

# SCL(FOL) Can Simulate Ground Nonredundant Ordered Resolution

Martin Desharnais

March 10, 2025

## Abstract

SCL(FOL) (i.e., Simple Clause Learning for First-Order Logic without equality) is known to be able to simulate the derivation of nonredundant clauses by the ground ordered resolution calculus [1]. Due to the space constraints of a 16-pages paper, the published proof is monolithic and hard to comprehend. In this work, we reuse the existing strategy for ground ordered resolution and present a new, simpler strategy for SCL(FOL). We prove a stronger bisimulation theorem between these two strategies (i.e., they both simulate each other). Our proof is modular: it consists of ten refinement steps focusing on different aspects of the two strategies.

## Contents

<b>1</b>	<b>Ground Resolution Calculus</b>	<b>4</b>
1.1	Ground Rules . . . . .	7
1.2	Ground Layer . . . . .	8
1.3	Correctness . . . . .	8
1.4	Redundancy Criterion . . . . .	10
1.5	Refutational Completeness . . . . .	12
1.6	Move to <i>HOL.Transitive-Closure</i> . . . . .	53
<b>2</b>	<b>Move to <i>HOL-Library.Multiset</i></b>	<b>57</b>
<b>3</b>	<b>Move to <i>HOL-Library.FSet</i></b>	<b>58</b>
<b>4</b>	<b>Move to <i>VeriComp.Simulation</i></b>	<b>59</b>
<b>5</b>	<b>Move to <i>Simple-Clause-Learning.SCL-FOL</i></b>	<b>61</b>
<b>6</b>	<b>Move to ground ordered resolution</b>	<b>64</b>
<b>7</b>	<b>Move somewhere?</b>	<b>66</b>

8	Ground ordered resolution for ground terms	74
9	Common definitions and lemmas	75
10	Lemmas about going between ground and first-order terms	91
11	SCL(FOL) for first-order terms	91
12	ORD-RES	94
13	ORD-RES-1 (deterministic)	95
14	Function for full factorization	99
15	ORD-RES-2 (full factorization)	109
16	Function for full resolution	121
17	ORD-RES-3 (full resolve)	145
18	Function for implicit full factorization	150
19	ORD-RES-4 (implicit factorization)	158
20	ORD-RES-5 (explicit model construction)	162
21	ORD-RES-6 (model backjump)	196
22	ORD-RES-7 (clause-guided literal trail construction)	233
23	ORD-RES-8 (atom-guided literal trail construction)	343
24	ORD-RES-9 (factorize when propagating)	374
25	ORD-RES-10 (propagate iff a conflict is produced)	384
26	ORD-RES-11 (SCL strategy)	412
27	ORD-RES-1 (deterministic)	443
28	ORD-RES-2 (full factorization)	445
29	ORD-RES-3 (full resolve)	461
30	ORD-RES-4 (implicit factorization)	478
31	ORD-RES-5 (explicit model construction)	483

<b>32 ORD-RES-6 (model backjump)</b>	<b>489</b>
<b>33 ORD-RES-7 (clause-guided literal trail construction)</b>	<b>501</b>
<b>34 ORD-RES-8 (atom-guided literal trail construction)</b>	<b>525</b>
<b>35 ORD-RES-9 (factorize when propagating)</b>	<b>562</b>
<b>36 ORD-RES-10 (propagate iff a conflict is produced)</b>	<b>564</b>
<b>37 ORD-RES-11 (SCL strategy)</b>	<b>572</b>
<b>38 ORD-RES-11 is a regular SCL strategy</b>	<b>588</b>
<b>theory Isabelle-2024-Compatibility</b>	
<b>imports</b>	
<i>Main</i>	
<i>HOL-Library.Multiset</i>	
<b>begin</b>	
<b>lemmas</b> <i>wfP-def = wfp-def</i>	
<b>lemmas</b> <i>wfP-if-convertible-to-wfP = wfp-if-convertible-to-wfp</i>	
<b>lemmas</b> <i>wfP-imp-asymp = wfp-imp-asymp</i>	
<b>lemmas</b> <i>wfP-induct-rule = wfp-induct-rule</i>	
<b>lemmas</b> <i>wfP-multp = wfp-multp</i>	
<b>end</b>	
<b>theory Ground-Ordered-Resolution</b>	
<b>imports</b>	
<i>Saturation-Framework.Calculus</i>	
<i>Saturation-Framework-Extensions.Clausal-Calculus</i>	
<i>Isabelle-2024-Compatibility</i>	
<i>First-Order-Clause.Ground-Context</i>	
<i>First-Order-Clause.HOL-Extra</i>	
<i>First-Order-Clause.Transitive-Closure-Extra</i>	
<i>Min-Max-Least-Greatest.Min-Max-Least-Greatest-FSet</i>	
<i>Min-Max-Least-Greatest.Min-Max-Least-Greatest-Multiset</i>	
<i>First-Order-Clause.Multiset-Extra</i>	
<i>Superposition-Calculus.Relation-Extra</i>	
<b>begin</b>	
<b>hide-type</b> <i>Inference-System.inference</i>	
<b>hide-const</b>	
<i>Inference-System.Infer</i>	
<i>Inference-System.prem-s-of</i>	
<i>Inference-System.concl-of</i>	
<i>Inference-System.main-prem-of</i>	
<b>primrec</b> <i>mset-lit :: 'a literal <math>\Rightarrow</math> 'a multiset where</i>	
<i>mset-lit (Pos A) = {#A#}  </i>	

$mset\text{-}lit (Neg A) = \{\#A, A\#\}$

**type-synonym**  $'t\ atom = 't$

## 1 Ground Resolution Calculus

**locale** *ground-ordered-resolution-calculus* =

**fixes**

$less\text{-}trm :: 'f\ gterm \Rightarrow 'f\ gterm \Rightarrow bool$  (**infix**  $\prec_t$  50) **and**

$select :: 'f\ gterm\ atom\ clause \Rightarrow 'f\ gterm\ atom\ clause$

**assumes**

$transp\text{-}less\text{-}trm[simp]: transp (\prec_t)$  **and**

$asympt\text{-}less\text{-}trm[intro]: asympt (\prec_t)$  **and**

$wfP\text{-}less\text{-}trm[intro]: wfP (\prec_t)$  **and**

$totalp\text{-}less\text{-}trm[intro]: totalp (\prec_t)$  **and**

$less\text{-}trm\text{-}compatible\text{-}with\text{-}gctxt[simp]: \bigwedge\ ctxt\ t\ t'. t \prec_t t' \Longrightarrow ctxt\langle t \rangle_G \prec_t ctxt\langle t' \rangle_G$

**and**

$less\text{-}trm\text{-}if\text{-}subterm[simp]: \bigwedge\ ctxt. ctxt \neq \square_G \Longrightarrow t \prec_t ctxt\langle t \rangle_G$  **and**

$select\text{-}subset: \bigwedge\ C. select\ C \subseteq\# C$  **and**

$select\text{-}negative\text{-}lits: \bigwedge\ C\ L. L \in\# select\ C \Longrightarrow is\text{-}neg\ L$

**begin**

**lemma** *irreflp-on-less-trm[simp]*: *irreflp-on*  $A (\prec_t)$

**by** (*metis* *asymptD* *asympt-less-trm* *irreflp-onI*)

**abbreviation** *lesseq-trm* (**infix**  $\preceq_t$  50) **where**

$lesseq\text{-}trm \equiv (\prec_t)^{==}$

**lemma** *lesseq-trm-if-subtermeq*:  $t \preceq_t ctxt\langle t \rangle_G$

**using** *less-trm-if-subterm*

**by** (*metis* *gctxt-ident-iff-eq-GHole* *reflclp-iff*)

**definition** *less-lit* ::

$'f\ gterm\ atom\ literal \Rightarrow 'f\ gterm\ atom\ literal \Rightarrow bool$  (**infix**  $\prec_l$  50) **where**

$less\text{-}lit\ L1\ L2 \equiv multp (\prec_t) (mset\text{-}lit\ L1) (mset\text{-}lit\ L2)$

**abbreviation** *lesseq-lit* (**infix**  $\preceq_l$  50) **where**

$lesseq\text{-}lit \equiv (\prec_l)^{==}$

**abbreviation** *less-cls* ::

$'f\ gterm\ atom\ clause \Rightarrow 'f\ gterm\ atom\ clause \Rightarrow bool$  (**infix**  $\prec_c$  50) **where**

$less\text{-}cls \equiv multp (\prec_l)$

**abbreviation** *lesseq-cls* (**infix**  $\preceq_c$  50) **where**

$lesseq\text{-}cls \equiv (\prec_c)^{==}$

**lemma** *transp-on-less-lit[simp]*: *transp-on*  $A (\prec_l)$

**by** (*smt* (*verit*, *best*) *less-lit-def* *transpE* *transp-less-trm* *transp-multp* *transp-onI*)

**corollary** *transp-less-lit*: *transp* ( $\prec_i$ )  
**by** *simp*

**lemma** *transp-less-cls*[*simp*]: *transp* ( $\prec_c$ )  
**by** (*simp add: transp-multp*)

**lemma** *asympt-on-less-lit*[*simp*]: *asympt-on* *A* ( $\prec_i$ )  
**by** (*metis asymptD asympt-less-trm asympt-multp<sub>HO</sub> asympt-onI less-lit-def multp-eq-multp<sub>HO</sub> transp-less-trm*)

**corollary** *asympt-less-lit*[*simp*]: *asympt* ( $\prec_i$ )  
**by** *simp*

**lemma** *asympt-less-cls*[*simp*]: *asympt* ( $\prec_c$ )  
**by** (*simp add: asympt-multp<sub>HO</sub> multp-eq-multp<sub>HO</sub>*)

**lemma** *irreflp-on-less-lit*[*simp*]: *irreflp-on* *A* ( $\prec_i$ )  
**by** (*simp only: asympt-on-less-lit irreflp-on-if-asympt-on*)

**lemma** *wfP-less-lit*[*simp*]: *wfP* ( $\prec_i$ )  
**unfolding** *less-lit-def*  
**using** *wfP-less-trm wfp-multp wfp-if-convertible-to-wfp* **by** *meson*

**lemma** *wfP-less-cls*[*simp*]: *wfP* ( $\prec_c$ )  
**using** *wfP-less-lit wfp-multp* **by** *blast*

**lemma** *totalp-on-less-lit*[*simp*]: *totalp-on* *A* ( $\prec_i$ )  
**proof** (*rule totalp-onI*)  
**fix** *L1 L2* :: '*f gterm atom literal*  
**assume** *L1*  $\neq$  *L2*

**show** *L1*  $\prec_i$  *L2*  $\vee$  *L2*  $\prec_i$  *L1*  
**unfolding** *less-lit-def*  
**proof** (*rule totalp-multp[THEN totalpD]*)  
**show** *totalp* ( $\prec_t$ )  
**using** *totalp-less-trm* .

**next**  
**show** *transp* ( $\prec_t$ )  
**using** *transp-less-trm* .

**next**  
**show** *mset-lit* *L1*  $\neq$  *mset-lit* *L2*  
**using**  $\langle L1 \neq L2 \rangle$   
**by** (*cases L1; cases L2*) (*auto simp add: add-eq-conv-ex*)

**qed**  
**qed**

**corollary** *totalp-less-lit*: *totalp* ( $\prec_i$ )  
**by** *simp*

**lemma** *totalp-less-cls*[simp]: *totalp* ( $\prec_c$ )  
**proof** (*rule totalp-multp*)  
  **show** *totalp* ( $\prec_l$ )  
    **by** *simp*  
**next**  
  **show** *transp* ( $\prec_l$ )  
    **by** *simp*  
**qed**

**interpretation** *term-order*: *linorder lesseq-trm less-trm*  
**proof** *unfold-locales*  
  **show**  $\bigwedge x y. (x \prec_t y) = (x \preceq_t y \wedge \neg y \preceq_t x)$   
    **by** (*metis asympD asymp-less-trm reflclp-iff*)  
**next**  
  **show**  $\bigwedge x. x \preceq_t x$   
    **by** *simp*  
**next**  
  **show**  $\bigwedge x y z. x \preceq_t y \implies y \preceq_t z \implies x \preceq_t z$   
    **by** (*meson transpE transp-less-trm transp-on-reflclp*)  
**next**  
  **show**  $\bigwedge x y. x \preceq_t y \implies y \preceq_t x \implies x = y$   
    **by** (*metis asympD asymp-less-trm reflclp-iff*)  
**next**  
  **show**  $\bigwedge x y. x \preceq_t y \vee y \preceq_t x$   
    **by** (*metis reflclp-iff totalpD totalp-less-trm*)  
**qed**

**interpretation** *literal-order*: *linorder lesseq-lit less-lit*  
**proof** *unfold-locales*  
  **show**  $\bigwedge x y. (x \prec_l y) = (x \preceq_l y \wedge \neg y \preceq_l x)$   
    **by** (*metis asympD asymp-less-lit reflclp-iff*)  
**next**  
  **show**  $\bigwedge x. x \preceq_l x$   
    **by** *simp*  
**next**  
  **show**  $\bigwedge x y z. x \preceq_l y \implies y \preceq_l z \implies x \preceq_l z$   
    **by** (*meson transpE transp-less-lit transp-on-reflclp*)  
**next**  
  **show**  $\bigwedge x y. x \preceq_l y \implies y \preceq_l x \implies x = y$   
    **by** (*metis asympD asymp-less-lit reflclp-iff*)  
**next**  
  **show**  $\bigwedge x y. x \preceq_l y \vee y \preceq_l x$   
    **by** (*metis reflclp-iff totalpD totalp-less-lit*)  
**qed**

**interpretation** *clause-order*: *linorder lesseq-cls less-cls*  
**proof** *unfold-locales*  
  **show**  $\bigwedge x y. (x \prec_c y) = (x \preceq_c y \wedge \neg y \preceq_c x)$   
    **by** (*metis asympD asymp-less-cls reflclp-iff*)

```

next
  show  $\bigwedge x. x \preceq_c x$ 
    by simp
next
  show  $\bigwedge x y z. x \preceq_c y \implies y \preceq_c z \implies x \preceq_c z$ 
    by (meson transpE transp-less-cls transp-on-reflclp)
next
  show  $\bigwedge x y. x \preceq_c y \implies y \preceq_c x \implies x = y$ 
    by (metis asympD asymp-less-cls reflclp-iff)
next
  show  $\bigwedge x y. x \preceq_c y \vee y \preceq_c x$ 
    by (metis reflclp-iff totalpD totalp-less-cls)
qed

```

```

lemma less-lit-simps[simp]:
  Pos A1 <l Pos A2  $\longleftrightarrow$  A1 <t A2
  Pos A1 <l Neg A2  $\longleftrightarrow$  A1  $\preceq_t$  A2
  Neg A1 <l Neg A2  $\longleftrightarrow$  A1 <t A2
  Neg A1 <l Pos A2  $\longleftrightarrow$  A1 <t A2
  by (auto simp add: less-lit-def)

```

## 1.1 Ground Rules

**abbreviation** *is-maximal-lit* :: 'f gterm literal  $\Rightarrow$  'f gterm clause  $\Rightarrow$  bool **where**  
*is-maximal-lit* L M  $\equiv$  *is-maximal-in-mset-wrt* (<<sub>l</sub>) M L

**abbreviation** *is-strictly-maximal-lit* :: 'f gterm literal  $\Rightarrow$  'f gterm clause  $\Rightarrow$  bool  
**where**  
*is-strictly-maximal-lit* L M  $\equiv$  *is-greatest-in-mset-wrt* (<<sub>l</sub>) M L

**inductive** *ground-resolution* ::  
'f gterm atom clause  $\Rightarrow$  'f gterm atom clause  $\Rightarrow$  'f gterm atom clause  $\Rightarrow$  bool  
**where**

```

ground-resolutionI:
  P1 = add-mset (Neg t) P1'  $\implies$ 
  P2 = add-mset (Pos t) P2'  $\implies$ 
  P2 <c P1  $\implies$ 
  select P1 = {#}  $\wedge$  is-maximal-lit (Neg t) P1  $\vee$  Neg t  $\in$  # select P1  $\implies$ 
  select P2 = {#}  $\implies$ 
  is-strictly-maximal-lit (Pos t) P2  $\implies$ 
  C = P1' + P2'  $\implies$ 
  ground-resolution P1 P2 C

```

**inductive** *ground-factoring* :: 'f gterm atom clause  $\Rightarrow$  'f gterm atom clause  $\Rightarrow$  bool  
**where**

```

ground-factoringI:
  P = add-mset (Pos t) (add-mset (Pos t) P')  $\implies$ 
  select P = {#}  $\implies$ 
  is-maximal-lit (Pos t) P  $\implies$ 

```

$C = \text{add-mset } (Pos\ t)\ P' \implies$   
 $\text{ground-factoring } P\ C$

## 1.2 Ground Layer

**definition**  $G\text{-Inf} :: 'f\ gterm\ atom\ clause\ inference\ set\ \mathbf{where}$

$G\text{-Inf} =$   
 $\{\text{Infer } [P_2, P_1]\ C \mid P_2\ P_1\ C.\ \text{ground-resolution } P_1\ P_2\ C\} \cup$   
 $\{\text{Infer } [P]\ C \mid P\ C.\ \text{ground-factoring } P\ C\}$

**abbreviation**  $G\text{-Bot} :: 'f\ gterm\ atom\ clause\ set\ \mathbf{where}$

$G\text{-Bot} \equiv \{\{\#\}\}$

**definition**  $G\text{-entails} :: 'f\ gterm\ atom\ clause\ set \Rightarrow 'f\ gterm\ atom\ clause\ set \Rightarrow bool$   
**where**

$G\text{-entails } N_1\ N_2 \iff (\forall (I :: 'f\ gterm\ set). I \models_s N_1 \longrightarrow I \models_s N_2)$

## 1.3 Correctness

**lemma** *soundness-ground-resolution:*

**assumes**

*step: ground-resolution P1 P2 C*

**shows**  $G\text{-entails } \{P_1, P_2\} \{C\}$

**using** *step*

**proof** (*cases P1 P2 C rule: ground-resolution.cases*)

**case** (*ground-resolutionI t P1' P2'*)

**show** *?thesis*

**unfolding** *G-entails-def true-cls-singleton*

**unfolding** *true-cls-insert*

**proof** (*intro allI impI, elim conjE*)

**fix**  $I :: 'f\ gterm\ set$

**assume**  $I \models P_1$  **and**  $I \models P_2$

**then obtain**  $K_1\ K_2 :: 'f\ gterm\ atom\ literal\ \mathbf{where}$

$K_1 \in\# P_1$  **and**  $I \models_l K_1$  **and**  $K_2 \in\# P_2$  **and**  $I \models_l K_2$

**by** (*auto simp: true-cls-def*)

**show**  $I \models C$

**proof** (*cases K1 = Neg t*)

**case** *K1-def: True*

**hence**  $I \models_l Neg\ t$

**using**  $\langle I \models_l K_1 \rangle$  **by** *simp*

**show** *?thesis*

**proof** (*cases K2 = Pos t*)

**case** *K2-def: True*

**hence**  $I \models_l Pos\ t$

**using**  $\langle I \models_l K_2 \rangle$  **by** *simp*

**hence** *False*

**using**  $\langle I \models_l Neg\ t \rangle$  **by** *simp*



```

    thus ?thesis ..
  next
    case False
    hence  $K2 \in\# P_2'$ 
      using  $\langle K2 \in\# P2 \rangle$ 
      unfolding ground-resolutionI by simp
    hence  $I \Vdash P_2'$ 
      using  $\langle I \Vdash K2 \rangle$  by blast
    thus ?thesis
      unfolding ground-resolutionI by simp
  qed
next
  case False
  hence  $K1 \in\# P_1'$ 
    using  $\langle K1 \in\# P1 \rangle$ 
    unfolding ground-resolutionI by simp
  hence  $I \Vdash P_1'$ 
    using  $\langle I \Vdash K1 \rangle$  by blast
  thus ?thesis
    unfolding ground-resolutionI by simp
  qed
qed
qed

```

lemma *soundness-ground-factoring*:

```

  assumes step: ground-factoring  $P C$ 
  shows  $G$ -entails  $\{P\} \{C\}$ 
  using step
proof (cases  $P C$  rule: ground-factoring.cases)
  case (ground-factoringI  $t P'$ )
  show ?thesis
    unfolding G-entails-def true-cls-singleton
  proof (intro allI impI)
    fix  $I :: 'f$  gterm set
    assume  $I \Vdash P$ 
    then obtain  $K :: 'f$  gterm atom literal where
       $K \in\# P$  and  $I \Vdash K$ 
      by (auto simp: true-cls-def)

    show  $I \Vdash C$ 
  proof (cases  $K = Pos\ t$ )
    case True
    hence  $I \Vdash Pos\ t$ 
      using  $\langle I \Vdash K \rangle$  by metis
    thus ?thesis
      unfolding ground-factoringI
      by (metis true-cls-add-mset)
  next
    case False

```

```

    hence  $K \in\# P'$ 
      using  $\langle K \in\# P \rangle$ 
      unfolding ground-factoringI
      by auto
    hence  $K \in\# C$ 
      unfolding ground-factoringI
      by simp
    thus ?thesis
      using  $\langle I \Vdash K \rangle$  by blast
  qed
qed
qed

```

**interpretation** *G*: *sound-inference-system G-Inf G-Bot G-entails*

**proof** *unfold-locales*

```

  show  $G\text{-Bot} \neq \{\}$ 

```

```

    by simp

```

**next**

```

  show  $\bigwedge B N. B \in G\text{-Bot} \implies G\text{-entails } \{B\} N$ 

```

```

    by (simp add: G-entails-def)

```

**next**

```

  show  $\bigwedge N2 N1. N2 \subseteq N1 \implies G\text{-entails } N1 N2$ 

```

```

    by (auto simp: G-entails-def elim!: true-cls-mono[rotated])

```

**next**

```

  fix  $N1 N2$  assume ball-G-entails:  $\forall C \in N2. G\text{-entails } N1 \{C\}$ 

```

```

  show  $G\text{-entails } N1 N2$ 

```

```

    unfolding G-entails-def

```

```

  proof (intro allI impI)

```

```

    fix  $I :: 'f \text{ gterm set}$ 

```

```

    assume  $I \Vdash N1$ 

```

```

    hence  $\forall C \in N2. I \Vdash \{C\}$ 

```

```

      using ball-G-entails by (simp add: G-entails-def)

```

```

    then show  $I \Vdash N2$ 

```

```

      by (simp add: true-cls-def)

```

```

  qed

```

**next**

```

  show  $\bigwedge N1 N2 N3. G\text{-entails } N1 N2 \implies G\text{-entails } N2 N3 \implies G\text{-entails } N1 N3$ 

```

```

    using G-entails-def by simp

```

**next**

```

  show  $\bigwedge \iota. \iota \in G\text{-Inf} \implies G\text{-entails } (\text{set } (\text{prems-of } \iota)) \{\text{concl-of } \iota\}$ 

```

```

    unfolding G-Inf-def

```

```

    using soundness-ground-resolution

```

```

    using soundness-ground-factoring

```

```

    by (auto simp: G-entails-def)

```

**qed**

## 1.4 Redundancy Criterion

**lemma** *ground-resolution-smaller-conclusion*:

```

assumes
  step: ground-resolution P1 P2 C
shows  $C \prec_c P1$ 
using step
proof (cases P1 P2 C rule: ground-resolution.cases)
  case (ground-resolutionI t P1' P2')
  have  $\forall k \in \#P_2'. k \prec_l Pos\ t$ 
    using  $\langle is\_strictly\_maximal\_lit\ (Pos\ t)\ P_2 \rangle \langle P_2 = add\_mset\ (Pos\ t)\ P_2' \rangle$ 
    by (simp add: literal-order.is-greatest-in-mset-iff)
  moreover have  $\bigwedge A. Pos\ A \prec_l Neg\ A$ 
    by (simp add: less-lit-def)
  ultimately have  $\forall k \in \#P_2'. k \prec_l Neg\ t$ 
    by (metis transp-def transp-less-lit)
  hence  $P_2' \prec_c \{\#Neg\ t\#}$ 
    using one-step-implies-multp[of  $\{\#Neg\ t\#$   $P_2'$   $(\prec_l)$   $\{\#\}$ ] by simp
  hence  $P_2' + P_1' \prec_c add\_mset\ (Neg\ t)\ P_1'$ 
    using multp-cancel[of  $(\prec_l)$   $P_1'$   $P_2'$   $\{\#Neg\ t\#$ ] by simp
  thus ?thesis
    unfolding ground-resolutionI
    by (simp only: add.commute)
qed

lemma ground-factoring-smaller-conclusion:
  assumes step: ground-factoring P C
  shows  $C \prec_c P$ 
  using step
proof (cases P C rule: ground-factoring.cases)
  case (ground-factoringI t P')
  then show ?thesis
    by (metis add-mset-add-single mset-subset-eq-exists-conv multi-self-add-other-not-self
      multp-subset-supersetI totalpD totalp-less-cls transp-less-lit)
qed

interpretation G: calculus-with-finitary-standard-redundancy G-Inf G-Bot G-entails
  ( $\prec_c$ )
proof unfold-locales
  show transp ( $\prec_c$ )
    using transp-less-cls .
  next
  show wfP ( $\prec_c$ )
    using wfP-less-cls .
  next
  show  $\bigwedge \iota. \iota \in G-Inf \implies prems-of\ \iota \neq []$ 
    by (auto simp: G-Inf-def)
  next
  fix  $\iota$ 
  have concl-of  $\iota \prec_c$  main-prem-of  $\iota$ 
    if  $\iota$ -def:  $\iota = Infer\ [P_2, P_1]\ C$  and
      infer: ground-resolution P1 P2 C

```

```

for  $P_2 P_1 C$ 
unfolding  $\iota$ -def
using infer
using ground-resolution-smaller-conclusion
by simp

moreover have  $\text{concl-of } \iota \prec_c \text{ main-prem-of } \iota$ 
if  $\iota$ -def:  $\iota = \text{Infer } [P] C$  and
  infer: ground-factoring  $P C$ 
for  $P C$ 
unfolding  $\iota$ -def
using infer
using ground-factoring-smaller-conclusion
by simp

ultimately show  $\iota \in G\text{-Inf} \implies \text{concl-of } \iota \prec_c \text{ main-prem-of } \iota$ 
unfolding  $G\text{-Inf-def}$ 
by fast
qed

```

## 1.5 Refutational Completeness

**context**

**fixes**  $N :: 'f \text{ gterm atom clause set}$

**begin**

**function**  $\text{production} :: 'f \text{ gterm atom clause} \Rightarrow 'f \text{ gterm set}$  **where**

$\text{production } C = \{A \mid A C'\}$ .

$C \in N \wedge$

$C = \text{add-mset } (\text{Pos } A) C' \wedge$

$\text{select } C = \{\#\} \wedge$

$\text{is-strictly-maximal-lit } (\text{Pos } A) C \wedge$

$\neg (\bigcup D \in \{D \in N. D \prec_c C\}. \text{production } D) \models C\}$

**by**  $\text{simp-all}$

**termination**  $\text{production}$

**proof** ( $\text{relation } \{(x, y). x \prec_c y\}$ )

**show**  $\text{wf } \{(x, y). x \prec_c y\}$

**using**  $\text{wfp-less-cls}$

**by** ( $\text{simp add: wfp-def}$ )

**next**

**show**  $\bigwedge C D. D \in \{D \in N. D \prec_c C\} \implies (D, C) \in \{(x, y). x \prec_c y\}$

**by**  $\text{simp}$

**qed**

**declare**  $\text{production.simps}[\text{simp del}]$

**end**

**lemma** *Uniq-strictly-maximal-lit-in-ground-clc:*

$\exists_{\leq 1} L. \text{is-strictly-maximal-lit } L \ C$

**proof** (rule *Uniq-is-greatest-in-mset-wrt*)

**show** *transp-on* (set-mset  $C$ ) ( $\prec_l$ )

by (auto intro: *transp-on-subset transp-less-lit*)

**next**

**show** *asypm-on* (set-mset  $C$ ) ( $\prec_l$ )

by (auto intro: *asypm-on-subset asypm-less-lit*)

**next**

**show** *totalp-on* (set-mset  $C$ ) ( $\prec_l$ )

by (auto intro: *totalp-on-subset totalp-less-lit*)

**qed**

**lemma** *production-eq-empty-or-singleton:*

$\text{production } N \ C = \{\} \vee (\exists A. \text{production } N \ C = \{A\})$

**proof** –

**have**  $\exists_{\leq 1} A. \text{is-strictly-maximal-lit } (\text{Pos } A) \ C$

using *Uniq-strictly-maximal-lit-in-ground-clc*

by (metis (mono-tags, lifting) *Uniq-def literal.inject(1)*)

**hence**  $\exists_{\leq 1} A. \exists C'. C = \text{add-mset } (\text{Pos } A) \ C' \wedge \text{is-strictly-maximal-lit } (\text{Pos } A)$

$C$

by (simp add: *Uniq-def*)

**hence** *Uniq-production*:  $\exists_{\leq 1} A. \exists C'.$

$C \in N \wedge$

$C = \text{add-mset } (\text{Pos } A) \ C' \wedge$

$\text{select } C = \{\#\} \wedge$

$\text{is-strictly-maximal-lit } (\text{Pos } A) \ C \wedge$

$\neg (\bigcup D \in \{D \in N. D \prec_c C\}. \text{production } N \ D) \Vdash C$

using *Uniq-antimono'*

by (smt (verit) *Uniq-def Uniq-prodI case-prod-conv*)

**show** *?thesis*

**unfolding** *production.simps*[of  $N \ C$ ]

using *Collect-eq-if-Uniq*[OF *Uniq-production*]

by (smt (verit, best) *Collect-cong Collect-empty-eq Uniq-def Uniq-production*

*case-prod-conv*

*insertCI mem-Collect-eq*)

**qed**

**lemma** *production-eq-singleton-if-atom-in-production:*

**assumes**  $A \in \text{production } N \ C$

**shows**  $\text{production } N \ C = \{A\}$

using *assms production-eq-empty-or-singleton*

by *force*

**definition** *interp where*

$\text{interp } N \ C \equiv (\bigcup D \in \{D \in N. D \prec_c C\}. \text{production } N \ D)$

**lemma** *interp-mempty*[simp]:  $\text{interp } N \ \{\#\} = \{\}$

**proof** –

**have**  $\# C. C \prec_c \{\#\}$   
**by** (*metis clause-order.order.asym subset-implies-multp subset-mset.gr-zeroI*)  
**hence**  $\{D \in N. D \prec_c \{\#\}\} = \{\}$   
**by** *simp*  
**thus** *?thesis*  
**unfolding** *interp-def* **by** *auto*  
**qed**

**lemma** *production-unfold*:  $\text{production } N C = \{A \mid A C'\}$ .  
 $C \in N \wedge$   
 $C = \text{add-mset } (\text{Pos } A) C' \wedge$   
 $\text{select } C = \{\#\} \wedge$   
 $\text{is-strictly-maximal-lit } (\text{Pos } A) C \wedge$   
 $\neg \text{interp } N C \models C\}$   
**by** (*simp add: production.simps[of N C] interp-def*)

**lemma** *production-unfold'*:  $\text{production } N C = \{A \mid A.$   
 $C \in N \wedge$   
 $\text{select } C = \{\#\} \wedge$   
 $\text{is-strictly-maximal-lit } (\text{Pos } A) C \wedge$   
 $\neg \text{interp } N C \models C\}$   
**unfolding** *production-unfold*  
**by** (*metis (mono-tags, lifting) literal-order.explode-greatest-in-mset*)

**lemma** *mem-productionE*:  
**assumes** *C-prod*:  $A \in \text{production } N C$   
**obtains**  $C'$  **where**  
 $C \in N$  **and**  
 $C = \text{add-mset } (\text{Pos } A) C'$  **and**  
 $\text{select } C = \{\#\}$  **and**  
 $\text{is-strictly-maximal-lit } (\text{Pos } A) C$  **and**  
 $\neg \text{interp } N C \models C$   
**using** *C-prod*  
**unfolding** *production.simps[of N C] mem-Collect-eq Let-def interp-def*  
**by** (*metis (no-types, lifting)*)

**lemma** *production-subset-if-less-cls*:  $C \prec_c D \implies \text{production } N C \subseteq \text{interp } N D$   
**unfolding** *interp-def*  
**using** *production-unfold* **by** *blast*

**lemma** *Uniq-production-eq-singleton*:  $\exists_{\leq 1} C. \text{production } N C = \{A\}$   
**proof** (*rule Uniq-I*)  
**fix**  $C D$   
**assume**  $\text{production } N C = \{A\}$   
**hence**  $A \in \text{production } N C$   
**by** *simp*  
**then obtain**  $C'$  **where**  
 $C \in N$   
 $C = \text{add-mset } (\text{Pos } A) C'$

*is-strictly-maximal-lit* (Pos A) C  
 $\neg \text{interp } N C \models C$   
**by** (*auto elim!*: *mem-productionE*)

**assume** *production* N D = {A}  
**hence**  $A \in \text{production } N D$   
**by** *simp*  
**then obtain** D' **where**  
 $D \in N$   
 $D = \text{add-mset } (\text{Pos } A) D'$   
*is-strictly-maximal-lit* (Pos A) D  
 $\neg \text{interp } N D \models D$   
**by** (*auto elim!*: *mem-productionE*)

**have**  $\neg (C \prec_c D)$   
**proof** (*rule notI*)  
**assume**  $C \prec_c D$   
**hence** *production* N C  $\subseteq$  *interp* N D  
**using** *production-subset-if-less-cls* **by** *metis*  
**hence**  $A \in \text{interp } N D$   
**unfolding**  $\langle \text{production } N C = \{A\} \rangle$  **by** *simp*  
**hence** *interp* N D  $\models D$   
**unfolding**  $\langle D = \text{add-mset } (\text{Pos } A) D' \rangle$  **by** *simp*  
**with**  $\langle \neg \text{interp } N D \models D \rangle$  **show** *False*  
**by** *metis*  
**qed**

**moreover have**  $\neg (D \prec_c C)$   
**proof** (*rule notI*)  
**assume**  $D \prec_c C$   
**hence** *production* N D  $\subseteq$  *interp* N C  
**using** *production-subset-if-less-cls* **by** *metis*  
**hence**  $A \in \text{interp } N C$   
**unfolding**  $\langle \text{production } N D = \{A\} \rangle$  **by** *simp*  
**hence** *interp* N C  $\models C$   
**unfolding**  $\langle C = \text{add-mset } (\text{Pos } A) C' \rangle$  **by** *simp*  
**with**  $\langle \neg \text{interp } N C \models C \rangle$  **show** *False*  
**by** *metis*  
**qed**

**ultimately show**  $C = D$   
**by** *order*  
**qed**

**lemma** *singleton-eq-CollectD*:  $\{x\} = \{y. P y\} \implies P x$   
**by** *blast*

**lemma** *subset-Union-mem-CollectI*:  $P x \implies f x \subseteq (\bigcup y \in \{z. P z\}. f y)$   
**by** *blast*

**lemma** *interp-subset-if-less-cls*:  $C \prec_c D \implies \text{interp } N C \subseteq \text{interp } N D$   
**by** (*smt (verit, best) UN-iff mem-Collect-eq interp-def subsetI transpD transp-less-cls*)

**lemma** *interp-subset-if-less-cls'*:  $C \prec_c D \implies \text{interp } N C \subseteq \text{interp } N D \cup \text{production } N D$   
**using** *interp-subset-if-less-cls* **by** *blast*

**lemma** *split-Union-production*:

**assumes** *D-in*:  $D \in N$

**shows**  $(\bigcup C \in N. \text{production } N C) =$

$\text{interp } N D \cup \text{production } N D \cup (\bigcup C \in \{C \in N. D \prec_c C\}. \text{production } N C)$

**proof** –

**have**  $N = \{C \in N. C \prec_c D\} \cup \{D\} \cup \{C \in N. D \prec_c C\}$

**proof** (*rule partition-set-around-element*)

**show** *totalp-on*  $N (\prec_c)$

**using** *totalp-less-cls totalp-on-subset* **by** *blast*

**next**

**show**  $D \in N$

**using** *D-in* **by** *simp*

**qed**

**hence**  $(\bigcup C \in N. \text{production } N C) =$

$(\bigcup C \in \{C \in N. C \prec_c D\}. \text{production } N C) \cup \text{production } N D \cup (\bigcup C \in \{C \in N. D \prec_c C\}. \text{production } N C)$

**by** *auto*

**thus**  $(\bigcup C \in N. \text{production } N C) =$

$\text{interp } N D \cup \text{production } N D \cup (\bigcup C \in \{C \in N. D \prec_c C\}. \text{production } N C)$

**by** (*simp add: interp-def*)

**qed**

**lemma** *split-Union-production'*:

**assumes** *D-in*:  $D \in N$

**shows**  $(\bigcup C \in N. \text{production } N C) = \text{interp } N D \cup (\bigcup C \in \{C \in N. D \preceq_c C\}. \text{production } N C)$

**using** *split-Union-production[OF D-in]* *D-in* **by** *auto*

**lemma** *split-interp*:

**assumes**  $C \in N$  **and** *D-in*:  $D \in N$  **and**  $D \prec_c C$

**shows**  $\text{interp } N C = \text{interp } N D \cup (\bigcup C' \in \{C' \in N. D \preceq_c C' \wedge C' \prec_c C\}. \text{production } N C')$

**proof** –

**have**  $\{D \in N. D \prec_c C\} =$

$\{y \in \{D \in N. D \prec_c C\}. y \prec_c D\} \cup \{D\} \cup \{y \in \{D \in N. D \prec_c C\}. D \prec_c y\}$

**proof** (*rule partition-set-around-element*)

**show** *totalp-on*  $\{D \in N. D \prec_c C\} (\prec_c)$

**using** *totalp-less-cls totalp-on-subset* **by** *blast*

**next**

**from** *D-in*  $\langle D \prec_c C \rangle$  **show**  $D \in \{D \in N. D \prec_c C\}$



by *simp*  
**qed**  
**also have**  $\dots = \{x \in N. x \prec_c C \wedge x \prec_c D\} \cup \{D\} \cup \{x \in N. D \prec_c x \wedge x \prec_c C\}$   
 by *auto*  
**also have**  $\dots = \{x \in N. x \prec_c D\} \cup \{D\} \cup \{x \in N. D \prec_c x \wedge x \prec_c C\}$   
 using  $\langle D \prec_c C \rangle$  *transp-less-cls*  
 by (*metis* (*no-types*, *opaque-lifting*) *transpD*)  
**finally have** *Collect-N-lt-C*:  $\{x \in N. x \prec_c C\} = \{x \in N. x \prec_c D\} \cup \{x \in N. D \preceq_c x \wedge x \prec_c C\}$   
 by *auto*

**have** *interp N C* =  $(\bigcup C' \in \{D \in N. D \prec_c C\}. \text{production } N C')$   
 by (*simp add: interp-def*)  
**also have**  $\dots = (\bigcup C' \in \{x \in N. x \prec_c D\}. \text{production } N C') \cup (\bigcup C' \in \{x \in N. D \preceq_c x \wedge x \prec_c C\}. \text{production } N C')$   
 unfolding *Collect-N-lt-C* by *simp*  
**finally show** *interp N C* = *interp N D*  $\cup \bigcup (\text{production } N ' \{C' \in N. D \preceq_c C' \wedge C' \prec_c C\})$   
 unfolding *interp-def* by *simp*  
**qed**

**lemma** *less-imp-Interp-subseteq-interp*:  $C \prec_c D \implies \text{interp } N C \cup \text{production } N C \subseteq \text{interp } N D$   
 using *interp-subset-if-less-cls* *production-subset-if-less-cls*  
 by (*simp add: interp-def*)

**lemma** *not-interp-to-Interp-imp-le*:  $A \notin \text{interp } N C \implies A \in \text{interp } N D \cup \text{production } N D \implies C \preceq_c D$   
 using *less-imp-Interp-subseteq-interp*  
 by (*metis* (*mono-tags*, *opaque-lifting*) *subsetD sup2CI totalpD totalp-less-cls*)

**lemma** *produces-imp-in-interp*:  
**assumes** *Neg A*  $\in \# C$  **and** *D-prod*:  $A \in \text{production } N D$   
**shows**  $A \in \text{interp } N C$   
**proof** –  
**from** *D-prod* **have** *Pos A*  $\in \# D$  **and** *is-strictly-maximal-lit* (*Pos A*) *D*  
 by (*auto elim: mem-productionE*)

**have**  $D \prec_c C$   
**proof** (*rule multp-if-maximal-of-lhs-is-less*)  
**show** *Pos A*  $\in \# D$   
 using  $\langle \text{Pos } A \in \# D \rangle$  .  
**next**  
**show** *Neg A*  $\in \# C$   
 using  $\langle \text{Neg } A \in \# C \rangle$  .  
**next**  
**show** *Pos A*  $\prec_l \text{Neg } A$   
 by (*simp add: less-lit-def subset-implies-multp*)

**next**  
**show** *is-maximal-lit* (Pos A) D  
**using**  $\langle \text{is-strictly-maximal-lit (Pos A) D} \rangle$  **by** *auto*  
**qed** *simp-all*  
**hence**  $\neg (\prec_c)^{==} C D$   
**by** (*metis asympD asymp-less-cls reflclp-iff*)  
**thus** *?thesis*  
**proof** (*rule contrapos-np*)  
**from** *D-prod* **show**  $A \notin \text{interp } N C \implies (\prec_c)^{==} C D$   
**using** *not-interp-to-Interp-imp-le* **by** *simp*  
**qed**  
**qed**

**lemma** *neg-notin-Interp-not-produce*:

*Neg A*  $\in \# C \implies A \notin \text{interp } N D \cup \text{production } N D \implies C \preceq_c D \implies A \notin$   
*production* *N D''*  
**using** *interp-subset-if-less-cls'*  
**by** (*metis Un-iff produces-imp-in-interp reflclp-iff sup.orderE*)

**lemma** *lift-interp-entails*:

**assumes**  
*D-in*:  $D \in N$  **and**  
*D-entailed*:  $\text{interp } N D \Vdash D$  **and**  
*C-in*:  $C \in N$  **and**  
*D-lt-C*:  $D \prec_c C$   
**shows**  $\text{interp } N C \Vdash D$   
**proof** –  
**from** *D-entailed* **obtain** *L A* **where**  
*L-in*:  $L \in \# D$  **and**  
*L-eq-disj-L-eq*:  $L = \text{Pos } A \wedge A \in \text{interp } N D \vee L = \text{Neg } A \wedge A \notin \text{interp } N D$   
**unfolding** *true-cls-def true-lit-iff* **by** *metis*

**have**  $\text{interp } N D \subseteq \text{interp } N C$   
**using** *interp-subset-if-less-cls[OF D-lt-C]* .

**from** *L-eq-disj-L-eq* **show**  $\text{interp } N C \Vdash D$

**proof** (*elim disjE conjE*)

**assume**  $L = \text{Pos } A$  **and**  $A \in \text{interp } N D$

**thus**  $\text{interp } N C \Vdash D$

**using** *L-in*  $\langle \text{interp } N D \subseteq \text{interp } N C \rangle$  **by** *auto*

**next**

**assume**  $L = \text{Neg } A$  **and**  $A \notin \text{interp } N D$

**hence**  $A \notin \text{interp } N C$

**using** *neg-notin-Interp-not-produce*

**by** (*smt (verit, ccfv-threshold) L-in UN-E interp-def produces-imp-in-interp*)

**thus**  $\text{interp } N C \Vdash D$

**using** *L-in*  $\langle L = \text{Neg } A \rangle$  **by** *blast*

**qed**

**qed**

**lemma** *lift-interp-entails-to-interp-production-entails:*

**assumes**

*C-in:*  $C \in N$  **and**

*D-in:*  $D \in N$  **and**

*C-lt-D:*  $D \prec_c C$  **and**

*D-entailed:*  $\text{interp } N C \models D$

**shows**  $\text{interp } N C \cup \text{production } N C \models D$

**proof** –

**from** *D-entailed* **obtain**  $L A$  **where**

*L-in:*  $L \in \# D$  **and**

*L-eq-disj-L-eq:*  $L = \text{Pos } A \wedge A \in \text{interp } N C \vee L = \text{Neg } A \wedge A \notin \text{interp } N C$

**unfolding** *true-cls-def true-lit-iff* **by** *metis*

**from** *L-eq-disj-L-eq* **show**  $\text{interp } N C \cup \text{production } N C \models D$

**proof** (*elim disjE conjE*)

**assume**  $L = \text{Pos } A$  **and**  $A \in \text{interp } N C$

**thus**  $\text{interp } N C \cup \text{production } N C \models D$

**using** *L-in* **by** *blast*

**next**

**assume**  $L = \text{Neg } A$  **and**  $A \notin \text{interp } N C$

**hence**  $A \notin \text{interp } N C \cup \text{production } N C$

**using** *neg-notin-Interp-not-produce*

**by** (*metis (no-types, opaque-lifting) C-lt-D L-in Un-iff interp-subset-if-less-cls produces-imp-in-interp subsetD*)

**thus**  $\text{interp } N C \cup \text{production } N C \models D$

**using** *L-in*  $\langle L = \text{Neg } A \rangle$  **by** *blast*

**qed**

**qed**

**lemma** *lift-entailment-to-Union:*

**fixes**  $N D$

**assumes**

*D-in:*  $D \in N$  **and**

*R<sub>D</sub>-entails-D:*  $\text{interp } N D \models D$

**shows**

$(\bigcup C \in N. \text{production } N C) \models D$

**using** *lift-interp-entails*

**by** (*smt (verit, best) D-in R<sub>D</sub>-entails-D UN-iff produces-imp-in-interp split-Union-production' subsetD sup-ge1 true-cls-def true-lit-iff*)

**lemma**

**assumes**

$D \preceq_c C$  **and**

*C-prod:*  $A \in \text{production } N C$  **and**

*L-in:*  $L \in \# D$

**shows**

*lesseq-trm-if-pos:*  $\text{is-pos } L \implies \text{atm-of } L \preceq_t A$  **and**

$less\text{-}trm\text{-}if\text{-}neg: is\text{-}neg\ L \implies atm\text{-}of\ L \prec_t A$   
**proof** –  
**from**  $C\text{-}prod$  **obtain**  $C'$  **where**  
 $C\text{-}def: C = add\text{-}mset\ (Pos\ A)\ C'$  **and**  
 $C\text{-}max\text{-}lit: is\text{-}strictly\text{-}maximal\text{-}lit\ (Pos\ A)\ C$   
**by**  $(auto\ elim: mem\text{-}productionE)$

**have**  $Pos\ A \prec_l L$  **if**  $is\text{-}pos\ L$  **and**  $\neg atm\text{-}of\ L \preceq_t A$   
**proof** –  
**from**  $that(2)$  **have**  $A \prec_t atm\text{-}of\ L$   
**using**  $totalp\text{-}less\text{-}trm[THEN\ totalpD]$  **by**  $auto$   
**hence**  $multp\ (\prec_t)\ \{\#A\#\}\ \{\#atm\text{-}of\ L\#\}$   
**by**  $(smt\ (verit,\ del\text{-}insts)\ add.\text{right}\text{-}neutral\ empty\text{-}iff\ insert\text{-}iff\ one\text{-}step\text{-}implies\text{-}multp\ set\text{-}mset\text{-}add\text{-}mset\text{-}insert\ set\text{-}mset\text{-}empty\ transpD\ transp\text{-}less\text{-}trm\ union\text{-}mset\text{-}add\text{-}mset\text{-}right)$   
**with**  $that(1)$  **show**  $Pos\ A \prec_l L$   
**by**  $(cases\ L)\ (simp\text{-}all\ add: less\text{-}lit\text{-}def)$   
**qed**

**moreover have**  $Pos\ A \prec_l L$  **if**  $is\text{-}neg\ L$  **and**  $\neg atm\text{-}of\ L \prec_t A$   
**proof** –  
**from**  $that(2)$  **have**  $A \preceq_t atm\text{-}of\ L$   
**using**  $totalp\text{-}less\text{-}trm[THEN\ totalpD]$  **by**  $auto$   
**hence**  $multp\ (\prec_t)\ \{\#A\#\}\ \{\#atm\text{-}of\ L,\ atm\text{-}of\ L\#\}$   
**by**  $(smt\ (z3)\ add\text{-}mset\text{-}add\text{-}single\ add\text{-}mset\text{-}remove\text{-}trivial\ add\text{-}mset\text{-}remove\text{-}trivial\text{-}iff\ empty\text{-}not\text{-}add\text{-}mset\ insert\text{-}DiffM\ insert\text{-}noteq\text{-}member\ one\text{-}step\text{-}implies\text{-}multp\ reflclp\text{-}iff\ transp\text{-}def\ transp\text{-}less\text{-}trm\ union\text{-}mset\text{-}add\text{-}mset\text{-}left\ union\text{-}mset\text{-}add\text{-}mset\text{-}right)$   
**with**  $that(1)$  **show**  $Pos\ A \prec_l L$   
**by**  $(cases\ L)\ (simp\text{-}all\ add: less\text{-}lit\text{-}def)$   
**qed**

**moreover have**  $False$  **if**  $Pos\ A \prec_l L$   
**proof** –  
**have**  $C \prec_c D$   
**proof**  $(rule\ multp\text{-}if\text{-}maximal\text{-}of\text{-}lhs\text{-}is\text{-}less)$   
**show**  $Pos\ A \in\#\ C$   
**by**  $(simp\ add: C\text{-}def)$   
**next**  
**show**  $L \in\#\ D$   
**using**  $L\text{-}in$  **by**  $simp$   
**next**  
**show**  $is\text{-}maximal\text{-}lit\ (Pos\ A)\ C$   
**using**  $C\text{-}max\text{-}lit$  **by**  $auto$   
**next**  
**show**  $Pos\ A \prec_l L$   
**using**  $that$  **by**  $simp$   
**qed**  $simp\text{-}all$   
**with**  $\langle D \preceq_c C \rangle$  **show**  $False$   
**by**  $(metis\ asympD\ reflclp\text{-}iff\ wfp\text{-}imp\text{-}asymp\ wfp\text{-}less\text{-}cls)$

qed

ultimately show  $is\text{-}pos\ L \implies atm\text{-}of\ L \preceq_t A$  and  $is\text{-}neg\ L \implies atm\text{-}of\ L \prec_t A$   
by *metis+*

qed

**lemma** *less-trm-iff-less-cls-if-mem-production:*

assumes  $C\text{-}prod: A_C \in production\ N\ C$  and  $D\text{-}prod: A_D \in production\ N\ D$   
shows  $A_C \prec_t A_D \iff C \prec_c D$

**proof** –

from  $C\text{-}prod$  obtain  $C'$  where

$C \in N$  and

$C\text{-}def: C = add\text{-}mset\ (Pos\ A_C)\ C'$  and

$is\text{-}strictly\text{-}maximal\text{-}lit\ (Pos\ A_C)\ C$

by (*elim mem-productionE*) *simp*

hence  $\forall L \in \# C'. L \prec_l Pos\ A_C$

by (*simp add: literal-order.is-greatest-in-mset-iff*)

from  $D\text{-}prod$  obtain  $D'$  where

$D \in N$  and

$D\text{-}def: D = add\text{-}mset\ (Pos\ A_D)\ D'$  and

$is\text{-}strictly\text{-}maximal\text{-}lit\ (Pos\ A_D)\ D$

by (*elim mem-productionE*) *simp*

hence  $\forall L \in \# D'. L \prec_l Pos\ A_D$

by (*simp add: literal-order.is-greatest-in-mset-iff*)

show *?thesis*

**proof** (*rule iffI*)

assume  $A_C \prec_t A_D$

hence  $Pos\ A_C \prec_l Pos\ A_D$

by (*simp add: less-lit-def*)

moreover hence  $\forall L \in \# C'. L \prec_l Pos\ A_D$

using  $\langle \forall L \in \# C'. L \prec_l Pos\ A_C \rangle$

by (*meson transp-less-lit transpD*)

ultimately show  $C \prec_c D$

using *one-step-implies-multp*[of  $D\ C - \{\#\}$ ]

by (*simp add: D-def C-def*)

**next**

assume  $C \prec_c D$

hence  $production\ N\ C \subseteq (\bigcup (production\ N\ \{x \in N. x \prec_c D\}))$

using  $\langle C \in N \rangle$  by *auto*

hence  $A_C \in (\bigcup (production\ N\ \{x \in N. x \prec_c D\}))$

using  $C\text{-}prod$  by *auto*

hence  $A_C \neq A_D$

by (*metis D-prod interp-def mem-productionE true-cls-add-mset true-lit-iff*)

moreover have  $\neg (A_D \prec_t A_C)$

by (*metis C-def D-prod*  $\langle C \prec_c D \rangle$  *asymptD asympt-less-trm lesseq-trm-if-pos*  
*literal.disc(1)*)

*literal.sel(1) reflclp-iff union-single-eq-member*)

**ultimately show**  $A_C \prec_t A_D$   
**using** *totalp-less-trm*[*THEN totalpD*]  
**using** *C-def D-def* **by** *auto*  
**qed**  
**qed**

**lemma** *false-cls-if-productive-production*:  
**assumes** *C-prod*:  $A \in \text{production } N \ C$  **and**  $D \in N$  **and**  $C \prec_c D$   
**shows**  $\neg \text{interp } N \ D \models C - \{\# \text{Pos } A\}$   
**proof** –  
**from** *C-prod* **obtain**  $C'$  **where**  
*C-in*:  $C \in N$  **and**  
*C-def*:  $C = \text{add-mset } (\text{Pos } A) \ C'$  **and**  
*select*  $C = \{\#\}$  **and**  
*Pox-A-max*: *is-strictly-maximal-lit*  $(\text{Pos } A) \ C$  **and**  
 $\neg \text{interp } N \ C \models C$   
**by** (*rule mem-productionE*) *blast*

**from**  $\langle D \in N \rangle \langle C \prec_c D \rangle$  **have**  $A \in \text{interp } N \ D$   
**using** *C-prod production-subset-if-less-cls* **by** *auto*

**from**  $\langle D \in N \rangle$  **have**  $\text{interp } N \ D \subseteq (\bigcup D \in N. \text{production } N \ D)$   
**by** (*auto simp: interp-def*)

**have**  $\neg \text{interp } N \ D \models C'$   
**unfolding** *true-cls-def Set.bex-simps*  
**proof** (*intro ballI*)  
**fix**  $L$  **assume** *L-in*:  $L \in \# \ C'$   
**hence**  $L \in \# \ C$   
**by** (*simp add: C-def*)

**have**  $C' \prec_c C$   
**by** (*simp add: C-def subset-implies-multp*)  
**hence**  $C' \preceq_c C$   
**by** *order*

**show**  $\neg \text{interp } N \ D \models_l L$   
**proof** (*cases L*)  
**case**  $(\text{Pos } A_L)$   
**moreover have**  $A_L \notin \text{interp } N \ D$   
**proof** –  
**have**  $\forall y \in \# \ C'. y \prec_l \text{Pos } A$   
**using** *Pox-A-max*  
**by** (*simp add: C-def literal-order.is-greatest-in-mset-iff*)  
**with** *Pos* **have**  $A_L \notin \text{insert } A \ (\text{interp } N \ C)$   
**using** *L-in*  $\langle \neg \text{interp } N \ C \models C \rangle$  *C-def*  
**by** *blast*

**moreover have**  $A_L \notin (\bigcup D' \in \{D' \in N. C \prec_c D' \wedge D' \prec_c D\}. \text{production})$

$N D'$   
**proof** –  
 have  $A_L \preceq_t A$   
 using *Pos lesseq-trm-if-pos*[*OF*  $\langle C' \preceq_c C \rangle$  *C-prod*  $\langle L \in \# C' \rangle$ ]  
 by *simp*  
 thus *?thesis*  
 using *less-trm-iff-less-cls-if-mem-production*  
 using *C-prod calculation interp-def* **by** *fastforce*  
**qed**

**moreover have**  $interp\ N\ D =$   
 $insert\ A\ (interp\ N\ C) \cup (\bigcup D' \in \{D' \in N. C \prec_c D' \wedge D' \prec_c D\}. production$   
 $N\ D')$

$N D'$   
**proof** –  
 have  $interp\ N\ D = (\bigcup D' \in \{D' \in N. D' \prec_c D\}. production\ N\ D')$   
 by (*simp only: interp-def*)  
 also have  $\dots = (\bigcup D' \in \{D' \in \{y \in N. y \prec_c C\} \cup \{C\} \cup \{y \in N. C \prec_c$   
 $y\}. D' \prec_c D\}. production\ N\ D')$   
 using *partition-set-around-element*[*of*  $N\ (\prec_c)$ , *OF* -  $\langle C \in N \rangle$ ]  
*totalp-on-subset*[*OF* *totalp-less-cls, simplified*]  
 by *simp*  
 also have  $\dots = (\bigcup D' \in \{y \in N. y \prec_c C \wedge y \prec_c D\} \cup \{C\} \cup \{y \in N.$   
 $C \prec_c y \wedge y \prec_c D\}. production\ N\ D')$   
 using  $\langle C \prec_c D \rangle$  **by** *auto*  
 also have  $\dots = (\bigcup D' \in \{y \in N. y \prec_c C\} \cup \{C\} \cup \{y \in N. C \prec_c y \wedge y$   
 $\prec_c D\}. production\ N\ D')$   
 by (*metis* (*no-types, opaque-lifting*) *assms*(3) *transpD transp-less-cls*)  
 also have  $\dots = interp\ N\ C \cup production\ N\ C \cup (\bigcup D' \in \{y \in N. C \prec_c$   
 $y \wedge y \prec_c D\}. production\ N\ D')$   
 by (*auto simp: interp-def*)  
**finally show** *?thesis*  
 using *C-prod*  
 by (*metis* (*no-types, lifting*) *empty-iff insertE insert-is-Un*  
*production-eq-empty-or-singleton sup-commute*)  
**qed**

**ultimately show** *?thesis*  
 by *simp*  
**qed**

**ultimately show** *?thesis*  
 by *simp*

**next**  
**case** ( $Neg\ A_L$ )  
**moreover have**  $A_L \in interp\ N\ D$   
 using *Neg*  $\langle L \in \# C \rangle \langle C \prec_c D \rangle \langle \neg interp\ N\ C \models C \rangle$  *interp-subset-if-less-cls*  
 by *blast*  
**ultimately show** *?thesis*  
 by *simp*  
**qed**

**qed**  
**thus**  $\neg \text{interp } N D \models C - \{\#Pos A\# \}$   
**by** (*simp add: C-def*)  
**qed**

**lemma** *production-subset-Union-production*:  
 $\bigwedge C N. C \in N \implies \text{production } N C \subseteq (\bigcup D \in N. \text{production } N D)$   
**by** *auto*

**lemma** *interp-subset-Union-production*:  
 $\bigwedge C N. C \in N \implies \text{interp } N C \subseteq (\bigcup D \in N. \text{production } N D)$   
**by** (*simp add: split-Union-production'*)

**lemma** *model-construction*:  
**fixes**  
 $N :: 'f \text{ gterm atom clause set and}$   
 $C :: 'f \text{ gterm atom clause}$   
**assumes** *G.saturated N and*  $\{\#\} \notin N$  **and** *C-in: C ∈ N*  
**shows**  
 $\text{production } N C = \{\} \longleftrightarrow \text{interp } N C \models C$   
 $(\bigcup D \in N. \text{production } N D) \models C$   
 $D \in N \implies C \prec_c D \implies \text{interp } N D \models C$   
**unfolding** *atomize-conj atomize-imp*  
**using** *wfP-less-cls C-in*  
**proof** (*induction C arbitrary: D rule: wfp-induct-rule*)  
**case** (*less C*)  
**note**  $IH = \text{less.IH}$

**from**  $\langle \{\#\} \notin N \rangle \langle C \in N \rangle$  **have**  $C \neq \{\#\}$   
**by** *metis*

**define**  $I :: 'f \text{ gterm set where}$   
 $I = \text{interp } N C$

**have**  $i: \text{interp } N C \models C \longleftrightarrow (\text{production } N C = \{\})$

**proof** (*rule iffI*)

**show**  $\text{interp } N C \models C \implies \text{production } N C = \{\}$

**by** (*smt (z3) Collect-empty-eq interp-def production.elims*)

**next**

**assume**  $\text{production } N C = \{\}$

**show**  $\text{interp } N C \models C$

**proof** (*cases*  $\exists A. \text{Neg } A \in \# C \wedge (\text{Neg } A \in \# \text{select } C \vee \text{select } C = \{\#\} \wedge \text{is-maximal-lit } (\text{Neg } A) C)$ )

**case** *ex-neg-lit-sel-or-max: True*

**then obtain**  $A$  **where**

$\text{Neg } A \in \# C$  **and**

$\text{sel-or-max: } \text{Neg } A \in \# \text{select } C \vee \text{select } C = \{\#\} \wedge \text{is-maximal-lit } (\text{Neg } A)$

$C$

**by** *metis*



```

then obtain  $C'$  where
   $C$ -def:  $C = \text{add-mset } (\text{Neg } A) C'$ 
  by (metis mset-add)

show ?thesis
proof (cases  $A \in \text{interp } N C$ )
  case True
    then obtain  $D$  where
       $A \in \text{production } N D$  and  $D \in N$  and  $D \prec_c C$ 
      unfolding interp-def by auto
    then obtain  $D'$  where
       $D$ -def:  $D = \text{add-mset } (\text{Pos } A) D'$  and
      sel-D: select  $D = \{\#\}$  and
      max-t-t': is-strictly-maximal-lit  $(\text{Pos } A) D$  and
       $\neg \text{interp } N D \models D$ 
      by (elim mem-productionE) fast

    have reso: ground-resolution  $C D (C' + D')$ 
    proof (rule ground-resolutionI)
      show  $C = \text{add-mset } (\text{Neg } A) C'$ 
      by (simp add: C-def)
    next
      show  $D = \text{add-mset } (\text{Pos } A) D'$ 
      by (simp add: D-def)
    next
      show  $D \prec_c C$ 
      using  $\langle D \prec_c C \rangle$  .
    next
      show select  $C = \{\#\} \wedge \text{is-maximal-lit } (\text{Neg } A) C \vee \text{Neg } A \in \#$  select  $C$ 
      using sel-or-max by auto
    next
      show select  $D = \{\#\}$ 
      using sel-D by blast
    next
      show is-strictly-maximal-lit  $(\text{Pos } A) D$ 
      using max-t-t' .
    qed simp-all

define  $\iota :: 'f \text{ gterm clause inference}$  where
   $\iota = \text{Infer } [D, C] (C' + D')$ 

have  $\iota \in G\text{-Inf}$ 
  using reso
  by (auto simp only: \iota-def G-Inf-def)

moreover have  $\bigwedge t. t \in \text{set } (\text{prems-of } \iota) \implies t \in N$ 
  using  $\langle C \in N \rangle \langle D \in N \rangle$ 
  by (auto simp add: \iota-def)

```

**ultimately have**  $\iota \in G.\text{Inf-from } N$   
**by** (*auto simp: G.Inf-from-def*)  
**hence**  $\iota \in G.\text{Red-I } N$   
**using**  $\langle G.\text{saturated } N \rangle$   
**by** (*auto simp: G.saturated-def*)  
**then obtain**  $DD$  **where**  
 $DD\text{-subset: } DD \subseteq N$  **and**  
 $\text{finite } DD$  **and**  
 $DD\text{-entails-CD: } G.\text{entails } (\text{insert } D \ DD) \ \{C' + D'\}$  **and**  
 $\text{ball-}DD\text{-lt-}C: \forall D \in DD. D \prec_c C$   
**unfolding**  $G.\text{Red-I-def } G.\text{redundant-infer-def mem-Collect-eq}$   
**by** (*auto simp:  $\iota$ -def*)

**moreover have**  $\forall D \in \text{insert } D \ DD. \text{interp } N \ C \models D$   
**using**  $IH[\text{THEN conjunct2}, \text{THEN conjunct2}, \text{rule-format}, \text{of-} C]$   
**using**  $\langle C \in N \rangle \langle D \in N \rangle \langle D \prec_c C \rangle DD\text{-subset ball-}DD\text{-lt-}C$   
**by** (*metis in-mono insert-iff*)

**ultimately have**  $\text{interp } N \ C \models C' + D'$   
**using**  $DD\text{-entails-}CD$   
**unfolding**  $G.\text{entails-def}$   
**by** (*simp add: I-def true-cls-def*)

**moreover have**  $\neg \text{interp } N \ D \models D'$   
**using**  $\langle \neg \text{interp } N \ D \models D \rangle$   
**using**  $D\text{-def}$  **by force**

**moreover have**  $D' \prec_c D$   
**unfolding**  $D\text{-def}$   
**by** (*simp add: subset-implies-multp*)

**moreover have**  $\neg \text{interp } N \ C \models D'$   
**using**  $\text{false-cls-if-productive-production}[\text{OF-} \langle C \in N \rangle \langle D \prec_c C \rangle]$   
**by** (*metis D-def  $\langle A \in \text{production } N \ D \rangle \text{remove1-mset-add-mset-If}$* )

**ultimately show**  $\text{interp } N \ C \models C$   
**unfolding**  $C\text{-def}$  **by fast**

**next**  
**case**  $\text{False}$   
**thus** *?thesis*  
**using**  $\langle \text{Neg } A \in \# \ C \rangle$   
**by** (*auto simp add: true-cls-def*)

**qed**

**next**  
**case**  $\text{False}$   
**hence**  $\text{select } C = \{\#\}$   
**using**  $\text{select-subset select-negative-lits}$   
**by** (*metis (no-types, opaque-lifting) Neg-atm-of-iff mset-subset-eqD multi-set-nonemptyE*)

```

    from False obtain A where Pos-A-in: Pos A ∈# C and max-Pos-A:
is-maximal-lit (Pos A) C
    using ex-maximal-in-mset-wrt[OF transp-on-less-lit asymp-on-less-lit ‹C ≠
{#}›]
    using is-maximal-in-mset-wrt-iff[OF transp-on-less-lit asymp-on-less-lit]
    by (metis Neg-atm-of-iff ‹select C = {#}› is-pos-def)
    then obtain C' where C-def: C = add-mset (Pos A) C'
    by (meson mset-add)

show ?thesis
proof (cases interp N C ≡ C')
  case True
  then show ?thesis
    using C-def by force
next
  case False
  show ?thesis
  proof (cases is-strictly-maximal-lit (Pos A) C)
    case strictly-maximal: True
    then show ?thesis
      using ‹production N C = {#}› ‹select C = {#}› less.prem
      unfolding production-unfold[of N C] Collect-empty-eq
      using C-def by blast
  next
    case False
    hence count C (Pos A) ≥ 2
      using max-Pos-A
  by (simp add: literal-order.count-ge-2-if-maximal-in-mset-and-not-greatest-in-mset)
  then obtain C' where C-def: C = add-mset (Pos A) (add-mset (Pos A)
C')
    by (metis two-le-countE)

define ι :: 'f gterm clause inference where
  ι = Infer [C] (add-mset (Pos A) C')

have eq-fact: ground-factoring C (add-mset (Pos A) C')
proof (rule ground-factoringI)
  show C = add-mset (Pos A) (add-mset (Pos A) C')
    by (simp add: C-def)
next
  show select C = {#}
    using ‹select C = {#}› .
next
  show is-maximal-lit (Pos A) C
    using max-Pos-A .
qed simp-all
hence ι ∈ G-Inf
  by (auto simp: ι-def G-Inf-def)

```

**moreover have**  $\bigwedge t. t \in \text{set} (\text{prems-of } \iota) \implies t \in N$   
**using**  $\langle C \in N \rangle$   
**by** (*auto simp add:  $\iota$ -def*)

**ultimately have**  $\iota \in G.\text{Inf-from } N$   
**by** (*auto simp: G.Inf-from-def*)  
**hence**  $\iota \in G.\text{Red-I } N$   
**using**  $\langle G.\text{saturated } N \rangle$   
**by** (*auto simp: G.saturated-def*)  
**then obtain**  $DD$  **where**  
 $DD\text{-subset: } DD \subseteq N$  **and**  
 $\text{finite } DD$  **and**  
 $DD\text{-entails-concl: } G\text{-entails } DD \{ \text{add-mset } (\text{Pos } A) C' \}$  **and**  
 $\text{ball-}DD\text{-lt-}C: \forall D \in DD. D \prec_c C$   
**unfolding**  $G.\text{Red-I-def } G.\text{redundant-infer-def mem-Collect-eq}$   
**by** (*auto simp:  $\iota$ -def*)

**moreover have**  $\forall D \in DD. \text{interp } N C \models D$   
**using**  $IH[\text{THEN } \text{conjunct2}, \text{THEN } \text{conjunct2}, \text{rule-format}, \text{of } - C]$   
**using**  $\langle C \in N \rangle DD\text{-subset ball-}DD\text{-lt-}C$   
**by** *blast*

**ultimately have**  $\text{interp } N C \models \text{add-mset } (\text{Pos } A) C'$   
**using**  $DD\text{-entails-concl}$   
**unfolding**  $G\text{-entails-def}$   
**by** (*simp add: I-def true-clss-def*)  
**then show** *?thesis*  
**by** (*simp add: C-def joinI-right pair-imageI*)

**qed**  
**qed**  
**qed**  
**qed**

**moreover have**  $\text{iaa: } (\bigcup (\text{production } N \text{ ' } N)) \models C$   
**using**  $\text{production-eq-empty-or-singleton[of } N C]$   
**proof** (*elim disjE exE*)  
**assume**  $\text{production } N C = \{ \}$   
**hence**  $\text{interp } N C \models C$   
**by** (*simp only: i*)  
**thus** *?thesis*  
**using**  $\text{lift-entailment-to-Union[OF } \langle C \in N \rangle]$  **by** *argo*

**next**  
**fix**  $A$   
**assume**  $\text{production } N C = \{A\}$   
**hence**  $\text{prod: } A \in \text{production } N C$   
**by** *simp*

**from**  $\text{prod}$  **have**  $\text{Pos } A \in \# C$

**unfolding** *production.simps*[of  $N$   $C$ ] *mem-Collect-eq*  
**by** *force*

**moreover from** *prod* **have**  $A \in \bigcup$  (*production*  $N$  '  $N$ )  
**using**  $\langle C \in N \rangle$  *production-subset-Union-production*  
**by** *fast*

**ultimately show** *?thesis*  
**by** *blast*

**qed**

**moreover have** *iib*: *interp*  $N$   $D \Vdash C$  **if**  $D \in N$  **and**  $C \prec_c D$   
**using** *production-eq-empty-or-singleton*[of  $N$   $C$ , *folded* ]

**proof** (*elim disjE exE*)

**assume** *production*  $N$   $C = \{\}$

**hence** *interp*  $N$   $C \Vdash C$

**unfolding** *i* **by** *simp*

**thus** *?thesis*

**using** *lift-interp-entails*[*OF*  $\langle C \in N \rangle$  - *that*] **by** *argo*

**next**

**fix**  $A$  **assume** *production*  $N$   $C = \{A\}$

**thus** *?thesis*

**by** (*metis Un-insert-right insertCI insert-subset less-imp-Interp-subseteq-interp*  
*mem-productionE pos-literal-in-imp-true-cls that(2) union-single-eq-member*)

**qed**

**ultimately show** *?case*  
**by** *argo*

**qed**

**lemma**

**assumes** *clause-order.is-least-in-fset*  $N$   $C$  **and**  $A \in$  *production* (*fset*  $N$ )  $C$

**shows**  $\bigwedge A'. A' \prec_t A \implies A' \notin (\bigcup D \in$  *fset*  $N.$  *production* (*fset*  $N$ )  $D$ )

**using** *assms less-trm-iff-less-cls-if-mem-production clause-order.is-least-in-fset-iff*

**by** *auto*

**lemma** *lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-productive*:

**assumes**  $A \in$  *production* (*fset*  $N$ )  $C$

**shows**  $\bigwedge A'. A' \prec_t A \implies A' \notin$  *interp* (*fset*  $N$ )  $C \implies A' \notin (\bigcup D \in$  *fset*  $N.$  *production* (*fset*  $N$ )  $D$ )

**using** *assms production-subset-if-less-cls less-trm-iff-less-cls-if-mem-production*

**by** *fastforce*

**lemma** *lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-not-productive*:

**assumes** *literal-order.is-maximal-in-mset*  $C$   $L$  **and** *production* (*fset*  $N$ )  $C = \{\}$

**shows**  $\bigwedge A'. A' \prec_t$  *atm-of*  $L \implies A' \notin$  *interp* (*fset*  $N$ )  $C \implies A' \notin (\bigcup D \in$  *fset*  $N.$  *production* (*fset*  $N$ )  $D$ )

**using** *assms*

**by** (*metis (no-types, lifting) UN-E UnCI less-trm-if-neg lesseq-trm-if-pos*)

*literal-order.is-maximal-in-mset-iff term-order.less-imp-not-less term-order.not-le  
not-interp-to-Interp-imp-le)*

**lemma** *lesser-atoms-not-in-previous-interp-are-not-in-final-interp:*

**fixes**  $A$

**assumes**

$L$ -max: *literal-order.is-maximal-in-mset*  $C L$  **and**

$A$ -less:  $A \prec_t$  *atm-of*  $L$  **and**

$A$ -no-in:  $A \notin$  *interp* (fset  $N$ )  $C$

**shows**  $A \notin (\bigcup D \in \text{fset } N. \text{production } (\text{fset } N) D)$

**proof** (*cases production* (fset  $N$ )  $C = \{\}$ )

**case** *True*

**thus** *?thesis*

**using**  $L$ -max  $A$ -less  $A$ -no-in

**using** *lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-not-productive*

**by** *metis*

**next**

**case** *False*

**then obtain**  $A'$  **where** *C-produces- $A'$* :  $A' \in$  *production* (fset  $N$ )  $C$

**by** *auto*

**hence** *is-strictly-maximal-lit* (Pos  $A'$ )  $C$

**unfolding** *production-unfold mem-Collect-eq* **by** *metis*

**hence** *literal-order.is-maximal-in-mset*  $C$  (Pos  $A'$ )

**using** *literal-order.is-maximal-in-mset-if-is-greatest-in-mset* **by** *metis*

**hence**  $L = \text{Pos } A'$

**using**  $L$ -max

**by** (*metis Uniq-D literal-order.Uniq-is-maximal-in-mset*)

**hence** *atm-of*  $L \in$  *production* (fset  $N$ )  $C$

**using** *C-produces- $A'$*  **by** *simp*

**thus** *?thesis*

**using**  $L$ -max  $A$ -less  $A$ -no-in

**using** *lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-productive*

**by** *metis*

**qed**

**lemma** *lesser-atoms-in-previous-interp-are-in-final-interp:*

**fixes**  $A$

**assumes**

$L$ -max: *literal-order.is-maximal-in-mset*  $C L$  **and**

$A$ -less:  $A \prec_t$  *atm-of*  $L$  **and**

$A$ -in:  $A \in$  *interp*  $N C$

**shows**  $A \in (\bigcup D \in N. \text{production } N D)$

**using**  $A$ -in *interp-def* **by** *fastforce*

**lemma** *interp-fixed-for-smaller-literals:*

**fixes**  $A$

**assumes**

$L$ -max: *literal-order.is-maximal-in-mset*  $C L$  **and**

$A$ -less:  $A \prec_t$  *atm-of*  $L$  **and**

$C \prec_c D$   
**shows**  $A \in \text{interp } N C \longleftrightarrow A \in \text{interp } N D$   
**proof** (*rule iffI*)  
**show**  $A \in \text{interp } N C \implies A \in \text{interp } N D$   
**using** *assms(3) interp-subset-if-less-cls* **by** *auto*  
**next**  
**assume**  $A \in \text{interp } N D$

**then obtain**  $E$  **where**  $A \in \text{production } N E$  **and**  $E \in N$  **and**  $E \prec_c D$   
**unfolding** *interp-def* **by** *auto*

**hence** *literal-order.is-greatest-in-mset*  $E$  (*Pos*  $A$ )  
**by** (*auto elim: mem-productionE*)

**moreover have** *Pos*  $A \prec_l L$   
**by** (*metis A-less Neg-atm-of-iff less-lit-simps(1) less-lit-simps(2)*  
*literal-order.dual-order.strict-trans term-order.order-eq-iff literal.collapse(1)*)

**ultimately have**  $E \prec_c C$   
**using** *L-max*  
**using** *literal-order.multip<sub>HO</sub>-if-maximal-less-that-maximal multip<sub>DM</sub>-imp-multip<sub>HO</sub>-imp-multip<sub>DM</sub>*  
**by** *blast*

**hence** *production*  $N E \subseteq \text{interp } N C$   
**by** (*simp add: production-subset-if-less-cls*)

**thus**  $A \in \text{interp } N C$   
**using**  $\langle A \in \text{production } N E \rangle$  **by** *auto*  
**qed**

**lemma** *neg-lits-not-in-model-stay-out-of-model:*  
**assumes**  
 $L\text{-in}: L \in \# C$  **and**  
 $L\text{-neg}: \text{is-neg } L$  **and**  
 $\text{atm-}L\text{-not-in}: \text{atm-of } L \notin \text{interp } N C$   
**shows**  $\text{atm-of } L \notin (\bigcup D \in N. \text{production } N D)$   
**using** *assms produces-imp-in-interp* **by** *force*

**lemma** *neg-lits-already-in-model-stay-in-model:*  
**assumes**  
 $L\text{-in}: L \in \# C$  **and**  
 $L\text{-neg}: \text{is-neg } L$  **and**  
 $\text{atm-}L\text{-not-in}: \text{atm-of } L \in \text{interp } N C$   
**shows**  $\text{atm-of } L \in (\bigcup D \in N. \text{production } N D)$   
**using** *atm-}L\text{-not-in interp-def}* **by** *auto*

**lemma** *image-eq-imageI:*  
**assumes**  $\bigwedge x. x \in X \implies f x = g x$

**shows**  $f \text{ ' } X = g \text{ ' } X$   
**using** *assms* **by** *auto*

**lemma** *production-swap-clause-set*:

**assumes**

*agree*:  $\{D \in N1. D \preceq_c C\} = \{D \in N2. D \preceq_c C\}$

**shows** *production*  $N1 \ C = \text{production } N2 \ C$

**using** *agree*

**proof** (*induction*  $C$  *rule*: *wfp-induct-rule*[*OF wfp-less-cl*])

**case** *hyps*:  $(1 \ C)$

**from** *hyps* **have** *AAA*:  $C \in N1 \longleftrightarrow C \in N2$

**by** *auto*

**from** *hyps* **have**  $\{D \in N1. D \prec_c C\} = \{D \in N2. D \prec_c C\}$

**by** *blast*

**have** *production*  $N1 \ \{D \in N2. D \prec_c C\} = \text{production } N2 \ \{D \in N2. D \prec_c C\}$

**proof** (*rule* *image-eq-imageI*)

**fix**  $x$  **assume**  $x \in \{D \in N2. D \prec_c C\}$

**hence**  $x \in N2$  **and**  $x \prec_c C$

**by** *simp-all*

**moreover** **have**  $\{D \in N1. (\prec_c)^{==} D \ x\} = \{D \in N2. (\prec_c)^{==} D \ x\}$

**using** *hyps.prem*s  $\langle x \prec_c C \rangle$  *clause-order.order.strict-trans1* **by** *blast*

**ultimately** **show** *production*  $N1 \ x = \text{production } N2 \ x$

**using** *hyps.IH*[*rule-format*, *of*  $x$ ] **by** *metis*

**qed**

**hence** *BBB*: *interp*  $N1 \ C = \text{interp } N2 \ C$

**unfolding** *interp-def*  $\langle \{D \in N1. D \prec_c C\} = \{D \in N2. D \prec_c C\} \rangle$

**by** *argo*

**show** *?case*

**unfolding** *production-unfold*

**unfolding** *AAA BBB*

**by** *simp*

**qed**

**lemma** *interp-swap-clause-set*:

**assumes** *agree*:  $\{D \in N1. D \prec_c C\} = \{D \in N2. D \prec_c C\}$

**shows** *interp*  $N1 \ C = \text{interp } N2 \ C$

**proof** –

**have** *BBB*: *production*  $N1 \ \{D \in N2. D \prec_c C\} = \text{production } N2 \ \{D \in N2. D \prec_c C\}$

**proof** (*intro* *image-eq-imageI* *production-swap-clause-set*)

**fix**  $x$

**assume**  $x \in \{D \in N2. D \prec_c C\}$

**thus**  $\{D \in N1. (\prec_c)^{==} D \ x\} = \{D \in N2. (\prec_c)^{==} D \ x\}$

**using** *agree* *clause-order.le-less-trans* **by** *blast*

**qed**



**show** *?thesis*  
**unfolding** *interp-def*  
**unfolding** *agree BBB*  
**by** *argo*  
**qed**

**definition** *interp' where*  
 $interp' N \equiv (\bigcup C \in N. production N C)$

**lemma** *interp-eq-interp'*:  $interp N D = interp' \{C \in N. C \prec_c D\}$

**proof** –

**have**  $interp N D = interp \{C \in N. C \prec_c D\} D$

**proof** (*rule interp-swap-clause-set*)

**show**  $\{Da \in N. Da \prec_c D\} = \{Da \in \{C \in N. C \prec_c D\}. Da \prec_c D\}$

**by** *blast*

**qed**

**also have**  $\dots = interp' \{C \in N. C \prec_c D\}$

**unfolding** *interp-def interp'-def* **by** *blast*

**finally show** *?thesis* .

**qed**

**lemma** *production-unfold''*:  $production N C = \{A \mid A.$

$C \in N \wedge select C = \{\#\} \wedge$

*is-strictly-maximal-lit* (*Pos A*)  $C \wedge$

$\neg interp' \{B \in N. B \prec_c C\} \models C\}$

**unfolding** *production-unfold interp-eq-interp'*

**using** *literal-order.explode-greatest-in-mset*

**by** *metis*

**lemma** *Interp-swap-clause-set*:

**assumes** *agree*:  $\{D \in N1. D \preceq_c C\} = \{D \in N2. D \preceq_c C\}$

**shows**  $interp N1 C \cup production N1 C = interp N2 C \cup production N2 C$

**using** *production-swap-clause-set[OF agree]*

**using** *interp-swap-clause-set*

**using** *agree*

**by** *blast*

**lemma** *production-insert-greater-clause*:

**assumes**  $C \prec_c D$

**shows**  $production (insert D N) C = production N C$

**proof** (*rule production-swap-clause-set*)

**show**  $\{Da \in insert D N. (\prec_c)^{==} Da C\} = \{D \in N. (\prec_c)^{==} D C\}$

**using**  $\langle C \prec_c D \rangle$  **by** *auto*

**qed**

**lemma** *interp-insert-greater-clause-strong*:

**assumes**  $C \preceq_c D$

**shows**  $\text{interp } (\text{insert } D \ N) \ C = \text{interp } N \ C$   
**proof** (rule *interp-swap-clause-set*)  
**show**  $\{x \in \text{insert } D \ N. x \prec_c C\} = \{x \in N. x \prec_c C\}$   
**using**  $\langle C \preceq_c D \rangle$  **by** *auto*  
**qed**

**lemma** *interp-insert-greater-clause*:  
**assumes**  $C \prec_c D$   
**shows**  $\text{interp } (\text{insert } D \ N) \ C = \text{interp } N \ C$   
**proof** (rule *interp-swap-clause-set*)  
**show**  $\{x \in \text{insert } D \ N. x \prec_c C\} = \{x \in N. x \prec_c C\}$   
**using**  $\langle C \prec_c D \rangle$  **by** *auto*  
**qed**

**lemma** *Interp-insert-greater-clause*:  
**assumes**  $C \prec_c D$   
**shows**  $\text{interp } (\text{insert } D \ N) \ C \cup \text{production } (\text{insert } D \ N) \ C = \text{interp } N \ C \cup \text{production } N \ C$   
**proof** (rule *Interp-swap-clause-set*)  
**show**  $\{Da \in \text{insert } D \ N. (\prec_c)^{==} Da \ C\} = \{D \in N. (\prec_c)^{==} D \ C\}$   
**using**  $\langle C \prec_c D \rangle$  **by** *auto*  
**qed**

**lemma** *production-add-irrelevant-clause-to-set0*:  
**assumes**  
*fin*: *finite*  $N$  **and**  
*D-irrelevant*:  $E \in N \ E \subset\# \ D \ \text{set-mset } D = \text{set-mset } E$  **and**  
*no-select*:  $\text{select } E = \{\#\}$   
**shows**  $\text{production } (\text{insert } D \ N) \ D = \{\}$   
**proof** –  
**from** *D-irrelevant* **have**  $E \prec_c D$   
**using** *subset-implies-multp* **by** *metis*  
**hence** *prod-E-subset*:  $\text{production } (\text{insert } D \ N) \ E \subseteq \text{interp } (\text{insert } D \ N) \ D$   
**using** *production-subset-if-less-cl* **by** *metis*

**show** *?thesis*  
**proof** (*cases*  $\text{production } (\text{insert } D \ N) \ E = \{\}$ )  
**case** *True*  
**hence**  $(\nexists A. \text{is-strictly-maximal-lit } (\text{Pos } A) \ E) \vee \text{interp } (\text{insert } D \ N) \ E \models E$   
**unfolding** *production-unfold*  
**using** *no-select*  
**by** (*smt* (*verit*, *del-insts*) *Collect-empty-eq*  $\langle E \in N \rangle$  *diff-single-eq-union* *insertI2* *literal-order.is-greatest-in-mset-iff*)  
**hence**  $(\nexists A. \text{is-strictly-maximal-lit } (\text{Pos } A) \ D) \vee \text{interp } (\text{insert } D \ N) \ D \models D$   
**proof** (*elim disjE*)  
**assume**  $\nexists A. \text{is-strictly-maximal-lit } (\text{Pos } A) \ E$   
**hence**  $\nexists A. \text{is-strictly-maximal-lit } (\text{Pos } A) \ D$   
**proof** (rule *contrapos-nn*)  
**show**  $\exists A. \text{is-strictly-maximal-lit } (\text{Pos } A) \ D \implies \exists A. \text{is-strictly-maximal-lit}$

```

(Pos A) E
  using ⟨E ⊂# D⟩ ⟨set-mset D = set-mset E⟩
  unfolding literal-order.is-greatest-in-mset-iff
  by (metis (no-types, opaque-lifting) add-mset-remove-trivial-eq
      insert-union-subset-iff)
qed
thus ?thesis ..
next
assume interp (insert D N) E ⊨ E
hence interp (insert D N) D ⊨ D
  using lift-interp-entails ⟨E ∈ N⟩ ⟨E <c D⟩
  by (metis ⟨set-mset D = set-mset E⟩ insert-iff true-cls-def)
thus ?thesis ..
qed
thus ?thesis
  unfolding production-unfold by auto
next
case False
hence interp (insert D N) D ⊨ D
  using prod-E-subset
by (metis ⟨set-mset D = set-mset E⟩ mem-productionE production-eq-empty-or-singleton
    insertCI insert-subset literal-order.is-greatest-in-mset-iff
    pos-literal-in-imp-true-cls)
thus ?thesis
  unfolding production-unfold by simp
qed
qed

lemma production-add-irrelevant-clause-to-set:
  assumes
    fin: finite N and
    C-in: C ∈ N and
    D-irrelevant: ∃ E ∈ N. E ⊂# D ∧ set-mset D = set-mset E and
    no-select: ⋀ C. select C = {#}
  shows production (insert D N) C = production N C
  using wfp-less-cls C-in
proof (induction C rule: wfp-induct-rule)
case (less C)
hence C-in-iff: C ∈ insert D N ⟷ C ∈ N
  by simp

have interp-insert-eq: interp (insert D N) C = interp N C
proof (cases D <c C)
case True
hence {x ∈ insert D N. x <c C} = insert D {x ∈ N. x <c C}
  by auto
hence ⋃ (production (insert D N) ‘ {x ∈ insert D N. x <c C} ) =
  production (insert D N) D ∪ ⋃ (production (insert D N) ‘ {D ∈ N. D <c
C})

```

```

    by simp
  also have ... =  $\bigcup$  (production (insert D N) ‘ {D ∈ N. D <_c C} )
    using production-add-irrelevant-clause-to-set0
    using D-irrelevant fin no-select by force
  also have ... =  $\bigcup$  (production N ‘ {D ∈ N. D <_c C} )
    using less.IH by simp
  finally show ?thesis
    unfolding interp-def .
next
  case False
  then show ?thesis
    unfolding interp-def
    by (smt (verit, best) Collect-cong image-eq-imageI insert-iff less.IH mem-Collect-eq)
qed

show ?case
  unfolding production-unfold C-in-iff interp-insert-eq by argo
qed

lemma production-add-irrelevant-clauses-to-set0:
  assumes
    fin: finite N finite N' and
    D-in: D ∈ N' and
    irrelevant:  $\forall D \in N'. \exists E \in N. E \subset\# D \wedge \text{set-mset } D = \text{set-mset } E$  and
    no-select:  $\bigwedge C. \text{select } C = \{\#\}$ 
  shows production (N  $\cup$  N') D = {}
  using ⟨finite N'⟩ D-in irrelevant
proof (induction N' rule: finite-induct)
  case empty
  hence False
  by simp
  thus ?case ..
next
  case (insert x F)
  then show ?case
    using production-add-irrelevant-clause-to-set0
    by (metis UnCI fin(1) finite-Un finite-insert production-add-irrelevant-clause-to-set
no-select)
qed

lemma production-add-irrelevant-clauses-to-set:
  assumes
    fin: finite N finite N' and
    C-in: C ∈ N and
    irrelevant:  $\forall D \in N'. \exists E \in N. E \subset\# D \wedge \text{set-mset } D = \text{set-mset } E$  and
    no-select:  $\bigwedge C. \text{select } C = \{\#\}$ 
  shows production (N  $\cup$  N') C = production N C
  using ⟨finite N'⟩ irrelevant
proof (induction N' rule: finite-induct)

```

```

case empty
show ?case
  by simp
next
case (insert x F)
then show ?case
  using production-add-irrelevant-clause-to-set
  by (metis C-in UnI1 Un-insert-right fin(1) finite-Un insertCI no-select)
qed

```

**lemma** *interp-add-irrelevant-clauses-to-set:*

```

assumes
  fin: finite N finite N' and
  C-in: C ∈ N and
  irrelevant: ∀ D ∈ N'. ∃ E ∈ N. E ⊂# D ∧ set-mset D = set-mset E and
  no-select: ∧ C. select C = {#}
shows interp (N ∪ N') C = interp N C
proof -
have interp (N ∪ N') C = ∪ (production (N ∪ N') ‘ {D ∈ N ∪ N'. D <_c C})
  unfolding interp-def ..
also have ... = ∪ (production (N ∪ N') ‘ {D ∈ N. D <_c C} ∪
  production (N ∪ N') ‘ {D ∈ N'. D <_c C})
  by auto
also have ... = ∪ (production (N ∪ N') ‘ {D ∈ N. D <_c C})
  using production-add-irrelevant-clauses-to-set0[OF fin - irrelevant no-select] by
simp
also have ... = ∪ (production N ‘ {D ∈ N. D <_c C})
  using production-add-irrelevant-clauses-to-set[OF fin - - no-select] irrelevant
  using subset-mset.less-le by fastforce
also have ... = interp N C
  unfolding interp-def ..
finally show ?thesis .
qed

```

**lemma** *interp-add-irrelevant-clauses-to-set':*

```

assumes
  fin: finite N finite N' and
  C-in: C ∈ N and
  irrelevant: ∀ D ∈ N'. ∃ E ∈ N. E ⊆# D ∧ set-mset D = set-mset E and
  no-select: ∧ C. select C = {#}
shows interp (N ∪ N') C = interp N C
proof -
define N'' where
  N'' = N' - N

from fin(2) have finite N''
  unfolding N''-def by simp

moreover from irrelevant have ∀ D ∈ N''. ∃ E ∈ N. E ⊂# D ∧ set-mset E =

```

*set-mset D*  
**unfolding**  $N''\text{-def}$   
**by** (*metis Diff-iff subset-mset.le-less*)

**moreover have**  $N \cup N' = N \cup N''$   
**unfolding**  $N''\text{-def}$  **by** *simp*

**ultimately show** *?thesis*  
**using** *assms interp-add-irrelevant-clauses-to-set* **by** *metis*  
**qed**

**lemma** *lesser-entailed-clause-stays-entailed'*:  
**assumes**  $C \preceq_c D$  **and**  $D\text{-lt}: D \prec_c E$  **and**  $C\text{-entailed}: \text{interp } N D \cup \text{production } N D \models C$   
**shows**  $\text{interp } N E \models C$   
**proof** –  
**from**  $C\text{-entailed}$  **obtain**  $L$  **where**  $L \in \# C$  **and**  $\text{interp } N D \cup \text{production } N D \models_l L$   
**by** (*auto simp: true-cls-def*)

**show** *?thesis*  
**proof** (*cases L*)  
**case** ( $Pos A$ )  
**hence**  $A \in \text{interp } N D \cup \text{production } N D$   
**using**  $\langle \text{interp } N D \cup \text{production } N D \models_l L \rangle$  **by** *simp*  
**moreover from**  $D\text{-lt}$  **have**  $\text{interp } N D \cup \text{production } N D \subseteq \text{interp } N E$   
**using** *less-imp-Interp-subseteq-interp* **by** *blast*  
**ultimately have**  $A \in \text{interp } N E$   
**by** *auto*  
**thus** *?thesis*  
**using**  $Pos \langle L \in \# C \rangle$  **by** *auto*

**next**  
**case** ( $Neg A$ )  
**then show** *?thesis*  
**using** *neg-lits-not-in-model-stay-out-of-model*  
**by** (*smt (verit, best) UN-E Un-iff*  $\langle L \in \# C \rangle \langle \text{interp } N D \cup \text{production } N D \models_l L \rangle$  *assms(1)*  
*interp-def clause-order.antisym-conv neg-literal-notin-imp-true-cls*  
*not-interp-to-Interp-imp-le produces-imp-in-interp true-lit-simps(2)*)

**qed**  
**qed**

**lemma** *lesser-entailed-clause-stays-entailed*:  
**assumes**  $C\text{-le}: C \preceq_c D$  **and**  $D\text{-lt}: D \prec_c E$  **and**  $C\text{-entailed}: \text{interp } N D \cup \text{production } N D \models C$   
**shows**  $\text{interp } N E \cup \text{production } N E \models C$   
**proof** –  
**from**  $C\text{-entailed}$  **obtain**  $L$  **where**  $L \in \# C$  **and**  $\text{interp } N D \cup \text{production } N D \models_l L$

**by** (*auto simp: true-cls-def*)

**show** *?thesis*

**proof** (*cases L*)

**case** (*Pos A*)

**hence**  $A \in \text{interp } N D \cup \text{production } N D$

**using**  $\langle \text{interp } N D \cup \text{production } N D \models_l L \rangle$  **by** *simp*

**moreover from** *D-lt* **have**  $\text{interp } N D \cup \text{production } N D \subseteq \text{interp } N E \cup$   
*production N E*

**using** *less-imp-Interp-subseteq-interp* **by** *blast*

**ultimately have**  $A \in \text{interp } N E \cup \text{production } N E$

**by** *auto*

**thus** *?thesis*

**using** *Pos*  $\langle L \in \# C \rangle$  **by** *auto*

**next**

**case** (*Neg A*)

**then show** *?thesis*

**using** *neg-lits-not-in-model-stay-out-of-model*

**by** (*smt (verit, best) UN-E Un-iff*  $\langle L \in \# C \rangle$   $\langle \text{interp } N D \cup \text{production } N D$   
 $\models_l L \rangle$  *assms(1)*

*interp-def clause-order.antisym-conv neg-literal-notin-imp-true-cls*  
*not-interp-to-Interp-imp-le produces-imp-in-interp true-lit-simps(2)*)

**qed**

**qed**

**lemma** *entailed-clause-stays-entailed'*:

**assumes** *C-lt*:  $C \prec_c D$  **and** *C-entailed*:  $\text{interp } N C \cup \text{production } N C \models C$

**shows**  $\text{interp } N D \models C$

**using** *lesser-entailed-clause-stays-entailed'*[*OF clause-order.order-refl assms*] .

**lemma** *entailed-clause-stays-entailed*:

**assumes** *C-lt*:  $C \prec_c D$  **and** *C-entailed*:  $\text{interp } N C \cup \text{production } N C \models C$

**shows**  $\text{interp } N D \cup \text{production } N D \models C$

**using** *lesser-entailed-clause-stays-entailed*[*OF clause-order.order-refl assms*] .

**lemma** *multp-if-all-left-smaller*:  $M2 \neq \{\#\} \implies \forall k \in \#M1. \exists j \in \#M2. R k j \implies$   
*multp R M1 M2*

**using** *one-step-implies-multp*

**by** (*metis add-0*)

**lemma**

**fixes**

$P1 :: 'f \text{ gterm}$  **and**

$C1 :: 'f \text{ gterm clause}$  **and**

$N :: 'f \text{ gterm clause set}$

**defines**

$C1 \equiv \{\#Neg P1\#}$  **and**

$N \equiv \{C1\}$

**assumes**

```

    no-select:  $\bigwedge C. \text{select } C = \{\#\}$ 
  shows
    False
proof -
  have interp N C1 = {}
    unfolding interp-def N-def by simp
  have production N C1 = {}
    unfolding production-unfold  $\langle \text{interp } N \ C1 = \{\}\rangle$  C1-def by simp
  hence interp N C1  $\cup$  production N C1  $\models$  C1
    unfolding  $\langle \text{interp } N \ C1 = \{\}\rangle$   $\langle \text{production } N \ C1 = \{\}\rangle$ 
    by (simp add: C1-def)
oops

lemma
  fixes
    P1 P2 :: 'f gterm and
    C1 :: 'f gterm clause and
    N :: 'f gterm clause set
  defines
    C1  $\equiv$   $\{\#\text{Pos } P1, \text{Neg } P2\#\}$  and
    N  $\equiv$   $\{C1\}$ 
  assumes
    term-order: P1  $\prec_t$  P2 and
    no-select:  $\bigwedge C. \text{select } C = \{\#\}$ 
  shows False
proof -
  have lit-order: Pos P1  $\prec_l$  Neg P1 Neg P1  $\prec_l$  Pos P2 Pos P2  $\prec_l$  Neg P2
    using term-order by simp-all
  have interp N C1 = {}
    unfolding interp-def N-def by simp
  have production N C1 = {}
    unfolding production-unfold
    using C1-def  $\langle \text{interp } N \ C1 = \{\}\rangle$  by simp
  hence interp N C1  $\cup$  production N C1  $\models$  C1
    unfolding  $\langle \text{interp } N \ C1 = \{\}\rangle$   $\langle \text{production } N \ C1 = \{\}\rangle$ 
    by (simp add: C1-def)
oops

lemma
  fixes
    P1 P2 P3 P4 :: 'f gterm and
    C1 C2 C3 C4 C5 :: 'f gterm clause and
    N :: 'f gterm clause set
  defines
    C1  $\equiv$   $\{\#\text{Neg } P1, \text{Neg } P2\#\}$  and
    C2  $\equiv$   $\{\#\text{Pos } P2, \text{Neg } P3\#\}$  and

```



$C3 \equiv \{\#Pos P1, Pos P2, Pos P4\# \}$  **and**  
 $C4 \equiv \{\#Pos P2, Pos P3, Pos P4\# \}$  **and**  
 $C5 \equiv \{\#Pos P2, Neg P4\# \}$  **and**  
 $N \equiv \{C1, C2, C3, C4, C5\}$

**assumes**

*term-order*:  $P1 \prec_t P2 P2 \prec_t P3 P3 \prec_t P4$  **and**  
*no-select*:  $\bigwedge C. select C = \{\#\}$

**shows**

$C1 \prec_c C2 C2 \prec_c C3 C3 \prec_c C4 C4 \prec_c C5$

**proof** –

**have** *lit-order*:  $Pos P1 \prec_l Neg P1 Neg P1 \prec_l Pos P2 Pos P2 \prec_l Neg P2 Neg P2$   
 $\prec_l Pos P3$   
 $Pos P3 \prec_l Neg P3 Neg P3 \prec_l Pos P4 Pos P4 \prec_l Neg P4$   
**using** *term-order* **by** *simp-all*

**show**  $C1 \prec_c C2$

**unfolding** *C1-def C2-def*

**proof** (*rule multp-if-all-left-smaller*)

**show**  $\{\#Pos P2, Neg P3\# \} \neq \{\#\}$

**by** *simp*

**next**

**show**  $\forall k \in \#\{\#Neg P1, Neg P2\#\}. \exists j \in \#\{\#Pos P2, Neg P3\#\}. k \prec_l j$

**using** *lit-order* **by** *simp*

**qed**

**moreover show**  $C2 \prec_c C3$

**unfolding** *C2-def C3-def*

**proof** (*rule multp-if-all-left-smaller*)

**show**  $\{\#Pos P1, Pos P2, Pos P4\# \} \neq \{\#\}$

**by** *simp*

**next**

**show**  $\forall k \in \#\{\#Pos P2, Neg P3\#\}. \exists j \in \#\{\#Pos P1, Pos P2, Pos P4\#\}. k \prec_l j$

**using** *lit-order* **by** *auto*

**qed**

**moreover show**  $C3 \prec_c C4$

**proof** –

**have**  $\{\#Pos P1, Pos P2\# \} \prec_c \{\#Pos P2, Pos P3\# \}$

**proof** (*rule multp-if-all-left-smaller*)

**show**  $\{\#Pos P2, Pos P3\# \} \neq \{\#\}$

**by** *simp*

**next**

**show**  $\forall k \in \#\{\#Pos P1, Pos P2\#\}. \exists j \in \#\{\#Pos P2, Pos P3\#\}. k \prec_l j$

**using** *lit-order* **by** *simp*

**qed**

**thus** *?thesis*

**unfolding** *C3-def C4-def*

**by** (*smt (verit, ccfv-SIG) add-mset-commute irreflp-on-less-lit multp-cancel-add-mset*)

```

      transp-less-lit)
qed

moreover show  $C_4 \prec_c C_5$ 
  unfolding  $C_4$ -def  $C_5$ -def
proof (rule multp-if-all-left-smaller)
  show  $\{\#Pos\ P_2, Neg\ P_4\} \neq \{\#\}$ 
  by simp
next
  show  $\forall k \in \#\{\#Pos\ P_2, Pos\ P_3, Pos\ P_4\}. \exists j \in \#\{\#Pos\ P_2, Neg\ P_4\}. k \prec_l$ 
j
  using lit-order by auto
qed

note cls-order = calculation this

have interp  $N\ C1 = \{\}$ 
  unfolding interp-def  $N$ -def
  using cls-order
  by (smt (verit, best) Collect-empty-eq bot-fset.rep-eq ccSUP-empty finsertCI
fset-simps(2)
  clause-order.dual-order.strict-implies-not-eq clause-order.is-minimal-in-fset-finsertI
  clause-order.is-minimal-in-fset-iff singletonD)
hence production  $N\ C1 = \{\}$ 
  unfolding production-unfold  $C1$ -def by simp
hence interp  $N\ C1 \cup$  production  $N\ C1 \models C1$ 
  unfolding  $\langle$ interp  $N\ C1 = \{\}\rangle$   $\langle$ production  $N\ C1 = \{\}\rangle$ 
  unfolding  $C1$ -def
  unfolding true-cls-def true-lit-def
  by (simp add:  $C1$ -def)

have  $\{D \in N. D \prec_c C2\} = \{C1\}$ 
  unfolding  $N$ -def
  using cls-order by auto
hence interp  $N\ C2 =$  interp  $N\ C1 \cup$  production  $N\ C1$ 
  unfolding  $\langle$ interp  $N\ C1 = \{\}\rangle$ 
  unfolding interp-def
  by simp
hence interp  $N\ C2 = \{\}$ 
  unfolding  $\langle$ interp  $N\ C1 = \{\}\rangle$   $\langle$ production  $N\ C1 = \{\}\rangle$  by simp
hence production  $N\ C2 = \{\}$ 
  unfolding production-unfold
  by (simp add:  $C2$ -def)
hence interp  $N\ C2 \cup$  production  $N\ C2 \models C2$ 
  using  $\langle$ interp  $N\ C2 = \{\}\rangle$ 
  by (simp add:  $C2$ -def)

have  $\{D \in N. D \prec_c C3\} = \{C1, C2\}$ 
  unfolding  $N$ -def

```

**using** *cls-order* **by** *auto*  
**hence**  $\text{interp } N \ C3 = \text{interp } N \ C2 \cup \text{production } N \ C2$   
**unfolding**  $\langle \text{interp } N \ C2 = \{\} \rangle$   
**unfolding** *interp-def*  
**by** (*simp add: production N C1 = {}*)  
**hence**  $\text{interp } N \ C3 = \{\}$   
**unfolding**  $\langle \text{interp } N \ C2 = \{\} \rangle \langle \text{production } N \ C2 = \{\} \rangle$  **by** *simp*  
**have**  $\text{production } N \ C3 = \{P4\}$   
**proof** –  
**have**  $C3 \in N$   
**by** (*simp add: N-def*)  
**moreover** **have**  $\exists C3'. C3 = \text{add-mset } (Pos \ P4) \ C3'$   
**by** (*auto simp: C3-def*)  
**moreover** **have** *is-strictly-maximal-lit (Pos P4) C3*  
**unfolding** *C3-def literal-order.is-greatest-in-mset-iff*  
**using** *lit-order* **by** *auto*  
**moreover** **have**  $\neg \text{interp } N \ C3 \models C3$   
**unfolding**  $\langle \text{interp } N \ C3 = \{\} \rangle$   
**unfolding** *C3-def*  
**by** *simp*  
**ultimately** **have**  $P4 \in \text{production } N \ C3$   
**unfolding** *production-unfold*  
**using** *no-select*  
**by** *simp*  
**thus** *?thesis*  
**using** *production-eq-empty-or-singleton* **by** *fastforce*  
**qed**  
**hence**  $\text{interp } N \ C3 \cup \text{production } N \ C3 \models C3$   
**using**  $\langle \text{interp } N \ C3 = \{\} \rangle$   
**by** (*simp add: C3-def*)  
  
**have**  $\{D \in N. D \prec_c C4\} = \{C1, C2, C3\}$   
**unfolding** *N-def*  
**using** *cls-order* **by** *auto*  
**hence**  $\text{interp } N \ C4 = \text{interp } N \ C3 \cup \text{production } N \ C3$   
**unfolding**  $\langle \text{interp } N \ C3 = \text{interp } N \ C2 \cup \text{production } N \ C2 \rangle$   
**unfolding**  $\langle \text{interp } N \ C2 = \text{interp } N \ C1 \cup \text{production } N \ C1 \rangle$   
**unfolding**  $\langle \text{interp } N \ C1 = \{\} \rangle$   
**unfolding** *interp-def*  
**by** *auto*  
**hence**  $\text{interp } N \ C4 = \{P4\}$   
**using**  $\langle \text{interp } N \ C3 = \{\} \rangle \langle \text{production } N \ C3 = \{P4\} \rangle$  **by** *simp*  
**hence**  $\text{interp } N \ C4 \models C4$   
**using** *C4-def* **by** *simp*  
**hence**  $\text{production } N \ C4 = \{\}$   
**unfolding** *production-unfold* **by** *simp*  
**hence**  $\text{interp } N \ C4 \cup \text{production } N \ C4 \models C4$   
**using**  $\langle \text{interp } N \ C4 = \{P4\} \rangle$   
**by** (*simp add: C4-def*)

**have**  $\{D \in N. D \prec_c C5\} = \{C1, C2, C3, C4\}$   
**unfolding** *N-def*  
**using** *cls-order by auto*  
**hence**  $\text{interp } N \ C5 = \text{interp } N \ C4 \cup \text{production } N \ C4$   
**unfolding**  $\langle \text{interp } N \ C4 = \text{interp } N \ C3 \cup \text{production } N \ C3 \rangle$   
**unfolding**  $\langle \text{interp } N \ C3 = \text{interp } N \ C2 \cup \text{production } N \ C2 \rangle$   
**unfolding**  $\langle \text{interp } N \ C2 = \text{interp } N \ C1 \cup \text{production } N \ C1 \rangle$   
**unfolding**  $\langle \text{interp } N \ C1 = \{\} \rangle$   
**unfolding** *interp-def*  
**by** *auto*  
**hence**  $\text{interp } N \ C5 = \{P4\}$   
**using**  $\langle \text{interp } N \ C4 = \{P4\} \rangle \langle \text{production } N \ C4 = \{\} \rangle$  **by** *simp*  
**have**  $\text{production } N \ C5 = \{\}$   
**proof** –  
**have** *is-strictly-maximal-lit (Neg P4) C5*  
**unfolding** *C5-def literal-order.is-greatest-in-mset-iff*  
**using** *lit-order by auto*  
**hence**  $\bigwedge A. \neg \text{is-strictly-maximal-lit } (Pos \ A) \ C5$   
**by** *(meson Uniq-D literal-order.Uniq-is-greatest-in-mset literal.distinct(1))*  
**thus** *?thesis*  
**unfolding** *production-unfold by simp*  
**qed**  
**hence**  $\neg \text{interp } N \ C5 \cup \text{production } N \ C5 \models C5$   
**unfolding**  $\langle \text{interp } N \ C5 = \{P4\} \rangle \langle \text{production } N \ C5 = \{\} \rangle$   
**unfolding** *C5-def*  
**using** *term-order*  
**by** *simp*  
**qed**

**interpretation** *G: statically-complete-calculus G-Bot G-Inf G-entails G.Red-I G.Red-F*

**proof** *unfold-locales*

**fix**  $B :: 'f \text{ gterm atom clause and } N :: 'f \text{ gterm atom clause set}$

**assume**  $B \in G\text{-Bot and } G.\text{saturated } N$

**hence**  $B = \{\#\}$

**by** *simp*

**assume**  $G\text{-entails } N \ \{B\}$

**hence**  $\{\#\} \in N$

**unfolding**  $\langle B = \{\#\} \rangle$

**proof** *(rule contrapos-pp)*

**assume**  $\{\#\} \notin N$

**define**  $I :: 'f \text{ gterm set where}$

$I = (\bigcup D \in N. \text{production } N \ D)$

**show**  $\neg G\text{-entails } N \ G\text{-Bot}$

**unfolding** *G-entails-def not-all not-imp*

```

proof (intro exI conjI)
  show  $I \models_s N$ 
    unfolding I-def
    using model-construction(2)[OF ⟨G.saturated N⟩ ⟨{#} ∉ N⟩]
    by (simp add: true-cls-def)
  next
    show  $\neg I \models_s G\text{-Bot}$ 
      by simp
    qed
  qed
thus  $\exists B' \in G\text{-Bot}. B' \in N$ 
  by auto
qed

end

end
theory Lower-Set
  imports Main
begin

definition is-lower-set-wrt :: ('a  $\Rightarrow$  'a  $\Rightarrow$  bool)  $\Rightarrow$  'a set  $\Rightarrow$  'a set  $\Rightarrow$  bool where
  transp-on X R  $\Longrightarrow$  asymp-on X R  $\Longrightarrow$ 
  is-lower-set-wrt R L X  $\longleftrightarrow$   $L \subseteq X \wedge (\forall l \in L. \forall x \in X. R x l \longrightarrow x \in L)$ 

definition is-strict-lower-set-wrt :: ('a  $\Rightarrow$  'a  $\Rightarrow$  bool)  $\Rightarrow$  'a set  $\Rightarrow$  'a set  $\Rightarrow$  bool
where
  transp-on X R  $\Longrightarrow$  asymp-on X R  $\Longrightarrow$ 
  is-strict-lower-set-wrt R L X  $\longleftrightarrow$   $L \subset X \wedge (\forall l \in L. \forall x \in X. R x l \longrightarrow x \in L)$ 

lemma is-lower-set-wrt-empty:
  fixes X :: 'a set and R :: 'a  $\Rightarrow$  'a  $\Rightarrow$  bool
  assumes transp-on X R and asymp-on X R
  shows is-lower-set-wrt R {} X
  unfolding is-lower-set-wrt-def[OF assms] by simp

lemma is-lower-set-wrt-refl:
  fixes X :: 'a set and R :: 'a  $\Rightarrow$  'a  $\Rightarrow$  bool
  assumes transp-on X R and asymp-on X R
  shows is-lower-set-wrt R X X
  unfolding is-lower-set-wrt-def[OF assms] by simp

lemma is-lower-set-wrt-trans:
  fixes X Y Z :: 'a set and R :: 'a  $\Rightarrow$  'a  $\Rightarrow$  bool
  assumes
    transp-on Z R and asymp-on Z R and
    is-lower-set-wrt R X Y and is-lower-set-wrt R Y Z
  shows is-lower-set-wrt R X Z
proof –

```

```

have  $Y \subseteq Z$  and  $Y\text{-lower}$ :  $\forall l \in Y. \forall x \in Z. R\ x\ l \longrightarrow x \in Y$ 
  using  $\langle is\text{-lower}\text{-set}\text{-wrt}\ R\ Y\ Z \rangle$ 
  unfolding  $is\text{-lower}\text{-set}\text{-wrt}\text{-def}[OF\ \langle transp\text{-on}\ Z\ R \rangle\ \langle asymp\text{-on}\ Z\ R \rangle]$ 
  unfolding  $atomize\text{-conj}$  .

have  $transp\text{-on}\ Y\ R$  and  $asymp\text{-on}\ Y\ R$ 
  unfolding  $atomize\text{-conj}$ 
  using  $\langle transp\text{-on}\ Z\ R \rangle\ \langle asymp\text{-on}\ Z\ R \rangle\ \langle Y \subseteq Z \rangle$ 
  by ( $metis\ asymp\text{-on}\text{-subset}\ transp\text{-on}\text{-subset}$ )

have  $X \subseteq Y$  and  $X\text{-lower}$ :  $\forall l \in X. \forall x \in Y. R\ x\ l \longrightarrow x \in X$ 
  using  $\langle is\text{-lower}\text{-set}\text{-wrt}\ R\ X\ Y \rangle$ 
  unfolding  $is\text{-lower}\text{-set}\text{-wrt}\text{-def}[OF\ \langle transp\text{-on}\ Y\ R \rangle\ \langle asymp\text{-on}\ Y\ R \rangle]$ 
  unfolding  $atomize\text{-conj}$  .

have  $X \subseteq Z$ 
  using  $\langle X \subseteq Y \rangle\ \langle Y \subseteq Z \rangle$  by  $order$ 

moreover have ( $\forall l \in X. \forall x \in Z. R\ x\ l \longrightarrow x \in X$ )
  using  $\langle X \subseteq Y \rangle\ X\text{-lower}\ Y\text{-lower}$  by  $blast$ 

ultimately show  $?thesis$ 
  unfolding  $is\text{-lower}\text{-set}\text{-wrt}\text{-def}[OF\ \langle transp\text{-on}\ Z\ R \rangle\ \langle asymp\text{-on}\ Z\ R \rangle]$  ..
qed

lemma  $is\text{-lower}\text{-set}\text{-wrt}\text{-antisym}$ :
  fixes  $X\ Y :: 'a\ set$  and  $R :: 'a \Rightarrow 'a \Rightarrow bool$ 
  assumes
     $transp\text{-on}\ Y\ R$  and  $asymp\text{-on}\ Y\ R$  and
     $is\text{-lower}\text{-set}\text{-wrt}\ R\ X\ Y$  and  $is\text{-lower}\text{-set}\text{-wrt}\ R\ Y\ X$ 
  shows  $X = Y$ 
proof –
  have  $X \subseteq Y$ 
    using  $\langle is\text{-lower}\text{-set}\text{-wrt}\ R\ X\ Y \rangle$ 
    unfolding  $is\text{-lower}\text{-set}\text{-wrt}\text{-def}[OF\ \langle transp\text{-on}\ Y\ R \rangle\ \langle asymp\text{-on}\ Y\ R \rangle]$  ..

  have  $transp\text{-on}\ X\ R$  and  $asymp\text{-on}\ X\ R$ 
    unfolding  $atomize\text{-conj}$ 
    using  $\langle transp\text{-on}\ Y\ R \rangle\ \langle asymp\text{-on}\ Y\ R \rangle\ \langle X \subseteq Y \rangle$ 
    by ( $metis\ asymp\text{-on}\text{-subset}\ transp\text{-on}\text{-subset}$ )

  have  $Y \subseteq X$ 
    using  $\langle is\text{-lower}\text{-set}\text{-wrt}\ R\ Y\ X \rangle$ 
    unfolding  $is\text{-lower}\text{-set}\text{-wrt}\text{-def}[OF\ \langle transp\text{-on}\ X\ R \rangle\ \langle asymp\text{-on}\ X\ R \rangle]$  ..

  show  $X = Y$ 
    using  $\langle X \subseteq Y \rangle\ \langle Y \subseteq X \rangle$  by ( $metis\ subset\text{-antisym}$ )
qed

```

**lemma** *order-is-lower-set-wrt*:  
**fixes**  $R :: 'a \Rightarrow 'a \Rightarrow \text{bool}$   
**assumes** *transp R and asymp R*  
**shows** *class.order (is-lower-set-wrt R) (is-strict-lower-set-wrt R)*  
**proof** *unfold-locales*  
**have** *trans:  $\bigwedge A. \text{transp-on } A \ R$*   
**using**  $\langle \text{transp } R \rangle$  **by** (*metis subset-UNIV transp-on-subset*)  
  
**have** *asym:  $\bigwedge A. \text{asymp-on } A \ R$*   
**using**  $\langle \text{asymp } R \rangle$  **by** (*metis subset-UNIV asymp-on-subset*)  
  
**show**  $\bigwedge x \ y. \text{is-strict-lower-set-wrt } R \ x \ y = (\text{is-lower-set-wrt } R \ x \ y \wedge \neg \text{is-lower-set-wrt } R \ y \ x)$   
**unfolding** *is-lower-set-wrt-def[OF trans asym]*  
**unfolding** *is-strict-lower-set-wrt-def[OF trans asym]*  
**by** (*metis order-le-less subset-not-subset-eq*)  
  
**show**  $\bigwedge x. \text{is-lower-set-wrt } R \ x \ x$   
**using** *is-lower-set-wrt-refl[OF trans asym]* .  
  
**show**  $\bigwedge x \ y \ z. \text{is-lower-set-wrt } R \ x \ y \Longrightarrow \text{is-lower-set-wrt } R \ y \ z \Longrightarrow \text{is-lower-set-wrt } R \ x \ z$   
**using** *is-lower-set-wrt-trans[OF trans asym]* .  
  
**show**  $\bigwedge x \ y. \text{is-lower-set-wrt } R \ x \ y \Longrightarrow \text{is-lower-set-wrt } R \ y \ x \Longrightarrow x = y$   
**using** *is-lower-set-wrt-antisym[OF trans asym]* .  
**qed**

**lemma** *is-lower-set-wrt-insertI*:  
**assumes** *transp-on (insert x X) R and asymp-on (insert x X) R and*  
 $x \in X$  **and**  $\forall w \in X. R \ w \ x \longrightarrow w \in L$  **and** *is-lower-set-wrt R L X*  
**shows** *is-lower-set-wrt R (insert x L) X*  
**proof** –  
**have** *transp-on X R and asymp-on X R*  
**unfolding** *atomize-conj*  
**using**  $\langle \text{transp-on } (\text{insert } x \ X) \ R \rangle \langle \text{asymp-on } (\text{insert } x \ X) \ R \rangle$   
**by** (*metis asymp-on-subset subset-insertI transp-on-subset*)  
  
**have** *is-lower-set-wrt R (insert x L) X  $\longleftrightarrow$*   
 $\text{insert } x \ L \subseteq X \wedge (\forall l \in \text{insert } x \ L. \forall xa \in X. R \ xa \ l \longrightarrow xa \in \text{insert } x \ L)$   
**unfolding** *is-lower-set-wrt-def[OF  $\langle \text{transp-on } X \ R \rangle \langle \text{asymp-on } X \ R \rangle$  ..*  
  
**also have**  $\dots \longleftrightarrow x \in X \wedge L \subseteq X \wedge (\forall l \in \text{insert } x \ L. \forall xa \in X. R \ xa \ l \longrightarrow xa \in \text{insert } x \ L)$   
**by** *simp*  
  
**also have**  $\dots \longleftrightarrow x \in X \wedge L \subseteq X \wedge (\forall w \in X. R \ w \ x \longrightarrow w \in \text{insert } x \ L) \wedge$   
 $(\forall l \in L. \forall xa \in X. R \ xa \ l \longrightarrow xa \in \text{insert } x \ L)$   
**by** *simp*

**also have**  $\dots \longleftrightarrow x \in X \wedge L \subseteq X \wedge (\forall w \in X. R w x \longrightarrow w \in L) \wedge$   
 $(\forall l \in L. \forall y \in X. R y l \longrightarrow y \in \text{insert } x L)$   
**using**  $\langle \text{asympt-on } (\text{insert } x X) R \rangle$  **by**  $(\text{smt } (\text{verit}) \text{ asympt-onD insertCI insertE})$

**also have**  $\dots \longleftrightarrow x \in X \wedge (\forall w \in X. R w x \longrightarrow w \in L) \wedge \text{is-lower-set-wrt } R L$   
 $X$   
**using**  $\langle \text{is-lower-set-wrt } R L X \rangle$   
**unfolding**  $\text{is-lower-set-wrt-def}[OF \langle \text{transp-on } X R \rangle \langle \text{asympt-on } X R \rangle]$   
**by**  $\text{blast}$

**finally show**  $?thesis$   
**using**  $\text{assms}(\mathcal{B}-)$  **by**  $\text{argo}$   
**qed**

**lemma**  $\text{lower-set-wrt-appendI}$ :  
**assumes**  
 $\text{trans: transp-on } (\text{set } (xs @ ys)) R$  **and**  
 $\text{asym: asympt-on } (\text{set } (xs @ ys)) R$  **and**  
 $\text{sorted: sorted-wrt } R (xs @ ys)$   
**shows**  $\text{is-lower-set-wrt } R (\text{set } xs) (\text{set } (xs @ ys))$   
**unfolding**  $\text{is-lower-set-wrt-def}[OF \text{trans asym}]$   
**proof**  $(\text{intro conjI})$   
**show**  $\text{set } xs \subseteq \text{set } (xs @ ys)$   
**by**  $\text{simp}$

**next**  
**show**  $\forall l \in \text{set } xs. \forall x \in \text{set } (xs @ ys). R x l \longrightarrow x \in \text{set } xs$   
**using**  $\text{sorted}$   
**by**  $(\text{metis Un-iff asym asympt-on-def set-append sorted-wrt-append})$   
**qed**

**lemma**  $\text{sorted-and-lower-set-wrt-appendD-left}$ :  
**assumes**  $\text{transp-on } A R$  **and**  $\text{asympt-on } A R$  **and**  
 $\text{sorted-wrt } R (xs @ ys)$  **and**  $\text{is-lower-set-wrt } R (\text{set } (xs @ ys)) A$   
**shows**  $\text{sorted-wrt } R xs$  **and**  $\text{is-lower-set-wrt } R (\text{set } xs) A$   
**unfolding**  $\text{atomize-conj}$   
**using**  $\text{assms}$   
**by**  $(\text{smt } (\text{verit}) \text{ Un-iff asympt-on-def is-lower-set-wrt-def le-sup-iff set-append sorted-wrt-append subsetD})$

**lemma**  $\text{sorted-and-lower-set-wrt-appendD-right}$ :  
**assumes**  $\text{transp-on } A R$  **and**  $\text{asympt-on } A R$  **and**  
 $\text{sorted-wrt } (\lambda x y. R y x) (xs @ ys)$  **and**  $\text{is-lower-set-wrt } R (\text{set } (xs @ ys)) A$   
**shows**  $\text{sorted-wrt } (\lambda x y. R y x) ys$  **and**  $\text{is-lower-set-wrt } R (\text{set } ys) A$   
**unfolding**  $\text{atomize-conj}$   
**using**  $\text{assms}$   
**by**  $(\text{smt } (\text{verit}, \text{ccfv-threshold}) \text{ Un-iff asympt-onD is-lower-set-wrt-def le-sup-iff set-append})$



*sorted-wrt-append subsetD*)

**lemma** *not-in-lower-set-wrtI*:

**fixes**  $R :: 'a \Rightarrow 'a \Rightarrow \text{bool}$

**assumes** *trans*: *transp-on*  $Y R$  **and** *asym*: *asymp-on*  $Y R$

**shows** *is-lower-set-wrt*  $R X Y \implies y \notin X \implies y \in Y \implies R y z \implies z \notin X$

**unfolding** *is-lower-set-wrt-def*[*OF trans asym*]

**by** *blast*

**abbreviation** (**in** *preorder*) *is-lower-set* **where**

$is-lower-set \equiv is-lower-set-wrt (<)$

**lemmas** (**in** *preorder*) *is-lower-set-iff* =

*is-lower-set-wrt-def*[*OF transp-on-less asymp-on-less*]

**context** *linorder* **begin**

**sublocale** *is-lower-set: order is-lower-set-wrt* ( $<$ ) *is-strict-lower-set-wrt* ( $<$ )

**proof** (*rule order-is-lower-set-wrt*)

**show** *transp* ( $<$ )

**using** *transp-on-less* .

**next**

**show** *asympt* ( $<$ )

**using** *asympt-on-less* .

**qed**

**end**

**lemmas** (**in** *preorder*) *is-lower-set-empty*[*simp*] =

*is-lower-set-wrt-empty*[*OF transp-on-less asymp-on-less*]

**lemmas** (**in** *preorder*) *is-lower-set-insertI* =

*is-lower-set-wrt-insertI*[*OF transp-on-less asymp-on-less*]

**lemmas** (**in** *preorder*) *lower-set-appendI* =

*lower-set-wrt-appendI*[*OF transp-on-less asymp-on-less*]

**lemmas** (**in** *preorder*) *sorted-and-lower-set-appendD-left* =

*sorted-and-lower-set-wrt-appendD-left*[*OF transp-on-less asymp-on-less*]

**lemmas** (**in** *preorder*) *sorted-and-lower-set-appendD-right* =

*sorted-and-lower-set-wrt-appendD-right*[*OF transp-on-less asymp-on-less*]

**lemmas** (**in** *preorder*) *not-in-lower-setI* =

*not-in-lower-set-wrtI*[*OF transp-on-less asymp-on-less*]

**end**

```

theory HOL-Extra-Extra
  imports First-Order-Clause.HOL-Extra
begin

no-notation restrict-map (infixl |' 110)

lemma
  assumes  $\exists_{\leq 1} x. P x$ 
  shows finite { $x. P x$ }
  using assms Collect-eq-if-Uniq by fastforce

lemma finite-if-Uniq-Uniq:
  assumes
     $\exists_{\leq 1} x. P x$ 
     $\forall x. \exists_{\leq 1} y. Q x y$ 
  shows finite { $y. \exists x. P x \wedge Q x y$ }
  using assms
  by (smt (verit, best) Collect-eq-if-Uniq UniqI Uniq-D finite.emptyI finite-insert)

lemma finite-if-finite-finite:
  assumes
    finite { $x. P x$ }
     $\forall x. \textit{finite}$  { $y. Q x y$ }
  shows finite { $y. \exists x. P x \wedge Q x y$ }
  using assms by auto

lemma (in order) greater-wfp-on-finite-set: finite  $\mathcal{X} \implies \textit{Wellfounded.wfp-on}$   $\mathcal{X}$ 
( $>$ )
  using strict-partial-order-wfp-on-finite-set[OF transp-on-greater asymp-on-greater]
  .

lemma (in order) less-wfp-on-finite-set: finite  $\mathcal{X} \implies \textit{Wellfounded.wfp-on}$   $\mathcal{X}$  ( $<$ )
  using strict-partial-order-wfp-on-finite-set[OF transp-on-less asymp-on-less] .

lemma distinct-if-sorted-wrt-asymp:
  assumes asymp-on (set xs)  $R$  and sorted-wrt  $R$  xs
  shows distinct xs
  using assms
proof (induction xs)
  case Nil
  show ?case
    unfolding distinct.simps ..
next
  case (Cons x xs)

  have R-x-asymp:  $\forall y \in \textit{set xs}. R x y \longrightarrow \neg R y x$  and asymp-on (set xs)  $R$ 
  using Cons.prem1
  unfolding atomize-conj
  by (metis asymp-on-def list.set-intros(1) list.set-intros(2))

```

```

have  $R$ - $x$ :  $\forall y \in \text{set } xs. R \ x \ y$  and sorted-wrt  $R \ xs$ 
  using Cons.premis(2)
  unfolding atomize-conj sorted-wrt.simps
  by argo

have  $x \notin \text{set } xs$ 
proof (intro notI)
  assume  $x$ -in:  $x \in \text{set } xs$ 

  have  $R \ x \ x$ 
    using  $R$ - $x$   $x$ -in by metis

  moreover hence  $\neg R \ x \ x$ 
    using  $R$ - $x$ -asym  $x$ -in by metis

  ultimately show False
    by contradiction
qed

moreover have distinct  $xs$ 
  using Cons.IH  $\langle \text{asympt-on } (\text{set } xs) \ R \rangle \langle \text{sorted-wrt } R \ xs \rangle$  by argo

ultimately show ?case
  unfolding distinct.simps by argo
qed

lemma dropWhile-append-eq-rhs:
  fixes  $xs \ ys :: 'a \ \text{list}$  and  $P :: 'a \Rightarrow \text{bool}$ 
  assumes
     $\bigwedge x. x \in \text{set } xs \Longrightarrow P \ x$  and
     $\bigwedge y. y \in \text{set } ys \Longrightarrow \neg P \ y$ 
  shows dropWhile  $P \ (xs \ @ \ ys) = ys$ 
  using assms by simp

lemma mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone:
  fixes  $R :: 'a \Rightarrow 'a \Rightarrow \text{bool}$  and  $xs :: 'a \ \text{list}$  and  $P :: 'a \Rightarrow \text{bool}$ 
  assumes sorted-wrt  $R \ xs$  and monotone-on  $(\text{set } xs) \ R \ (\geq) \ P$ 
  shows  $x \in \text{set } (\text{dropWhile } P \ xs) \longleftrightarrow \neg P \ x \wedge x \in \text{set } xs$ 
  using assms
proof (induction xs)
  case Nil
  show ?case
    by simp
next
  case (Cons y xs)
  have  $\forall z \in \text{set } xs. R \ y \ z$  and sorted-wrt  $R \ xs$ 
    using  $\langle \text{sorted-wrt } R \ (y \ # \ xs) \rangle$  by simp-all

```

```

moreover have monotone-on (set xs) R ( $\geq$ ) P
  using ‹monotone-on (set (y # xs)) R ( $\geq$ ) P›
  by (metis monotone-on-subset set-subset-Cons)

ultimately have IH: (x ∈ set (dropWhile P xs)) = (¬ P x ∧ x ∈ set xs)
  using Cons.IH ‹sorted-wrt R xs› by metis

show ?case
proof (cases P y)
  case True
  thus ?thesis
    unfolding dropWhile.simps
    unfolding if-P[OF True]
    using IH by auto
  next
  case False
  then show ?thesis
    unfolding dropWhile.simps
    unfolding if-not-P[OF False]
    by (metis (full-types) Cons.prem1 Cons.prem2 le-boolD list.set-intros(1)
      monotone-on-def
      set-ConsD sorted-wrt.simps(2))
  qed
qed

lemma ball-set-dropWhile-if-sorted-wrt-and-monotone-on:
  fixes R :: 'a ⇒ 'a ⇒ bool and xs :: 'a list and P :: 'a ⇒ bool
  assumes sorted-wrt R xs and monotone-on (set xs) R ( $\geq$ ) P
  shows ∀ x ∈ set (dropWhile P xs). ¬ P x
  using mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone[OF assms] by
metis

lemma filter-set-eq-filter-set-minus-singleton:
  assumes ¬ P y
  shows {x ∈ X. P x} = {x ∈ X - {y}. P x}
  using assms by blast

lemma ex1-subset-eq-image-if-bij-betw:
  fixes f :: 'a ⇒ 'b and X :: 'a set and Y :: 'b set
  assumes bij-betw f X Y and Y' ⊆ Y
  shows ∃!X'. X' ⊆ X ∧ Y' = f ' X'
  using assms
  by (metis bij-betw-def inv-into-image-cancel subset-image-iff)

lemma Collect-eq-image-filter-Collect-if-bij-betw:
  fixes f :: 'a ⇒ 'b and X :: 'a set and Y :: 'b set
  assumes bij: bij-betw f X Y and sub: {y. P y} ⊆ Y
  shows {y. P y} = f ' {x. x ∈ X ∧ P (f x)}
  using ex1-subset-eq-image-if-bij-betw[OF bij sub]

```

by (smt (verit, best) Collect-cong image-def in-mono mem-Collect-eq)

**lemma** *restrict-map-ident-if-dom-subset*:  $\text{dom } \mathcal{M} \subseteq A \implies \text{restrict-map } \mathcal{M} A = \mathcal{M}$   
by (metis domIff ext in-mono restrict-map-def)

## 1.6 Move to HOL.Transitive-Closure

**lemma** *relpowp-right-unique*:  
fixes  $R :: 'a \Rightarrow 'a \Rightarrow \text{bool}$  and  $n :: \text{nat}$  and  $x y z :: 'a$   
assumes *runique*:  $\bigwedge x y z. R x y \implies R x z \implies y = z$   
shows  $(R \overset{\sim}{\sim} n) x y \implies (R \overset{\sim}{\sim} n) x z \implies y = z$   
**proof** (induction n arbitrary: x y z)  
case 0  
thus ?case  
by simp  
next  
case (Suc n')  
then obtain  $x' :: 'a$  where  
 $(R \overset{\sim}{\sim} n') x x'$  and  $R x' y$  and  $R x' z$   
by auto  
thus  $y = z$   
using *runique* by simp  
qed

**lemma** *Uniq-relpowp*:  
fixes  $n :: \text{nat}$  and  $R :: 'a \Rightarrow 'a \Rightarrow \text{bool}$   
assumes *runiq*:  $\forall x. \exists_{\leq 1} y. R x y$   
shows  $\exists_{\leq 1} y. (R \overset{\sim}{\sim} n) x y$   
**proof** (rule *Uniq-I*)  
fix y z  
assume  $(R \overset{\sim}{\sim} n) x y$  and  $(R \overset{\sim}{\sim} n) x z$   
show  $y = z$   
**proof** (rule *relpowp-right-unique*)  
show  $\bigwedge x y z. R x y \implies R x z \implies y = z$   
using *runiq* by (auto dest: *Uniq-D*)  
next  
show  $(R \overset{\sim}{\sim} n) x y$   
using  $\langle (R \overset{\sim}{\sim} n) x y \rangle$  .  
next  
show  $(R \overset{\sim}{\sim} n) x z$   
using  $\langle (R \overset{\sim}{\sim} n) x z \rangle$  .  
qed  
qed

**lemma** *relpowp-plus-of-right-unique*:  
assumes  
*right-unique*  $R$   
 $(R \overset{\sim}{\sim} m) x y$  and

```

  (R ~ (m + n)) x z
shows (R ~ n) y z
using assms(2,3)
proof (induction m arbitrary: x)
  case 0
  thus ?case
  by simp
next
  case (Suc m)
  then show ?case
  by (metis add-Suc assms(1) relpowp-Suc-E2 right-uniqueD)
qed

```

```

lemma relpowp-plusD:
  assumes (R ~ (m + n)) x z
  shows  $\exists y. (R ~ m) x y \wedge (R ~ n) y z$ 
  using assms
proof (induction m arbitrary: x)
  case 0
  thus ?case
  by simp
next
  case (Suc m)

```

```

  obtain y where R x y and (R ~ (m + n)) y z
  using Suc.prem1 by (metis add-Suc relpowp-Suc-D2)

```

```

  obtain y' where (R ~ m) y y' and (R ~ n) y' z
  using Suc.IH[OF <(R ~ (m + n)) y z>] by metis

```

```

  show ?case
  proof (intro exI conjI)
    show (R ~ Suc m) x y'
    using <R x y> <(R ~ m) y y'> by (metis relpowp-Suc-I2)
  next
    show (R ~ n) y' z
    using <(R ~ n) y' z> .
  qed
qed

```

```

lemma relpowp-Suc-of-right-unique:
  assumes
    right-unique R
    R x y and
    (R ~ Suc n) x z
  shows (R ~ n) y z
  using assms
  by (metis relpowp-Suc-D2 right-uniqueD)

```

```

lemma tranclp-if-relpowp:  $n \neq 0 \implies (R \overset{\sim}{\sim} n) x y \implies R^{++} x y$ 
  by (meson bot-nat-0.not-eq-extremum tranclp-power)

lemma transp-on-singleton[simp]: transp-on {x} R
  by (simp add: transp-on-def)

lemma rtranclp-rtranclp-compose-if-right-unique:
  assumes runique: right-unique R and  $R^{**} a b$  and  $R^{**} a c$ 
  shows  $R^{**} a b \wedge R^{**} b c \vee R^{**} a c \wedge R^{**} c b$ 
  using assms(2,3)
proof (induction b arbitrary: c rule: rtranclp-induct)
  case base
  thus ?case
    by simp
next
  case (step a' b)
  with runique show ?case
    by (metis converse-rtranclpE right-uniqueD rtranclp.rtrancl-into-rtrancl)
qed

lemma right-unique-terminating-rtranclp:
  assumes right-unique R
  shows right-unique ( $\lambda x y. R^{**} x y \wedge (\nexists z. R y z)$ )
  unfolding right-unique-def
  using rtranclp-rtranclp-compose-if-right-unique[OF <right-unique R>]
  by (metis converse-rtranclpE)

end
theory The-Optional
  imports Main
begin

definition The-optional :: ('a  $\implies$  bool)  $\implies$  'a option where
  The-optional P = (if  $\exists!x. P x$  then Some (THE x. P x) else None)

lemma The-optional-eq-SomeD: The-optional P = Some x  $\implies$  P x
  unfolding The-optional-def
  by (metis option.discI option.inject theI-unique)

lemma Some-eq-The-optionalD: Some x = The-optional P  $\implies$  P x
  using The-optional-eq-SomeD by metis

lemma The-optional-eq-NoneD: The-optional P = None  $\implies$   $\nexists!x. P x$ 
  unfolding The-optional-def
  by (metis option.discI)

lemma None-eq-The-optionalD: None = The-optional P  $\implies$   $\nexists!x. P x$ 
  unfolding The-optional-def
  by (metis option.discI)

```

```

lemma The-optional-eq-SomeI:
  assumes  $\exists_{\leq 1} x. P x$  and  $P x$ 
  shows The-optional P = Some x
  using assms by (metis The-optional-def the1-equality)

end
theory Full-Run
  imports
    VeriComp.Transfer-Extras
    HOL-Extra-Extra
begin

definition full-run where
  full-run  $\mathcal{R} x y \longleftrightarrow \mathcal{R}^{**} x y \wedge (\nexists z. \mathcal{R} y z)$ 

lemma Uniq-full-run:
  assumes Uniq-R:  $\bigwedge x. \exists_{\leq 1} y. R x y$ 
  shows  $\exists_{\leq 1} y. \text{full-run } R x y$ 
  unfolding full-run-def
  using assms
  by (smt (verit, best) Uniq-I right-unique-iff rtranclp-complete-run-right-unique)

lemma ex1-full-run:
  assumes Uniq-R:  $\bigwedge x. \exists_{\leq 1} y. R x y$  and wfP-R:  $wfP R^{-1-1}$ 
  shows  $\exists! y. \text{full-run } R x y$ 
proof –
  have  $\exists_{\leq 1} y. \text{full-run } R x y$ 
    using Uniq-full-run[of R x] Uniq-R by argo

  moreover have  $\exists y. \text{full-run } R x y$ 
    using ex-terminating-rtranclp[OF wfP-R, of x, folded full-run-def] .

  ultimately show ?thesis
    using Uniq-implies-ex1 by metis
qed

lemma full-run-preserves-invariant:
  assumes
    run: full-run  $R x y$  and
    P-init:  $P x$  and
    R-preserves-P:  $\bigwedge x y. R x y \implies P x \implies P y$ 
  shows  $P y$ 
proof –
  from run have  $R^{**} x y$ 
    unfolding full-run-def by simp
  thus  $P y$ 
    using P-init
  proof (induction x rule: converse-rtranclp-induct)

```



```

    case base
    thus ?case
      by assumption
  next
  case (step x x')
  then show ?case
    using R-preserves-P by metis
  qed
qed

end
theory Background
imports
  Simple-Clause-Learning.SCL-FOL
  Simple-Clause-Learning.Correct-Termination
  Simple-Clause-Learning.Initial-Literals-Generalize-Learned-Literals
  Simple-Clause-Learning.Termination
  Ground-Ordered-Resolution
  Min-Max-Least-Greatest.Min-Max-Least-Greatest-FSet
  First-Order-Clause.Multiset-Extra
  VeriComp.Compiler
  HOL-ex.Sketch-and-Explore
  HOL-Library.FuncSet
  Lower-Set
  HOL-Extra-Extra
  The-Optional
  Full-Run
begin

lemma I  $\models$  L  $\longleftrightarrow$  (is-pos L  $\longleftrightarrow$  atm-of L  $\in$  I)
  by (cases L) simp-all

```

## 2 Move to *HOL-Library.Multiset*

```

lemmas strict-subset-implies-multp = subset-implies-multp
hide-fact subset-implies-multp

```

```

lemma subset-implies-reflclp-multp: A  $\subseteq$ # B  $\implies$  (multp R)== A B
  by (metis reflclp-iff strict-subset-implies-multp subset-mset.le-imp-less-or-eq)

```

```

lemma member-mset-repeat-msetD: L  $\in$ # repeat-mset n M  $\implies$  L  $\in$ # M
  by (induction n) auto

```

```

lemma member-mset-repeat-mset-Suc[simp]: L  $\in$ # repeat-mset (Suc n) M  $\longleftrightarrow$  L
 $\in$ # M
  by (metis member-mset-repeat-msetD repeat-mset-Suc union-iff)

```

```

lemma image-msetI: x  $\in$ # M  $\implies$  f x  $\in$ # image-mset f M
  by (metis imageI in-image-mset)

```

**lemma** *inj-image-mset-mem-iff*:  $\text{inj } f \implies f x \in\# \text{ image-mset } f M \longleftrightarrow x \in\# M$   
**by** (*simp add: inj-image-mem-iff*)

### 3 Move to *HOL-Library.FSet*

**declare** *wfP-pfsubset*[*intro*]

**syntax**

*-FFilter* :: *p*trn  $\Rightarrow$  'a *fset*  $\Rightarrow$  bool  $\Rightarrow$  'a *fset* ((1{|- | $\in$ | -/ -|}))

**translations**

{|x | $\in$ | X. P|} == *CONST ffilter* ( $\lambda x. P$ ) X

**lemma** *fimage-ffUnion*:  $f \mid\uparrow \text{ffUnion } SS = \text{ffUnion } ((\mid\uparrow) f \mid\uparrow SS)$

**proof** (*intro fsubset-antisym fsubsetI*)

**fix** x **assume** x | $\in$ | f | $\uparrow$  ffUnion SS

**then obtain** y **where** y | $\in$ | ffUnion SS **and** x = f y

**by** *auto*

**thus** x | $\in$ | ffUnion ((| $\uparrow$ ) f | $\uparrow$  SS)

**unfolding** *fmember-ffUnion-iff*

**by** (*metis UN-E ffUnion.rep-eq fimage-eqI*)

**next**

**fix** x **assume** x | $\in$ | ffUnion ((| $\uparrow$ ) f | $\uparrow$  SS)

**then obtain** S **where** S | $\in$ | SS **and** x | $\in$ | f | $\uparrow$  S

**unfolding** *fmember-ffUnion-iff* **by** *auto*

**then show** x | $\in$ | f | $\uparrow$  ffUnion SS

**by** (*metis ffUnion-fsubset-iff fimage-mono fin-mono fsubsetI*)

**qed**

**lemma** *ffilter-eq-ffilter-minus-singleton*:

**assumes**  $\neg P y$

**shows** {|x | $\in$ | X. P x|} = {|x | $\in$ | X - {|y|}. P x|}

**using** *assms* **by** (*induction X*) *auto*

**lemma** *fun-upd-fimage*:  $f(x := y) \mid\uparrow A = (\text{if } x \mid\in\mid A \text{ then } \text{finsert } y (f \mid\uparrow (A - \{|x|\})) \text{ else } f \mid\uparrow A)$

**using** *fun-upd-image*

**by** (*smt (verit) bot-fset.rep-eq finsert.rep-eq fset.set-map fset-cong minus-fset.rep-eq*)

**lemma** *ffilter-fempty*[*simp*]: *ffilter* P {|}| = {|}|

**by** (*metis ex-fin-conv fmember-filter*)

**lemma** *fstrict-subset-iff-fset-strict-subset-fset*:

**fixes** X Y :: - *fset*

**shows** X | $\subset$ | Y  $\longleftrightarrow$  *fset* X  $\subset$  *fset* Y

**by** *blast*

**lemma** (**in** *linorder*) *ex1-sorted-list-for-fset*:

$\exists!xs. \text{sorted-wrt } (<) xs \wedge \text{fset-of-list } xs = X$   
**using** *ex1-sorted-list-for-set-if-finite*  
**by** (*metis finite-fset fset-cong fset-of-list.rep-eq*)

**lemma** (**in** *linorder*) *is-least-in-fset-filterD*:  
**assumes** *is-least-in-fset-wrt* (<) (*ffilter P X*) *x*  
**shows**  $x \in X \wedge P x$   
**using** *assms*  
**by** (*simp-all add: is-least-in-fset-wrt-iff*)

## 4 Move to *VeriComp.Simulation*

**locale** *forward-simulation-with-measuring-function* =  
*L1: semantics step1 final1* +  
*L2: semantics step2 final2*  
**for**  
*step1* :: *'state1*  $\Rightarrow$  *'state1*  $\Rightarrow$  *bool* **and**  
*step2* :: *'state2*  $\Rightarrow$  *'state2*  $\Rightarrow$  *bool* **and**  
*final1* :: *'state1*  $\Rightarrow$  *bool* **and**  
*final2* :: *'state2*  $\Rightarrow$  *bool* +  
**fixes**  
*match* :: *'state1*  $\Rightarrow$  *'state2*  $\Rightarrow$  *bool* **and**  
*measure* :: *'state1*  $\Rightarrow$  *'index* **and**  
*order* :: *'index*  $\Rightarrow$  *'index*  $\Rightarrow$  *bool* (**infix**  $\square$  70)  
**assumes**  
*wfp-order*:  
*wfp* ( $\square$ ) **and**  
*match-final*:  
*match s1 s2*  $\Longrightarrow$  *final1 s1*  $\Longrightarrow$  *final2 s2* **and**  
*simulation*:  
*match s1 s2*  $\Longrightarrow$  *step1 s1 s1'*  $\Longrightarrow$   
 $(\exists s2'. \text{step2}^{++} s2 s2' \wedge \text{match } s1' s2') \vee (\text{match } s1' s2 \wedge \text{measure } s1' \square$   
*measure s1)*  
**begin**  
**sublocale** *forward-simulation where*  
*step1 = step1 and step2 = step2 and final1 = final1 and final2 = final2 and*  
*order = order and*  
*match =  $\lambda i x y. i = \text{measure } x \wedge \text{match } x y$*   
**proof** *unfold-locales*  
**show**  $\bigwedge i s1 s2. i = \text{measure } s1 \wedge \text{match } s1 s2 \Longrightarrow \text{final1 } s1 \Longrightarrow \text{final2 } s2$   
**using** *match-final by metis*  
**next**  
**show**  $\bigwedge i s1 s2 s1'. i = \text{measure } s1 \wedge \text{match } s1 s2 \Longrightarrow \text{step1 } s1 s1' \Longrightarrow$   
 $(\exists i' s2'. \text{step2}^{++} s2 s2' \wedge i' = \text{measure } s1' \wedge \text{match } s1' s2') \vee$   
 $(\exists i'. (i' = \text{measure } s1' \wedge \text{match } s1' s2) \wedge i' \square i)$   
**using** *simulation by metis*  
**next**  
**show** *wfp* ( $\square$ )

```

    using wfp-order .
qed

end

locale backward-simulation-with-measuring-function =
  L1: semantics step1 final1 +
  L2: semantics step2 final2
for
  step1 :: 'state1  $\Rightarrow$  'state1  $\Rightarrow$  bool and
  step2 :: 'state2  $\Rightarrow$  'state2  $\Rightarrow$  bool and
  final1 :: 'state1  $\Rightarrow$  bool and
  final2 :: 'state2  $\Rightarrow$  bool +
fixes
  match :: 'state1  $\Rightarrow$  'state2  $\Rightarrow$  bool and
  measure :: 'state2  $\Rightarrow$  'index and
  order :: 'index  $\Rightarrow$  'index  $\Rightarrow$  bool (infix  $\sqsubset$  70)
assumes
  wfp-order:
    wfp ( $\sqsubset$ ) and
  match-final:
    match s1 s2  $\Longrightarrow$  final2 s2  $\Longrightarrow$  final1 s1 and
  simulation:
    match s1 s2  $\Longrightarrow$  step2 s2 s2'  $\Longrightarrow$ 
      ( $\exists$  s1'. step1++ s1 s1'  $\wedge$  match s1' s2')  $\vee$  (match s1 s2'  $\wedge$  measure s2'  $\sqsubset$ 
measure s2)
begin

sublocale backward-simulation where
  step1 = step1 and step2 = step2 and final1 = final1 and final2 = final2 and
  order = order and
  match =  $\lambda$ i x y. i = measure y  $\wedge$  match x y
proof unfold-locales
  show  $\bigwedge$ i s1 s2. i = measure s2  $\wedge$  match s1 s2  $\Longrightarrow$  final2 s2  $\Longrightarrow$  final1 s1
    using match-final by metis
next
  show  $\bigwedge$ i1 s1 s2 s2'. i1 = measure s2  $\wedge$  match s1 s2  $\Longrightarrow$  step2 s2 s2'  $\Longrightarrow$ 
    ( $\exists$  i2 s1'. step1++ s1 s1'  $\wedge$  i2 = measure s2'  $\wedge$  match s1' s2')  $\vee$ 
    ( $\exists$  i2. (i2 = measure s2'  $\wedge$  match s1 s2')  $\wedge$  i2  $\sqsubset$  i1)
    using simulation by metis
next
  show wfp ( $\sqsubset$ )
    using wfp-order .
qed

end

```

## 5 Move to *Simple-Clause-Learning.SCL-FOL*

**definition** *trail-true-lit* :: (- literal × - option) list ⇒ - literal ⇒ bool **where**  
*trail-true-lit*  $\Gamma$   $L \longleftrightarrow L \in \text{fst } \text{' set } \Gamma$

**definition** *trail-false-lit* :: (- literal × - option) list ⇒ - literal ⇒ bool **where**  
*trail-false-lit*  $\Gamma$   $L \longleftrightarrow \neg L \in \text{fst } \text{' set } \Gamma$

**definition** *trail-true-cls* :: (- literal × - option) list ⇒ - clause ⇒ bool **where**  
*trail-true-cls*  $\Gamma$   $C \longleftrightarrow (\exists L \in \# C. \text{trail-true-lit } \Gamma L)$

**definition** *trail-false-cls* :: (- literal × - option) list ⇒ - clause ⇒ bool **where**  
*trail-false-cls*  $\Gamma$   $C \longleftrightarrow (\forall L \in \# C. \text{trail-false-lit } \Gamma L)$

**lemma** *trail-false-cls-mempty[simp]*: *trail-false-cls*  $\Gamma$  {#}  
**by** (*simp add: trail-false-cls-def*)

**definition** *trail-defined-lit* :: (- literal × - option) list ⇒ - literal ⇒ bool **where**  
*trail-defined-lit*  $\Gamma$   $L \longleftrightarrow (L \in \text{fst } \text{' set } \Gamma \vee \neg L \in \text{fst } \text{' set } \Gamma)$

**lemma** *trail-defined-lit-iff*: *trail-defined-lit*  $\Gamma$   $L \longleftrightarrow \text{atm-of } L \in \text{atm-of } \text{' fst } \text{' set } \Gamma$   
**by** (*simp add: atm-of-in-atm-of-set-iff-in-set-or-uminus-in-set trail-defined-lit-def*)

**definition** *trail-defined-cls* :: (- literal × - option) list ⇒ - clause ⇒ bool **where**  
*trail-defined-cls*  $\Gamma$   $C \longleftrightarrow (\forall L \in \# C. \text{trail-defined-lit } \Gamma L)$

**lemma** *trail-defined-lit-iff-true-or-false*:  
*trail-defined-lit*  $\Gamma$   $L \longleftrightarrow \text{trail-true-lit } \Gamma L \vee \text{trail-false-lit } \Gamma L$   
**unfolding** *trail-defined-lit-def trail-false-lit-def trail-true-lit-def* **by** (*rule refl*)

**lemma** *trail-true-or-false-cls-if-defined*:  
*trail-defined-cls*  $\Gamma$   $C \implies \text{trail-true-cls } \Gamma C \vee \text{trail-false-cls } \Gamma C$   
**unfolding** *trail-defined-cls-def trail-false-cls-def trail-true-cls-def*  
**unfolding** *trail-defined-lit-iff-true-or-false*  
**by** *blast*

**lemma** *subtrail-falseI*:  
**assumes** *tr-false*: *trail-false-cls*  $((L, Cl) \# \Gamma) C$  **and** *L-not-in*:  $\neg L \notin \# C$   
**shows** *trail-false-cls*  $\Gamma C$   
**unfolding** *trail-false-cls-def*  
**proof** (*rule ballI*)  
**fix**  $M$   
**assume** *M-in*:  $M \in \# C$

**from** *M-in L-not-in* **have** *M-neq-L*:  $M \neq \neg L$  **by** *auto*

**from** *M-in tr-false* **have** *tr-false-lit-M*: *trail-false-lit*  $((L, Cl) \# \Gamma) M$   
**unfolding** *trail-false-cls-def* **by** *simp*

**thus** *trail-false-lit*  $\Gamma$   $M$   
**unfolding** *trail-false-lit-def*  
**using**  $M\text{-neq-}L$   
**by** (*cases*  $L$ ; *cases*  $M$ ) (*simp-all add: trail-interp-def trail-false-lit-def*)  
**qed**

**inductive** *trail-consistent* :: ('a literal  $\times$  'b option) list  $\Rightarrow$  bool **where**  
*Nil[*simp*]: trail-consistent [] |*  
*Cons:  $\neg$  trail-defined-lit  $\Gamma$   $L \Longrightarrow$  trail-consistent  $\Gamma \Longrightarrow$  trail-consistent (( $L$ ,  $u$ ) #  $\Gamma$ )*

**lemma** *distinct-lits-if-trail-consistent*:  
*trail-consistent  $\Gamma \Longrightarrow$  distinct (map fst  $\Gamma$ )*  
**by** (*induction*  $\Gamma$  *rule: trail-consistent.induct*)  
*(simp-all add: image-comp trail-defined-lit-def)*

**lemma** *trail-true-lit-if-trail-true-suffix*:  
*suffix  $\Gamma' \Gamma \Longrightarrow$  trail-true-lit  $\Gamma' K \Longrightarrow$  trail-true-lit  $\Gamma K$*   
**by** (*meson image-mono set-mono-suffix subsetD trail-true-lit-def*)

**lemma** *trail-true-cls-if-trail-true-suffix*:  
*suffix  $\Gamma' \Gamma \Longrightarrow$  trail-true-cls  $\Gamma' C \Longrightarrow$  trail-true-cls  $\Gamma C$*   
**using** *trail-true-cls-def trail-true-lit-if-trail-true-suffix* **by** *metis*

**lemma** *trail-false-lit-if-trail-false-suffix*:  
*suffix  $\Gamma' \Gamma \Longrightarrow$  trail-false-lit  $\Gamma' K \Longrightarrow$  trail-false-lit  $\Gamma K$*   
**by** (*meson image-mono set-mono-suffix subsetD trail-false-lit-def*)

**lemma** *trail-false-cls-if-trail-false-suffix*:  
*suffix  $\Gamma' \Gamma \Longrightarrow$  trail-false-cls  $\Gamma' C \Longrightarrow$  trail-false-cls  $\Gamma C$*   
**using** *trail-false-cls-def trail-false-lit-if-trail-false-suffix* **by** *metis*

**lemma** *trail-defined-lit-if-trail-defined-suffix*:  
*suffix  $\Gamma' \Gamma \Longrightarrow$  trail-defined-lit  $\Gamma' K \Longrightarrow$  trail-defined-lit  $\Gamma K$*   
**unfolding** *trail-defined-lit-def*  
**by** (*metis (no-types) Un-iff image-Un set-append suffix-def*)

**lemma** *trail-defined-cls-if-trail-defined-suffix*:  
*suffix  $\Gamma' \Gamma \Longrightarrow$  trail-defined-cls  $\Gamma' C \Longrightarrow$  trail-defined-cls  $\Gamma C$*   
**unfolding** *trail-defined-cls-def* **by** (*metis trail-defined-lit-if-trail-defined-suffix*)

**lemma** *not-trail-true-lit-and-trail-false-lit*:  
**fixes**  $\Gamma$  :: ('a literal  $\times$  'b option) list **and**  $L$  :: 'a literal  
**shows** *trail-consistent  $\Gamma \Longrightarrow \neg$  (trail-true-lit  $\Gamma L \wedge$  trail-false-lit  $\Gamma L$ )*  
**proof** (*induction*  $\Gamma$  *rule: trail-consistent.induct*)  
**case** *Nil*  
**show** *?case*  
**by** (*simp add: trail-true-cls-def trail-false-cls-def trail-true-lit-def trail-false-lit-def*)  
**next**

```

case (Cons  $\Gamma$  K annot)
then show ?case
  unfolding trail-defined-lit-def trail-false-lit-def trail-true-lit-def
  by (metis (no-types, opaque-lifting) fst-conv image-insert insertE list.simps(15)
uminus-not-id'
      uminus-of-uminus-id)
qed

lemma not-trail-true-cls-and-trail-false-cls:
  fixes  $\Gamma :: ('a \text{ literal} \times 'b \text{ option}) \text{ list}$  and C :: 'a clause
  shows trail-consistent  $\Gamma \implies \neg (\text{trail-true-cls } \Gamma \ C \wedge \text{trail-false-cls } \Gamma \ C)$ 
proof (induction  $\Gamma$  rule: trail-consistent.induct)
  case Nil
  show ?case
  by (simp add: trail-true-cls-def trail-false-cls-def trail-true-lit-def trail-false-lit-def)
next
  case (Cons  $\Gamma$  L u)
  thus ?case
  using not-trail-true-lit-and-trail-false-lit
  by (metis trail-consistent.Cons trail-false-cls-def trail-true-cls-def)
qed

lemma not-lit-and-comp-lit-false-if-trail-consistent:
  assumes trail-consistent  $\Gamma$ 
  shows  $\neg (\text{trail-false-lit } \Gamma \ L \wedge \text{trail-false-lit } \Gamma \ (\neg L))$ 
  using assms
proof (induction  $\Gamma$ )
  case Nil
  show ?case
  by (simp add: trail-false-lit-def)
next
  case (Cons  $\Gamma$  K u)
  show ?case
  proof (cases  $K = L \vee K = \neg L$ )
    case True
    thus ?thesis
    unfolding trail-false-lit-def uminus-of-uminus-id
    unfolding de-Morgan-conj list.set image-insert prod.sel
    by (metis Cons.hyps(1) insertE trail-defined-lit-def uminus-not-id' umi-
minus-of-uminus-id)
  next
  case False
  thus ?thesis
  unfolding trail-false-lit-def uminus-of-uminus-id
  by (metis (no-types, lifting) Cons.IH fst-conv image-iff set-ConsD trail-false-lit-def
      uminus-of-uminus-id)
qed
qed

```

**lemma** *not-both-lit-and-comp-lit-in-false-clause-if-trail-consistent*:  
**assumes**  $\Gamma$ -consistent: trail-consistent  $\Gamma$  **and**  $C$ -false: trail-false-clc  $\Gamma$   $C$   
**shows**  $\neg (L \in\# C \wedge -L \in\# C)$   
**proof** (rule notI)  
**assume**  $L \in\# C \wedge -L \in\# C$   
  
**hence** trail-false-lit  $\Gamma$   $L \wedge$  trail-false-lit  $\Gamma$   $(-L)$   
**using**  $C$ -false **unfolding** trail-false-clc-def **by** metis  
  
**thus** False  
**using**  $\Gamma$ -consistent not-lit-and-comp-lit-false-if-trail-consistent **by** metis  
**qed**

## 6 Move to ground ordered resolution

**lemma** (in ground-ordered-resolution-calculus) *unique-ground-resolution*:  
**shows**  $\exists_{\leq 1} C$ . ground-resolution  $P1$   $P2$   $C$   
**proof** (intro Uniq-I)  
**fix**  $C$   $C'$   
**assume** ground-resolution  $P1$   $P2$   $C$  **and** ground-resolution  $P1$   $P2$   $C'$   
**thus**  $C = C'$   
**unfolding** ground-resolution.simps  
**apply** (elim exE conjE)  
**apply** simp  
**by** (metis asymp-on-less-lit is-maximal-in-mset-wrt-if-is-greatest-in-mset-wrt  
is-maximal-in-mset-wrt-iff literal.inject(1) totalpD totalp-on-less-lit transp-on-less-lit  
union-single-eq-diff)  
**qed**

**lemma** (in ground-ordered-resolution-calculus) *unique-ground-factoring*:  
**shows**  $\exists_{\leq 1} C$ . ground-factoring  $P$   $C$   
**proof** (intro Uniq-I)  
**fix**  $P$   $C$   $C'$   
**assume** ground-factoring  $P$   $C$  **and** ground-factoring  $P$   $C'$   
**thus**  $C = C'$   
**unfolding** ground-factoring.simps  
**by** (metis asymp-on-less-lit is-maximal-in-mset-wrt-iff totalpD totalp-less-lit  
transp-on-less-lit union-single-eq-diff)  
**qed**

**lemma** (in ground-ordered-resolution-calculus) *termination-ground-factoring*:  
**shows** wfp ground-factoring<sup>-1-1</sup>  
**proof** (rule wfp-if-convertible-to-wfp)  
**show**  $\bigwedge x y$ . ground-factoring<sup>-1-1</sup>  $x$   $y \implies x \prec_c y$   
**using** ground-factoring-smaller-conclusion **by** simp  
**next**  
**show** wfp ( $\prec_c$ )  
**by** simp  
**qed**



**lemma** (in *ground-ordered-resolution-calculus*) *atms-of-concl-subset-if-ground-resolution*:  
**assumes** *ground-resolution*  $P_1 P_2 C$   
**shows**  $\text{atms-of } C \subseteq \text{atms-of } P_1 \cup \text{atms-of } P_2$   
**using** *assms* **by** (*cases*  $P_1 P_2 C$  *rule: ground-resolution.cases*) (*auto simp add: atms-of-def*)

**lemma** (in *ground-ordered-resolution-calculus*) *strict-subset-mset-if-ground-factoring*:  
**assumes** *ground-factoring*  $P C$   
**shows**  $C \subset\# P$   
**using** *assms* **by** (*cases*  $P C$  *rule: ground-factoring.cases*) *simp*

**lemma** (in *ground-ordered-resolution-calculus*) *set-mset-eq-set-mset-if-ground-factoring*:  
**assumes** *ground-factoring*  $P C$   
**shows**  $\text{set-mset } P = \text{set-mset } C$   
**using** *assms* **by** (*cases*  $P C$  *rule: ground-factoring.cases*) *simp*

**lemma** (in *ground-ordered-resolution-calculus*) *atms-of-concl-eq-if-ground-factoring*:  
**assumes** *ground-factoring*  $P C$   
**shows**  $\text{atms-of } C = \text{atms-of } P$   
**using** *assms* **by** (*cases*  $P C$  *rule: ground-factoring.cases*) *simp*

**lemma** (in *ground-ordered-resolution-calculus*) *ground-factoring-preserves-maximal-literal*:  
**assumes** *ground-factoring*  $P C$   
**shows**  $\text{is-maximal-lit } L P = \text{is-maximal-lit } L C$   
**using** *assms* **by** (*cases*  $P C$  *rule: ground-factoring.cases*) (*simp add: is-maximal-in-mset-wrt-iff*)

**lemma** (in *ground-ordered-resolution-calculus*) *ground-factorings-preserves-maximal-literal*:  
**assumes** *ground-factoring\*\**  $P C$   
**shows**  $\text{is-maximal-lit } L P = \text{is-maximal-lit } L C$   
**using** *assms*  
**by** (*induction*  $P$  *rule: converse-rtranclp-induct*)  
(*simp-all add: ground-factoring-preserves-maximal-literal*)

**lemma** (in *ground-ordered-resolution-calculus*) *ground-factoring-reduces-maximal-pos-lit*:  
**assumes** *ground-factoring*  $P C$  **and** *is-pos*  $L$  **and**  
*is-maximal-lit*  $L P$  **and** *count*  $P L = \text{Suc } (\text{Suc } n)$   
**shows** *is-maximal-lit*  $L C$  **and** *count*  $C L = \text{Suc } n$   
**unfolding** *atomize-conj*  
**using**  $\langle \text{ground-factoring } P C \rangle$   
**proof** (*cases*  $P C$  *rule: ground-factoring.cases*)  
**case** (*ground-factoringI*  $t P'$ )  
**then show**  $\text{is-maximal-lit } L C \wedge \text{count } C L = \text{Suc } n$   
**by** (*metis* *assms*(1) *assms*(3) *assms*(4) *asympt-on-less-lit count-add-mset*  
*ground-factoring-preserves-maximal-literal is-maximal-in-mset-wrt-iff nat.inject*  
*totalpD*  
*totalp-less-lit transp-on-less-lit*)

**qed**

**lemma** (in *ground-ordered-resolution-calculus*) *ground-factorings-reduces-maximal-pos-lit*:  
**assumes** (*ground-factoring*  $\sim m$ )  $P C$  **and**  $m \leq \text{Suc } n$  **and** *is-pos*  $L$  **and**  
*is-maximal-lit*  $L P$  **and**  $\text{count } P L = \text{Suc } (\text{Suc } n)$   
**shows** *is-maximal-lit*  $L C$  **and**  $\text{count } C L = \text{Suc } (\text{Suc } n - m)$   
**unfolding** *atomize-conj*  
**using**  $\langle (\text{ground-factoring } \sim m) P C \rangle \langle m \leq \text{Suc } n \rangle$   
**proof** (*induction*  $m$  *arbitrary*:  $C$ )  
**case**  $0$   
**hence**  $P = C$   
**by** *simp*  
**then show** *?case*  
**using** *assms(4,5)* **by** *simp*  
**next**  
**case** ( $\text{Suc } m$ )  
**then show** *?case*  
**by** (*metis Suc-diff-le Suc-leD assms(3) diff-Suc-Suc ground-factoring-reduces-maximal-pos-lit(2)*  
*ground-factorings-preserves-maximal-literal relpowp-Suc-E relpowp-imp-rtranclp*)  
**qed**

**lemma** (in *ground-ordered-resolution-calculus*) *full-ground-factorings-reduces-maximal-pos-lit*:  
**assumes** *steps*: (*ground-factoring*  $\sim \text{Suc } n$ )  $P C$  **and** *L-pos*: *is-pos*  $L$  **and**  
*L-max*: *is-maximal-lit*  $L P$  **and** *L-count*:  $\text{count } P L = \text{Suc } (\text{Suc } n)$   
**shows** *is-maximal-lit*  $L C$  **and**  $\text{count } C L = \text{Suc } 0$   
**unfolding** *atomize-conj*  
**using** *steps L-max L-count*  
**proof** (*induction*  $n$  *arbitrary*:  $P$ )  
**case**  $0$   
**then show** *?case*  
**using** *L-pos ground-factorings-reduces-maximal-pos-lit[of Suc 0 P C 0]* **by** *simp*  
**next**  
**case** ( $\text{Suc } n$ )  
**from** *Suc.prem*s **obtain**  $P'$  **where**  
*ground-factoring*  $P P'$  **and** (*ground-factoring*  $\sim \text{Suc } n$ )  $P' C$   
**by** (*metis relpowp-Suc-D2*)  
  
**from** *Suc.prem*s **have** *is-maximal-lit*  $L P'$  **and**  $\text{count } P' L = \text{Suc } (\text{Suc } n)$   
**using** *ground-factoring-reduces-maximal-pos-lit[OF  $\langle \text{ground-factoring } P P' \rangle$*   
*L-pos]* **by** *simp-all*  
  
**thus** *?case*  
**using** *Suc.IH[OF  $\langle (\text{ground-factoring } \sim \text{Suc } n) P' C \rangle$*  **by** *metis*  
**qed**

## 7 Move somewhere?

**lemma** *true-cls-imp-neq-empty*:  $\mathcal{I} \models C \implies C \neq \{\#\}$   
**by** *blast*

**lemma** *lift-tranclp-to-pairs-with-constant-fst*:

$(R x)^{++} y z \implies (\lambda(x, y) (x', z). x = x' \wedge R x y z)^{++} (x, y) (x, z)$   
**by** (*induction* *z arbitrary*: *rule*: *tranclp-induct*) (*auto simp*: *tranclp.trancl-into-trancl*)

**abbreviation** (*in preorder*) *is-lower-fset* **where**  
*is-lower-fset*  $X Y \equiv \text{is-lower-set-wrt } (<) (fset X) (fset Y)$

**lemma** *lower-set-wrt-prefixI*:

**assumes**

*trans*: *transp-on* (*set zs*) *R* **and**

*asym*: *asympt-on* (*set zs*) *R* **and**

*sorted*: *sorted-wrt* *R* *zs* **and**

*prefix*: *prefix* *xs zs*

**shows** *is-lower-set-wrt* *R* (*set xs*) (*set zs*)

**proof** –

**obtain** *ys* **where**  $zs = xs @ ys$

**using** *prefix* **by** (*auto elim*: *prefixE*)

**show** *?thesis*

**using** *trans* *asym* *sorted*

**unfolding**  $\langle zs = xs @ ys \rangle$

**by** (*metis* *lower-set-wrt-appendI*)

**qed**

**lemmas** (*in preorder*) *lower-set-prefixI* =  
*lower-set-wrt-prefixI*[*OF transp-on-less asympt-on-less*]

**lemma** *lower-set-wrt-suffixI*:

**assumes**

*trans*: *transp-on* (*set zs*) *R* **and**

*asym*: *asympt-on* (*set zs*) *R* **and**

*sorted*: *sorted-wrt*  $R^{-1-1}$  *zs* **and**

*suffix*: *suffix* *ys zs*

**shows** *is-lower-set-wrt* *R* (*set ys*) (*set zs*)

**proof** –

**obtain** *xs* **where**  $zs = xs @ ys$

**using** *suffix* **by** (*auto elim*: *suffixE*)

**show** *?thesis*

**using** *trans* *asym* *sorted*

**unfolding**  $\langle zs = xs @ ys \rangle$

**by** (*smt* (*verit*, *del-insts*) *Un-iff*  $\langle zs = xs @ ys \rangle$  *asympt-onD* *asympt-on-conversep*  
*conversepI*

*is-lower-set-wrt-def* *set-append* *set-mono-suffix* *sorted-wrt-append* *suffix*)

**qed**

**lemmas** (*in preorder*) *lower-set-suffixI* =  
*lower-set-wrt-suffixI*[*OF transp-on-less asympt-on-less*]

**lemma** *true-cls-repeat-mset-Suc[simp]*:  $I \models \text{repeat-mset } (\text{Suc } n) C \longleftrightarrow I \models C$   
**by** (*induction n*) *simp-all*

**lemma** (*in backward-simulation*)  
**assumes** *match i S1 S2* **and**  $\neg L1.\text{inf-step } S1$   
**shows**  $\neg L2.\text{inf-step } S2$   
**using** *assms match-inf* **by** *metis*

**lemma** (*in scl-fol-calculus*) *grounding-of-cls-ground*:  
**assumes** *is-ground-cls N*  
**shows** *grounding-of-cls N = N*  
**proof** –  
**have** *grounding-of-cls N =  $(\bigcup C \in N. \text{grounding-of-cls } C)$*   
**unfolding** *grounding-of-cls-def* **by** *simp*  
**also have**  $\dots = (\bigcup C \in N. \{C\})$   
**using**  $\langle \text{is-ground-cls } N \rangle$   
**by** (*simp add: is-ground-cls-def grounding-of-cls-ground*)  
**also have**  $\dots = N$   
**by** *simp*  
**finally show** *?thesis* .  
**qed**

**lemma** (*in scl-fol-calculus*) *propagateI'*:  
 $C \models N \mid U \implies C = \text{add-mset } L C' \implies \text{is-ground-cls } (C \cdot \gamma) \implies$   
 $\forall K \in \# C \cdot \gamma. \text{atm-of } K \preceq_B \beta \implies$   
 $C_0 = \{\#K \in \# C'. K \cdot l \gamma \neq L \cdot l \gamma\} \implies C_1 = \{\#K \in \# C'. K \cdot l \gamma = L \cdot l$   
 $\gamma\} \implies$   
 $\text{SCL-FOL.trail-false-cls } \Gamma (C_0 \cdot \gamma) \implies \neg \text{SCL-FOL.trail-defined-lit } \Gamma (L \cdot l \gamma)$   
 $\implies$   
 $\text{is-ingu } \mu \{\text{atm-of ' set-mset (add-mset } L C_1)\} \implies$   
 $\Gamma' = \text{trail-propagate } \Gamma (L \cdot l \mu) (C_0 \cdot \mu) \gamma \implies$   
 $\text{propagate } N \beta (\Gamma, U, \text{None}) (\Gamma', U, \text{None})$   
**using** *propagateI* **by** *metis*

**lemma** (*in scl-fol-calculus*) *decideI'*:  
 $\text{is-ground-lit } (L \cdot l \gamma) \implies \neg \text{SCL-FOL.trail-defined-lit } \Gamma (L \cdot l \gamma) \implies \text{atm-of } L \cdot a$   
 $\gamma \preceq_B \beta \implies$   
 $\Gamma' = \text{trail-decide } \Gamma (L \cdot l \gamma) \implies$   
 $\text{decide } N \beta (\Gamma, U, \text{None}) (\Gamma', U, \text{None})$   
**by** (*auto intro!: decideI*)

**lemma** *ground-iff-vars-term-empty*:  $\text{ground } t \longleftrightarrow \text{vars-term } t = \{\}$   
**proof** (*rule iffI*)  
**show**  $\text{ground } t \implies \text{vars-term } t = \{\}$   
**by** (*rule ground-vars-term-empty*)  
**next**  
**show**  $\text{vars-term } t = \{\} \implies \text{ground } t$   
**by** (*induction t*) *simp-all*  
**qed**

**lemma** *is-ground-atm-eq-ground*[*iff*]: *is-ground-atm* = *ground*  
**proof** (*rule ext*)  
**fix** *t* :: ('*v*, '*f*) *Term.term*  
**show** *is-ground-atm t* = *ground t*  
**by** (*simp only: is-ground-atm-iff-vars-empty ground-iff-vars-term-empty*)  
**qed**

**definition** *lit-of-glit* :: '*f gterm literal* ⇒ ('*f*, '*v*) *term literal* **where**  
*lit-of-glit* = *map-literal term-of-gterm*

**definition** *glit-of-lit* **where**  
*glit-of-lit* = *map-literal gterm-of-term*

**definition** *cls-of-gcls* **where**  
*cls-of-gcls* = *image-mset lit-of-glit*

**definition** *gcls-of-cls* **where**  
*gcls-of-cls* = *image-mset glit-of-lit*

**lemma** *inj-lit-of-glit*: *inj lit-of-glit*  
**proof** (*rule injI*)  
**fix** *L K*  
**show** *lit-of-glit L* = *lit-of-glit K* ⇒ *L* = *K*  
**by** (*metis lit-of-glit-def literal.expand literal.map-disc-iff literal.map-sel term-of-gterm-inv*)  
**qed**

**lemma** *atm-of-lit-of-glit-conv*: *atm-of (lit-of-glit L)* = *term-of-gterm (atm-of L)*  
**by** (*cases L*) (*simp-all add: lit-of-glit-def*)

**lemma** *ground-atm-of-lit-of-glit*[*simp*]: *Term-Context.ground (atm-of (lit-of-glit L))*  
**by** (*simp add: atm-of-lit-of-glit-conv*)

**lemma** *is-ground-lit-lit-of-glit*[*simp*]: *is-ground-lit (lit-of-glit L)*  
**by** (*simp add: is-ground-lit-def atm-of-lit-of-glit-conv*)

**lemma** *is-ground-cls-cls-of-gcls*[*simp*]: *is-ground-cls (cls-of-gcls C)*  
**by** (*auto simp add: is-ground-cls-def cls-of-gcls-def*)

**lemma** *glit-of-lit-lit-of-glit*[*simp*]: *glit-of-lit (lit-of-glit L)* = *L*  
**by** (*simp add: glit-of-lit-def lit-of-glit-def literal.map-comp comp-def literal.map-ident*)

**lemma** *gcls-of-cls-cls-of-gcls*[*simp*]: *gcls-of-cls (cls-of-gcls L)* = *L*  
**by** (*simp add: gcls-of-cls-def cls-of-gcls-def multiset.map-comp comp-def multiset.map-ident*)

**lemma** *lit-of-glit-glit-of-lit-ident*[*simp*]: *is-ground-lit L* ⇒ *lit-of-glit (glit-of-lit L)*  
 = *L*  
**by** (*simp add: is-ground-lit-def lit-of-glit-def glit-of-lit-def literal.map-comp lit-*

*eral.expand*  
*literal.map-sel*)

**lemma** *cls-of-gcls-gcls-of-cls-ident*[simp]: *is-ground-cls*  $D \implies \text{cls-of-gcls } (\text{gcls-of-cls } D) = D$

**by** (*simp add: is-ground-cls-def cls-of-gcls-def gcls-of-cls-def*)

**lemma** *vars-lit-lit-of-glit*[simp]: *vars-lit* (*lit-of-glit*  $L$ ) =  $\{\}$   
**by** *simp*

**lemma** *vars-cls-cls-of-gcls*[simp]: *vars-cls* (*cls-of-gcls*  $C$ ) =  $\{\}$   
**by** (*metis is-ground-cls-cls-of-gcls is-ground-cls-iff-vars-empty*)

**definition** *atms-of-cls* :: 'a *clause*  $\Rightarrow$  'a *fset* **where**  
*atms-of-cls*  $C = \text{atm-of } |\uparrow| \text{ fset-mset } C$

**definition** *atms-of-clss* :: 'a *clause* *fset*  $\Rightarrow$  'a *fset* **where**  
*atms-of-clss*  $N = \text{ffUnion } (\text{atms-of-cls } |\uparrow| N)$

**lemma** *atms-of-clss-fempty*[simp]: *atms-of-clss*  $\{\|\}$  =  $\{\|\}$   
**unfolding** *atms-of-clss-def* **by** *simp*

**lemma** *atms-of-clss-finsert*[simp]:  
*atms-of-clss* (*finsert*  $C N$ ) = *atms-of-cls*  $C \cup | \text{atms-of-clss } N$   
**unfolding** *atms-of-clss-def* **by** *simp*

**definition** *lits-of-clss* :: 'a *clause* *fset*  $\Rightarrow$  'a *literal* *fset* **where**  
*lits-of-clss*  $N = \text{ffUnion } (\text{fset-mset } |\uparrow| N)$

**definition** *lit-occures-in-clss* **where**  
*lit-occures-in-clss*  $L N \longleftrightarrow \text{fBex } N (\lambda C. L \in \# C)$

**inductive** *constant-context* **for**  $R$  **where**  
 $R C \mathcal{D} \mathcal{D}' \implies \text{constant-context } R (C, \mathcal{D}) (C, \mathcal{D}')$

**lemma** *rtranclp-constant-context*:  $(R C)^{**} \mathcal{D} \mathcal{D}' \implies (\text{constant-context } R)^{**} (C, \mathcal{D}) (C, \mathcal{D}')$

**by** (*induction*  $\mathcal{D}'$  *rule: rtranclp-induct*) (*auto intro: constant-context.intros rtranclp.intros*)

**lemma** *tranclp-constant-context*:  $(R C)^{++} \mathcal{D} \mathcal{D}' \implies (\text{constant-context } R)^{++} (C, \mathcal{D}) (C, \mathcal{D}')$

**by** (*induction*  $\mathcal{D}'$  *rule: tranclp-induct*) (*auto intro: constant-context.intros tranclp.intros*)

**lemma** *right-unique-constant-context*:  
**assumes**  $R\text{-ru}$ :  $\bigwedge C. \text{right-unique } (R C)$   
**shows** *right-unique* (*constant-context*  $R$ )  
**proof** (*rule right-uniqueI*)

```

fix x y z
show constant-context R x y  $\implies$  constant-context R x z  $\implies$  y = z
  using R-ru[THEN right-uniqueD]
  by (elim constant-context.cases) (metis prod.inject)
qed

```

**lemma** safe-state-constant-context-if-invars:

```

fixes N s
assumes
   $\mathcal{R}$ -preserves- $\mathcal{I}$ :
     $\bigwedge N s s'. \mathcal{R} N s s' \implies \mathcal{I} N s \implies \mathcal{I} N s'$  and
  ex- $\mathcal{R}$ -if-not-final:
     $\bigwedge N s. \neg \mathcal{F} (N, s) \implies \mathcal{I} N s \implies \exists s'. \mathcal{R} N s s'$ 
assumes invars:  $\mathcal{I} N s$ 
shows safe-state (constant-context  $\mathcal{R}$ )  $\mathcal{F} (N, s)$ 
proof -
  {
    fix S'
    assume (constant-context  $\mathcal{R}$ )** (N, s) S' and stuck-state (constant-context  $\mathcal{R}$ )
    S'
    then obtain s' where S' = (N, s') and ( $\mathcal{R} N$ )** s s' and  $\mathcal{I} N s'$ 
    using invars
    proof (induction (N, s) arbitrary: N s rule: converse-rtranclp-induct)
      case base
      thus ?case by simp
    next
      case (step z)
      thus ?case
      by (metis (no-types, opaque-lifting) Pair-inject  $\mathcal{R}$ -preserves- $\mathcal{I}$  constant-context.cases
        converse-rtranclp-into-rtranclp)
    qed
    hence  $\neg \mathcal{F} (N, s') \implies \exists s''. \mathcal{R} N s' s''$ 
    using ex- $\mathcal{R}$ -if-not-final[of N s'] by argo
    hence  $\neg \mathcal{F} S' \implies \exists S''. \text{constant-context } \mathcal{R} S' S''$ 
    unfolding  $\langle S' = (N, s') \rangle$  using constant-context.intros by metis
    hence  $\mathcal{F} S'$ 
    by (metis  $\langle$ stuck-state (constant-context  $\mathcal{R}) S'\rangle$  stuck-state-def)
  }
  thus ?thesis
  by (metis (no-types) safe-state-def stuck-state-def)
qed

```

```

primrec trail-atms :: (- literal  $\times$  -) list  $\Rightarrow$  - fset where
  trail-atms [] = {} |
  trail-atms (Ln #  $\Gamma$ ) = finsert (atm-of (fst Ln)) (trail-atms  $\Gamma$ )

```

```

lemma fset-trail-atms: fset (trail-atms  $\Gamma$ ) = atm-of 'fst ' set  $\Gamma$ 
by (induction  $\Gamma$ ) simp-all

```

**lemma** *trail-defined-lit-iff-trail-defined-atm:*  
*trail-defined-lit*  $\Gamma$   $L \longleftrightarrow \text{atm-of } L \mid \in \mid \text{trail-atms } \Gamma$

**proof** (*induction*  $\Gamma$ )  
**case** *Nil*  
**show** *?case*  
**by** (*simp add: trail-defined-lit-def*)

**next**  
**case** (*Cons Ln*  $\Gamma$ )

**have** *trail-defined-lit* ( $Ln \# \Gamma$ )  $L \longleftrightarrow L = \text{fst } Ln \vee - L = \text{fst } Ln \vee \text{trail-defined-lit}$   
 $\Gamma$   $L$   
**unfolding** *trail-defined-lit-def* **by** *auto*

**also have**  $\dots \longleftrightarrow \text{atm-of } L = \text{atm-of } (\text{fst } Ln) \vee \text{trail-defined-lit } \Gamma$   $L$   
**by** (*cases L; cases fst Ln*) *simp-all*

**also have**  $\dots \longleftrightarrow \text{atm-of } L = \text{atm-of } (\text{fst } Ln) \vee \text{atm-of } L \mid \in \mid \text{trail-atms } \Gamma$   
**unfolding** *Cons.IH ..*

**also have**  $\dots \longleftrightarrow \text{atm-of } L \mid \in \mid \text{trail-atms } (Ln \# \Gamma)$   
**by** *simp*

**finally show** *?case .*

**qed**

**lemma** *trail-atms-subset-if-suffix:*  
**assumes** *suffix*  $\Gamma' \Gamma$   
**shows** *trail-atms*  $\Gamma' \mid \subseteq \mid \text{trail-atms } \Gamma$

**proof** –  
**obtain**  $\Gamma_0$  **where**  $\Gamma = \Gamma_0 @ \Gamma'$   
**using** *assms* **unfolding** *suffix-def* **by** *metis*

**show** *?thesis*  
**unfolding**  $\langle \Gamma = \Gamma_0 @ \Gamma' \rangle$   
**by** (*induction*  $\Gamma_0$ ) *auto*

**qed**

**lemma** *dom-model-eq-trail-interp:*  
**assumes**  
 $\forall A C. \mathcal{M} A = \text{Some } C \longleftrightarrow \text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C)$  **and**  
 $\forall Ln \in \text{set } \Gamma. \forall L. Ln = (L, \text{None}) \longrightarrow \text{is-neg } L$

**shows** *dom*  $\mathcal{M} = \text{trail-interp } \Gamma$

**proof** –  
**have** *dom*  $\mathcal{M} = \{A. \exists C. \mathcal{M} A = \text{Some } C\}$   
**unfolding** *dom-def* **by** *simp*  
**also have**  $\dots = \{A. \exists C. \text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C)\}$   
**using** *assms(1)* **by** *metis*  
**also have**  $\dots = \{A. \exists \text{opt}. \text{map-of } \Gamma (\text{Pos } A) = \text{Some } \text{opt}\}$   
**proof** (*rule Collect-cong*)



```

show  $\bigwedge A. (\exists C. \text{map-of } \Gamma (Pos A) = \text{Some } (Some C)) \longleftrightarrow (\exists \text{opt. map-of } \Gamma$ 
 $(Pos A) = \text{Some opt})$ 
using assms(2)
by (metis literal.disc(1) map-of-SomeD option.exhaust)
qed
also have  $\dots = \text{trail-interp } \Gamma$ 
proof (induction  $\Gamma$ )
case Nil
thus ?case
by (simp add: trail-interp-def)
next
case (Cons Ln  $\Gamma$ )

have  $\{A. \exists \text{opt. map-of } (Ln \# \Gamma) (Pos A) = \text{Some opt}\} =$ 
 $\{A. \exists \text{opt. map-of } [Ln] (Pos A) = \text{Some opt}\} \cup \{A. \exists \text{opt. map-of } \Gamma (Pos A)$ 
 $= \text{Some opt}\}$ 
by auto

also have  $\dots = \{A. Pos A = \text{fst } Ln\} \cup \text{trail-interp } \Gamma$ 
unfolding Cons.IH by simp

also have  $\dots = \text{trail-interp } [Ln] \cup \text{trail-interp } \Gamma$ 
by (cases fst Ln) (simp-all add: trail-interp-def)

also have  $\dots = \text{trail-interp } (Ln \# \Gamma)$ 
unfolding trail-interp-Cons[of Ln  $\Gamma$ ] ..

finally show ?case .
qed
finally show ?thesis .
qed

```

```

type-synonym 'f gliteral = 'f gterm literal
type-synonym 'f gclause = 'f gterm clause

```

```

locale simulation-SCLFOL-ground-ordered-resolution =
  renaming-apart renaming-vars
for renaming-vars :: 'v set  $\Rightarrow$  'v  $\Rightarrow$  'v +
fixes
  less-trm :: 'f gterm  $\Rightarrow$  'f gterm  $\Rightarrow$  bool (infix  $\prec_t$  50)
assumes
  transp-less-trm[simp]: transp  $(\prec_t)$  and
  asypm-less-trm[intro]: asypm  $(\prec_t)$  and
  wfP-less-trm[intro]: wfP  $(\prec_t)$  and
  totalp-less-trm[intro]: totalp  $(\prec_t)$  and

```

*finite-less-trm*:  $\bigwedge \beta. \text{finite } \{x. x \prec_t \beta\}$  **and**  
*less-trm-compatible-with-gctxt[simp]*:  $\bigwedge \text{cxt} t t'. t \prec_t t' \implies \text{cxt} \langle t \rangle_G \prec_t \text{cxt} \langle t' \rangle_G$   
**and**  
*less-trm-if-subterm[simp]*:  $\bigwedge t \text{cxt}. \text{cxt} \neq \square_G \implies t \prec_t \text{cxt} \langle t \rangle_G$

## 8 Ground ordered resolution for ground terms

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**sublocale** *ord-res*: *ground-ordered-resolution-calculus*  $(\prec_t) \lambda\cdot. \{\#\}$   
**by** *unfold-locales auto*

**sublocale** *linorder-trm*: *linorder*  $(\preceq_t) (\prec_t)$

**proof** *unfold-locales*

**show**  $\bigwedge x y. (x \prec_t y) = (x \preceq_t y \wedge \neg y \preceq_t x)$   
**by** (*metis asympD asymp-less-trm reflclp-iff*)

**next**

**show**  $\bigwedge x. x \preceq_t x$   
**by** *simp*

**next**

**show**  $\bigwedge x y z. x \preceq_t y \implies y \preceq_t z \implies x \preceq_t z$   
**by** (*meson transpE transp-less-trm transp-on-reflclp*)

**next**

**show**  $\bigwedge x y. x \preceq_t y \implies y \preceq_t x \implies x = y$   
**by** (*metis asympD asymp-less-trm reflclp-iff*)

**next**

**show**  $\bigwedge x y. x \preceq_t y \vee y \preceq_t x$   
**by** (*metis reflclp-iff totalpD totalp-less-trm*)

**qed**

**sublocale** *linorder-lit*: *linorder*  $(\preceq_l) (\prec_l)$

**proof** *unfold-locales*

**show**  $\bigwedge x y. (x \prec_l y) = (x \preceq_l y \wedge \neg y \preceq_l x)$   
**by** (*metis asympD ord-res.asymp-less-lit reflclp-iff*)

**next**

**show**  $\bigwedge x. x \preceq_l x$   
**by** *simp*

**next**

**show**  $\bigwedge x y z. x \preceq_l y \implies y \preceq_l z \implies x \preceq_l z$   
**by** (*meson transpE ord-res.transp-less-lit transp-on-reflclp*)

**next**

**show**  $\bigwedge x y. x \preceq_l y \implies y \preceq_l x \implies x = y$   
**by** (*metis asympD ord-res.asymp-less-lit reflclp-iff*)

**next**

**show**  $\bigwedge x y. x \preceq_l y \vee y \preceq_l x$   
**by** (*metis reflclp-iff totalpD ord-res.totalp-less-lit*)

**qed**

**sublocale** *linorder-clc*: *linorder*  $(\preceq_c) (\prec_c)$

```

proof unfold-locales
  show  $\bigwedge x y. (x \prec_c y) = (x \preceq_c y \wedge \neg y \preceq_c x)$ 
    by (metis asympD ord-res.asymp-less-cls reflclp-iff)
next
  show  $\bigwedge x. x \preceq_c x$ 
    by simp
next
  show  $\bigwedge x y z. x \preceq_c y \implies y \preceq_c z \implies x \preceq_c z$ 
    by (meson transpE ord-res.transp-less-cls transp-on-reflclp)
next
  show  $\bigwedge x y. x \preceq_c y \implies y \preceq_c x \implies x = y$ 
    by (metis asympD ord-res.asymp-less-cls reflclp-iff)
next
  show  $\bigwedge x y. x \preceq_c y \vee y \preceq_c x$ 
    by (metis reflclp-iff totalpD ord-res.totalp-less-cls)
qed

declare linorder-trm.is-least-in-fset-ffilterD[no-atp]
declare linorder-lit.is-least-in-fset-ffilterD[no-atp]
declare linorder-cls.is-least-in-fset-ffilterD[no-atp]

end

```

## 9 Common definitions and lemmas

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**abbreviation** *ord-res-Interp* **where**

*ord-res-Interp*  $N C \equiv \text{ord-res.interp } N C \cup \text{ord-res.production } N C$

**definition** *is-least-false-clause* **where**

*is-least-false-clause*  $N C \longleftrightarrow$

*linorder-cls.is-least-in-fset*  $\{|C| \in |N|. \neg \text{ord-res-Interp } (fset N) C \models C\} C$

**lemma** *is-least-false-clause-finsert-smaller-false-clause*:

**assumes**

*D-least*: *is-least-false-clause*  $N D$  **and**

$C \prec_c D$  **and**

*C-false*:  $\neg \text{ord-res-Interp } (fset (finsert C N)) C \models C$

**shows** *is-least-false-clause*  $(finsert C N) C$

**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*

**proof** (*intro conjI ballI impI*)

**show**  $C \in |finsert C N|$

**by** *simp*

**next**

**show**  $\neg \text{ord-res-Interp } (fset (finsert C N)) C \models C$

**using** *assms* **by** *metis*

**next**

**fix**  $y$

**assume**  $y \in | \text{finsert } C \ N$  **and**  $y \neq C$  **and**  $y\text{-false: } \neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C \ N)) \ y \models y$   
**hence**  $y \in | \ N$   
**by** *simp*

**have**  $\neg (y \prec_c C)$   
**proof** (*rule notI*)  
**assume**  $y \prec_c C$   
**hence**  $\text{ord-res-Interp } (\text{fset } (\text{finsert } C \ N)) \ y = \text{ord-res-Interp } (\text{fset } N) \ y$   
**using** *ord-res.Interp-insert-greater-clause* **by** *simp*

**hence**  $\neg \text{ord-res-Interp } (\text{fset } N) \ y \models y$   
**using** *y-false* **by** *argo*

**moreover have**  $y \prec_c D$   
**using**  $\langle y \prec_c C \rangle \langle C \prec_c D \rangle$  **by** *order*

**ultimately show** *False*  
**using** *D-least*  
**by** (*metis (mono-tags, lifting) \langle y \in | \ N \rangle linorder-cls.is-least-in-ffilter-iff linorder-cls.less-asym' is-least-false-clause-def*)

**qed**  
**thus**  $C \prec_c y$   
**using**  $\langle y \neq C \rangle$  **by** *order*

**qed**

**lemma** *is-least-false-clause-swap-swap-compatible-fsets:*  
**assumes**  $\{|x \in | \ N1. x \preceq_c C|\} = \{|x \in | \ N2. x \preceq_c C|\}$   
**shows** *is-least-false-clause*  $N1 \ C \longleftrightarrow \text{is-least-false-clause } N2 \ C$   
**proof** –

**have** *is-least-false-clause*  $N2 \ C$  **if**  
*subsets-agree: \{|x \in | \ N1. x \preceq\_c C|\} = \{|x \in | \ N2. x \preceq\_c C|\}* **and**  
*C-least: is-least-false-clause*  $N1 \ C$  **for**  $N1 \ N2 \ C$   
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*

**proof** (*intro conjI ballI impI*)  
**have**  $C \in | \ N1$   
**using** *C-least*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
**by** *argo*

**thus**  $C \in | \ N2$   
**using** *subsets-agree* **by** *auto*

**next**  
**have**  $\neg \text{ord-res-Interp } (\text{fset } N1) \ C \models C$   
**using** *C-least*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
**by** *argo*

**moreover have**  $\text{ord-res-Interp } (\text{fset } N1) \ C = \text{ord-res-Interp } (\text{fset } N2) \ C$   
**using** *subsets-agree* **by** (*auto intro!: ord-res.Interp-swap-clause-set*)  
**ultimately show**  $\neg \text{ord-res-Interp } (\text{fset } N2) \ C \models C$

by *argo*  
 next  
 fix  $y$  assume  $y \in N2$  and  $y \neq C$   
 show  $\neg \text{ord-res-Interp } (fset\ N2)\ y \Vdash y \implies C \prec_c y$   
 proof (*erule contrapos-np*)  
 assume  $\neg C \prec_c y$   
 hence  $y \preceq_c C$   
 by *order*  
 hence  $y \in N1$   
 using  $\langle y \in N2 \rangle$  using *subsets-agree* by *auto*  
 hence  $\neg \text{ord-res-Interp } (fset\ N1)\ y \Vdash y \longrightarrow C \prec_c y$   
 using  $\langle \text{is-least-false-clause } N1\ C \rangle \langle y \neq C \rangle$   
 unfolding *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
 by *metis*  
 moreover have  $\text{ord-res-Interp } (fset\ N1)\ y = \text{ord-res-Interp } (fset\ N2)\ y$   
 proof (*rule ord-res.Interp-swap-clause-set*)  
 show  $\{D. D \in N1 \wedge (\prec_c)^{==} D\ y\} = \{D. D \in N2 \wedge (\prec_c)^{==} D\ y\}$   
 using *subsets-agree*  $\langle y \preceq_c C \rangle$  by *fastforce*  
 qed  
 ultimately show  $\text{ord-res-Interp } (fset\ N2)\ y \Vdash y$   
 using  $\langle y \preceq_c C \rangle$  by *auto*  
 qed  
 qed  
 thus *?thesis*  
 using *assms* by *metis*  
 qed

**lemma** *Uniq-is-least-false-clause*:  $\exists \leq_1 C. \text{is-least-false-clause } N\ C$   
**proof** (*rule Uniq-I*)  
 show  $\bigwedge x\ y\ z. \text{is-least-false-clause } x\ y \implies \text{is-least-false-clause } x\ z \implies y = z$   
 unfolding *is-least-false-clause-def*  
 by (*meson Uniq-D linorder-cls.Uniq-is-least-in-fset*)  
 qed

**lemma** *mempty-lesseq-cl[simp]*:  $\{\#\} \preceq_c C$  for  $C$   
 by (*cases C*) (*simp-all add: strict-subset-implies-multp*)

**lemma** *is-least-false-clause-mempty*:  $\{\#\} \in N \implies \text{is-least-false-clause } N\ \{\#\}$   
 using *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff mempty-lesseq-cl*  
 by *fastforce*

**lemma** *production-union-unproductive-strong*:  
**assumes**  
*fin*: *finite N1 finite N2* and  
*N2-unproductive*:  $\forall x \in N2 - N1. \text{ord-res.production } (N1 \cup N2)\ x = \{\}$  and  
*C-in*:  $C \in N1$   
**shows**  $\text{ord-res.production } (N1 \cup N2)\ C = \text{ord-res.production } N1\ C$   
**using** *ord-res.wfP-less-cl C-in*  
**proof** (*induction C rule: wfp-induct-rule*)

**case** (*less C*)  
**hence** *C-in-iff*:  $C \in N1 \cup N2 \iff C \in N1$   
**by** *simp*

**have** *interp-eq*:  $\text{ord-res.interp } (N1 \cup N2) C = \text{ord-res.interp } N1 C$   
**proof** –  
**have**  $\text{ord-res.interp } (N1 \cup N2) C = \bigcup (\text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N1 \cup N2. D \prec_c C\}$ )  
**unfolding** *ord-res.interp-def ..*  
**also have**  $\dots = \bigcup (\text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N1. D \prec_c C\} \cup \text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N2 - N1. D \prec_c C\})$   
**by** *auto*  
**also have**  $\dots = \bigcup (\text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N1. D \prec_c C\})$   
**using** *N2-unproductive by simp*  
**also have**  $\dots = \bigcup (\text{ord-res.production } N1 \text{ ‘ } \{D \in N1. D \prec_c C\})$   
**using** *less.IH by simp*  
**also have**  $\dots = \text{ord-res.interp } N1 C$   
**unfolding** *ord-res.interp-def ..*  
**finally show**  $\text{ord-res.interp } (N1 \cup N2) C = \text{ord-res.interp } N1 C$  .  
**qed**

**show** *?case*  
**unfolding** *ord-res.production-unfold C-in-iff interp-eq by argo*  
**qed**

**lemma** *production-union-unproductive*:

**assumes**

*fin*: *finite N1 finite N2 and*

*N2-unproductive*:  $\forall x \in N2. \text{ord-res.production } (N1 \cup N2) x = \{\}$  **and**

*C-in*:  $C \in N1$

**shows**  $\text{ord-res.production } (N1 \cup N2) C = \text{ord-res.production } N1 C$

**using** *production-union-unproductive-strong assms by simp*

**lemma** *interp-union-unproductive*:

**assumes**

*fin*: *finite N1 finite N2 and*

*N2-unproductive*:  $\forall x \in N2. \text{ord-res.production } (N1 \cup N2) x = \{\}$

**shows**  $\text{ord-res.interp } (N1 \cup N2) = \text{ord-res.interp } N1$

**proof** (*rule ext*)

**fix** *C*

**have**  $\text{ord-res.interp } (N1 \cup N2) C = \bigcup (\text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N1 \cup N2. D \prec_c C\})$

**unfolding** *ord-res.interp-def ..*

**also have**  $\dots = \bigcup (\text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N1. D \prec_c C\} \cup \text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N2. D \prec_c C\})$

**by** *auto*

**also have**  $\dots = \bigcup (\text{ord-res.production } (N1 \cup N2) \text{ ‘ } \{D \in N1. D \prec_c C\})$

**using** *N2-unproductive by simp*

**also have**  $\dots = \bigcup (\text{ord-res.production } N1 \text{ ‘ } \{D \in N1. D \prec_c C\})$

**using** *production-union-unproductive*[*OF fin N2-unproductive*] **by** *simp*  
**also have**  $\dots = \text{ord-res.interp } N1 \ C$   
**unfolding** *ord-res.interp-def ..*  
**finally show**  $\text{ord-res.interp } (N1 \cup N2) \ C = \text{ord-res.interp } N1 \ C$  .  
**qed**

**lemma** *Interp-union-unproductive:*

**assumes**

*fin: finite N1 finite N2 and*

*N2-unproductive:  $\forall x \in N2. \text{ord-res.production } (N1 \cup N2) \ x = \{\}$*

**shows**  $\text{ord-res-Interp } (N1 \cup N2) \ C = \text{ord-res-Interp } N1 \ C$

**unfolding** *interp-union-unproductive*[*OF assms*]

**using** *production-union-unproductive*[*OF assms*]

**using** *N2-unproductive*[*rule-format*]

**by** (*metis (no-types, lifting) Un-iff empty-Collect-eq ord-res.production-unfold*)

**lemma** *Interp-insert-unproductive:*

**assumes**

*fin: finite N1 and*

*x-unproductive:  $\text{ord-res.production } (\text{insert } x \ N1) \ x = \{\}$*

**shows**  $\text{ord-res-Interp } (\text{insert } x \ N1) \ C = \text{ord-res-Interp } N1 \ C$

**using** *assms Interp-union-unproductive*

**by** (*metis Un-commute finite.emptyI finite.insertI insert-is-Un singletonD*)

**lemma** *extended-partial-model-entails-iff-partial-model-entails:*

**assumes**

*fin: finite N finite N' and*

*irrelevant:  $\forall D \in N'. \exists E \in N. E \subset\# D \wedge \text{set-mset } D = \text{set-mset } E$  and*

*C-in:  $C \in N$*

**shows**  $\text{ord-res-Interp } (N \cup N') \ C \models C \longleftrightarrow \text{ord-res-Interp } N \ C \models C$

**using** *ord-res.interp-add-irrelevant-clauses-to-set*[*OF fin C-in irrelevant*]

**using** *ord-res.production-add-irrelevant-clauses-to-set*[*OF fin C-in irrelevant*]

**by** *metis*

**lemma** *nex-strictly-maximal-pos-lit-if-factorizable:*

**assumes** *ord-res.ground-factoring C C'*

**shows**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C$

**by** (*metis Uniq-D add-mset-remove-trivial assms linorder-lit. Uniq-is-maximal-in-mset  
linorder-lit.dual-order.order-iff-strict linorder-lit.is-greatest-in-mset-iff  
linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset linorder-lit.not-less  
ord-res.ground-factoring.cases union-single-eq-member*)

**lemma** *unproductive-if-nex-strictly-maximal-pos-lit:*

**assumes**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C$

**shows**  $\text{ord-res.production } N \ C = \{\}$

**using** *assms by (simp add: ord-res.production-unfold)*

**lemma** *ball-unproductive-if-nex-strictly-maximal-pos-lit:*

**assumes**  $\forall C \in N'. \nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C$

**shows**  $\forall C \in N'. \text{ord-res.production } (N \cup N') C = \{\}$   
**using** *assms unproductive-if-nex-strictly-maximal-pos-lit* **by** *metis*

**lemma** *is-least-false-clause-finsert-cancel:*

**assumes**

*C-unproductive:*  $\text{ord-res.production } (\text{fset } (\text{finsert } C N)) C = \{\}$  **and**

*C-entailed-by-smaller:*  $\exists D \mid \in \mid N. D \prec_c C \wedge \{D\} \models_e \{C\}$

**shows**  $\text{is-least-false-clause } (\text{finsert } C N) = \text{is-least-false-clause } N$

**proof** (*intro ext iffI*)

**fix**  $E$

**assume**  $E\text{-least: is-least-false-clause } (\text{finsert } C N) E$

**hence**

*E-in:*  $E \mid \in \mid \text{finsert } C N$  **and**

*E-false:*  $\neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C N)) E \models E$  **and**

$\models y \longrightarrow E \prec_c y$   
*E-least:*  $(\forall y \mid \in \mid \text{finsert } C N. y \neq E \longrightarrow \neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C N)) y$

**unfolding** *atomize-conj is-least-false-clause-def linorder-cls.is-least-in-filter-iff*  
**by** *metis*

**obtain**  $D$  **where**

$D \mid \in \mid N$  **and**  $D \prec_c C$  **and**  $\{D\} \models_e \{C\}$

**using** *C-entailed-by-smaller* **by** *metis*

**show**  $\text{is-least-false-clause } N E$

**proof** (*cases C = E*)

**case** *True*

**have**  $E \prec_c D$

**proof** (*rule E-least[rule-format]*)

**show**  $D \mid \in \mid \text{finsert } C N$

**using**  $\langle D \mid \in \mid N \rangle$  **by** *simp*

**next**

**show**  $D \neq E$

**using**  $\langle D \prec_c C \rangle \langle C = E \rangle$  **by** *order*

**next**

**show**  $\neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C N)) D \models D$

**using** *E-false*

**proof** (*rule contrapos-nn*)

**assume**  $\text{ord-res-Interp } (\text{fset } (\text{finsert } C N)) D \models D$

**thus**  $\text{ord-res-Interp } (\text{fset } (\text{finsert } C N)) E \models E$

**using**  $\langle D \prec_c C \rangle \langle C = E \rangle \langle \{D\} \models_e \{C\} \rangle$  *ord-res.entailed-clause-stays-entailed*

**by** *auto*

**qed**

**qed**

**hence** *False*

**using**  $\langle D \prec_c C \rangle \langle C = E \rangle$  **by** *order*

**thus** *?thesis ..*

**next**

**case** *False*



```

show ?thesis
  unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
  show  $E \in N$ 
    using E-in  $\langle C \neq E \rangle$  by simp
next
  have ord-res-Interp (fset (finsert C N)) E = ord-res-Interp (fset N) E
    using C-unproductive Interp-insert-unproductive by simp
  thus  $\neg$  ord-res-Interp (fset N) E  $\models$  E
    using E-false by argo
next
  show  $\bigwedge y. y \in N \implies y \neq E \implies \neg$  ord-res-Interp (fset N) y  $\models$  y  $\implies$  E  $\prec_c$ 
    y
    using E-least C-unproductive Interp-insert-unproductive by auto
  qed
qed
next
fix E
assume is-least-false-clause N E
hence
  E-in:  $E \in N$  and
  E-false:  $\neg$  ord-res-Interp (fset N) E  $\models$  E and
  E-least:  $(\forall y \in N. y \neq E \implies \neg$  ord-res-Interp (fset N) y  $\models$  y  $\implies$  E  $\prec_c$  y)
  unfolding atomize-conj is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
  by metis

show is-least-false-clause (finsert C N) E
  unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
  show  $E \in$  finsert C N
    using E-in by simp
next
  show  $\neg$  ord-res-Interp (fset (finsert C N)) E  $\models$  E
    using E-least E-false C-unproductive Interp-insert-unproductive by simp
next
  fix y
  assume  $y \in$  finsert C N and  $y \neq E$  and  $\neg$  ord-res-Interp (fset (finsert C N))
    y  $\models$  y
  show E  $\prec_c$  y
  proof (cases y = C)
    case True
      thus ?thesis
      using E-least  $\langle \neg$  ord-res-Interp (fset (finsert C N)) y  $\models$  y  $\rangle$ 
    by (metis (no-types, lifting) C-entailed-by-smaller C-unproductive Interp-insert-unproductive
      finite-fset fset-simps(2) linorder-cls.dual-order.strict-trans
      ord-res.entailed-clause-stays-entailed true-clss-singleton)
next
  case False
  thus ?thesis

```

**using**  $E\text{-least } \langle y \mid \in \mid \text{finsert } C \ N \rangle \langle y \neq E \rangle \langle \neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C \ N)) \ y \models y \rangle$   
**using**  $C\text{-unproductive Interp-insert-unproductive by auto}$   
**qed**  
**qed**  
**qed**

**lemma**  $is\text{-least-false-clause-funion-cancel-right-strong}$ :

**assumes**  
 $\forall C \mid \in \mid N2 - N1. \forall U. \text{ord-res.production } U \ C = \{\}$  **and**  
 $\forall C \mid \in \mid N2 - N1. \exists D \mid \in \mid N1. D \prec_c C \wedge \{D\} \models_e \{C\}$   
**shows**  $is\text{-least-false-clause } (N1 \mid \cup \mid N2) = is\text{-least-false-clause } N1$   
**using**  $assms$   
**proof** ( $induction \ N2$ )  
**case**  $empty$   
**thus**  $?case$   
**by**  $simp$   
**next**  
**case** ( $insert \ C \ N2$ )

**have**  $IH: is\text{-least-false-clause } (N1 \mid \cup \mid N2) = is\text{-least-false-clause } N1$

**proof** ( $rule \ insert.IH$ )

**show**  $\forall C \mid \in \mid N2 \mid - \mid N1. \forall U. \text{ord-res.production } U \ C = \{\}$

**using**  $insert.prem1$  **by**  $auto$

**next**

**show**  $\forall C \mid \in \mid N2 \mid - \mid N1. \exists D \mid \in \mid N1. D \prec_c C \wedge \{D\} \models_e \{C\}$

**using**  $insert.prem2$  **by**  $auto$

**qed**

**show**  $?case$

**proof** ( $cases \ C \ \mid \in \mid \ N1$ )

**case**  $True$

**hence**  $is\text{-least-false-clause } (N1 \mid \cup \mid \text{finsert } C \ N2) = is\text{-least-false-clause } (N1 \mid \cup \mid N2)$

**by** ( $simp \ add: \ \text{finsert-absorb}$ )

**also have**  $\dots = is\text{-least-false-clause } N1$

**using**  $IH$  .

**finally show**  $?thesis$  .

**next**

**case**  $False$

**then show**  $?thesis$

**using**  $is\text{-least-false-clause-finsert-cancel } IH$

**by** ( $metis \ \text{finsertCI} \ \text{fminusI} \ \text{funionI1} \ \text{funion-finsert-right} \ \text{insert.prem1} \ \text{insert.prem2}$ )

**qed**

**qed**

**lemma**  $is\text{-least-false-clause-funion-cancel-right}$ :

**assumes**

$\forall C \mid \in \mid N2. \forall U. \text{ord-res.production } U C = \{\}$  **and**  
 $\forall C \mid \in \mid N2. \exists D \mid \in \mid N1. D \prec_c C \wedge \{D\} \Vdash_e \{C\}$   
**shows** *is-least-false-clause*  $(N1 \mid \cup \mid N2) = \text{is-least-false-clause } N1$   
**using** *assms is-least-false-clause-funion-cancel-right-strong* **by** *simp*

**definition** *ex-false-clause* **where**

$\text{ex-false-clause } N = (\exists C \in N. \neg \text{ord-res.interp } N C \cup \text{ord-res.production } N C$   
 $\Vdash C)$

**lemma** *obtains-least-false-clause-if-ex-false-clause*:

**assumes** *ex-false-clause*  $(\text{fset } N)$

**obtains**  $C$  **where** *is-least-false-clause*  $N C$

**using** *assms*

**by**  $(\text{metis } (\text{mono-tags, lifting}) \text{bot-fset.rep-eq emptyE ex-false-clause-def ffilter-member-filter})$

$\text{linorder-cls.ex-least-in-fset is-least-false-clause-def})$

**lemma** *ex-false-clause-if-least-false-clause*:

**assumes** *is-least-false-clause*  $N C$

**shows** *ex-false-clause*  $(\text{fset } N)$

**using** *assms*

**by**  $(\text{metis } (\text{no-types, lifting}) \text{ex-false-clause-def is-least-false-clause-def})$

$\text{linorder-cls.is-least-in-fset-ffilterD(1) linorder-cls.is-least-in-fset-ffilterD(2)})$

**lemma** *ex-false-clause-iff*: *ex-false-clause*  $(\text{fset } N) \longleftrightarrow (\exists C. \text{is-least-false-clause } N C)$

**using** *obtains-least-false-clause-if-ex-false-clause ex-false-clause-if-least-false-clause*  
**by** *metis*

**definition** *ord-res-model* **where**

$\text{ord-res-model } N = (\bigcup D \in N. \text{ord-res.production } N D)$

**lemma** *ord-res-model-eq-interp-union-production-of-greatest-clause*:

**assumes** *C-greatest*:  $\text{linorder-cls.is-greatest-in-set } N C$

**shows**  $\text{ord-res.model } N = \text{ord-res.interp } N C \cup \text{ord-res.production } N C$

**proof** –

**have**  $\text{ord-res.model } N = (\bigcup D \in N. \text{ord-res.production } N D)$

**unfolding** *ord-res-model-def* ..

**also have**  $\dots = (\bigcup D \in \{D \in N. D \preceq_c C\}. \text{ord-res.production } N D)$

**using** *C-greatest linorder-cls.is-greatest-in-set-iff* **by** *auto*

**also have**  $\dots = (\bigcup D \in \{D \in N. D \prec_c C\} \cup \{C\}. \text{ord-res.production } N D)$

**using** *C-greatest linorder-cls.is-greatest-in-set-iff* **by** *auto*

**also have**  $\dots = (\bigcup D \in \{D \in N. D \prec_c C\}. \text{ord-res.production } N D) \cup \text{ord-res.production } N C$

**by** *auto*

**also have**  $\dots = \text{ord-res.interp } N C \cup \text{ord-res.production } N C$

**unfolding** *ord-res.interp-def* ..

**finally show** *?thesis* .

**qed**

**lemma** *ord-res-model-entails-clauses-if-nex-false-clause*:  
**assumes** *finite N and  $N \neq \{\}$  and  $\neg$  ex-false-clause N*  
**shows** *ord-res-model N  $\models_s$  N*  
**unfolding** *true-clss-def*  
**proof** (*intro ballI*)  
**from**  $\langle \neg$  *ex-false-clause N*  $\rangle$  **have** *ball-true*:  
 $\forall C \in N. \text{ord-res.interp } N \ C \cup \text{ord-res.production } N \ C \models C$   
**by** (*simp add: ex-false-clause-def*)  
  
**from**  $\langle$  *finite N*  $\rangle$   $\langle$   *$N \neq \{\}$*   $\rangle$  **obtain** *D where*  
*D-greatest: linorder-cls.is-greatest-in-set N D*  
**using** *linorder-cls.ex-greatest-in-set* **by** *metis*  
  
**fix** *C* **assume**  $C \in N$   
**hence** *ord-res.interp N C  $\cup$  ord-res.production N C  $\models$  C*  
**using** *ball-true by metis*  
  
**moreover** **have**  $C \prec_c D$   
**using**  $\langle C \in N \rangle$  *D-greatest[unfolded linorder-cls.is-greatest-in-set-iff]* **by** *auto*  
  
**ultimately** **have** *ord-res.interp N D  $\cup$  ord-res.production N D  $\models$  C*  
**using** *ord-res.entailed-clause-stays-entailed* **by** *auto*  
  
**thus** *ord-res-model N  $\models$  C*  
**using** *ord-res-model-eq-interp-union-production-of-greatest-clause[OF D-greatest]*  
**by** *argo*  
**qed**

**lemma** *pos-lit-not-greatest-if-maximal-in-least-false-clause*:  
**assumes**  
*C-least: linorder-cls.is-least-in-fset  $\{|C| \in| N. \neg \text{ord-res-Interp (fset N) C} \models C\}$  C and*  
*C-max-lit: ord-res.is-maximal-lit (Pos A) C*  
**shows**  $\neg \text{ord-res.is-strictly-maximal-lit (Pos A) C}$   
**proof** –  
**from** *C-max-lit* **obtain** *C'* **where** *C-def: C = add-mset (Pos A) C'*  
**by** (*meson linorder-lit.is-maximal-in-mset-iff mset-add*)  
  
**from** *C-least* **have**  
*C-in: C  $| \in|$  N and*  
*C-false:  $\neg \text{ord-res-Interp (fset N) C} \models C$  and*  
*C-lt:  $\forall y | \in| N. y \neq C \longrightarrow \neg \text{ord-res-Interp (fset N) y} \models y \longrightarrow C \prec_c y$*   
**unfolding** *linorder-cls.is-least-in-filter-iff* **by** *auto*  
  
**have**  $\exists A. A \in \text{ord-res.production (fset N) C}$   
**proof** (*rule notI, elim exE*)  
**fix** *A* **assume**  $A \in \text{ord-res.production (fset N) C}$   
**have**  $\text{Pos } A \in \# C$

**using** *A-in* **by** (*auto elim: ord-res.mem-productionE*)  
**moreover have**  $A \in \text{ord-res-Interp } (fset\ N)\ C$   
**using** *A-in C-in* **by** *blast*  
**ultimately show** *False*  
**using** *C-false* **by** *auto*  
**qed**  
**hence** *C-unproductive: ord-res.production (fset N) C = {}*  
**using** *ord-res.production-eq-empty-or-singleton[of fset N C]* **by** *simp*  
  
**have**  $\{D \in fset\ N. D \preceq_c C\} = \{D \in fset\ N. D \prec_c C\} \cup \{D \in fset\ N. D = C\}$   
**by** *fastforce*  
**with** *C-unproductive* **have**  
 $\text{ord-res-Interp } (fset\ N)\ C = \text{ord-res.interp } (fset\ N)\ C$   
**by** *simp*  
**with** *C-false* **have** *C-false'*:  $\neg \text{ord-res.interp } (fset\ N)\ C \models C$   
**by** *simp*  
  
**from** *C-unproductive* **have**  $A \notin \text{ord-res.production } (fset\ N)\ C$   
**by** *simp*  
**thus** *?thesis*  
**proof** (*rule contrapos-nn*)  
**assume** *ord-res.is-strictly-maximal-lit (Pos A) C*  
**then show**  $A \in \text{ord-res.production } (fset\ N)\ C$   
**unfolding** *ord-res.production-unfold[of fset N C] mem-Collect-eq*  
**using** *C-in C-def C-false'*  
**by** *metis*  
**qed**  
**qed**  
  
**lemma** *ex-ground-factoringI*:  
**assumes**  
 $C\text{-max-lit: ord-res.is-maximal-lit } (Pos\ A)\ C$  **and**  
 $C\text{-not-max-lit: } \neg \text{ord-res.is-strictly-maximal-lit } (Pos\ A)\ C$   
**shows**  $\exists C'. \text{ord-res.ground-factoring } C\ C'$   
**proof** –  
**from** *C-max-lit C-not-max-lit* **have**  $\text{count } C\ (Pos\ A) \geq 2$   
**using** *linorder-lit.count-ge-2-if-maximal-in-mset-and-not-greatest-in-mset* **by**  
*metis*  
**then obtain** *C'* **where** *C-def*:  $C = \text{add-mset } (Pos\ A)\ (\text{add-mset } (Pos\ A)\ C')$   
**by** (*metis two-le-countE*)  
  
**show** *?thesis*  
**proof** (*rule exI*)  
**show** *ord-res.ground-factoring C (add-mset (Pos A) C')*  
**using** *ord-res.ground-factoringI[of C A C']*  
**using** *C-def C-max-lit* **by** *metis*  
**qed**  
**qed**

**lemma** *true-cls-if-true-lit-in*:  $L \in\# C \implies I \models_l L \implies I \models C$   
**by** *auto*

**lemma** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*:  
**assumes**

*C-least-false*: *is-least-false-clause*  $N C$  **and**

*Neg-A-max*: *ord-res.is-maximal-lit* ( $Neg A$ )  $C$

**shows** *fBex*  $N (\lambda D. D \prec_c C \wedge \text{ord-res.is-strictly-maximal-lit } (Pos A) D \wedge$   
*ord-res.production* (*fset*  $N$ )  $D = \{A\}$ )

**proof** –

**from** *C-least-false* **have**

*C-in*:  $C \in N$  **and**

*C-false*:  $\neg \text{ord-res-Interp } (\text{fset } N) C \models C$  **and**

*C-min*:  $\forall y \in N. y \neq C \longrightarrow \neg \text{ord-res-Interp } (\text{fset } N) y \models y \longrightarrow C \prec_c y$

**unfolding** *atomize-conj is-least-false-clause-def*

**unfolding** *linorder-cls.is-least-in-filter-iff*

**by** *argo*

**have**  $\exists A. A \in \text{ord-res.production } (\text{fset } N) C$

**proof** (*rule notI, elim exE*)

**fix**  $A$  **assume** *A-in*:  $A \in \text{ord-res.production } (\text{fset } N) C$

**have**  $Pos A \in\# C$

**using** *A-in* **by** (*auto elim: ord-res.mem-productionE*)

**moreover** **have**  $A \in \text{ord-res-Interp } (\text{fset } N) C$

**using** *A-in C-in* **by** *blast*

**ultimately** **show** *False*

**using** *C-false* **by** *auto*

**qed**

**hence** *C-unproductive*:  $\text{ord-res.production } (\text{fset } N) C = \{\}$

**using** *ord-res.production-eq-empty-or-singleton*[*of fset N C*] **by** *simp*

**from** *Neg-A-max* **have**  $Neg A \in\# C$

**by** (*simp add: linorder-lit.is-maximal-in-mset-iff*)

**from** *C-false* **have**  $\neg \text{ord-res-Interp } (\text{fset } N) C \models_l Neg A$

**using** *true-cls-if-true-lit-in*[*OF <Neg A ∈# C>*]

**by** *meson*

**hence**  $A \in \text{ord-res-Interp } (\text{fset } N) C$

**by** *simp*

**with** *C-unproductive* **have**  $A \in \text{ord-res.interp } (\text{fset } N) C$

**by** *blast*

**then** **obtain**  $D$  **where**

*D-in*:  $D \in N$  **and** *D-lt-C*:  $D \prec_c C$  **and** *D-productive*:  $A \in \text{ord-res.production } (\text{fset } N) D$

**unfolding** *ord-res.interp-def* **by** *auto*

**from** *D-productive* **have**  $\text{ord-res.is-strictly-maximal-lit } (Pos A) D$

**using** *ord-res.mem-productionE* **by** *metis*

**moreover have** *ord-res.production* (*fset N*)  $D = \{A\}$   
**using** *D-productive ord-res.production-eq-empty-or-singleton* **by** *fastforce*

**ultimately show** *?thesis*  
**using** *D-in D-lt-C* **by** *metis*

**qed**

**lemma** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit'*:

**assumes**

*C-least-false: is-least-false-clause N C* **and**

*L-max: ord-res.is-maximal-lit L C* **and**

*L-neg: is-neg L*

**shows**  $fBex\ N\ (\lambda D. D \prec_c C \wedge ord-res.is-strictly-maximal-lit\ (-\ L)\ D \wedge$   
*ord-res.production* (*fset N*)  $D = \{atn-of\ L\}$ )

**using** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit'* [*OF C-least-false*]

**by** (*simp add: L-max L-neg uminus-literal-def*)

**lemma** *ex-ground-resolutionI*:

**assumes**

*C-max-lit: ord-res.is-maximal-lit (Neg A) C* **and**

*D-lt: D  $\prec_c$  C* **and**

*D-max-lit: ord-res.is-strictly-maximal-lit (Pos A) D*

**shows**  $\exists CD. ord-res.ground-resolution\ C\ D\ CD$

**proof** –

**from** *C-max-lit* **obtain**  $C'$  **where** *C-def: C = add-mset (Neg A) C'*  
**by** (*meson linorder-lit.is-maximal-in-mset-iff mset-add*)

**from** *D-max-lit* **obtain**  $D'$  **where** *D-def: D = add-mset (Pos A) D'*  
**by** (*meson linorder-lit.is-greatest-in-mset-iff mset-add*)

**show** *?thesis*

**proof** (*rule exI*)

**show** *ord-res.ground-resolution C D (C' + D')*

**using** *ord-res.ground-resolutionI* [*of C A C' D D'*]

**using** *C-def D-def D-lt C-max-lit D-max-lit* **by** *metis*

**qed**

**qed**

**lemma**

**fixes**  $N\ N'$

**assumes**

*fin: finite N finite N'* **and**

*irrelevant:  $\forall D \in N'. \exists E \in N. E \subset\# D \wedge set-mset\ D = set-mset\ E$*  **and**

*C-in: C  $\in$  N* **and**

*C-not-entailed:  $\neg ord-res.interp\ N\ C \cup ord-res.production\ N\ C \models C$*

**shows**  $\neg ord-res.interp\ (N \cup N')\ C \cup ord-res.production\ (N \cup N')\ C \models C$

**using** *C-not-entailed*

**proof** (*rule contrapos-nn*)

**assume**  $ord-res.interp (N \cup N') C \cup ord-res.production (N \cup N') C \models C$   
**then show**  $ord-res.interp N C \cup ord-res.production N C \models C$   
**using**  $ord-res.interp-add-irrelevant-clauses-to-set[OF\ fin\ C-in\ irrelevant]$   
**using**  $ord-res.production-add-irrelevant-clauses-to-set[OF\ fin\ C-in\ irrelevant]$   
**by** *metis*  
**qed**

**lemma** *trail-consistent-if-sorted-wrt-atoms*:  
**assumes**  $sorted-wrt (\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$   
**shows** *trail-consistent*  $\Gamma$   
**using** *assms*  
**proof** (*induction*  $\Gamma$ )  
**case** *Nil*  
**show** *?case*  
**by** *simp*  
**next**  
**case** (*Cons*  $Ln \Gamma$ )  
  
**obtain**  $L\ opt$  **where**  
 $Ln = (L, opt)$   
**by** *fastforce*

**show** *?case*  
**unfolding**  $\langle Ln = (L, opt) \rangle$   
**proof** (*rule* *trail-consistent.Cons*)  
**have**  $\forall x \in set \Gamma. atm-of (fst x) \prec_t atm-of (fst Ln)$   
**using** *Cons.prem* **by** *simp*

**hence**  $\forall x \in set \Gamma. atm-of (fst x) \neq atm-of L$   
**unfolding**  $\langle Ln = (L, opt) \rangle$  **by** *fastforce*

**thus**  $\neg trail-defined-lit \Gamma L$   
**unfolding** *trail-defined-lit-def* **by** *fastforce*

**next**  
**show** *trail-consistent*  $\Gamma$   
**using** *Cons* **by** *simp*

**qed**  
**qed**

**lemma** *mono-atms-lt*: *monotone-on* ( $set \Gamma$ )  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) (\lambda x y. y \leq x)$   
 $(\lambda x. atm-of K \preceq_t atm-of (fst x))$  **for**  $K$   
**proof** (*intro* *monotone-onI*, *unfold* *le-bool-def*, *intro* *impI*)  
**fix**  $x y$   
**assume**  $atm-of (fst y) \prec_t atm-of (fst x)$  **and**  $atm-of K \preceq_t atm-of (fst y)$   
**thus**  $atm-of K \preceq_t atm-of (fst x)$   
**by** *order*  
**qed**



**lemma** *in-trail-atms-dropWhileI*:

**assumes**  
*sorted-wrt*  $R \Gamma$  **and**  
*monotone-on* (*set*  $\Gamma$ )  $R (\geq) (\lambda x. P (atm-of (fst x)))$  **and**  
 $\neg P A$  **and**  
 $A \in trail-atms \Gamma$

**shows**  $A \in trail-atms (dropWhile (\lambda Ln. P (atm-of (fst Ln)))) \Gamma$

**using** *assms(1,2,4)*

**proof** (*induction*  $\Gamma$ )

**case** *Nil*

**thus** *?case*  
**by** *simp*

**next**

**case** (*Cons*  $Ln \Gamma$ )

**show** *?case*

**proof** (*cases*  $P (atm-of (fst Ln))$ )

**case** *True*

**have**  $A \in trail-atms (dropWhile (\lambda Ln. P (atm-of (fst Ln)))) \Gamma$

**proof** (*rule* *Cons.IH*)

**show** *sorted-wrt*  $R \Gamma$   
**using** *Cons.prem(1)* **by** *simp*

**next**

**show** *monotone-on* (*set*  $\Gamma$ )  $R (\lambda x y. y \leq x) (\lambda x. P (atm-of (fst x)))$   
**using** *Cons.prem(2)* **by** (*meson monotone-on-subset set-subset-Cons*)

**next**

**have**  $\neg P A$   
**using** *assms* **by** *metis*

**hence**  $A \neq atm-of (fst Ln)$   
**using** *True* **by** *metis*

**moreover** **have**  $A \in trail-atms (Ln \# \Gamma)$   
**using** *Cons.prem(3)* **by** *metis*

**ultimately show**  $A \in trail-atms \Gamma$   
**by** (*simp add: trail-defined-lit-def*)

**qed**

**thus** *?thesis*  
**using** *True* **by** *simp*

**next**

**case** *False*

**thus** *?thesis*  
**using** *Cons.prem(3)* **by** *simp*

**qed**

**qed**

**lemma** *trail-defined-lit-dropWhileI*:

**assumes**  
*sorted-wrt*  $R \Gamma$  **and**

$monotone-on (set \Gamma) R (\geq) (\lambda x. P (fst x))$  **and**  
 $\neg P L \wedge \neg P (- L)$  **and**  
 $trail-defined-lit \Gamma L$   
**shows**  $trail-defined-lit (dropWhile (\lambda Ln. P (fst Ln)) \Gamma) L$   
**using**  $assms in-trail-atms-dropWhileI$   
**by** ( $smt (verit) imageE image-eqI mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone$   
 $trail-defined-lit-def trail-defined-lit-iff-trail-defined-atm$ )

**lemma**  $trail-defined-cls-dropWhileI$ :

**assumes**  
 $sorted-wrt R \Gamma$  **and**  
 $monotone-on (set \Gamma) R (\geq) (\lambda x. P (fst x))$  **and**  
 $\forall L \in \# C. \neg P L \wedge \neg P (- L)$  **and**  
 $trail-defined-cls \Gamma C$   
**shows**  $trail-defined-cls (dropWhile (\lambda Ln. P (fst Ln)) \Gamma) C$   
**using**  $assms trail-defined-lit-dropWhileI$   
**by** ( $metis trail-defined-cls-def$ )

**lemma**  $nbeX-less-than-least-in-fset$ :  $\neg (\exists w \mid \in \mid X. w \prec_c x)$   
**if**  $linorder-cls.is-least-in-fset X x$  **for**  $X x$   
**using**  $that$  **unfolding**  $linorder-cls.is-least-in-fset-iff$  **by**  $auto$

**lemma**  $clause-le-if-lt-least-greater$ :

**fixes**  $N U_{er} \mathcal{F} C D$   
**defines**  
 $C \equiv The-optional (linorder-cls.is-least-in-fset (ffilter ((\prec_c) D) N))$   
**assumes**  
 $C-lt: \bigwedge E. C = Some E \implies C \prec_c E$  **and**  
 $C-in: C \mid \in \mid N$   
**shows**  $C \preceq_c D$   
**proof** ( $cases C$ )  
**case**  $None$

**hence**  $\neg (\exists E. linorder-cls.is-least-in-fset (ffilter ((\prec_c) D) N) E)$   
**using**  $C-def$   
**by** ( $metis None-eq-The-optionalD Uniq-D linorder-cls.Uniq-is-least-in-fset$ )

**hence**  $\neg (\exists E \mid \in \mid N. D \prec_c E)$   
**by** ( $metis femptyE ffmembers-filter linorder-cls.ex1-least-in-fset$ )

**thus**  $?thesis$   
**using**  $C-in linorder-cls.less-linear$  **by**  $blast$

**next**  
**case** ( $Some E$ )

**hence**  $linorder-cls.is-least-in-fset (ffilter ((\prec_c) D) N) E$   
**using**  $C-def$  **by** ( $metis Some-eq-The-optionalD$ )

**hence**  $C \prec_c D \vee C = D$

```

    by (metis C-in C-lt Some fmember-filter linorder-cls.neqE nbex-less-than-least-in-fset)

    thus ?thesis
      by simp
  qed

end

```

## 10 Lemmas about going between ground and first-order terms

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**lemma** *ex1-gterm-of-term*:

```

  fixes t :: 'f gterm
  shows  $\exists!(t' :: ('f, 'v) \text{ term}). \text{ground } t' \wedge t = \text{gterm-of-term } t'$ 
  proof (rule ex1I)
    show  $\text{ground } (\text{term-of-gterm } t) \wedge t = \text{gterm-of-term } (\text{term-of-gterm } t)$ 
      by simp
  next
    fix t' :: ('f, 'v) term
    show  $\text{ground } t' \wedge t = \text{gterm-of-term } t' \implies t' = \text{term-of-gterm } t$ 
      by (induction t') (simp-all add: map-idI)
  qed

```

**lemma** *inj-betw-gterm-of-term*:  $\text{bij-betw } \text{gterm-of-term } \{t. \text{ground } t\} \text{ UNIV}$

```

  unfolding bij-betw-def
  proof (rule conjI)
    show  $\text{inj-on } \text{gterm-of-term } \{t. \text{ground } t\}$ 
      by (metis gterm-of-term-inj mem-Collect-eq)
  next
    show  $\text{gterm-of-term } \{t. \text{ground } t\} = \text{UNIV}$ 
      proof (rule Set.subset-antisym)
        show  $\text{gterm-of-term } \{t. \text{Term-Context.ground } t\} \subseteq \text{UNIV}$ 
          by simp
      next
        show  $\text{UNIV} \subseteq \text{gterm-of-term } \{t. \text{Term-Context.ground } t\}$ 
          by (metis (mono-tags, opaque-lifting) ground-term-of-gterm image-iff mem-Collect-eq
              subsetI term-of-gterm-inv)
      qed
    qed
  qed

end

```

## 11 SCL(FOL) for first-order terms

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**definition** *less-B* where

*less-B*  $x\ y \longleftrightarrow \text{ground } x \wedge \text{ground } y \wedge \text{gterm-of-term } x \prec_t \text{gterm-of-term } y$

**sublocale** *order-less-B*: *order less-B<sup>==</sup> less-B*

by *unfold-locales (auto simp add: less-B-def)*

**sublocale** *scl-fol*: *scl-fol-calculus renaming-vars less-B*

**proof** *unfold-locales*

**fix**  $\beta :: ('f, 'v) \text{ term}$

**have** *Collect-gterms-eq*:  $\bigwedge P. \{y. P\ y\} = \text{gterm-of-term } \{t. \text{ground } t \wedge P (\text{gterm-of-term } t)\}$

**using** *Collect-eq-image-filter-Collect-if-bij-betw*[*OF binj-betw-gterm-of-term subset-UNIV*]

**by** *auto*

**have**  $\{t_G. t_G \prec_t \text{gterm-of-term } \beta\} = \text{gterm-of-term } \{x. \text{ground } x \wedge \text{gterm-of-term } x \prec_t \text{gterm-of-term } \beta\}$

**using** *Collect-gterms-eq*[*of*  $\lambda t_G. t_G \prec_t \text{gterm-of-term } \beta$ ].

**hence** *finite* ( $\text{gterm-of-term } \{x. \text{ground } x \wedge \text{gterm-of-term } x \prec_t \text{gterm-of-term } \beta\}$ )

**using** *finite-less-trm*[*of*  $\text{gterm-of-term } \beta$ ] **by** *metis*

**moreover** **have** *inj-on*  $\text{gterm-of-term } \{x. \text{ground } x \wedge \text{gterm-of-term } x \prec_t \text{gterm-of-term } \beta\}$

**by** (*rule* *gterm-of-term-inj*) *simp*

**ultimately** **have** *finite*  $\{x. \text{ground } x \wedge \text{gterm-of-term } x \prec_t \text{gterm-of-term } \beta\}$

**using** *finite-imageD* **by** *blast*

**thus** *finite*  $\{x. \text{less-B } x\ \beta\}$

**unfolding** *less-B-def*

**using** *not-finite-existsD* **by** *force*

**qed**

**end**

**lemma** *trail-atms-eq-trail-atms-if-same-lits*:

**assumes** *list-all2*  $(\lambda x\ y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10}$

**shows** *trail-atms*  $\Gamma_9 = \text{trail-atms } \Gamma_{10}$

**using** *assms*

**proof** (*induction*  $\Gamma_9 \Gamma_{10}$  *rule*: *list.rel-induct*)

**case** *Nil*

**show** *?case*

**by** (*simp* *add*: *trail-atms-def*)

**next**

**case** (*Cons*  $Ln_9 \Gamma_9' Ln_{10} \Gamma_{10}'$ )

**thus** *?case*

**by** (*simp* *add*: *trail-atms-def*)

**qed**

```

lemma trail-false-lit-eq-trail-false-lit-if-same-lits:
  assumes list-all2 ( $\lambda x y. \text{fst } x = \text{fst } y$ )  $\Gamma_9 \Gamma_{10}$ 
  shows trail-false-lit  $\Gamma_9 = \text{trail-false-lit } \Gamma_{10}$ 
  using assms
proof (induction  $\Gamma_9 \Gamma_{10}$  rule: list.rel-induct)
  case Nil
  show ?case
    by (simp add: trail-false-lit-def)
next
  case (Cons  $Ln_9 \Gamma_9' Ln_{10} \Gamma_{10}'$ )
  thus ?case
    unfolding trail-false-lit-def
    unfolding list.set image-insert
    by (metis insert-iff)
qed

lemma trail-false-cls-eq-trail-false-cls-if-same-lits:
  assumes list-all2 ( $\lambda x y. \text{fst } x = \text{fst } y$ )  $\Gamma_9 \Gamma_{10}$ 
  shows trail-false-cls  $\Gamma_9 = \text{trail-false-cls } \Gamma_{10}$ 
  unfolding trail-false-cls-def
  unfolding trail-false-lit-eq-trail-false-lit-if-same-lits [OF assms] ..

lemma trail-defined-lit-eq-trail-defined-lit-if-same-lits:
  assumes list-all2 ( $\lambda x y. \text{fst } x = \text{fst } y$ )  $\Gamma_9 \Gamma_{10}$ 
  shows trail-defined-lit  $\Gamma_9 = \text{trail-defined-lit } \Gamma_{10}$ 
  using assms
proof (induction  $\Gamma_9 \Gamma_{10}$  rule: list.rel-induct)
  case Nil
  show ?case
    by (simp add: trail-defined-lit-def)
next
  case (Cons  $Ln_9 \Gamma_9' Ln_{10} \Gamma_{10}'$ )
  thus ?case
    unfolding trail-defined-lit-def
    unfolding list.set image-insert
    by (metis (no-types, opaque-lifting) insert-iff)
qed

lemma trail-defined-cls-eq-trail-defined-cls-if-same-lits:
  assumes list-all2 ( $\lambda x y. \text{fst } x = \text{fst } y$ )  $\Gamma_9 \Gamma_{10}$ 
  shows trail-defined-cls  $\Gamma_9 = \text{trail-defined-cls } \Gamma_{10}$ 
  unfolding trail-defined-cls-def
  unfolding trail-defined-lit-eq-trail-defined-lit-if-same-lits [OF assms] ..

end
theory ORD-RES
  imports Background
begin

```

## 12 ORD-RES

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**lemma** *ex-false-clause*  $N \longleftrightarrow \neg (\forall C \in N. \text{ord-res-Interp } N C \models C)$   
**unfolding** *ex-false-clause-def* **by** *metis*

**lemma**  $\neg \text{ex-false-clause } N \longleftrightarrow (\forall C \in N. \text{ord-res-Interp } N C \models C)$   
**unfolding** *ex-false-clause-def* **by** *metis*

**definition** *ord-res-final* **where**

*ord-res-final*  $N \longleftrightarrow \{\#\} \notin N \vee \neg \text{ex-false-clause } (\text{fset } N)$

**inductive** *ord-res* **where**

*factoring*:  $\{\#\} \notin N \implies \text{ex-false-clause } (\text{fset } N) \implies$   
 — Maybe write  $\neg \text{ord-res-final } N$  instead?  
 $P \notin N \implies \text{ord-res.ground-factoring } P C \implies$   
 $N' = \text{finsert } C N \implies$   
 $\text{ord-res } N N' \mid$

*resolution*:  $\{\#\} \notin N \implies \text{ex-false-clause } (\text{fset } N) \implies$   
 $P1 \notin N \implies P2 \notin N \implies \text{ord-res.ground-resolution } P1 P2 C \implies$   
 $N' = \text{finsert } C N \implies$   
 $\text{ord-res } N N'$

**inductive** *ord-res-load* **where**

$N \neq \{\|\}$   $\implies \text{ord-res-load } N N$

**sublocale** *ord-res-antics: semantics* **where**

*step* = *ord-res* **and**  
*final* = *ord-res-final*

**proof** *unfold-locales*

**fix**  $N :: 'f \text{ gterm clause fset}$

**assume** *ord-res-final*  $N$

**thus** *finished* *ord-res*  $N$

**unfolding** *ord-res-final-def*

**proof** (*elim disjE*)

**show**  $\{\#\} \notin N \implies \text{finished } \text{ord-res } N$

**by** (*simp add: finished-def ord-res.simps*)

**next**

**show**  $\neg \text{ex-false-clause } (\text{fset } N) \implies \text{finished } \text{ord-res } N$

**by** (*simp add: finished-def ord-res.simps*)

**qed**

**qed**

**sublocale** *ord-res-language: language* **where**

*step* = *ord-res* **and**

*final* = *ord-res-final* **and**

*load* = *ord-res-load*

```

    by unfold-locales

end

end
theory ORD-RES-1
  imports ORD-RES
begin

```

### 13 ORD-RES-1 (deterministic)

```

context simulation-SCLFOL-ground-ordered-resolution begin

```

```

inductive ord-res-1 where

```

```

  factoring:

```

```

    is-least-false-clause  $N C \implies$ 
    linorder-lit.is-maximal-in-mset  $C L \implies$ 
    is-pos  $L \implies$ 
    ord-res.ground-factoring  $C C' \implies$ 
     $N' = \text{finsert } C' N \implies$ 
    ord-res-1  $N N' \mid$ 

```

```

  resolution:

```

```

    is-least-false-clause  $N C \implies$ 
    linorder-lit.is-maximal-in-mset  $C L \implies$ 
    is-neg  $L \implies$ 
     $D \mid \in \mid N \implies$ 
     $D \prec_c C \implies$ 
    ord-res.production (fset  $N$ )  $D = \{\text{atm-of } L\} \implies$ 
    ord-res.ground-resolution  $C D CD \implies$ 
     $N' = \text{finsert } CD N \implies$ 
    ord-res-1  $N N'$ 

```

```

lemma

```

```

  assumes ord-res.ground-resolution  $C D CD$ 
  shows  $D \prec_c C$ 
  using assms
  by (auto simp add: ord-res.ground-resolution.simps)

```

```

lemma right-unique-ord-res-1: right-unique ord-res-1

```

```

proof (rule right-uniqueI)

```

```

  fix  $N N' N'' :: 'f \text{ gterm clause fset}$ 
  assume step1: ord-res-1  $N N'$  and step2: ord-res-1  $N N''$ 
  from step1 show  $N' = N''$ 
  proof (cases  $N N'$  rule: ord-res-1.cases)
    case hyp1: (factoring  $C1 L1 C1'$ )
    from step2 show ?thesis
    proof (cases  $N N''$  rule: ord-res-1.cases)
      case hyp2: (factoring  $C2 L2 C2'$ )

```

```

from hyps1 hyps2 have  $C1 = C2$ 
  by (meson Uniq-D Uniq-is-least-false-clause)
with hyps1 hyps2 have  $C1' = C2'$ 
  by (metis (no-types, lifting) Uniq-D ord-res.unique-ground-factoring)
with hyps1 hyps2 show ?thesis
  by argo
next
case hyps2: (resolution C2 L2 D2 CD2)
from hyps1 hyps2 have  $C1 = C2$ 
  by (meson Uniq-D Uniq-is-least-false-clause)
with hyps1 hyps2 have  $L1 = L2$ 
  by (meson Uniq-D linorder-lit.Uniq-is-maximal-in-mset)
with hyps1 hyps2 have False
  by metis
thus ?thesis ..
qed
next
case hyps1: (resolution C1 L1 D1 CD1)
from step2 show ?thesis
proof (cases N N'' rule: ord-res-1.cases)
  case hyps2: (factoring C2 L2 C2')
from hyps1 hyps2 have  $C1 = C2$ 
  by (meson Uniq-D Uniq-is-least-false-clause)
with hyps1 hyps2 have  $L1 = L2$ 
  by (meson Uniq-D linorder-lit.Uniq-is-maximal-in-mset)
with hyps1 hyps2 have False
  by metis
thus ?thesis ..
next
case hyps2: (resolution C2 L2 D2 CD2)
from hyps1 hyps2 have  $C1 = C2$ 
  by (meson Uniq-D Uniq-is-least-false-clause)
with hyps1 hyps2 have  $L1 = L2$ 
  by (meson Uniq-D linorder-lit.Uniq-is-maximal-in-mset)
with hyps1 hyps2 have  $D1 = D2$ 
  by (metis (mono-tags) Uniq-D ord-res.Uniq-production-eq-singleton)
with hyps1 hyps2 have  $CD1 = CD2$ 
by (metis (no-types, lifting) Uniq-D ⟨C1 = C2⟩ ord-res.unique-ground-resolution)
with hyps1 hyps2 show ?thesis
  by argo
qed
qed
qed

```

**definition** *ord-res-1-final* **where**  
*ord-res-1-final N*  $\longleftrightarrow$  *ord-res-final N*

**inductive** *ord-res-1-load* **where**  
 $N \neq \{\|\}$   $\implies$  *ord-res-1-load N N*



```

sublocale ord-res-1-semantics: semantics where
  step = ord-res-1 and
  final = ord-res-1-final
proof unfold-locale
  fix N :: 'f gterm clause fset
  assume ord-res-1-final N
  thus finished ord-res-1 N
    unfolding ord-res-1-final-def ord-res-final-def
proof (elim disjE)
  assume {#} |∈| N
  have False if ord-res-1 N N' for N'
    using that
proof (cases N N' rule: ord-res-1.cases)
  case hyps: (factoring C L C')
  from hyps ⟨{#} |∈| N⟩ have C = {#}
    unfolding is-least-false-clause-def
  by (metis (no-types, lifting) emptyE fmember-filter linorder-cls.dual-order.strict-trans
    linorder-cls.is-least-in-fset-iff linorder-cls.less-irrefl
    ord-res.mulp-if-all-left-smaller set-mset-empty true-cls-empty)
  moreover from hyps have L ∈# C
    using linorder-lit.is-maximal-in-mset-iff by blast
  ultimately show False
    by simp
next
  case hyps: (resolution C L D CD)
  from hyps ⟨{#} |∈| N⟩ have C = {#}
    unfolding is-least-false-clause-def
  by (metis (no-types, lifting) emptyE fmember-filter linorder-cls.dual-order.strict-trans
    linorder-cls.is-least-in-fset-iff linorder-cls.less-irrefl
    ord-res.mulp-if-all-left-smaller set-mset-empty true-cls-empty)
  moreover from hyps have L ∈# C
    using linorder-lit.is-maximal-in-mset-iff by blast
  ultimately show False
    by simp
qed
thus finished ord-res-1 N
  unfolding finished-def by metis
next
assume ¬ ex-false-clause (fset N)
have False if ord-res-1 N N' for N'
  using that
proof (cases N N' rule: ord-res-1.cases)
  case (factoring C L C')
  with ⟨¬ ex-false-clause (fset N)⟩ show False
    unfolding ex-false-clause-def is-least-false-clause-def
    using linorder-cls.is-least-in-fset-iff by force
next
  case (resolution C L D CD)

```

**with**  $\langle \neg \text{ex-false-clause } (fset N) \rangle$  **show** *False*  
**unfolding** *ex-false-clause-def is-least-false-clause-def*  
**using** *linorder-cls.is-least-in-fset-iff* **by** *force*  
**qed**  
**thus** *finished ord-res-1 N*  
**unfolding** *finished-def* **by** *metis*  
**qed**  
**qed**

**sublocale** *ord-res-1-language: language* **where**  
*step = ord-res-1* **and**  
*final = ord-res-1-final* **and**  
*load = ord-res-1-load*  
**by** *unfold-locales*

**lemma** *ex-ord-res-1-if-not-final:*  
**assumes**  $\neg \text{ord-res-1-final } N$   
**shows**  $\exists N'. \text{ord-res-1 } N N'$   
**proof** –  
**from** *assms* **have**  $\{\#\} \notin N$  **and** *ex-false-clause (fset N)*  
**unfolding** *ord-res-1-final-def ord-res-final-def* **by** *argo+*

**obtain** *C* **where** *C-least-false: is-least-false-clause N C*  
**using**  $\langle \text{ex-false-clause } (fset N) \rangle$  *obtains-least-false-clause-if-ex-false-clause* **by**  
*metis*

**hence**  $C \neq \{\#\}$   
**using**  $\langle \{\#\} \notin N \rangle$   
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
**by** *metis*

**then obtain** *L* **where** *C-max: linorder-lit.is-maximal-in-mset C L*  
**using** *linorder-lit.ex-maximal-in-mset* **by** *metis*

**show** *?thesis*  
**proof** (*cases L*)  
**case** (*Pos A*)

**hence**  $\neg \text{linorder-lit.is-greatest-in-mset } C L$   
**using** *pos-lit-not-greatest-if-maximal-in-least-false-clause*  
**using** *C-least-false is-least-false-clause-def* **by** *blast*

**then obtain** *C'* **where** *ord-res.ground-factoring C C'*  
**using** *ex-ground-factoringI C-max Pos* **by** *metis*

**thus** *?thesis*  
**using** *ord-res-1.factoring*  
**by** (*metis*  $\langle \text{is-least-false-clause } N C \rangle$  *is-pos-def ord-res.ground-factoring.cases*)  
**next**

```

case (Neg A)
then obtain D where
  D |∈| N and
  D  $\prec_c$  C and
  ord-res.is-strictly-maximal-lit (Pos A) D and
  ord-res.production (fset N) D = {A}
  using bx-smaller-productive-clause-if-least-false-clause-has-negative-max-lit
  using C-least-false C-max by metis

moreover then obtain CD where
  ord-res.ground-resolution C D CD
  using ex-ground-resolutionI C-max Neg by metis

ultimately show ?thesis
  using ord-res-1.resolution[of N C L D CD finset CD N]
  using C-least-false C-max Neg by auto
qed
qed

corollary ord-res-1-safe: ord-res-1-final N  $\vee$  ( $\exists N'$ . ord-res-1 N N')
  using ex-ord-res-1-if-not-final by metis

```

**end**

**end**

**theory** *Exhaustive-Factorization*

**imports** *Background*

**begin**

## 14 Function for full factorization

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**definition** *efac* :: '*f gterm clause*  $\Rightarrow$  '*f gterm clause* **where**

*efac C* = (*THE C'*. *ord-res.ground-factorizing\*\* C C'  $\wedge$  ( $\exists C''$ . ord-res.ground-factorizing C' C'')*)

The function *efac* performs exhaustive factorization of its input clause.

**lemma** *ex1-efac-eq-factorizing-chain*:

$\exists! C'$ . *efac C* = *C'  $\wedge$  ord-res.ground-factorizing\*\* C C'  $\wedge$  ( $\exists C''$ . ord-res.ground-factorizing C' C'')*

**proof** –

**have** *right-unique* ( $\lambda x y$ . *ord-res.ground-factorizing\*\* x y  $\wedge$  ( $\exists z$ . ord-res.ground-factorizing y z)*)

**using** *ord-res.unique-ground-factorizing right-unique-terminating-rtranclp right-unique-iff*  
  **by** *blast*

**moreover obtain** *C'* **where** *ord-res.ground-factorizing\*\* C C'  $\wedge$  ( $\exists C''$ . ord-res.ground-factorizing C' C'')*

**using** *ex-terminating-rtranclp*[*OF ord-res.termination-ground-factoring*]  
**by** *metis*

**ultimately have**  $\text{efac } C = C'$   
**by** (*simp add: efac-def right-unique-def the-equality*)

**then show** *?thesis*  
**using**  $\langle \text{ord-res.ground-factoring}^{**} C C' \wedge (\# C''). \text{ord-res.ground-factoring } C' C'' \rangle$  **by** *blast*  
**qed**

**lemma** *efac-eq-disj*:  
 $\text{efac } C = C \vee (\exists! C'. \text{efac } C = C' \wedge \text{ord-res.ground-factoring}^{**} C C' \wedge (\# C''). \text{ord-res.ground-factoring } C' C'')$   
**using** *ex1-efac-eq-factoring-chain*  
**by** (*metis is-pos-def*)

**lemma** *member-mset-if-count-eq-Suc*:  $\text{count } X x = \text{Suc } n \implies x \in \# X$   
**by** (*simp add: count-inI*)

**lemmas** *member-fsetE = mset-add*

**lemma** *ord-res-ground-factoring-iff*:  $\text{ord-res.ground-factoring } C C' \longleftrightarrow (\exists A. \text{ord-res.is-maximal-lit } (Pos A) C \wedge (\exists n. \text{count } C (Pos A) = \text{Suc } (\text{Suc } n) \wedge C' = C - \{\#Pos A\}))$   
**proof** (*rule iffI*)  
**assume**  $\text{ord-res.ground-factoring } C C'$   
**thus**  $\exists A. \text{ord-res.is-maximal-lit } (Pos A) C \wedge (\exists n. \text{count } C (Pos A) = \text{Suc } (\text{Suc } n) \wedge C' = C - \{\#Pos A\})$   
**proof** (*cases C C' rule: ord-res.ground-factoring.cases*)  
**case** (*ground-factoringI A P'*)  
**show** *?thesis*  
**proof** (*intro exI conjI*)  
**show**  $\text{ord-res.is-maximal-lit } (Pos A) C$   
**using**  $\langle \text{ord-res.is-maximal-lit } (Pos A) C \rangle$  .  
**next**  
**show**  $\text{count } C (Pos A) = \text{Suc } (\text{Suc } (\text{count } P' (Pos A)))$   
**unfolding**  $\langle C = \text{add-mset } (Pos A) (\text{add-mset } (Pos A) P') \rangle$  **by** *simp*  
**next**  
**show**  $C' = \text{remove1-mset } (Pos A) C$   
**unfolding**  $\langle C = \text{add-mset } (Pos A) (\text{add-mset } (Pos A) P') \rangle \langle C' = \text{add-mset } (Pos A) P' \rangle$  **by** *simp*  
**qed**  
**qed**  
**next**  
**assume**  $\exists A. \text{ord-res.is-maximal-lit } (Pos A) C \wedge (\exists n. \text{count } C (Pos A) = \text{Suc } (\text{Suc } n) \wedge C' = C - \{\#Pos A\})$   
**then obtain** *A n* **where**  
 $\text{ord-res.is-maximal-lit } (Pos A) C$  **and**

$\text{count } C \text{ (Pos } A) = \text{Suc (Suc } n)$  **and**  
 $C' = C - \{\# \text{Pos } A\}$   
**by** *metis*

**have**  $\text{Pos } A \in \# C$   
**using**  $\langle \text{count } C \text{ (Pos } A) = \text{Suc (Suc } n) \rangle$  *member-mset-if-count-eq-Suc* **by** *metis*  
**then obtain**  $C''$  **where**  $C = \text{add-mset (Pos } A) C''$   
**by** (*auto elim: member-fsetE*)  
**with**  $\langle \text{count } C \text{ (Pos } A) = \text{Suc (Suc } n) \rangle$  **have**  $\text{count } C'' \text{ (Pos } A) = \text{Suc } n$   
**by** *simp*  
**hence**  $\text{Pos } A \in \# C''$   
**using** *member-mset-if-count-eq-Suc* **by** *metis*  
**then obtain**  $C'''$  **where**  $C'' = \text{add-mset (Pos } A) C'''$   
**by** (*auto elim: member-fsetE*)

**show** *ord-res.ground-factorizing*  $C C'$   
**proof** (*rule ord-res.ground-factorizingI*)  
**show**  $C = \text{add-mset (Pos } A) (\text{add-mset (Pos } A) C''')$   
**using**  $\langle C = \text{add-mset (Pos } A) C'' \rangle \langle C'' = \text{add-mset (Pos } A) C''' \rangle$  **by** *metis*  
**next**  
**show** *ord-res.is-maximal-lit*  $(\text{Pos } A) C$   
**using**  $\langle \text{ord-res.is-maximal-lit (Pos } A) C \rangle$  .  
**next**  
**show**  $C' = \text{add-mset (Pos } A) C'''$   
**using**  $\langle C' = C - \{\# \text{Pos } A\} \rangle \langle C = \text{add-mset (Pos } A) C'' \rangle \langle C'' = \text{add-mset (Pos } A) C''' \rangle$  **by** *simp*  
**qed** *simp-all*  
**qed**

**lemma** *tranclp-ord-res-ground-factorizing-iff*:  
 $\text{ord-res.ground-factorizing}^{++} C C' \wedge (\nexists C''. \text{ord-res.ground-factorizing } C' C'') \longleftrightarrow$   
 $(\exists A. \text{ord-res.is-maximal-lit (Pos } A) C \wedge (\exists n. \text{count } C \text{ (Pos } A) = \text{Suc (Suc } n) \wedge$   
 $C' = C - \text{replicate-mset (Suc } n) \text{ (Pos } A)))$

**proof** (*intro iffI; elim exE conjE*)  
**assume**  $\text{ord-res.ground-factorizing}^{++} C C'$  **and**  $(\nexists C''. \text{ord-res.ground-factorizing } C' C'')$   
**then show**  $\exists A. \text{ord-res.is-maximal-lit (Pos } A) C \wedge (\exists n. \text{count } C \text{ (Pos } A) = \text{Suc (Suc } n) \wedge$   
 $C' = C - \text{replicate-mset (Suc } n) \text{ (Pos } A))$   
**proof** (*induction C rule: converse-tranclp-induct*)  
**case** (*base C*)  
**from** *base.hyps* **obtain**  $A n$  **where**  
 $\text{ord-res.is-maximal-lit (Pos } A) C$  **and**  
 $\text{count } C \text{ (Pos } A) = \text{Suc (Suc } n)$  **and**  
 $C' = \text{remove1-mset (Pos } A) C$   
**unfolding** *ord-res-ground-factorizing-iff* **by** *auto*

**moreover have**  $n = 0$   
**proof** (*rule ccontr*)

```

assume  $n \neq 0$ 
then obtain  $C''$  where  $C' = \text{add-mset } (Pos\ A) (\text{add-mset } (Pos\ A)\ C'')$ 
  by (metis (no-types, lifting) Zero-not-Suc calculation(2,3) count-add-mset
count-inI
  diff-Suc-1 insert-DiffM)
hence ord-res.ground-factorizing  $C' (\text{add-mset } (Pos\ A)\ C'')$ 
  using ord-res.ground-factorizingI
by (metis calculation(1,3) linorder-lit.is-maximal-in-mset-iff more-than-one-mset-mset-diff
union-single-eq-member)
with base.prems show False
  by metis
qed

ultimately show ?case
  by (metis replicate-mset-0 replicate-mset-Suc)
next
case (step C C'')
from step.IH step.prems obtain  $A\ n$  where
  ord-res.is-maximal-lit  $(Pos\ A)\ C''$  and
  count  $C'' (Pos\ A) = Suc\ (Suc\ n)$  and
   $C' = C'' - \text{replicate-mset } (Suc\ n) (Pos\ A)$ 
by metis

from step.hyps(1) obtain  $A'\ m$  where
  ord-res.is-maximal-lit  $(Pos\ A')\ C$  and
  count  $C (Pos\ A') = Suc\ (Suc\ m)$  and
   $C'' = \text{remove1-mset } (Pos\ A')\ C$ 
unfolding ord-res.ground-factorizing-iff by metis

have  $A' = A$ 
  using  $\langle \text{ord-res.is-maximal-lit } (Pos\ A)\ C'' \rangle \langle \text{ord-res.is-maximal-lit } (Pos\ A')\ C \rangle$ 
by (metis  $\langle C'' = \text{remove1-mset } (Pos\ A')\ C \rangle \langle \text{count } C (Pos\ A') = Suc\ (Suc\ m) \rangle$ 
  add-mset-remove-trivial-eq count-add-mset count-greater-zero-iff diff-Suc-1
linorder-lit.antisym-conv3 linorder-lit.is-maximal-in-mset-iff literal.inject(1)
zero-less-Suc)

have  $m = Suc\ n$ 
  using  $\langle \text{count } C'' (Pos\ A) = Suc\ (Suc\ n) \rangle \langle \text{count } C (Pos\ A') = Suc\ (Suc\ m) \rangle$ 
unfolding  $\langle C'' = \text{remove1-mset } (Pos\ A')\ C \rangle \langle A' = A \rangle$ 
by simp

show ?case
proof (intro exI conjI)
  show ord-res.is-maximal-lit  $(Pos\ A)\ C$ 
    using  $\langle \text{ord-res.is-maximal-lit } (Pos\ A')\ C \rangle \langle A' = A \rangle$  by metis
next
  show count  $C (Pos\ A) = Suc\ (Suc\ m)$ 

```

```

    using ⟨count C (Pos A') = Suc (Suc m)⟩ ⟨A' = A⟩ by metis
  next
    show C' = C - replicate-mset (Suc m) (Pos A)
    unfolding ⟨C' = C'' - replicate-mset (Suc n) (Pos A)⟩ ⟨C'' = remove1-mset
(Pos A') C⟩
      ⟨A' = A⟩ ⟨m = Suc n⟩
    by simp
  qed
  qed
next
  fix A n assume ord-res.is-maximal-lit (Pos A) C
  thus count C (Pos A) = Suc (Suc n) ⇒ C' = C - replicate-mset (Suc n) (Pos
A) ⇒
    ord-res.ground-factorizing++ C C' ∧ (∄ C''. ord-res.ground-factorizing C' C'')
  proof (induction n arbitrary: C)
  case 0
  hence (ord-res.is-maximal-lit (Pos A) C ∧
    (count C (Pos A) = Suc (Suc 0) ∧
      C' = remove1-mset (Pos A) C))
    by (metis replicate-mset-0 replicate-mset-Suc)
  hence ord-res.ground-factorizing C C' ∧ (∄ a. ord-res.ground-factorizing C' a)
  unfolding ord-res.ground-factorizing-iff
  by (metis Zero-not-Suc add-mset-remove-trivial-eq count-add-mset count-inI
    linorder-lit.antisym-conv3 linorder-lit.is-maximal-in-mset-iff nat.inject)
  thus ?case
  by blast
next
  case (Suc n)
  have ord-res.ground-factorizing++ (C - {#Pos A#}) C' ∧ (∄ a. ord-res.ground-factorizing
C' a)
  proof (rule Suc.IH)
  show count (remove1-mset (Pos A) C) (Pos A) = Suc (Suc n)
  using Suc.prem by simp
  next
  show C' = remove1-mset (Pos A) C - replicate-mset (Suc n) (Pos A)
  using Suc.prem by simp
  next
  show ord-res.is-maximal-lit (Pos A) (remove1-mset (Pos A) C)
  using Suc.prem
  by (smt (verit, ccfv-SIG) Zero-not-Suc add-diff-cancel-left' add-mset-remove-trivial-eq
    count-add-mset count-inI linorder-lit.is-maximal-in-mset-iff plus-1-eq-Suc)
  qed
  moreover have ord-res.ground-factorizing C (C - {#Pos A#})
  unfolding ord-res.ground-factorizing-iff
  proof (intro exI conjI)
  show ord-res.is-maximal-lit (Pos A) C
  using Suc.prem by metis
  next

```

```

    show count C (Pos A) = Suc (Suc (Suc n))
      using Suc.premis by metis
  next
    show remove1-mset (Pos A) C = remove1-mset (Pos A) C ..
  qed

  ultimately show ?case
    by auto
  qed
qed

lemma minus-mset-replicate-mset-eq-add-mset-filter-mset:
  assumes count X x = Suc n
  shows X - replicate-mset n x = add-mset x {#y ∈# X. y ≠ x#}
  using assms
  by (metis add-diff-cancel-left' add-mset-diff-bothsides filter-mset-eq filter-mset-neq
    multiset-partition replicate-mset-Suc union-mset-add-mset-right)

lemma minus-mset-replicate-mset-eq-add-mset-add-mset-filter-mset:
  assumes count X x = Suc (Suc n)
  shows X - replicate-mset n x = add-mset x (add-mset x {#y ∈# X. y ≠ x#})
  using assms
  by (metis add-diff-cancel-left' add-mset-diff-bothsides filter-mset-eq filter-mset-neq
    multiset-partition replicate-mset-Suc union-mset-add-mset-right)

lemma rtrancl-ground-factoring-iff:
  shows ord-res.ground-factoring** C C' ∧ (∄ C''. ord-res.ground-factoring C' C'')
  ⇔
  ((∄ A. ord-res.is-maximal-lit (Pos A) C ∧ count C (Pos A) ≥ 2) ∧ C = C' ∨
  (∃ A. ord-res.is-maximal-lit (Pos A) C ∧ C' = add-mset (Pos A) {#L ∈# C.
  L ≠ Pos A#}))
  proof (intro iffI; elim exE conjE disjE)
    show ord-res.ground-factoring** C C' ⇒ ∄ C''. ord-res.ground-factoring C' C''
    ⇒
      (∄ A. ord-res.is-maximal-lit (Pos A) C ∧ 2 ≤ count C (Pos A)) ∧ C = C' ∨
      (∃ A. ord-res.is-maximal-lit (Pos A) C ∧ C' = add-mset (Pos A) {#L ∈# C.
      L ≠ Pos A#})
    proof (induction C rule: converse-rtranclp-induct)
      case base
      thus ?case
      by (metis add-2-eq-Suc le-Suc-ex ord-res-ground-factoring-iff)
    next
      case (step y z)
      hence ord-res.ground-factoring++ y C' ∧ (∄ x. ord-res.ground-factoring C' x)
      by simp
      thus ?case
      unfolding tranclp-ord-res-ground-factoring-iff
      by (metis minus-mset-replicate-mset-eq-add-mset-filter-mset)
    qed
  qed

```



```

next
  assume  $\nexists A. \text{ord-res.is-maximal-lit } (Pos A) C \wedge 2 \leq \text{count } C (Pos A)$  and  $C = C'$ 
  thus  $\text{ord-res.ground-factoring}^{**} C C' \wedge (\nexists C''. \text{ord-res.ground-factoring } C' C'')$ 
    by (metis One-nat-def Suc-1 Suc-le-eq Suc-le-mono ord-res-ground-factoring-iff rtranclp.rtrancl-refl zero-less-Suc)
next
  fix A assume  $\text{ord-res.is-maximal-lit } (Pos A) C$ 
  then obtain n where  $\text{count } C (Pos A) = \text{Suc } n$ 
    by (meson in-countE linorder-lit.is-maximal-in-mset-iff)
  with  $\langle \text{ord-res.is-maximal-lit } (Pos A) C \rangle$  show  $C' = \text{add-mset } (Pos A) \{\#L \in \#C. L \neq Pos A\# \} \implies$ 
     $\text{ord-res.ground-factoring}^{**} C C' \wedge (\nexists C''. \text{ord-res.ground-factoring } C' C'')$ 
  proof (induction n arbitrary: C)
    case 0

      have  $(\nexists a. \text{ord-res.ground-factoring } C a)$ 
      proof (intro notI, elim exE)
        fix D assume  $\text{ord-res.ground-factoring } C D$ 
        thus False
      proof (cases rule: ord-res.ground-factoring.cases)
        case (ground-factoringI A' P')
          hence  $A' = A$ 
          using  $\langle \text{ord-res.is-maximal-lit } (Pos A) C \rangle$ 
          using linorder-lit.Uniq-is-maximal-in-mset
          by (metis Uniq-D literal.inject(1))
        thus False
      using  $\langle \text{count } C (Pos A) = \text{Suc } 0 \rangle \langle C = \text{add-mset } (Pos A) (\text{add-mset } (Pos A) P') \rangle$  by simp
      qed
      qed
      thus ?case
      by (metis 0.prem(1) 0.prem(3) diff-zero minus-mset-replicate-mset-eq-add-mset-filter-mset replicate-mset-0 rtranclp.rtrancl-refl)
    next
      case (Suc x)
      then show ?case
      by (metis minus-mset-replicate-mset-eq-add-mset-filter-mset tranclp-into-rtranclp tranclp-ord-res-ground-factoring-iff)
    qed
  qed

lemma efac-spec:  $\text{efac } C = C \vee$ 
  ( $\exists A. \text{ord-res.is-maximal-lit } (Pos A) C \wedge \text{efac } C = \text{add-mset } (Pos A) \{\#L \in \#C. L \neq Pos A\# \}$ )
  using efac-eq-disj[of C]
proof (elim disjE)
  assume  $\text{efac } C = C$ 

```

**thus**  $\text{efac } C = C \vee$   
 $(\exists A. \text{ord-res.is-maximal-lit } (\text{Pos } A) C \wedge \text{efac } C = \text{add-mset } (\text{Pos } A) \{\#L \in \#$   
 $C. L \neq \text{Pos } A\#\})$   
**by** *metis*  
**next**  
**assume**  $\exists! C'. \text{efac } C = C' \wedge \text{ord-res.ground-factorizing}^{**} C C' \wedge$   
 $(\nexists C''. \text{ord-res.ground-factorizing } C' C'')$   
**then obtain**  $C'$  **where**  
 $\text{efac } C = C'$  **and**  
 $\text{ord-res.ground-factorizing}^{**} C C' \wedge (\nexists C''. \text{ord-res.ground-factorizing } C' C'')$   
**by** *metis*  
**thus**  $\text{efac } C = C \vee$   
 $(\exists A. \text{ord-res.is-maximal-lit } (\text{Pos } A) C \wedge \text{efac } C = \text{add-mset } (\text{Pos } A) \{\#L \in \#$   
 $C. L \neq \text{Pos } A\#\})$   
**unfolding** *rtrancl-ground-factorizing-iff*  
**by** *metis*  
**qed**

**lemma** *efac-spec-if-pos-lit-is-maximal*:

**assumes**  $L\text{-pos}: \text{is-pos } L$  **and**  $L\text{-max}: \text{ord-res.is-maximal-lit } L C$   
**shows**  $\text{efac } C = \text{add-mset } L \{\#K \in \# C. K \neq L\#\}$   
**proof** –  
**from** *assms* **obtain**  $C'$  **where**  
 $\text{efac } C = C'$  **and**  
 $\text{ord-res.ground-factorizing}^{**} C C' \wedge (\nexists C''. \text{ord-res.ground-factorizing } C' C'')$   
**using** *ex1-efac-eq-factorizing-chain* **by** *metis*  
**thus** *?thesis*  
**unfolding** *rtrancl-ground-factorizing-iff*  
**proof** (*elim disjE conjE*)  
**assume** *hyps*:  $\nexists A. \text{ord-res.is-maximal-lit } (\text{Pos } A) C \wedge 2 \leq \text{count } C (\text{Pos } A)$   
 $C = C'$   
**with** *assms* **have**  $\text{count } C L = 1$   
**by** (*metis One-nat-def in-countE is-pos-def le-less-linear less-2-cases-iff*  
*linorder-lit.is-maximal-in-mset-iff nat-less-le zero-less-Suc*)  
**hence**  $C = \text{add-mset } L \{\#K \in \# C. K \neq L\#\}$   
**by** (*metis One-nat-def diff-zero minus-mset-replicate-mset-eq-add-mset-filter-mset*  
*replicate-mset-0*)  
**thus**  $\text{efac } C = \text{add-mset } L \{\#K \in \# C. K \neq L\#\}$   
**using**  $\langle \text{efac } C = C' \rangle \langle C = C' \rangle$  **by** *argo*  
**next**  
**assume**  $\exists A. \text{ord-res.is-maximal-lit } (\text{Pos } A) C \wedge C' = \text{add-mset } (\text{Pos } A) \{\#L$   
 $\in \# C. L \neq \text{Pos } A\#\}$   
**thus**  $\text{efac } C = \text{add-mset } L \{\#K \in \# C. K \neq L\#\}$   
**by** (*metis L-max Uniq-D*  $\langle \text{efac } C = C' \rangle$  *linorder-lit.Uniq-is-maximal-in-mset*)  
**qed**  
**qed**

**lemma** *efac-empty[simp]*:  $\text{efac } \{\#\} = \{\#\}$

**by** (*metis empty-iff linorder-lit.is-maximal-in-mset-iff set-mset-empty efac-spec*)

```

lemma set-mset-efac[simp]: set-mset (efac C) = set-mset C
  using efac-spec[of C]
proof (elim disjE exE conjE)
  show efac C = C  $\implies$  set-mset (efac C) = set-mset C
    by simp
next
  fix A
  assume ord-res.is-maximal-lit (Pos A) C
  hence Pos A  $\in\#$  C
    by (simp add: linorder-lit.is-maximal-in-mset-iff)

  assume efac-C-eq: efac C = add-mset (Pos A) {#L  $\in\#$  C. L  $\neq$  Pos A#}
  show set-mset (efac C) = set-mset C
  proof (intro Set.subset-antisym Set.subsetI)
    fix L assume L  $\in\#$  efac C
    then show L  $\in\#$  C
      unfolding efac-C-eq
      using  $\langle$ Pos A  $\in\#$  C $\rangle$  by auto
    next
    fix L assume L  $\in\#$  C
    then show L  $\in\#$  efac C
      unfolding efac-C-eq
      by simp
  qed
qed

lemma efac-subset: efac C  $\subseteq\#$  C
  using efac-spec[of C]
proof (elim disjE exE conjE)
  show efac C = C  $\implies$  efac C  $\subseteq\#$  C
    by simp
next
  fix A
  assume ord-res.is-maximal-lit (Pos A) C and
    efac-C-eq: efac C = add-mset (Pos A) {#L  $\in\#$  C. L  $\neq$  Pos A#}
  then show efac C  $\subseteq\#$  C
    by (smt (verit, ccfv-SIG) filter-mset-add-mset insert-DiffM insert-subset-eq-iff
      linorder-lit.is-maximal-in-mset-iff multiset-filter-subset)
qed

lemma true-cls-efac-iff[simp]:
  fixes  $\mathcal{I} :: 'f$  gterm set and  $C :: 'f$  gclause
  shows  $\mathcal{I} \models efac C \longleftrightarrow \mathcal{I} \models C$ 
  by (metis set-mset-efac true-cls-iff-set-mset-eq)

lemma obtains-positive-greatest-lit-if-efac-not-ident:
  assumes efac C  $\neq$  C
  obtains L where is-pos L and linorder-lit.is-greatest-in-mset (efac C) L

```

**proof** –  
**from**  $\langle \text{efac } C \neq C \rangle$  **obtain**  $A$  **where**  
*Pos-A-maximal*:  $\text{linorder-lit.is-maximal-in-mset } C \text{ (Pos } A)$  **and**  
*efac-C-eq*:  $\text{efac } C = \text{add-mset (Pos } A) \{\#L \in\# C. L \neq \text{Pos } A\# \}$   
**using** *efac-spec* **by** *metis*

**assume** *hyp*:  $\bigwedge L. \text{is-pos } L \implies \text{linorder-lit.is-greatest-in-mset (efac } C) L \implies$   
*thesis*  
**show** *thesis*  
**proof** (*rule hyp*)  
**show** *is-pos* (Pos A)  
**by** *simp*  
**next**  
**show**  $\text{linorder-lit.is-greatest-in-mset(efac } C) \text{ (Pos } A)$   
**unfolding** *efac-C-eq*  $\text{linorder-lit.is-greatest-in-mset-iff}$   
**using** *Pos-A-maximal*[*unfolded linorder-lit.is-maximal-in-mset-iff*]  
**by** *auto*  
**qed**  
**qed**

**lemma** *mempty-in-image-efac-iff*[*simp*]:  $\{\#\} \in \text{efac } N \iff \{\#\} \in N$   
**by** (*metis empty-iff imageE image-eqI linorder-lit.is-maximal-in-mset-iff set-mset-empty set-mset-efac efac-spec*)

**lemma** *greatest-literal-in-efacI*:  
**assumes** *is-pos* L **and** *C-max-lit*:  $\text{linorder-lit.is-maximal-in-mset } C L$   
**shows**  $\text{linorder-lit.is-greatest-in-mset (efac } C) L$   
**unfolding** *efac-spec-if-pos-lit-is-maximal*[*OF assms*]  $\text{linorder-lit.is-greatest-in-mset-iff}$   
**proof** (*intro conjI ballI*)  
**show**  $L \in\# \text{add-mset } L \{\#K \in\# C. K \neq L\# \}$   
**by** *simp*  
**next**  
**fix**  $y :: 'f \text{ gterm literal}$   
**assume**  $y \in\# \text{remove1-mset } L (\text{add-mset } L \{\#K \in\# C. K \neq L\# \})$   
**then show**  $y \prec_l L$   
**using** *C-max-lit*[*unfolded linorder-lit.is-maximal-in-mset-iff*]  
**by** *auto*  
**qed**

**lemma**  $\text{linorder-lit.is-maximal-in-mset (efac } C) L \iff \text{linorder-lit.is-maximal-in-mset } C L$   
**by** (*simp add: linorder-lit.is-maximal-in-mset-iff*)

**lemma**  
**assumes** *is-pos* L  
**shows**  $\text{linorder-lit.is-greatest-in-mset (efac } C) L \iff \text{linorder-lit.is-maximal-in-mset } C L$   
**by** (*metis (no-types, opaque-lifting) Uniq-D assms efac-spec greatest-literal-in-efacI linorder-lit.Uniq-is-greatest-in-mset linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)

*literal.disc(1))*

**lemma** *factorizable-if-neq-efac*:

**assumes**  $C \neq \text{efac } C$

**shows**  $\exists C'. \text{ord-res.ground-factoring } C C'$

**using** *assms*

**by** (*metis converse-rtranclpE ex1-efac-eq-factoring-chain*)

**lemma** *nex-strictly-maximal-pos-lit-if-neq-efac*:

**assumes**  $C \neq \text{efac } C$

**shows**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L C$

**using** *assms factorizable-if-neq-efac nex-strictly-maximal-pos-lit-if-factorizable* **by** *metis*

**lemma** *efac-properties-if-not-ident*:

**assumes**  $\text{efac } C \neq C$

**shows**  $\text{efac } C \prec_c C$  **and**  $\{\text{efac } C\} \models_e \{C\}$

**proof** –

**have**  $\text{efac } C \subseteq\# C$

**using** *efac-subset* .

**hence**  $\text{efac } C \preceq_c C$

**using** *subset-implies-reflclp-multp* **by** *blast*

**thus**  $\text{efac } C \prec_c C$

**using**  $\langle \text{efac } C \neq C \rangle$  **by** *order*

**show**  $\{\text{efac } C\} \models_e \{C\}$

**using**  $\langle \text{efac } C \subseteq\# C \rangle$  *true-cls-subclause* **by** *metis*

**qed**

**end**

**end**

**theory** *ORD-RES-2*

**imports**

*ORD-RES*

*Exhaustive-Factorization*

**begin**

## 15 ORD-RES-2 (full factorization)

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-2* **where**

*factoring*:

*is-least-false-clause*  $(N \mid \cup \mid U_r \mid \cup \mid U_{ef}) C \implies$

*linorder-lit.is-maximal-in-mset*  $C L \implies$

*is-pos*  $L \implies$

$U_{ef}' = \text{finsert } (\text{efac } C) U_{ef} \implies$

*ord-res-2*  $N (U_r, U_{ef}) (U_r, U_{ef}') \mid$

*resolution:*

*is-least-false-clause*  $(N \mid \cup \mid U_r \mid \cup \mid U_{ef}) C \implies$   
*linorder-lit.is-maximal-in-mset*  $C L \implies$   
*is-neg*  $L \implies$   
 $D \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \implies$   
 $D \prec_c C \implies$   
*ord-res.production*  $(fset (N \mid \cup \mid U_r \mid \cup \mid U_{ef})) D = \{atm-of L\} \implies$   
*ord-res.ground-resolution*  $C D CD \implies$   
 $U_r' = finsert CD U_r \implies$   
*ord-res-2*  $N (U_r, U_{ef}) (U_r', U_{ef})$

**inductive ord-res-2-step where**

*ord-res-2*  $N S S' \implies$  *ord-res-2-step*  $(N, S) (N, S')$

**inductive ord-res-2-final where**

*ord-res-final*  $(N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \implies$  *ord-res-2-final*  $(N, (U_r, U_{ef}))$

**inductive ord-res-2-load where**

$N \neq \{\mid\} \implies$  *ord-res-2-load*  $N (N, (\{\mid\}, \{\mid\}))$

**sublocale ord-res-2-semantic: semantics where**

*step* = *ord-res-2-step* **and**

*final* = *ord-res-2-final*

**proof** *unfold-locales*

**fix**  $S :: 'f gterm clause fset \times 'f gterm clause fset \times 'f gterm clause fset$

**obtain**  $N U_r U_{ef} :: 'f gterm clause fset$  **where**

*S-def*:  $S = (N, (U_r, U_{ef}))$

**by** (*metis prod.exhaust*)

**assume** *ord-res-2-final*  $S$

**hence**  $\{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \vee \neg$  *ex-false-clause*  $(fset (N \mid \cup \mid U_r \mid \cup \mid U_{ef}))$

**by** (*simp add: S-def ord-res-2-final.simps ord-res-final-def*)

**thus** *finished* *ord-res-2-step*  $S$

**proof** (*elim disjE*)

**assume**  $\{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}$

**have** *False* **if** *ord-res-2*  $N (U_r, U_{ef}) x$  **for**  $x$

**using** *that[unfolded S-def]*

**proof** (*cases*  $N (U_r, U_{ef}) x$  *rule: ord-res-2.cases*)

**case** *hyps*: (*factoring*  $C L U_{ef}'$ )

**from** *hyps* **have**  $C = \{\#\}$

**using** *is-least-false-clause-empty[OF  $\langle \{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \rangle$ ]*

**by** (*metis Uniq-D Uniq-is-least-false-clause*)

**moreover** **from** *hyps* **have**  $L \in \# C$

**using** *linorder-lit.is-maximal-in-mset-iff* **by** *blast*

**ultimately** **show** *False*

**by** *simp*

**next**

```

    case hyps: (resolution C L D CD Uef')
  from hyps ⟨{#} |∈| N |∪| Ur |∪| Uef⟩ have C = {#}
    using is-least-false-clause-mempty[OF ⟨{#} |∈| N |∪| Ur |∪| Uef⟩]
    by (metis Uniq-D Uniq-is-least-false-clause)
  moreover from hyps have L ∈# C
    using linorder-lit.is-maximal-in-mset-iff by blast
  ultimately show False
    by simp
qed
thus finished ord-res-2-step S
  unfolding finished-def ord-res-2-step.simps S-def
  by (metis prod.inject)
next
assume no-false-cls: ¬ ex-false-clause (fset (N |∪| Ur |∪| Uef))
have False if ord-res-2 N (Ur, Uef) x for x
  using that[unfolded S-def]
proof (cases N (Ur, Uef) x rule: ord-res-2.cases)
  case hyps: (factoring C L Uef')
  thus False
    using no-false-cls[unfolded ex-false-clause-def]
    using is-least-false-clause-def linorder-cls.is-least-in-fset-iff by auto
next
  case hyps: (resolution C L D CD Uef')
  thus False
    using no-false-cls[unfolded ex-false-clause-def]
    using is-least-false-clause-def linorder-cls.is-least-in-fset-iff by auto
qed
thus finished ord-res-2-step S
  unfolding finished-def ord-res-2-step.simps S-def
  by (metis prod.inject)
qed
qed

sublocale ord-res-2-language: language where
  step = ord-res-2-step and
  final = ord-res-2-final and
  load = ord-res-2-load
  by unfold-locales

lemma is-least-in-fset-with-irrelevant-clauses-if-is-least-in-fset:
  assumes
    irrelevant: ∀ D |∈| N'. ∃ E |∈| N. E ⊂# D ∧ set-mset D = set-mset E and
    C-least: linorder-cls.is-least-in-fset {C |∈| N. ¬ ord-res-Interp (fset N) C ⊨
C|} C
  shows linorder-cls.is-least-in-fset {C |∈| N |∪| N'. ¬ ord-res-Interp (fset (N |∪|
N')) C ⊨ C|} C
proof -
  have
    C-in: C |∈| N and

```

*C-not-entailed*:  $\neg \text{ord-res-Interp } (fset\ N)\ C \models C$  **and**  
*C-lt*:  $\forall x \mid \in \mid N. x \neq C \longrightarrow \neg \text{ord-res-Interp } (fset\ N)\ x \models x \longrightarrow C \prec_c x$   
**using** *C-least linorder-cls.is-least-in-ffilter-iff* **by** *simp-all*

**have**  $C \mid \in \mid N \mid \cup \mid N'$   
**using** *C-in* **by** *simp*

**moreover have**  $\neg \text{ord-res-Interp } (fset\ (N \mid \cup \mid N'))\ C \models C$   
**using** *extended-partial-model-entails-iff-partial-model-entails*[  
*of fset N fset N', OF finite-fset finite-fset irrelevant*]  
**using** *C-in C-not-entailed*  
**by** *simp*

**moreover have**  $C \prec_c x$

**if**

*x-in*:  $x \mid \in \mid N \mid \cup \mid N'$  **and**

*x-neg*:  $x \neq C$  **and**

*x-not-entailed*:  $\neg \text{ord-res-Interp } (fset\ (N \mid \cup \mid N'))\ x \models x$

**for**  $x$

**proof** –

**from** *x-in* **have**  $x \mid \in \mid N \vee x \mid \in \mid N'$

**by** *simp*

**thus**  $C \prec_c x$

**proof** (*elim disjE*)

**assume** *x-in*:  $x \mid \in \mid N$

**moreover have**  $\neg \text{ord-res-Interp } (fset\ N)\ x \models x$

**using** *extended-partial-model-entails-iff-partial-model-entails*[

*of fset N fset N', OF finite-fset finite-fset irrelevant x-in*]

**using** *x-not-entailed* **by** *simp*

**ultimately show**  $C \prec_c x$

**using** *C-lt*[*rule-format, of x*] *x-neg* **by** *argo*

**next**

**assume**  $x \mid \in \mid N'$

**then obtain**  $x'$  **where**  $x' \mid \in \mid N$  **and**  $x' \subset \# x$  *set-mset*  $x' = \text{set-mset } x$

**using** *irrelevant* **by** *metis*

**have**  $x' \prec_c x$

**using**  $\langle x' \subset \# x \rangle$  **by** (*metis strict-subset-implies-multip*)

**moreover have**  $C \preceq_c x'$

**proof** (*cases*  $x' = C$ )

**case** *True*

**thus** *?thesis*

**by** *order*

**next**

**case** *False*



```

have C <_c x'
proof (rule C-lt[rule-format])
  show x' |∈| N
    using ⟨x' |∈| N⟩ .
next
show x' ≠ C
  using False .
next
have ¬ ord-res-Interp (fset (N |∪| N')) x ⊨ x'
  using x-not-entailed ⟨set-mset x' = set-mset x⟩
  by (metis true-cls-def)
hence ¬ ord-res-Interp (fset (N |∪| N')) x' ⊨ x'
  by (metis ⟨x' <_c x⟩ ord-res.entailed-clause-stays-entailed)
thus ¬ ord-res-Interp (fset N) x' ⊨ x'
  using extended-partial-model-entails-iff-partial-model-entails[
    of fset N fset N' x', OF finite-fset finite-fset irrelevant]
  using ⟨x' |∈| N⟩ by simp
qed
thus ?thesis
  by order
qed

ultimately show C <_c x
  by order
qed

ultimately show linorder-cls.is-least-in-fset {C |∈| N |∪| N'}
  ¬ ord-res-Interp (fset (N |∪| N')) C ⊨ C } C
  using C-in C-not-entailed
  unfolding linorder-cls.is-least-in-filter-iff by metis
qed

primrec fset-upto :: nat ⇒ nat ⇒ nat fset where
  fset-upto i 0 = (if i = 0 then {0} else {}) |
  fset-upto i (Suc n) = (if i ≤ Suc n then finsert (Suc n) (fset-upto i n) else {})

lemma
  fset-upto 0 0 = {0}
  fset-upto 0 1 = {0, 1}
  fset-upto 0 2 = {0, 1, 2}
  fset-upto 0 3 = {0, 1, 2, 3}
  fset-upto 1 3 = {1, 2, 3}
  fset-upto 2 3 = {2, 3}
  fset-upto 3 3 = {3}
  fset-upto 4 3 = {}
  unfolding numeral-2-eq-2 numeral-3-eq-3
  by auto

```

```

lemma  $i \leq 1 + j \implies \text{List.upto } i (1 + j) = \text{List.upto } i j @ [1 + j]$ 
  using upto-rec2 by simp

lemma fset-of-append-singleton:  $\text{fset-of-list } (xs @ [x]) = \text{finsert } x (\text{fset-of-list } xs)$ 
  by simp

lemma fset-of-list (List.upto (int i) (int j)) = int |'| fset-upto i j
proof (induction j)
  case 0
  show ?case
    by simp
next
  case (Suc j)
  show ?case
  proof (cases  $i \leq \text{Suc } j$ )
    case True
    hence AAA:  $\text{int } i \leq 1 + \text{int } j$ 
      by presburger

    from True show ?thesis
      apply simp
      unfolding Suc.IH[symmetric]
      unfolding upto-rec2[OF AAA] fset-of-append-singleton
      by simp
    next
    case False
    thus ?thesis
      by simp
  qed
qed

lemma fset-fset-upto[simp]:  $\text{fset } (\text{fset-upto } m n) = \{m..n\}$ 
  apply (induction n)
  apply simp
  apply simp
  using atLeastAtMostSuc-conv by presburger

lemma minus-mset-replicate-strict-subset-minus-msetI:
  assumes  $m < n$  and  $n < \text{count } C L$ 
  shows  $C - \text{replicate-mset } n L \subset\# C - \text{replicate-mset } m L$ 
proof -
  from  $\langle m < n \rangle$  obtain k1 where n-def:  $n = m + \text{Suc } k1$ 
    using less-natE by auto

  with  $\langle n < \text{count } C L \rangle$  obtain k2 where
    count-eq:  $\text{count } C L = m + \text{Suc } k1 + \text{Suc } k2$ 
    by (metis add.commute add-Suc group-cancel.add1 less-natE)

```

**define**  $C_0$  **where**  
 $C_0 = \{\#K \in\# C. K \neq L\# \}$

**have**  $C$ -eq:  $C = C_0 + \text{replicate-mset } m \ L + \text{replicate-mset } (\text{Suc } k1) \ L + \text{replicate-mset } (\text{Suc } k2) \ L$   
**using**  $C_0$ -def count-eq  
**by** (metis (mono-tags, lifting) filter-mset-eq group-cancel.add1 replicate-mset-plus union-filter-mset-complement)

**have**  $C - \text{replicate-mset } n \ L = C_0 + \text{replicate-mset } (\text{Suc } k2) \ L$   
**unfolding**  $C$ -eq  $n$ -def  
**by** (simp add: replicate-mset-plus)

**also have**  $\dots \subset\# C_0 + \text{replicate-mset } (\text{Suc } k1) \ L + \text{replicate-mset } (\text{Suc } k2) \ L$   
**by** simp

**also have**  $\dots = C - \text{replicate-mset } m \ L$   
**unfolding**  $C$ -eq  
**by** (simp add: replicate-mset-plus)

**finally show** ?thesis .

**qed**

**lemma** factoring-all-is-between-efac-and-original-clause:

**fixes**  $z$   
**assumes**  
 $\text{is-pos } L$  **and**  $\text{ord-res.is-maximal-lit } L \ C$  **and**  $\text{count } C \ L = \text{Suc } (\text{Suc } n)$   
 $m' \leq n$  **and**  
 $z$ -in:  $z \in | \ (\lambda i. C - \text{replicate-mset } i \ L) \ |^{\uparrow} \ \text{fset-upto } 0 \ m'$

**shows**  $\text{efac } C \subset\# z$  **and**  $z \subseteq\# C$

**proof** –

**from**  $z$ -in **obtain**  $i$  **where**  
 $i \leq m'$  **and**  
 $z$ -def:  $z = C - \text{replicate-mset } i \ L$   
**by** auto

**have**  $i \leq n$   
**using**  $\langle i \leq m' \rangle \ \langle m' \leq n \rangle$  **by** presburger

**hence**  $i < \text{count } C \ L$   
**using**  $\langle \text{count } C \ L = \text{Suc } (\text{Suc } n) \rangle$  **by** presburger

**thus**  $z \subseteq\# C$   
**unfolding**  $z$ -def **by** simp

**show**  $\text{efac } C \subset\# z$

**proof** –

**have**  $\text{efac } C = \text{add-mset } L \ \{\#K \in\# C. K \neq L\# \}$   
**using**  $\text{efac-spec-if-pos-lit-is-maximal}[\text{OF } \langle \text{is-pos } L \rangle \ \langle \text{ord-res.is-maximal-lit } L \ C \rangle]$  .

**also have**  $\dots \subset\# \text{add-mset } L \ (\text{add-mset } L \ \{\#K \in\# C. K \neq L\# \})$   
**by** simp

**also have**  $\dots = C - \text{replicate-mset } n \ L$   
**using**  $\text{minus-mset-replicate-mset-eq-add-mset-add-mset-filter-mset}[\dots]$

```

      OF ⟨count C L = Suc (Suc n)⟩ ..
    also have ... ⊆# C - replicate-mset i L
      using ⟨i ≤ n⟩ by (simp add: subseteq-mset-def)
    also have ... = z
      using z-def ..
    finally show ?thesis .
  qed
qed

lemma
  assumes
    linorder-cls.is-least-in-fset {x | ∈| N1. P N1 x} x and
    linorder-cls.is-least-in-fset N2 y and
    ∀ z | ∈| N2. z ≼c x and
    P (N1 |∪| N2) y and
    ∀ z | ∈| N1. z ≺c x ⟶ ¬ P (N1 |∪| N2) z
  shows linorder-cls.is-least-in-fset {x | ∈| N1 |∪| N2. P (N1 |∪| N2) x} y
proof -
  show ?thesis
    unfolding linorder-cls.is-least-in-filter-iff
  proof (intro conjI ballI impI)
    from assms(2) show y | ∈| N1 |∪| N2
      unfolding linorder-cls.is-least-in-fset-iff by simp
  next
    from assms(4) show P (N1 |∪| N2) y
      by argo
  next
    fix z
    assume z-in: z | ∈| N1 |∪| N2 and z ≠ y and P (N1 |∪| N2) z
    show y ≺c z
      using z-in[unfolded union-iff]
    proof (elim disjE)
      from assms(2,3,5) show z | ∈| N1 ⟹ y ≺c z
      by (metis ⟨P (N1 |∪| N2) z⟩ ⟨z ≠ y⟩ linorder-cls.dual-order.not-eq-order-implies-strict
        linorder-cls.is-least-in-fset-iff linorder-cls.less-linear
        linorder-cls.order.strict-trans)
    next
      from assms(2) show z | ∈| N2 ⟹ y ≺c z
      using ⟨z ≠ y⟩ linorder-cls.is-least-in-fset-iff by blast
    qed
  qed
qed
qed

lemma ground-factoring-preserves-efac:
  assumes ord-res.ground-factoring P C
  shows efac P = efac C
  using assms
  by (smt (verit, ccfv-threshold) filter-mset-add-mset is-pos-def ord-res.ground-factoring.cases
    ord-res.ground-factoring-preserves-maximal-literal efac-spec-if-pos-lit-is-maximal)

```

**lemma** *ground-factorings-preserves-efac*:  
**assumes** *ord-res.ground-factoring\*\* P C*  
**shows** *efac P = efac C*  
**using** *assms*  
**by** (*induction P rule: converse-rtranclp-induct*)  
(*simp-all add: ground-factoring-preserves-efac*)

**lemma** *ex-ord-res-2-if-not-final*:  
**assumes**  $\neg$  *ord-res-2-final S*  
**shows**  $\exists S'$ . *ord-res-2-step S S'*

**proof** –

**from** *assms* **obtain**  $N U_r U_{ef}$  **where**  
*S-def*:  $S = (N, (U_r, U_{ef}))$  **and**  
 $\{\#\} \notin N \cup U_r \cup U_{ef}$  **and**  
*ex-false-clause* (*fset* ( $N \cup U_r \cup U_{ef}$ ))  
**by** (*metis ord-res-2-final.intros ord-res-final-def surj-pair*)

**obtain**  $C$  **where** *C-least-false: is-least-false-clause* ( $N \cup U_r \cup U_{ef}$ )  $C$   
**using**  $\langle$ *ex-false-clause* (*fset* ( $N \cup U_r \cup U_{ef}$ )) $\rangle$  *obtains-least-false-clause-if-ex-false-clause*  
**by** *metis*

**hence**  $C \neq \{\#\}$   
**using**  $\langle$  $\{\#\} \notin N \cup U_r \cup U_{ef}$  $\rangle$   
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
**by** *metis*

**then obtain**  $L$  **where** *C-max: linorder-lit.is-maximal-in-mset*  $C L$   
**using** *linorder-lit.ex-maximal-in-mset* **by** *metis*

**show** *?thesis*

**proof** (*cases L*)

**case** (*Pos A*)

**thus** *?thesis*

**using** *ord-res-2.factoring[OF C-least-false C-max]* *S-def is-pos-def*

**by** (*metis ord-res-2-step.intros*)

**next**

**case** (*Neg A*)

**then obtain**  $D$  **where**

$D \in N \cup U_r \cup U_{ef}$  **and**

$D \prec_c C$  **and**

*ord-res.is-strictly-maximal-lit* (*Pos A*)  $D$  **and**

*ord-res.production* (*fset* ( $N \cup U_r \cup U_{ef}$ ))  $D = \{A\}$

**using** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*

**using** *C-least-false C-max* **by** *metis*

**moreover then obtain**  $CD$  **where**

*ord-res.ground-resolution*  $C D CD$

**using** *ex-ground-resolutionI C-max Neg* **by** *metis*

**ultimately show** *?thesis*  
**using** *ord-res-2.resolution[OF C-least-false C-max]*  
**by** (*metis Neg S-def literal.disc(2) literal.sel(2) ord-res-2-step.intros*)

**qed**  
**qed**

**corollary** *ord-res-2-step-safe: ord-res-2-final S  $\vee$  ( $\exists S'$ . ord-res-2-step S S')*  
**using** *ex-ord-res-2-if-not-final* **by** *metis*

**lemma** *is-least-false-clause-if-is-least-false-clause-in-union-unproductive:*  
**assumes**  
*N2-unproductive:  $\forall C \in N2$ . ord-res.production (fset (N1  $\cup$  N2)) C = {}*  
**and**  
*C-in: C  $\in$  N1* **and**  
*C-least-false: is-least-false-clause (N1  $\cup$  N2) C*  
**shows** *is-least-false-clause N1 C*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-filter-iff*  
**proof** (*intro conjI ballI impI*)  
**show** *C  $\in$  N1*  
**using** *C-in* .

**next**  
**have**  $\neg$  *ord-res-Interp (fset (N1  $\cup$  N2)) C  $\models$  C*  
**using** *C-least-false[unfolded is-least-false-clause-def linorder-cls.is-least-in-filter-iff]*  
**by** *argo*  
**thus**  $\neg$  *ord-res.interp (fset N1) C  $\cup$  ord-res.production (fset N1) C  $\models$  C*  
**unfolding** *Interp-union-unproductive[of fset N1 fset N2, folded union-fset,*  
*OF finite-fset finite-fset N2-unproductive]* .

**next**  
**fix** *D*  
**have**  $\forall D \in N1 \cup N2$ . *D  $\neq$  C  $\longrightarrow$   $\neg$  ord-res-Interp (fset (N1  $\cup$  N2)) D  $\models$*   
*D  $\longrightarrow$  C  $\prec_c$  D*  
**using** *C-least-false[unfolded is-least-false-clause-def linorder-cls.is-least-in-filter-iff]*  
**by** *argo*

**moreover assume** *D  $\in$  N1* **and** *D  $\neq$  C* **and**  $\neg$  *ord-res-Interp (fset N1) D  $\models$*   
*D*

**ultimately show** *C  $\prec_c$  D*  
**using** *Interp-union-unproductive[of fset N1 fset N2, folded union-fset,*  
*OF finite-fset finite-fset N2-unproductive]*  
**by** *simp*

**qed**

**lemma** *ground-factoring-replicate-max-pos-lit:*  
*ord-res.ground-factoring*  
*(C<sub>0</sub> + replicate-mset (Suc (Suc n)) (Pos A))*  
*(C<sub>0</sub> + replicate-mset (Suc n) (Pos A))*

```

    if ord-res.is-maximal-lit (Pos A) (C0 + replicate-mset (Suc (Suc n)) (Pos A))
  for A C0 n
proof (rule ord-res.ground-factoringI)
  show C0 + replicate-mset (Suc (Suc n)) (Pos A) =
    add-mset (Pos A) (add-mset (Pos A) (C0 + replicate-mset n (Pos A)))
    by simp
next
  show ord-res.is-maximal-lit (Pos A) (C0 + replicate-mset (Suc (Suc n)) (Pos
A))
    using that .
next
  show C0 + replicate-mset (Suc n) (Pos A) =
    add-mset (Pos A) (C0 + replicate-mset n (Pos A))
    by simp
qed simp

```

**lemma** *ground-factorings-replicate-max-pos-lit:*

```

assumes
  ord-res.is-maximal-lit (Pos A) (C0 + replicate-mset (Suc (Suc n)) (Pos A))
shows m ≤ Suc n ⇒ (ord-res.ground-factoring  $\overset{\sim}{\sim}$  m)
  (C0 + replicate-mset (Suc (Suc n)) (Pos A))
  (C0 + replicate-mset (Suc (Suc n - m)) (Pos A))
proof (induction m)
  case 0
  show ?case
    by simp
next
  case (Suc m')
  then show ?case
    apply (cases m')
    using assms ground-factoring-replicate-max-pos-lit apply auto[1]
    by (metis (no-types, lifting) Suc-diff-le Suc-leD assms diff-Suc-Suc
ground-factoring-replicate-max-pos-lit ord-res.ground-factorings-preserves-maximal-literal
relpowp-Suc-I relpowp-imp-rtranclp)
qed

```

**lemma** *ord-res-Interp-entails-if-greatest-lit-is-pos:*

```

assumes C-in: C ∈ N and L-greatest: linorder-lit.is-greatest-in-mset C L and
L-pos: is-pos L
shows ord-res-Interp N C ⊨ C
proof (cases ord-res.interp N C ⊨ C)
  case True
  hence ord-res.production N C = {}
    by (simp add: ord-res.production-unfold)
  with True show ?thesis
    by simp
next
  case False

```

```

from L-pos obtain A where L-def:  $L = Pos\ A$ 
  by (cases L) simp-all

from L-greatest obtain C' where C-def:  $C = add\_mset\ L\ C'$ 
  unfolding linorder-lit.is-greatest-in-mset-iff
  by (metis insert-DiffM)

with C-in L-greatest have  $A \in ord\_res.production\ N\ C$ 
  unfolding L-def ord-res.production-unfold
  using False
  by (simp add: linorder-lit.is-greatest-in-mset-iff multi-member-split)
thus ?thesis
  by (simp add: true-cls-def C-def L-def)
qed

lemma right-unique-ord-res-2: right-unique (ord-res-2 N)
proof (rule right-uniqueI)
  fix  $s\ s'\ s'' :: 'f\ gterm\ clause\ fset \times 'f\ gterm\ clause\ fset$ 
  assume step1: ord-res-2 N s s' and step2: ord-res-2 N s s''
  show  $s' = s''$ 
    using step1
  proof (cases N s s' rule: ord-res-2.cases)
    case hyp1: (factoring Ur1 Uef1 C1 L1 Uef'1)
      show ?thesis
        using step2
    proof (cases N s s'' rule: ord-res-2.cases)
      case (factoring Ur2 Uef2 C2 L2 Uef'2)
        with hyp1 show ?thesis
          by (metis Uniq-D Uniq-is-least-false-clause prod.inject)
    next
      case (resolution Ur Uef C L D CD Ur')
        with hyp1 have False
        by (metis Pair-inject Uniq-is-least-false-clause linorder-lit.Uniq-is-maximal-in-mset
          the1-equality')
        thus ?thesis ..
    qed
  next
    case hyp1: (resolution Ur1 Uef1 C1 L1 D1 CD1 Ur'1)
      show ?thesis
        using step2
    proof (cases N s s'' rule: ord-res-2.cases)
      case (factoring Ur2 Uef2 C2 L2 Uef'2)
        with hyp1 have False
        by (metis Pair-inject Uniq-is-least-false-clause linorder-lit.Uniq-is-maximal-in-mset
          the1-equality')
        thus ?thesis ..
    next
      case (resolution Ur Uef C L D CD Ur')
        with hyp1 show ?thesis

```



```

    by (metis (mono-tags, lifting) Uniq-is-least-false-clause
        linorder-lit.Uniq-is-maximal-in-mset ord-res.Uniq-production-eq-singleton
        ord-res.unique-ground-resolution prod.inject the1-equality')
  qed
qed
qed

lemma right-unique-ord-res-2-step: right-unique ord-res-2-step
proof (rule right-uniqueI)
  fix x y z
  show ord-res-2-step x y  $\implies$  ord-res-2-step x z  $\implies$  y = z
    apply (cases x; cases y; cases z)
    apply (simp add: ord-res-2-step.simps)
    using right-unique-ord-res-2[THEN right-uniqueD]
    by blast
qed

end

end
theory Exhaustive-Resolution
  imports Background
begin

```

## 16 Function for full resolution

```

context simulation-SCLFOL-ground-ordered-resolution begin

definition ground-resolution where
  ground-resolution D C CD = ord-res.ground-resolution C D CD

lemma Uniq-ground-resolution:  $\exists_{\leq 1} DC$ . ground-resolution D C DC
  by (simp add: ground-resolution-def ord-res.unique-ground-resolution)

lemma ground-resolution-terminates: wfp (ground-resolution D)-1-1
proof (rule wfp-if-convertible-to-wfp)
  show wfp ( $\prec_c$ )
    using ord-res.wfp-less-cls .
next
  show  $\bigwedge x y$ . (ground-resolution D)-1-1 x y  $\implies$  x  $\prec_c$  y
    unfolding ground-resolution-def conversep-iff
    using ord-res.ground-resolution-smaller-conclusion by metis
qed

lemma not-ground-resolution-mempty-left:  $\neg$  ground-resolution {#} C x
  by (auto simp: ground-resolution-def elim: ord-res.ground-resolution.cases)

lemma not-ground-resolution-mempty-right:  $\neg$  ground-resolution C {#} x
  by (auto simp: ground-resolution-def elim: ord-res.ground-resolution.cases)

```

**lemma** *not-tranclp-ground-resolution-mempty-left*:  $\neg$  (*ground-resolution*  $\{\#\}$ )<sup>++</sup>  
 $C\ x$

**by** (*metis not-ground-resolution-mempty-left tranclpD*)

**lemma** *not-tranclp-ground-resolution-mempty-right*:  $\neg$  (*ground-resolution*  $C$ )<sup>++</sup>  $\{\#\}$   
 $x$

**by** (*metis not-ground-resolution-mempty-right tranclpD*)

**lemma** *left-premise-lt-right-premise-if-ground-resolution*:

*ground-resolution*  $D\ C\ DC \implies D \prec_c C$

**by** (*auto simp: ground-resolution-def elim: ord-res.ground-resolution.cases*)

**lemma** *left-premise-lt-right-premise-if-tranclp-ground-resolution*:

(*ground-resolution*  $D$ )<sup>++</sup>  $C\ DC \implies D \prec_c C$

**by** (*induction DC rule: tranclp-induct*)

(*auto simp add: left-premise-lt-right-premise-if-ground-resolution*)

**lemma** *resolvent-lt-right-premise-if-ground-resolution*:

*ground-resolution*  $D\ C\ DC \implies DC \prec_c C$

**by** (*simp add: ground-resolution-def ord-res.ground-resolution-smaller-conclusion*)

**lemma** *resolvent-lt-right-premise-if-tranclp-ground-resolution*:

(*ground-resolution*  $D$ )<sup>++</sup>  $C\ DC \implies DC \prec_c C$

**proof** (*induction DC rule: tranclp-induct*)

**case** (*base y*)

**thus** *?case*

**by** (*simp add: resolvent-lt-right-premise-if-ground-resolution*)

**next**

**case** (*step y z*)

**have**  $z \prec_c y$

**using** *step.hyps resolvent-lt-right-premise-if-ground-resolution by metis*

**thus** *?case*

**using** *step.IH by order*

**qed**

Exhaustive resolution

**definition** *eres where*

*eres*  $D\ C = (\text{THE } DC. \text{full-run } (\text{ground-resolution } D)\ C\ DC)$

The function *eres* performs exhaustive resolution between its two input clauses. The first clause is repeatedly used, while the second clause is only use to start the resolution chain.

**lemma** *eres-ident-iff*:  $\text{eres } D\ C = C \iff (\exists DC. \text{ground-resolution } D\ C\ DC)$

**proof** (*rule iffI*)

**assume**  $\text{eres } D\ C = C$

**thus**  $\exists DC. \text{ground-resolution } D\ C\ DC$

**unfolding** *eres-def*

**by** (*metis Uniq-full-run Uniq-ground-resolution full-run-def ground-resolution-terminates*)

$ex1\text{-full-run } the1\text{-equality}'$   
**next**  
**assume**  $stuck: \nexists DC. \text{ground-resolution } D \ C \ DC$   
**have**  $(\text{ground-resolution } D)^{**} \ C \ C$   
**by**  $auto$   
  
**with**  $stuck$  **have**  $\text{full-run } (\text{ground-resolution } D) \ C \ C$   
**unfolding**  $\text{full-run-def}$  **by**  $argo$   
  
**moreover** **have**  $Uniq: \exists_{\leq 1} y. \text{full-run } (\text{ground-resolution } D) \ C \ y$   
**by**  $(metis \ Uniq\text{-ground-resolution } \ Uniq\text{-full-run})$   
  
**ultimately** **show**  $eres \ D \ C = C$   
**unfolding**  $eres\text{-def}$  **by**  $(metis \ the1\text{-equality}'$   
**qed**

**lemma**  
**assumes**  
 $step1: \text{ground-resolution } D \ C \ DC$  **and**  
 $stuck: \nexists DDC. \text{ground-resolution } D \ DC \ DDC$   
**shows**  $eres \ D \ C = DC$   
**proof** –  
**from**  $step1$  **have**  $(\text{ground-resolution } D)^{**} \ C \ DC$   
**by**  $auto$   
  
**with**  $stuck$  **have**  $\text{full-run } (\text{ground-resolution } D) \ C \ DC$   
**unfolding**  $\text{full-run-def}$  **by**  $argo$   
  
**moreover** **have**  $Uniq: \exists_{\leq 1} y. \text{full-run } (\text{ground-resolution } D) \ C \ y$   
**by**  $(metis \ Uniq\text{-ground-resolution } \ Uniq\text{-full-run})$   
  
**ultimately** **show**  $?thesis$   
**unfolding**  $eres\text{-def}$  **by**  $(metis \ the1\text{-equality}'$   
**qed**

**lemma**  
**assumes**  
 $step1: \text{ground-resolution } D \ C \ DC$  **and**  
 $step2: \text{ground-resolution } D \ DC \ DDC$  **and**  
 $stuck: \nexists DDDC. \text{ground-resolution } D \ DDC \ DDDC$   
**shows**  $eres \ D \ C = DDC$   
**proof** –  
**from**  $step1$  **have**  $(\text{ground-resolution } D)^{**} \ C \ DC$   
**by**  $auto$   
  
**with**  $step2$  **have**  $(\text{ground-resolution } D)^{**} \ C \ DDC$   
**by**  $(metis \ rtranclp.\text{sims})$   
  
**with**  $stuck$  **have**  $\text{full-run } (\text{ground-resolution } D) \ C \ DDC$

**unfolding** *full-run-def* **by** *argo*

**moreover have** *Uniq:  $\exists_{\leq 1} y. \text{full-run (ground-resolution D) C y}$*   
**by** (*metis Uniq-ground-resolution Uniq-full-run*)

**ultimately show** *?thesis*  
**unfolding** *eres-def* **by** (*metis the1-equality'*)  
**qed**

**lemma**  
**assumes**  
*step1: ground-resolution D C DC and*  
*step2: ground-resolution D DC DDC and*  
*step3: ground-resolution D DDC DDDC and*  
*stuck:  $\nexists$  DDDDC. ground-resolution D DDDC DDDDC*  
**shows** *eres D C = DDDC*

**proof** –  
**from** *step1* **have** (*ground-resolution D*)<sup>\*\*</sup> *C DC*  
**by** *auto*

**with** *step2* **have** (*ground-resolution D*)<sup>\*\*</sup> *C DDC*  
**by** (*metis rtranclp.simps*)

**with** *step3* **have** (*ground-resolution D*)<sup>\*\*</sup> *C DDDC*  
**by** (*metis rtranclp.simps*)

**with** *stuck* **have** *full-run (ground-resolution D) C DDDC*  
**unfolding** *full-run-def* **by** *argo*

**moreover have** *Uniq:  $\exists_{\leq 1} y. \text{full-run (ground-resolution D) C y}$*   
**by** (*metis Uniq-ground-resolution Uniq-full-run*)

**ultimately show** *?thesis*  
**unfolding** *eres-def* **by** (*metis the1-equality'*)  
**qed**

**lemma** *eres-empty-left[simp]: eres {#} C = C*  
**unfolding** *eres-def*  
**by** (*metis Uniq-full-run Uniq-ground-resolution full-run-def not-ground-resolution-mempty-left rtranclp.rtrancl-refl the1-equality'*)

**lemma** *eres-empty-right[simp]: eres C {#} = {#}*  
**unfolding** *eres-def*  
**by** (*metis Uniq-full-run Uniq-ground-resolution full-run-def not-ground-resolution-mempty-right rtranclp.rtrancl-refl the1-equality'*)

**lemma** *ex1-eres-eq-full-run-ground-resolution:  $\exists! DC. \text{eres D C} = DC \wedge \text{full-run (ground-resolution D) C DC}$*   
**using** *ex1-full-run[of ground-resolution D C]*

by (metis Uniq-ground-resolution eres-def ground-resolution-terminates theI')

**lemma** *eres-le*:  $eres\ D\ C \preceq_c\ C$

**proof** –

**have** *full-run* (ground-resolution D) C (eres D C)

**using** *ex1-eres-eq-full-run-ground-resolution* **by** *metis*

**thus** *?thesis*

**proof** (rule *full-run-preserves-invariant*)

**show**  $C \preceq_c C$

**by** *simp*

**next**

**show**  $\bigwedge x\ y.\ ground\_resolution\ D\ x\ y \implies x \preceq_c C \implies y \preceq_c C$

**unfolding** *ground-resolution-def*

**using** *ord-res.ground-resolution-smaller-conclusion* **by** *fastforce*

**qed**

**qed**

**lemma** *clause-lt-clause-if-max-lit-comp*:

**assumes** *E-max-lit*: *linorder-lit.is-maximal-in-mset* E L **and** *L-neg*: *is-neg* L **and**

*D-max-lit*: *linorder-lit.is-maximal-in-mset* D (– L)

**shows**  $D \prec_c E$

**proof** –

**have**  $–\ L \prec_l L$

**using** *L-neg* **by** (cases L) *simp-all*

**thus** *?thesis*

**using** *D-max-lit* *E-max-lit*

**by** (metis *linorder-lit.multip-if-maximal-less-that-maximal*)

**qed**

**lemma** *eres-lt-if*:

**assumes** *E-max-lit*: *ord-res.is-maximal-lit* L E **and** *L-neg*: *is-neg* L **and**

*D-max-lit*: *linorder-lit.is-greatest-in-mset* D (– L)

**shows**  $eres\ D\ E \neq E$

**proof** –

**have**  $eres\ D\ E \neq E$

**unfolding** *eres-ident-iff not-not ground-resolution-def*

**proof** (rule *ex-ground-resolutionI*)

**show** *ord-res.is-maximal-lit* (Neg (atm-of L)) E

**using** *E-max-lit* *L-neg* **by** (cases L) *simp-all*

**next**

**show**  $D \prec_c E$

**using** *E-max-lit* *D-max-lit* *L-neg*

**by** (metis *clause-lt-clause-if-max-lit-comp*

*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)

**next**

**show** *ord-res.is-strictly-maximal-lit* (Pos (atm-of L)) D

**using** *D-max-lit*  $\langle is-neg\ L \rangle$

**by** (cases L) *simp-all*

**qed**

**thus**  $eres\ D\ E \prec_c\ E$   
**using**  $eres-le[of\ D\ E]$  **by**  $order$

**qed**

**lemma**  $eres-eq-after-ground-resolution$ :

**assumes**  $ground-resolution\ D\ C\ DC$

**shows**  $eres\ D\ C = eres\ D\ DC$

**using**  $assms$

**by** ( $metis\ (no-types,\ opaque-lifting)\ Uniq-def\ Uniq-full-run\ Uniq-ground-resolution$   
 $converse-rtranclpE\ ex1-eres-eq-full-run-ground-resolution\ full-run-def$ )

**lemma**  $eres-eq-after-rtranclp-ground-resolution$ :

**assumes**  $(ground-resolution\ D)^{**}\ C\ DC$

**shows**  $eres\ D\ C = eres\ D\ DC$

**using**  $assms$

**by** ( $induction\ DC\ rule:\ rtranclp-induct$ ) ( $simp-all\ add:\ eres-eq-after-ground-resolution$ )

**lemma**  $eres-eq-after-tranclp-ground-resolution$ :

**assumes**  $(ground-resolution\ D)^{++}\ C\ DC$

**shows**  $eres\ D\ C = eres\ D\ DC$

**using**  $assms$

**by** ( $induction\ DC\ rule:\ tranclp-induct$ ) ( $simp-all\ add:\ eres-eq-after-ground-resolution$ )

**lemma**  $resolvable-if-neq-eres$ :

**assumes**  $C \neq eres\ D\ C$

**shows**  $\exists!DC.\ ground-resolution\ D\ C\ DC$

**using**  $assms\ ex1-eres-eq-full-run-ground-resolution$

**by** ( $metis\ (no-types,\ opaque-lifting)\ Uniq-def\ Uniq-full-run\ Uniq-ground-resolution$   
 $full-run-def$

$rtranclp.rtrancl-refl$ )

**lemma**  $nex-maximal-pos-lit-if-resolvable$ :

**assumes**  $ground-resolution\ D\ C\ DC$

**shows**  $\nexists L.\ is-pos\ L \wedge ord-res.is-maximal-lit\ L\ C$

**using**  $assms\ unfolding\ ground-resolution-def$

**by** ( $metis\ Uniq-D\ empty-iff\ is-pos-def\ linorder-lit.\ Uniq-is-maximal-in-mset$   
 $literal.simps(4)\ ord-res.ground-resolution.cases\ set-mset-empty$ )

**corollary**  $nex-strictly-maximal-pos-lit-if-resolvable$ :

**assumes**  $ground-resolution\ D\ C\ DC$

**shows**  $\nexists L.\ is-pos\ L \wedge ord-res.is-strictly-maximal-lit\ L\ C$

**using**  $assms\ nex-maximal-pos-lit-if-resolvable$  **by**  $blast$

**corollary**  $nex-maximal-pos-lit-if-neq-eres$ :

**assumes**  $C \neq eres\ D\ C$

**shows**  $\nexists L.\ is-pos\ L \wedge ord-res.is-maximal-lit\ L\ C$

**using**  $assms\ resolvable-if-neq-eres\ nex-maximal-pos-lit-if-resolvable$  **by**  $metis$

**corollary** *nex-strictly-maximal-pos-lit-if-neq-eres*:

**assumes**  $C \neq \text{eres } D \ C$

**shows**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C$

**using** *assms resolvable-if-neq-eres nex-strictly-maximal-pos-lit-if-resolvable* **by** *metis*

**lemma** *ground-resolutionD*:

**assumes** *ground-resolution*  $D \ C \ DC$

**shows**  $\exists m \ A \ D' \ C'$ .

*linorder-lit.is-greatest-in-mset*  $D \ (\text{Pos } A) \wedge$

*linorder-lit.is-maximal-in-mset*  $C \ (\text{Neg } A) \wedge$

$D = \text{add-mset } (\text{Pos } A) \ D' \wedge$

$C = \text{replicate-mset } (\text{Suc } m) \ (\text{Neg } A) + C' \wedge \text{Neg } A \notin\# C' \wedge$

$DC = D' + \text{replicate-mset } m \ (\text{Neg } A) + C'$

**using** *assms*

**unfolding** *ground-resolution-def*

**proof** (*cases*  $C \ D \ DC$  *rule: ord-res.ground-resolution.cases*)

**case** (*ground-resolutionI*  $A \ C' \ D'$ )

**then obtain**  $m$  **where**  $\text{count } C \ (\text{Neg } A) = \text{Suc } m$

**by** *simp*

**define**  $C''$  **where**

$C'' = \{\#L \in\# C. L \neq \text{Neg } A\# \}$

**have**  $C = \text{replicate-mset } (\text{Suc } m) \ (\text{Neg } A) + C''$

**using**  $\langle \text{count } C \ (\text{Neg } A) = \text{Suc } m \rangle \ C''\text{-def}$

**by** (*metis filter-eq-replicate-mset union-filter-mset-complement*)

**show** *?thesis*

**proof** (*intro exI conjI*)

**show** *linorder-lit.is-greatest-in-mset*  $D \ (\text{Pos } A)$

**using**  $\langle \text{linorder-lit.is-greatest-in-mset } D \ (\text{Pos } A) \rangle .$

**next**

**show** *linorder-lit.is-maximal-in-mset*  $C \ (\text{Neg } A)$

**using** *ground-resolutionI* **by** *simp*

**next**

**show**  $D = \text{add-mset } (\text{Pos } A) \ D'$

**using**  $\langle D = \text{add-mset } (\text{Pos } A) \ D' \rangle .$

**next**

**show**  $C = \text{replicate-mset } (\text{Suc } m) \ (\text{Neg } A) + C''$

**using**  $\langle C = \text{replicate-mset } (\text{Suc } m) \ (\text{Neg } A) + C'' \rangle .$

**next**

**show**  $\text{Neg } A \notin\# C''$

**by** (*simp add: C''-def*)

**next**

**show**  $DC = D' + \text{replicate-mset } m \ (\text{Neg } A) + C''$

**using**  $\langle DC = C' + D' \rangle \langle C = \text{add-mset } (\text{Neg } A) \ C' \rangle \langle C = \text{replicate-mset } (\text{Suc}$

$m) (Neg A) + C''$

by *simp*

qed

qed

**lemma** *relpowp-ground-resolutionD*:

**assumes**  $n \neq 0$  **and**  $(ground-resolution D \rightsquigarrow n) C DnC$

**shows**  $\exists m A D' C'. Suc\ m \geq n \wedge$

$linorder-lit.is-greatest-in-mset\ D\ (Pos\ A) \wedge$

$linorder-lit.is-maximal-in-mset\ C\ (Neg\ A) \wedge$

$D = add-mset\ (Pos\ A)\ D' \wedge$

$C = replicate-mset\ (Suc\ m)\ (Neg\ A) + C' \wedge Neg\ A \notin\# C' \wedge$

$DnC = repeat-mset\ n\ D' + replicate-mset\ (Suc\ m - n)\ (Neg\ A) + C'$

**using** *assms*

**proof** (*induction n arbitrary: C*)

**case** 0

**hence** *False*

by *simp*

**thus** *?case ..*

**next**

**case**  $(Suc\ n')$

**then obtain** *DC* **where**

$ground-resolution\ D\ C\ DC$  **and**  $(ground-resolution\ D \rightsquigarrow n')\ DC\ DnC$

by (*metis relpowp-Suc-E2*)

**then obtain**  $m\ A\ D'\ C'$  **where**

$linorder-lit.is-greatest-in-mset\ D\ (Pos\ A)$  **and**

$linorder-lit.is-maximal-in-mset\ C\ (Neg\ A)$

$D = add-mset\ (Pos\ A)\ D'$  **and**

$C = replicate-mset\ (Suc\ m)\ (Neg\ A) + C'$  **and**

$Neg\ A \notin\# C'$  **and**

$DC = D' + replicate-mset\ m\ (Neg\ A) + C'$

**using**  $\langle ground-resolution\ D\ C\ DC \rangle [THEN\ ground-resolutionD]$  by *metis*

**have**  $Neg\ A \notin\# D'$

**using**  $\langle linorder-lit.is-greatest-in-mset\ D\ (Pos\ A) \rangle$

**unfolding**  $\langle D = add-mset\ (Pos\ A)\ D' \rangle$

**unfolding**  $linorder-lit.is-greatest-in-mset-iff$

by *auto*

**show** *?case*

**proof** (*cases n'*)

**case** 0

**hence**  $DnC = DC$

**using**  $\langle (ground-resolution\ D \rightsquigarrow n')\ DC\ DnC \rangle$  by *simp*

**show** *?thesis*

**unfolding** 0  $\langle DnC = DC \rangle$

**unfolding** *repeat-mset-Suc repeat-mset-0 empty-neutral*



**unfolding** *diff-Suc-Suc minus-nat.diff-0*  
**using**  $\langle C = \text{replicate-mset } (\text{Suc } m) (\text{Neg } A) + C' \rangle \langle D = \text{add-mset } (\text{Pos } A) D' \rangle$   
 $\langle DC = D' + \text{replicate-mset } m (\text{Neg } A) + C' \rangle \langle \text{Neg } A \notin\# C' \rangle$   
 $\langle \text{linorder-lit.is-greatest-in-mset } D (\text{Pos } A) \rangle \langle \text{linorder-lit.is-maximal-in-mset } C (\text{Neg } A) \rangle$   
**using** *linorder-lit.is-greatest-in-mset-iff*  
**by** *blast*  
**next**  
**case**  $(\text{Suc } n')$   
**hence**  $n' \neq 0$   
**by** *presburger*  
**then obtain**  $m' A' D'' DC'$  **where**  $n' \leq \text{Suc } m'$  **and**  
 $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A') D$  **and**  
 $\text{ord-res.is-maximal-lit } (\text{Neg } A') DC$  **and**  
 $D = \text{add-mset } (\text{Pos } A') D''$  **and**  
 $DC = \text{replicate-mset } (\text{Suc } m') (\text{Neg } A') + DC'$  **and**  
 $\text{Neg } A' \notin\# DC'$  **and**  
 $DnC = \text{repeat-mset } n' D'' + \text{replicate-mset } (\text{Suc } m' - n') (\text{Neg } A') + DC'$   
**using** *Suc.IH[OF - (ground-resolution D  $\overset{\sim}{\sim}$  n') DC DnC]*  
**by** *metis*  
  
**have**  $A' = A$   
**using**  $\langle \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A') D \rangle \langle \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) D \rangle$   
**by**  $(\text{meson } \text{Uniq-D } \text{linorder-lit.Uniq-is-maximal-in-mset } \text{linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset } \text{literal.inject}(1))$   
  
**hence**  $D'' = D'$   
**using**  $\langle D = \text{add-mset } (\text{Pos } A') D'' \rangle \langle D = \text{add-mset } (\text{Pos } A) D' \rangle$  **by** *auto*  
  
**have**  $m = \text{Suc } m'$   
**using**  $\langle DC = D' + \text{replicate-mset } m (\text{Neg } A) + C' \rangle \langle DC = \text{replicate-mset } (\text{Suc } m') (\text{Neg } A') + DC' \rangle$   
 $\langle \text{Neg } A \notin\# D' \rangle \langle \text{Neg } A \notin\# C' \rangle \langle \text{Neg } A' \notin\# DC' \rangle$   
**unfolding**  $\langle A' = A \rangle$   
**by**  $(\text{metis } \text{add-0 } \text{count-eq-zero-iff } \text{count-replicate-mset } \text{count-union } \text{union-commute})$   
  
**hence**  $DC' = D' + C'$   
**using**  $\langle DC = D' + \text{replicate-mset } m (\text{Neg } A) + C' \rangle \langle DC = \text{replicate-mset } (\text{Suc } m') (\text{Neg } A') + DC' \rangle$   
**by**  $(\text{simp } \text{add: } \langle A' = A \rangle)$   
  
**show** *?thesis*  
**proof**  $(\text{intro } \text{exI } \text{conjI})$   
**show**  $\text{Suc } n' \leq \text{Suc } (\text{Suc } m')$   
**using**  $\langle n' \leq \text{Suc } m' \rangle$  **by** *presburger*  
**next**  
**show**  $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) D$

```

    using ⟨ord-res.is-strictly-maximal-lit (Pos A) D⟩ .
  next
  show ord-res.is-maximal-lit (Neg A) C
    using ⟨ord-res.is-maximal-lit (Neg A) C⟩ by metis
  next
  show D = add-mset (Pos A) D'
    using ⟨D = add-mset (Pos A) D'⟩ .
  next
  show C = replicate-mset (Suc (Suc m')) (Neg A) + C'
    using ⟨C = replicate-mset (Suc m) (Neg A) + C'⟩ ⟨m = Suc m'⟩ by argo
  next
  show Neg A  $\notin$ # C'
    using ⟨Neg A  $\notin$ # C'⟩ .
  next
  show DnC = repeat-mset (Suc n') D' + replicate-mset (Suc (Suc m') - Suc
n') (Neg A) + C'
    using ⟨DnC = repeat-mset n' D'' + replicate-mset (Suc m' - n') (Neg A)
+ DC'⟩
    unfolding ⟨A' = A⟩ ⟨D'' = D'⟩ diff-Suc-Suc ⟨DC' = D' + C'⟩
    by simp
  qed
qed
qed

```

**lemma** *tranclp-ground-resolutionD*:

```

  assumes (ground-resolution D)++ C DnC
  shows  $\exists n m A D' C'. \text{Suc } m \geq \text{Suc } n \wedge$ 
    linorder-lit.is-greatest-in-mset D (Pos A)  $\wedge$ 
    linorder-lit.is-maximal-in-mset C (Neg A)  $\wedge$ 
    D = add-mset (Pos A) D'  $\wedge$ 
    C = replicate-mset (Suc m) (Neg A) + C'  $\wedge$  Neg A  $\notin$ # C'  $\wedge$ 
    DnC = repeat-mset (Suc n) D' + replicate-mset (Suc m - Suc n) (Neg A) +
C'

```

**proof** –

```

  from assms obtain n :: nat where
    (ground-resolution D  $\sim\sim$  Suc n) C DnC
  by (metis Suc-pred tranclp-power)
  thus ?thesis
  using assms relpowp-ground-resolutionD
  by (meson nat.discI)

```

qed

**lemma** *eres-not-identD*:

```

  assumes eres D C  $\neq$  C
  shows  $\exists m A D' C'.$ 
    linorder-lit.is-greatest-in-mset D (Pos A)  $\wedge$ 
    linorder-lit.is-maximal-in-mset C (Neg A)  $\wedge$ 
    D = add-mset (Pos A) D'  $\wedge$ 

```

$C = \text{replicate-mset } (Suc\ m) (Neg\ A) + C' \wedge Neg\ A \notin\# C' \wedge$   
 $\text{eres } D\ C = \text{repeat-mset } (Suc\ m) D' + C'$

**proof** –

**have**  $\wedge n. Suc\ n \neq 0$   
**by** *presburger*

**obtain**  $n$  **where**  
*steps*:  $(\text{ground-resolution } D \rightsquigarrow Suc\ n) C (\text{eres } D\ C)$  **and**  
*stuck*:  $\nexists x. \text{ground-resolution } D (\text{eres } D\ C) x$   
**using**  $\langle \text{eres } D\ C \neq C \rangle \text{ex1-eres-eq-full-run-ground-resolution}$   
**by**  $(\text{metis full-run-def } gr0\ conv\ Suc\ rtranclpD\ tranclp\ power)$

**obtain**  $m\ A\ D'\ C'$  **where**  
 $Suc\ n \leq Suc\ m$  **and**  
*D-max-lit*:  $\text{ord-res.is-strictly-maximal-lit } (Pos\ A) D$  **and**  
*C-max-lit*:  $\text{ord-res.is-maximal-lit } (Neg\ A) C$  **and**  
*D-eq*:  $D = \text{add-mset } (Pos\ A) D'$  **and**  
*C-eq*:  $C = \text{replicate-mset } (Suc\ m) (Neg\ A) + C'$  **and**  
 $Neg\ A \notin\# C'$  **and**  
*eres-eq*:  $\text{eres } D\ C = \text{repeat-mset } (Suc\ n) D' + \text{replicate-mset } (Suc\ m - Suc\ n)$   
 $(Neg\ A) + C'$   
**using**  $\text{relpowp-ground-resolutionD}[of\ Suc\ n, OF\ \langle Suc\ n \neq 0 \rangle\ steps]$  **by** *metis*

**from** *stuck* **have**  $\text{count } (\text{eres } D\ C) (Neg\ A) = 0$

**proof**  $(\text{rule contrapos-np})$   
**assume**  $\text{count } (\text{eres } D\ C) (Neg\ A) \neq 0$   
**then obtain**  $ERES'$  **where**  $\text{eres } D\ C = \text{add-mset } (Neg\ A) ERES'$   
**by**  $(\text{meson count-eq-zero-iff mset-add})$

**moreover have**  $\text{ord-res.is-maximal-lit } (Neg\ A) (\text{eres } D\ C)$   
**unfolding**  $\text{linorder-lit.is-maximal-in-mset-iff}$

**proof**  $(\text{intro conjI ballI impI})$   
**show**  $Neg\ A \in\# \text{eres } D\ C$   
**unfolding**  $\langle \text{eres } D\ C = \text{add-mset } (Neg\ A) ERES' \rangle$  **by** *simp*

**next**  
**fix**  $L$   
**assume**  $L \in\# \text{eres } D\ C$  **and**  $L \neq Neg\ A$   
**hence**  $L \in\# \text{repeat-mset } (Suc\ n) D' \vee L \in\# C'$   
**unfolding** *eres-eq*  
**by**  $(\text{metis Zero-not-Suc count-replicate-mset in-countE union-iff})$

**thus**  $\neg Neg\ A \prec_l L$

**proof**  $(\text{elim disjE})$   
**assume**  $L \in\# \text{repeat-mset } (Suc\ n) D'$   
**hence**  $L \in\# D'$   
**using**  $\text{member-mset-repeat-msetD}$  **by** *metis*  
**hence**  $L \prec_l Pos\ A$   
**using** *D-max-lit*  
**unfolding**  $D\text{-eq linorder-lit.is-greatest-in-mset-iff}$  **by** *simp*  
**also have**  $Pos\ A \prec_l Neg\ A$

```

      by simp
    finally show ?thesis
      by order
  next
    assume  $L \in\# C'$ 
    thus ?thesis
      using  $C\text{-eq } \langle L \neq \text{Neg } A \rangle C\text{-max-lit linorder-lit.is-maximal-in-mset-iff}$  by
auto
  qed
  qed

  moreover have  $D \prec_c \text{eres } D C$ 
    using  $D\text{-max-lit}$ 
    using  $\langle \text{ord-res.is-maximal-lit } (\text{Neg } A) (\text{eres } D C) \rangle$ 
    using  $\text{linorder-lit.mulp}_{HO}\text{-if-maximal-less-than-maximal}$ [of  $D \text{ Pos } A \text{ eres } D$ 
 $C \text{ Neg } A$ , simplified]
    using  $\text{mulp}_{DM}\text{-imp-mulp mulp}_{HO}\text{-imp-mulp}_{DM}$  by blast

  ultimately show  $\exists x. \text{ground-resolution } D (\text{eres } D C) x$ 
    unfolding  $\text{ground-resolution-def}$ 
    using  $D\text{-eq } D\text{-max-lit}$ 
    using  $\text{ord-res.ground-resolutionI}$ [of  $\text{eres } D C A \text{ ERES}' D D' \text{ ERES}' + D'$ ]
    by metis
  qed

  hence  $m = n$ 
    using  $\langle \text{eres } D C = \text{repeat-mset } (\text{Suc } n) D' + \text{replicate-mset } (\text{Suc } m - \text{Suc } n) (\text{Neg } A) + C' \rangle$ 
    using  $\langle \text{Suc } n \leq \text{Suc } m \rangle$  by auto

  show ?thesis
  proof (intro exI conjI)
    show  $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) D$ 
      using  $D\text{-max-lit}$  .
    next
      show  $\text{ord-res.is-maximal-lit } (\text{Neg } A) C$ 
        using  $C\text{-max-lit}$  .
    next
      show  $D = \text{add-mset } (\text{Pos } A) D'$ 
        using  $D\text{-eq}$  .
    next
      show  $C = \text{replicate-mset } (\text{Suc } m) (\text{Neg } A) + C'$ 
        using  $C\text{-eq}$  .
    next
      show  $\text{Neg } A \notin\# C'$ 
        using  $\langle \text{Neg } A \notin\# C' \rangle$  .
    next
      show  $\text{eres } D C = \text{repeat-mset } (\text{Suc } m) D' + C'$ 
        using  $\text{eres-eq}$  unfolding  $\langle m = n \rangle$  by simp
  end

```

qed  
qed

**lemma** *lit-in-one-of-resolvents-if-in-eres:*

**fixes**  $L :: 'f \text{ gterm literal}$  **and**  $C D :: 'f \text{ gclause}$

**assumes**  $L \in\# \text{ eres } C D$

**shows**  $L \in\# C \vee L \in\# D$

**proof** (*cases eres C D = D*)

**assume**  $\text{eres } C D = D$

**thus**  $L \in\# C \vee L \in\# D$

**using**  $\langle L \in\# \text{ eres } C D \rangle$  **by** *argo*

**next**

**assume**  $\text{eres } C D \neq D$

**thus**  $L \in\# C \vee L \in\# D$

**using**  $\langle L \in\# \text{ eres } C D \rangle$

**by** (*metis eres-not-identD member-mset-repeat-msetD repeat-mset-distrib-add-mset union-iff*)

qed

**lemma** *strong-lit-in-one-of-resolvents-if-in-eres:*

**fixes**  $L :: 'f \text{ gterm literal}$  **and**  $C D :: 'f \text{ gclause}$

**assumes**

*D-max-lit: linorder-lit.is-maximal-in-mset D L* **and**

*K-in: K \in\# eres C D*

**shows**  $K \in\# C \wedge K \neq -L \vee K \in\# D$

**proof** (*cases eres C D = D*)

**assume**  $\text{eres } C D = D$

**thus**  $K \in\# C \wedge K \neq -L \vee K \in\# D$

**using** *K-in* **by** *argo*

**next**

**assume**  $\text{eres } C D \neq D$

**then obtain**  $m :: \text{nat}$  **and**  $A :: 'f \text{ gterm}$  **and**  $C' D' :: 'f \text{ gterm literal multiset}$

**where**

*C-max-lit: ord-res.is-strictly-maximal-lit (Pos A) C* **and**

*D-max-lit': ord-res.is-maximal-lit (Neg A) D* **and**

*C-eq: C = add-mset (Pos A) C'* **and**

*D-eq: D = replicate-mset (Suc m) (Neg A) + D'* **and**

*Neg A \notin\# D'* **and**

*eres C D = repeat-mset (Suc m) C' + D'*

**using** *eres-not-identD* **by** *metis*

**have** *L-eq: L = Neg A*

**using** *D-max-lit D-max-lit'* **by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

**have**  $K \in\# \text{ repeat-mset } (Suc\ m) C' + D'$

**using** *K-in* **unfolding**  $\langle \text{eres } C D = \text{repeat-mset } (Suc\ m) C' + D' \rangle$  .

**hence**  $K \in\# \text{ repeat-mset } (Suc\ m) C' \vee K \in\# D'$

**unfolding** *Multiset.union-iff* .

**hence**  $K \in\# C' \vee K \in\# D'$   
**unfolding** *member-mset-repeat-mset-Suc* .

**thus**  $K \in\# C \wedge K \neq -L \vee K \in\# D$   
**proof** (*elim disjE*)  
**assume**  $K \in\# C'$

**hence**  $K \in\# C \wedge K \neq -L$   
**using** *C-max-lit*  
**unfolding** *C-eq L-eq linorder-lit.is-greatest-in-mset-iff* **by** *auto*

**thus**  $K \in\# C \wedge K \neq -L \vee K \in\# D$  ..  
**next**  
**assume**  $K \in\# D'$

**hence**  $K \in\# D$   
**unfolding** *D-eq* **by** *simp*

**thus**  $K \in\# C \wedge K \neq -L \vee K \in\# D$  ..  
**qed**  
**qed**

**lemma** *stronger-lit-in-one-of-resolvents-if-in-eres*:  
**fixes**  $K L :: 'f$  *gterm literal* **and**  $C D :: 'f$  *gclause*  
**assumes** *eres*  $C D \neq D$  **and**  
*D-max-lit: linorder-lit.is-maximal-in-mset*  $D L$  **and**  
*K-in-eres: K ∈# eres C D*  
**shows**  $K \in\# C \wedge K \neq -L \vee K \in\# D \wedge K \neq L$   
**proof** –

**obtain**  $m :: nat$  **and**  $A :: 'f$  *gterm* **and**  $C' D' :: 'f$  *gterm literal multiset* **where**  
*C-max-lit: ord-res.is-strictly-maximal-lit* (*Pos A*)  $C$  **and**  
*C-def: C = add-mset (Pos A) C'* **and**  
*D = replicate-mset (Suc m) (Neg A) + D'* **and**  
*Neg A ∉# D'* **and**  
*eres C D = repeat-mset (Suc m) C' + D'*  
**using**  $\langle eres C D \neq D \rangle$  [*THEN eres-not-identD*] **by** *metis*

**have**  $L = Neg A$   
**using** *assms(1) D-max-lit C-max-lit*  
**by** (*metis ground-resolutionD linorder-lit.Uniq-is-greatest-in-mset*  
*linorder-lit.Uniq-is-maximal-in-mset resolvable-if-neq-eres the1-equality' umi-*  
*nus-Pos*)

**have**  $K \in\# repeat-mset (Suc m) C' + D'$   
**using** *K-in-eres* **unfolding**  $\langle eres C D = repeat-mset (Suc m) C' + D' \rangle$  .

**hence**  $K \in\# repeat-mset (Suc m) C' \vee K \in\# D'$   
**unfolding** *Multiset.union-iff* .

**hence**  $K \in\# C' \vee K \in\# D'$   
**unfolding** *member-mset-repeat-mset-Suc* .

**thus**  $K \in\# C \wedge K \neq -L \vee K \in\# D \wedge K \neq L$   
**proof** (*elim disjE*)  
**assume**  $K \in\# C'$

**hence**  $K \in\# C \wedge K \neq -L$   
**using** *C-max-lit[unfolded linorder-lit.is-greatest-in-mset-iff]*  
**unfolding**  $\langle C = \text{add-mset} (\text{Pos } A) C' \rangle \langle L = \text{Neg } A \rangle$   
**by** *auto*

**thus** *?thesis*  
**by** *argo*

**next**  
**assume**  $K \in\# D'$

**hence**  $K \in\# D \wedge K \neq L$   
**unfolding**  $\langle D = \text{replicate-mset} (\text{Suc } m) (\text{Neg } A) + D' \rangle \langle L = \text{Neg } A \rangle$   
**using**  $\langle \text{Neg } A \notin\# D' \rangle$   
**by** *auto*

**thus** *?thesis*  
**by** *argo*

**qed**  
**qed**

**lemma** *lit-in-eres-lt-greatest-lit-in-greatest-resolvant:*  
**fixes**  $K L :: 'f \text{ gterm literal}$  **and**  $C D :: 'f \text{ gclause}$   
**assumes**  $\text{eres } C D \neq D$  **and**  
*D-max-lit: linorder-lit.is-maximal-in-mset D L* **and**  
 $-L \notin\# D$  **and**  
*K-in-eres: K ∈# eres C D*  
**shows** *atm-of K*  $\prec_t$  *atm-of L*  
**proof** –

**obtain**  $m :: \text{nat}$  **and**  $A :: 'f \text{ gterm}$  **and**  $C' D' :: 'f \text{ gterm literal multiset}$  **where**  
*C-max-lit: ord-res.is-strictly-maximal-lit (Pos A) C* **and**  
*C-def: C = add-mset (Pos A) C'* **and**  
*D = replicate-mset (Suc m) (Neg A) + D'* **and**  
 $\text{Neg } A \notin\# D'$  **and**  
*eres C D = repeat-mset (Suc m) C' + D'*  
**using**  $\langle \text{eres } C D \neq D \rangle$  [*THEN eres-not-identD*] **by** *metis*

**have**  $L = \text{Neg } A$   
**using** *assms(1) D-max-lit C-max-lit*  
**by** (*metis ground-resolutionD linorder-lit.Uniq-is-greatest-in-mset*  
*linorder-lit.Uniq-is-maximal-in-mset resolvable-if-neq-eres the1-equality' umi-*  
*nus-Pos*)

**have**  $K \in\# \text{repeat-mset } (\text{Suc } m) C' + D'$   
**using**  $K\text{-in-eres unfolding } \langle \text{eres } C D = \text{repeat-mset } (\text{Suc } m) C' + D' \rangle$  .

**hence**  $K \in\# \text{repeat-mset } (\text{Suc } m) C' \vee K \in\# D'$   
**unfolding**  $\text{Multiset.union-iff}$  .

**hence**  $K \in\# C' \vee K \in\# D'$   
**unfolding**  $\text{member-mset-repeat-mset-Suc}$  .

**thus**  $\text{atm-of } K \prec_t \text{atm-of } L$   
**proof**  $(\text{elim disjE})$   
**assume**  $K \in\# C'$   
**hence**  $K \prec_l \text{Pos } A$   
**using**  $C\text{-max-lit } C\text{-def } \langle L = \text{Neg } A \rangle$   
**unfolding**  $\text{linorder-lit.is-greatest-in-mset-iff}$   
**by**  $\text{simp}$   
**thus**  $\text{atm-of } K \prec_t \text{atm-of } L$   
**unfolding**  $\langle L = \text{Neg } A \rangle \text{literal.sel}$   
**by**  $(\text{cases } K) \text{simp-all}$   
**next**  
**assume**  $K \in\# D'$   
**hence**  $K \prec_l \text{Neg } A$   
**by**  $(\text{metis } D\text{-max-lit } \langle D = \text{replicate-mset } (\text{Suc } m) (\text{Neg } A) + D' \rangle \langle L = \text{Neg } A \rangle \langle \text{Neg } A \notin\# D' \rangle$   
 $\text{linorder-lit.is-maximal-in-mset-iff linorder-lit.neqE union-iff})$

**moreover have**  $K \neq \text{Pos } A$   
**using**  $\langle - L \notin\# D \rangle$   
**using**  $\langle D = \text{replicate-mset } (\text{Suc } m) (\text{Neg } A) + D' \rangle \langle K \in\# D' \rangle \langle L = \text{Neg } A \rangle$   
**by**  $\text{fastforce}$

**ultimately have**  $K \prec_l \text{Pos } A$   
**by**  $(\text{metis linorder-lit.less-asym linorder-lit.less-linear literal.exhaust ord-res.less-lit-simps(1) ord-res.less-lit-simps(3) ord-res.less-lit-simps(4)})$

**thus**  $\text{atm-of } K \prec_t \text{atm-of } L$   
**unfolding**  $\langle L = \text{Neg } A \rangle \text{literal.sel}$   
**by**  $(\text{cases } K) \text{simp-all}$   
**qed**  
**qed**

**lemma**  $\text{eres-entails-resolvent}$ :  
**fixes**  $C D :: 'f \text{gterm clause}$   
**assumes**  $(\text{ground-resolution } C)^{++} D_0 D$   
**shows**  $\{\text{eres } C D_0\} \Vdash_e \{D\}$   
**unfolding**  $\text{true-cls-singleton}$   
**proof**  $(\text{intro allI impI})$   
**have**  $\text{eres } C D_0 = \text{eres } C D$



using *assms eres-eq-after-tranclp-ground-resolution* by *metis*

**obtain**  $n\ m :: \text{nat}$  **and**  $A :: 'f\ \text{gterm}$  **and**  $C'\ D_0' :: 'f\ \text{gterm clause}$  **where**  
 $Suc\ n \leq Suc\ m$  **and**  
*ord-res.is-strictly-maximal-lit* ( $Pos\ A$ )  $C$  **and**  
*ord-res.is-maximal-lit* ( $Neg\ A$ )  $D_0$  **and**  
 $C = \text{add-mset}\ (Pos\ A)\ C'$  **and**  
 $D_0 = \text{replicate-mset}\ (Suc\ m)\ (Neg\ A) + D_0'$  **and**  
 $Neg\ A \notin\# D_0'$  **and**  
 $D = \text{repeat-mset}\ (Suc\ n)\ C' + \text{replicate-mset}\ (Suc\ m - Suc\ n)\ (Neg\ A) + D_0'$   
**using**  $\langle(\text{ground-resolution}\ C)^{++}\ D_0\ D\rangle[\text{THEN}\ \text{tranclp-ground-resolution}D]$  **by**  
*metis*

**fix**  $I :: 'f\ \text{gterm set}$   
**assume**  $I \models\ \text{eres}\ C\ D_0$   
**show**  $I \models\ D$   
**proof** (*cases eres C D<sub>0</sub> = D*)  
**case** *True*  
**thus** *?thesis*  
**using**  $\langle I \models\ \text{eres}\ C\ D_0 \rangle$  **by** *argo*  
**next**  
**case** *False*  
**then obtain**  $k :: \text{nat}$  **and**  $D' :: 'f\ \text{gterm clause}$  **where**  
*ord-res.is-strictly-maximal-lit* ( $Pos\ A$ )  $C$  **and**  
 $C = \text{add-mset}\ (Pos\ A)\ C'$  **and**  
 $D = \text{replicate-mset}\ (Suc\ k)\ (Neg\ A) + D'$  **and**  
 $Neg\ A \notin\# D'$  **and**  
 $\text{eres}\ C\ D = \text{repeat-mset}\ (Suc\ k)\ C' + D'$   
**unfolding**  $\langle\text{eres}\ C\ D_0 = \text{eres}\ C\ D\rangle$   
**using** *eres-not-identD*  
**using**  $\langle\text{ord-res.is-strictly-maximal-lit}\ (Pos\ A)\ C\rangle\ \langle C = \text{add-mset}\ (Pos\ A)\ C'\rangle$   
**by** (*metis Uniq-D add-mset-remove-trivial linorder-lit. Uniq-is-greatest-in-mset literal.sel(1)*)

**have**  $I \models\ \text{repeat-mset}\ (Suc\ k)\ C' + D'$   
**using**  $\langle I \models\ \text{eres}\ C\ D_0 \rangle$   
**unfolding**  $\langle\text{eres}\ C\ D_0 = \text{eres}\ C\ D\rangle\ \langle\text{eres}\ C\ D = \text{repeat-mset}\ (Suc\ k)\ C' + D'\rangle$  .

**hence**  $I \models\ D' \vee I \models\ \text{repeat-mset}\ (Suc\ k)\ C'$   
**by** *auto*

**thus**  $I \models\ D$   
**proof** (*elim disjE*)  
**assume**  $I \models\ D'$   
**thus**  $I \models\ D$   
**unfolding**  $\langle D = \text{replicate-mset}\ (Suc\ k)\ (Neg\ A) + D'\rangle$   
**by** *simp*  
**next**

```

assume  $I \Vdash \text{repeat-mset } (\text{Suc } k) C'$ 
thus  $I \Vdash D$ 
  using  $\langle D = \text{replicate-mset } (\text{Suc } k) (\text{Neg } A) + D' \rangle$ 
  using  $\langle D = \text{repeat-mset } (\text{Suc } n) C' + \text{replicate-mset } (\text{Suc } m - \text{Suc } n) (\text{Neg } A) + D_0' \rangle$ 
  by  $(\text{metis member-mset-repeat-msetD repeat-mset-Suc true-cls-def true-cls-union})$ 
qed
qed
qed

```

**lemma** *clause-true-if-resolved-true:*

```

assumes
   $(\text{ground-resolution } D)^{++} C DC$  and
   $D\text{-productive: ord-res.production } N D \neq \{\}$  and
   $C\text{-true: ord-res-Interp } N DC \Vdash DC$ 
shows  $\text{ord-res-Interp } N C \Vdash C$ 
proof -
  obtain  $n$  where
     $\text{steps: } (\text{ground-resolution } D \rightsquigarrow \text{Suc } n) C DC$ 
    using  $\langle (\text{ground-resolution } D)^{++} C DC \rangle$ 
    by  $(\text{metis less-not-refl not0-implies-Suc tranclp-power})$ 

```

**obtain**  $m A D' C'$  **where**

```

 $n \leq m$  and
 $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) D$  and
 $\text{ord-res.is-maximal-lit } (\text{Neg } A) C$  and
 $D = \text{add-mset } (\text{Pos } A) D'$  and
 $C = \text{replicate-mset } (\text{Suc } m) (\text{Neg } A) + C'$  and
 $\text{Neg } A \notin \# C'$  and
 $DC = \text{repeat-mset } (\text{Suc } n) D' + \text{replicate-mset } (m - n) (\text{Neg } A) + C'$ 
using  $\text{relpoup-ground-resolutionD}[OF \text{Suc-not-Zero steps}]$ 
by  $(\text{metis diff-Suc-Suc Suc-le-mono})$ 

```

**have**  $\text{Neg } A \notin \# D'$

```

by  $(\text{metis } \langle D = \text{add-mset } (\text{Pos } A) D' \rangle \langle \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) D \rangle$ 
 $\text{ord-res.less-lit-simps}(4) \text{linorder-lit.is-greatest-in-mset-iff linorder-trm.eq-refl}$ 
 $\text{linorder-trm.leD remove1-mset-add-mset-If})$ 

```

**have**  $DC \prec_c C$

**proof**  $(\text{cases } m = n)$

**case** *True*

**show** *?thesis*

**proof**  $(\text{intro one-step-implies-multp}[of - - \{\#\}, \text{simplified}] \text{ballI})$

**show**  $C \neq \{\#\}$

**by**  $(\text{simp add: } \langle C = \text{replicate-mset } (\text{Suc } m) (\text{Neg } A) + C' \rangle)$

**next**

```

fix L
assume L ∈# DC
hence L ∈# D' ∨ L ∈# C'
  unfolding ⟨DC = repeat-mset (Suc n) D' + replicate-mset (m - n) (Neg
A) + C'⟩ ⟨m = n⟩
  using member-mset-repeat-msetD by fastforce
  hence L <l Neg A
  using ⟨ord-res.is-strictly-maximal-lit (Pos A) D⟩ ⟨ord-res.is-maximal-lit (Neg
A) C⟩
  unfolding ⟨D = add-mset (Pos A) D'⟩ ⟨C = replicate-mset (Suc m) (Neg
A) + C'⟩
  unfolding linorder-lit.is-maximal-in-mset-iff linorder-lit.is-greatest-in-mset-iff
  by (metis ⟨Neg A ∉# C'⟩ add-mset-remove-trivial ord-res.less-lit-simps(4)
linorder-lit.antisym-conv3 linorder-lit.dual-order.strict-trans
linorder-trm.dual-order.asym union-iff)

moreover have Neg A ∈# C
  by (simp add: ⟨C = replicate-mset (Suc m) (Neg A) + C'⟩)

ultimately show ∃ K ∈# C. L <l K
  by metis
qed
next
case False
hence n < m
  using ⟨n ≤ m⟩ by presburger

have multpHO (<l) DC C
proof (rule linorder-lit.multpHO-if-same-maximal-and-count-lt)
  show ord-res.is-maximal-lit (Neg A) DC
  unfolding linorder-lit.is-maximal-in-mset-iff
  proof (intro conjI ballI impI)
  show Neg A ∈# DC
  unfolding ⟨DC = repeat-mset (Suc n) D' + replicate-mset (m - n) (Neg
A) + C'⟩
  using ⟨n < m⟩ by simp
  next
  fix L
  assume L ∈# DC and L ≠ Neg A
  hence L ∈# D' ∨ L ∈# C'
  unfolding ⟨DC = repeat-mset (Suc n) D' + replicate-mset (m - n) (Neg
A) + C'⟩
  by (metis in-replicate-mset member-mset-repeat-msetD union-iff)
  thus ¬ Neg A <l L
  using ⟨ord-res.is-strictly-maximal-lit (Pos A) D⟩ ⟨ord-res.is-maximal-lit
(Neg A) C⟩
  unfolding ⟨D = add-mset (Pos A) D'⟩ ⟨C = replicate-mset (Suc m) (Neg
A) + C'⟩
  unfolding linorder-lit.is-maximal-in-mset-iff linorder-lit.is-greatest-in-mset-iff

```

```

    by (metis ⟨L ≠ Neg A⟩ add-mset-diff-bothsides diff-zero
        linorder-lit.dual-order.strict-trans linorder-trm.less-irrefl
        ord-res.less-lit-simps(4) union-iff)
  qed
next
  show ord-res.is-maximal-lit (Neg A) C
    using ⟨ord-res.is-maximal-lit (Neg A) C⟩ .
next
  have count DC (Neg A) = count (repeat-mset (Suc n) D') (Neg A) +
    count (replicate-mset (m - n) (Neg A)) (Neg A) + count C' (Neg A)
    unfolding ⟨DC = repeat-mset (Suc n) D' + replicate-mset (m - n) (Neg
A) + C'⟩ by simp
    also have ... = count D' (Neg A) * Suc n + count {#Neg A#} (Neg A) *
(m - n) + count C' (Neg A)
      by simp
    also have ... = 0 * Suc n + 1 * (m - n) + 0
      by (simp add: ⟨Neg A ∉# C'⟩ ⟨Neg A ∉# D'⟩ count-eq-zero-iff)
    also have ... = m - n
      by presburger
    also have ... < Suc m
      by presburger
    also have ... = 1 * Suc m + 0
      by presburger
    also have ... = count {#Neg A#} (Neg A) * Suc m + count C' (Neg A)
      by (simp add: ⟨Neg A ∉# C'⟩ count-eq-zero-iff)
    also have ... = count (replicate-mset (Suc m) (Neg A)) (Neg A) + count C'
(Neg A)
      by simp
    also have ... = count C (Neg A)
      unfolding ⟨C = replicate-mset (Suc m) (Neg A) + C'⟩ by simp
    finally show count DC (Neg A) < count C (Neg A) .
  qed
  thus ?thesis
    by (simp add: multpDM-imp-multp multpHO-imp-multpDM)
qed

with C-true have ord-res-Interp N C ⊨ DC
  using ord-res.entailed-clause-stays-entailed by metis

thus ord-res-Interp N C ⊨ C
  unfolding true-cls-def
proof (elim bexE)
  fix L
  assume
    L-in: L ∈# DC and
    L-true: ord-res-Interp N C ⊨ l L

  from L-in have L ∈# D' ∨ L = Neg A ∨ L ∈# C'
    unfolding ⟨DC = repeat-mset (Suc n) D' + replicate-mset (m - n) (Neg A)

```

+  $C'$   
**by** (*metis in-replicate-mset member-mset-repeat-msetD union-iff*)

**moreover have**  $L \notin \# D'$   
**proof** (*rule notI*)  
**assume**  $L \in \# D'$

**moreover have**  $\neg \text{ord-res.interp } N \text{ (add-mset (Pos A) D')} \Vdash \text{add-mset (Pos A) D'}$   
**using** *D-productive[unfolded <D = add-mset (Pos A) D']*  
**unfolding** *ord-res.production-unfold*  
**by** *fast*

**ultimately have**  $\neg \text{ord-res.interp } N \text{ (add-mset (Pos A) D')} \Vdash L$   
**by** *auto*

**have**  $L \prec_l \text{Pos A}$   
**using**  $\langle D = \text{add-mset (Pos A) D'} \rangle \langle L \in \# D' \rangle \langle \text{ord-res.is-strictly-maximal-lit (Pos A) D} \rangle$   
*linorder-lit.is-greatest-in-mset-iff* **by** *fastforce*

**have**  $\neg \text{ord-res.Interp } N C \Vdash L$   
**proof** (*cases L*)  
**case** (*Pos B*)  
**hence**  $B \notin \text{ord-res.interp } N \text{ (add-mset (Pos A) D')}$   
**using**  $\langle \neg \text{ord-res.interp } N \text{ (add-mset (Pos A) D')} \Vdash L \rangle$  **by** *simp*

**moreover have**  $\text{add-mset (Pos A) D'} \prec_c C$   
**by** (*metis <D = add-mset (Pos A) D'> <\wedge thesis. (\wedge m A D' C'. [n \leq m; ord-res.is-strictly-maximal-lit (Pos A) D; ord-res.is-maximal-lit (Neg A) C; D = add-mset (Pos A) D'; C = replicate-mset (Suc m) (Neg A) + C'; Neg A \notin \# C'; DC = repeat-mset (Suc n) D' + replicate-mset (m - n) (Neg A) + C'] \implies thesis) \implies thesis>* *linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset linorder-lit.multip<sub>HO</sub>-if-maximal-less-than-maximal multip<sub>DM</sub>-imp-multip multip<sub>HO</sub>-imp-multip<sub>DM</sub> ord-res.less-lit-simps(2) reflclp-iff*)

**ultimately have**  $B \notin \text{ord-res.interp } N C$   
**using**  $\langle L \prec_l \text{Pos A} \rangle$  [*unfolded Pos, simplified*]  
**using** *ord-res.interp-fixed-for-smaller-literals*  
**by** (*metis <D = add-mset (Pos A) D'> <ord-res.is-strictly-maximal-lit (Pos A) D>*)  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset literal.sel(1)*)

**moreover have**  $B \notin \text{ord-res.production } N C$   
**by** (*metis Uniq-D <ord-res.is-maximal-lit (Neg A) C> ground-ordered-resolution-calculus.mem-production linorder-lit.Uniq-is-maximal-in-mset linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset literal.simps(4) ord-res.ground-ordered-resolution-calculus-axioms*)

**ultimately show** *?thesis*

**unfolding**  $Pos$  **by**  $simp$   
**next**  
**case**  $(Neg\ B)$   
**hence**  $B \in ord-res.interp\ N\ (add-mset\ (Pos\ A)\ D')$   
**using**  $\langle \neg\ ord-res.interp\ N\ (add-mset\ (Pos\ A)\ D') \models_l L \rangle$  **by**  $simp$

**moreover have**  $add-mset\ (Pos\ A)\ D' \prec_c\ C$   
**by**  $(metis\ \langle D = add-mset\ (Pos\ A)\ D' \rangle\ \langle \wedge thesis.\ (\wedge m\ A\ D'\ C'.\ \llbracket n \leq m; ord-res.is-strictly-maximal-lit\ (Pos\ A)\ D; ord-res.is-maximal-lit\ (Neg\ A)\ C; D = add-mset\ (Pos\ A)\ D'; C = replicate-mset\ (Suc\ m)\ (Neg\ A) + C'; Neg\ A \notin\# C'; DC = repeat-mset\ (Suc\ n)\ D' + replicate-mset\ (m - n)\ (Neg\ A) + C \rrbracket \implies thesis \rangle \implies thesis \rangle\ linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset\ linorder-lit.multip_{HO}-if-maximal-less-than-maximal\ multip_{DM}-imp-multip\ multip_{HO}-imp-multip_{DM}\ ord-res.less-lit-simps(2)\ reflclp-iff)$

**ultimately have**  $B \in ord-res.interp\ N\ C$   
**by**  $(metis\ Un-iff\ ord-res.not-interp-to-Interp-imp-le\ linorder-cls.leD)$

**then show**  $?thesis$   
**unfolding**  $Neg$   
**by**  $simp$   
**qed**

**with**  $L$ -**true show**  $False$   
**by**  $contradiction$   
**qed**

**ultimately have**  $L \in\# C$   
**unfolding**  $\langle C = replicate-mset\ (Suc\ m)\ (Neg\ A) + C' \rangle$  **by**  $simp$

**with**  $L$ -**true show**  $\exists L \in\# C.\ ord-res-Interp\ N\ C \models_l L$   
**by**  $metis$   
**qed**  
**qed**

**lemma**  $clause-true-if-eres-true$ :  
**assumes**  
 $(ground-resolution\ D1)^{++}\ D2\ C$  **and**  
 $C \neq eres\ D1\ C$  **and**  
 $eres-C-true: ord-res-Interp\ N\ (eres\ D1\ C) \models eres\ D1\ C$   
**shows**  $ord-res-Interp\ N\ C \models C$   
**proof** –  
**obtain**  $n$  **where**  
 $steps: (ground-resolution\ D1 \rightsquigarrow Suc\ n)\ D2\ C$   
**using**  $\langle (ground-resolution\ D1)^{++}\ D2\ C \rangle$   
**by**  $(metis\ less-not-refl\ not0-implies-Suc\ tranclp-power)$

**obtain**  $m\ A\ D'\ C'$  **where**  
 $n \leq m$  **and**

*ord-res.is-strictly-maximal-lit* (Pos A) D1 **and**  
*ord-res.is-maximal-lit* (Neg A) D2 **and**  
D1 = *add-mset* (Pos A) D' **and**  
D2 = *replicate-mset* (Suc m) (Neg A) + C' **and**  
Neg A  $\notin\#$  C' **and**  
C = *repeat-mset* (Suc n) D' + *replicate-mset* (m - n) (Neg A) + C'  
**using** *relpwp-ground-resolutionD*[OF Suc-not-Zero steps]  
**by** (*metis diff-Suc-Suc Suc-le-mono*)

**have** Neg A  $\notin\#$  D'  
**by** (*metis*  $\langle$ D1 = *add-mset* (Pos A) D' $\rangle$  *ord-res.is-strictly-maximal-lit* (Pos A) D1 $\rangle$   
*ord-res.less-lit-simps*(4) *linorder-lit.is-greatest-in-mset-iff linorder-trm.eq-refl linorder-trm.leD remove1-mset-add-mset-If*)

**obtain** m' C'' **where**  
C = *replicate-mset* (Suc m') (Neg A) + C'' **and**  
Neg A  $\notin\#$  C'' **and**  
eres D1 C = *repeat-mset* (Suc m') D' + C''  
**using**  $\langle$ C  $\neq$  eres D1 C $\rangle$  *eres-not-identD*  
**using**  $\langle$ *ord-res.is-strictly-maximal-lit* (Pos A) D1 $\rangle$  *linorder-lit.Uniq-is-greatest-in-mset*  
**using**  $\langle$ D1 = *add-mset* (Pos A) D' $\rangle$   
**by** (*metis Uniq-D add-mset-remove-trivial literal.inject*(1))

**have** m - n = Suc m'  
**proof** -  
**have** *count* C (Neg A) = *count* (*repeat-mset* (Suc n) D') (Neg A) +  
*count* (*replicate-mset* (m - n) (Neg A)) (Neg A) + *count* C' (Neg A)  
**using**  $\langle$ C = *repeat-mset* (Suc n) D' + *replicate-mset* (m - n) (Neg A) + C' $\rangle$   
**by** *simp*  
**also have** ... = *count* D' (Neg A) \* Suc n + *count* {#Neg A#} (Neg A) \* (m - n) +  
*count* C' (Neg A)  
**by** *simp*  
**also have** ... = 0 \* Suc n + 1 \* (m - n) + 0  
**using**  $\langle$ Neg A  $\notin\#$  D' $\rangle$   $\langle$ Neg A  $\notin\#$  C' $\rangle$  **by** (*simp add: count-eq-zero-iff*)  
**also have** ... = m - n  
**by** *presburger*  
**finally have** *count* C (Neg A) = m - n .

**have** *count* C (Neg A) = *count* (*replicate-mset* (Suc m') (Neg A)) (Neg A) +  
*count* C'' (Neg A)  
**using**  $\langle$ C = *replicate-mset* (Suc m') (Neg A) + C'' $\rangle$  **by** *simp*  
**also have** ... = *count* {#Neg A#} (Neg A) \* Suc m' + *count* C'' (Neg A)  
**by** *simp*  
**also have** ... = 1 \* Suc m' + 0  
**using**  $\langle$ Neg A  $\notin\#$  C'' $\rangle$  **by** (*simp add: count-eq-zero-iff*)  
**also have** ... = Suc m'  
**by** *presburger*

**finally have**  $\text{count } C \text{ (Neg } A) = \text{Suc } m'$  .

**show** *?thesis*  
**using**  $\langle \text{count } C \text{ (Neg } A) = m - n \rangle \langle \text{count } C \text{ (Neg } A) = \text{Suc } m' \rangle$  **by** *argo*  
**qed**

**hence**  $C'' = \text{repeat-mset } (\text{Suc } n) D' + C'$   
**using**  $\langle C = \text{repeat-mset } (\text{Suc } n) D' + \text{replicate-mset } (m - n) (\text{Neg } A) + C' \rangle$   
 $\langle C = \text{replicate-mset } (\text{Suc } m') (\text{Neg } A) + C'' \rangle$   
**by** *simp*

**hence** *eres-D1-C-eq*:  $\text{eres } D1 \ C = \text{repeat-mset } (\text{Suc } m' + \text{Suc } n) D' + C'$   
**using**  $\langle \text{eres } D1 \ C = \text{repeat-mset } (\text{Suc } m') D' + C'' \rangle$  **by** *simp*

**have** *ord-res-Interp N*  $(\text{eres } D1 \ C) \models \text{eres } D1 \ C$   
**using** *eres-C-true* .

**moreover have**  $\text{eres } D1 \ C \prec_c C$   
**using** *eres-le[of D1 C]*  $\langle C \neq \text{eres } D1 \ C \rangle$  **by** *order*

**ultimately have** *ord-res-Interp N C*  $\models \text{eres } D1 \ C$   
**using** *ord-res.entailedd-clause-stays-entailedd* **by** *metis*

**thus** *ord-res-Interp N C*  $\models C$   
**unfolding** *true-cls-def*  
**proof** (*elim bexE*)  
**fix**  $L$   
**assume**  
 $L\text{-in}: L \in \# \text{eres } D1 \ C$  **and**  
 $L\text{-true}: \text{ord-res-Interp } N \ C \models_l L$

**from**  $L\text{-in}$  **have**  $L \in \# D' \vee L \in \# C'$   
**unfolding** *eres-D1-C-eq*  
**using** *member-mset-repeat-msetD* **by** *fastforce*  
**hence**  $L \in \# C$   
**by** (*auto simp*:  $\langle C = \text{repeat-mset } (\text{Suc } n) D' + \text{replicate-mset } (m - n) (\text{Neg } A) + C' \rangle$ )  
**with**  $L\text{-true}$  **show**  $\exists L \in \# C. \text{ord-res-Interp } N \ C \models_l L$   
**by** *metis*

**qed**  
**qed**

**end**

**end**  
**theory** *ORD-RES-3*  
**imports**  
*ORD-RES*  
*Exhaustive-Factorization*



**begin**

## 17 ORD-RES-3 (full resolve)

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-3* **where**

*factoring:*

*is-least-false-clause*  $(N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}) C \implies$   
*linorder-lit.is-maximal-in-mset*  $C L \implies$   
*is-pos*  $L \implies$   
 $U_{ef}' = \text{finsert } (efac C) U_{ef} \implies$   
*ord-res-3*  $N (U_{er}, U_{ef}) (U_{er}, U_{ef}') \mid$

*resolution:*

*is-least-false-clause*  $(N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}) C \implies$   
*linorder-lit.is-maximal-in-mset*  $C L \implies$   
*is-neg*  $L \implies$   
 $D \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef} \implies$   
 $D \prec_c C \implies$   
*ord-res.production*  $(fset (N \mid \cup \mid U_{er} \mid \cup \mid U_{ef})) D = \{atm-of L\} \implies$   
 $U_{er}' = \text{finsert } (eres D C) U_{er} \implies$   
*ord-res-3*  $N (U_{er}, U_{ef}) (U_{er}', U_{ef})$

**inductive** *ord-res-3-step* **where**

*ord-res-3*  $N s s' \implies$  *ord-res-3-step*  $(N, s) (N, s')$

**inductive** *ord-res-3-final* **where**

*ord-res-final*  $(N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}) \implies$  *ord-res-3-final*  $(N, (U_{rr}, U_{ef}))$

**inductive** *ord-res-3-load* **where**

$N \neq \{\mid\} \implies$  *ord-res-3-load*  $N (N, (\{\mid\}, \{\mid\}))$

**sublocale** *ord-res-3-semantic: semantics* **where**

*step* = *ord-res-3-step* **and**

*final* = *ord-res-3-final*

**proof** *unfold-locales*

**fix**  $S3 :: 'f gterm clause fset \times 'f gterm clause fset \times 'f gterm clause fset$

**obtain**  $N U_{rr} U_{ef} :: 'f gterm clause fset$  **where**

*S3-def:*  $S3 = (N, (U_{rr}, U_{ef}))$

**by** (*metis prod.exhaust*)

**assume** *ord-res-3-final*  $S3$

**hence**  $\{\#\} \mid \in \mid N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef} \vee \neg$  *ex-false-clause*  $(fset (N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}))$

**by** (*simp add: S3-def ord-res-3-final.simps ord-res-final-def*)

**thus** *finished* *ord-res-3-step*  $S3$

**proof** (*elim disjE*)

**assume**  $\{\#\} \mid \in \mid N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}$   
**hence** *is-least-false-clause*  $(N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}) \{\#\}$   
**using** *is-least-false-clause-mempty* **by** *metis*  
**hence**  $\nexists C L. \text{is-least-false-clause } (N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}) C \wedge \text{linorder-lit.is-maximal-in-mset}$   
*C L*  
**by** (*metis Uniq-D Uniq-is-least-false-clause bot-fset.rep-eq fBex-fempty*  
*linorder-lit.is-maximal-in-mset-iff set-mset-empty*)  
**hence**  $\nexists x. \text{ord-res-3 } N (U_{rr}, U_{ef}) x$   
**by** (*auto simp: S3-def elim: ord-res-3.cases*)  
**thus** *?thesis*  
**by** (*simp add: finished-def ord-res-3-step.simps S3-def*)  
**next**  
**assume**  $\neg \text{ex-false-clause } (fset (N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}))$   
**hence**  $\nexists C. \text{is-least-false-clause } (N \mid \cup \mid U_{rr} \mid \cup \mid U_{ef}) C$   
**unfolding** *ex-false-clause-def is-least-false-clause-def*  
**by** (*metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD(1)*  
*linorder-cls.is-least-in-fset-ffilterD(2)*)  
**hence**  $\nexists x. \text{ord-res-3 } N (U_{rr}, U_{ef}) x$   
**by** (*auto simp: S3-def elim: ord-res-3.cases*)  
**thus** *?thesis*  
**by** (*simp add: finished-def ord-res-3-step.simps S3-def*)  
**qed**  
**qed**

**sublocale** *ord-res-3-language: language where*  
*step = ord-res-3-step and*  
*final = ord-res-3-final and*  
*load = ord-res-3-load*  
**by** *unfold-locales*

**lemma** *is-least-false-clause-conv-if-partial-resolution-invariant:*

**assumes**  $\forall C \mid \in \mid U_{pr}. \exists D1 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}. \exists D2 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}.$   
 $(\text{ground-resolution } D1)^{++} D2 C \wedge C \neq \text{eres } D1 D2 \wedge \text{eres } D1 D2 \mid \in \mid U_{er}$   
**shows** *is-least-false-clause*  $(N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef}) = \text{is-least-false-clause}$   
 $(N \mid \cup \mid U_{er} \mid \cup \mid U_{ef})$   
**proof** –  
**have** *is-least-false-clause*  $(N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef}) = \text{is-least-false-clause } (N$   
 $\mid \cup \mid U_{er} \mid \cup \mid U_{ef} \mid \cup \mid U_{pr})$   
**by** (*simp add: sup-commute sup-left-commute*)  
**also have**  $\dots = \text{is-least-false-clause } (N \mid \cup \mid U_{er} \mid \cup \mid U_{ef})$   
**proof** (*rule is-least-false-clause-union-cancel-right*)  
**show**  $\forall C \mid \in \mid U_{pr}. \forall U. \text{ord-res.production } U C = \{\}$   
**proof** (*intro ballI*)  
**fix** *C*  
**assume**  $C \mid \in \mid U_{pr}$   
**hence**  $\exists D \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}. (\exists C'. \text{ground-resolution } D C C')$   
**using** *assms* **by** (*metis eres-eq-after-tranclp-ground-resolution resolvable-if-neq-eres*)  
**hence**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L C$   
**using** *nex-strictly-maximal-pos-lit-if-resolvable* **by** *metis*

```

thus  $\forall U. \text{ord-res.production } U \ C = \{\}$ 
using unproductive-if-nex-strictly-maximal-pos-lit by metis
qed
next
show  $\forall C \in | U_{pr}. \exists D \in | N \cup | U_{er} \cup | U_{ef}. D \prec_c C \wedge \{D\} \models_e \{C\}$ 
proof (intro ballI)
  fix  $C$ 
  assume  $C \in | U_{pr}$ 
  then obtain  $D1 \ D2$  where
     $D1 \in | N \cup | U_{er} \cup | U_{ef}$  and
     $D2 \in | N \cup | U_{er} \cup | U_{ef}$  and
    (ground-resolution  $D1$ )++  $D2 \ C$  and
     $C \neq \text{eres } D1 \ D2$  and
     $\text{eres } D1 \ D2 \in | U_{er}$ 
  using assms by metis

  have  $\text{eres } D1 \ D2 = \text{eres } D1 \ C$ 
  using  $\langle (\text{ground-resolution } D1)^{++} \ D2 \ C \rangle$  eres-eq-after-tranclp-ground-resolution
by metis

  show  $\exists D \in | N \cup | U_{er} \cup | U_{ef}. D \prec_c C \wedge \{D\} \models_e \{C\}$ 
  proof (intro bexI conjI)
    have  $\text{eres } D1 \ C \preceq_c C$ 
    using eres-le .
    thus  $\text{eres } D1 \ D2 \prec_c C$ 
    using  $\langle C \neq \text{eres } D1 \ D2 \rangle \langle \text{eres } D1 \ D2 = \text{eres } D1 \ C \rangle$  by order
  next
  show  $\{\text{eres } D1 \ D2\} \models_e \{C\}$ 
  using  $\langle (\text{ground-resolution } D1)^{++} \ D2 \ C \rangle$  eres-entails-resolvent by metis
  next
  show  $\text{eres } D1 \ D2 \in | N \cup | U_{er} \cup | U_{ef}$ 
  using  $\langle \text{eres } D1 \ D2 \in | U_{er} \rangle$  by simp
  qed
qed
qed
finally show ?thesis .
qed

lemma right-unique-ord-res-3: right-unique (ord-res-3 N)
proof (rule right-uniqueI)
  fix  $s \ s' \ s'' :: 'f \ \text{gterm} \ \text{clause} \ \text{fset} \times 'f \ \text{gterm} \ \text{clause} \ \text{fset}$ 
  assume step1: ord-res-3 N s s' and step2: ord-res-3 N s s''
  show  $s' = s''$ 
  using step1
  proof (cases N s s' rule: ord-res-3.cases)
  case hyps1: (factoring Urr1 Uef1 C1 L1 Uef1')
  show ?thesis
  using step2
  proof (cases N s s'' rule: ord-res-3.cases)

```

```

    case (factoring  $U_{rr2}$   $U_{ef2}$   $C2$   $L2$   $U_{ef2}$ ')
    with hyps1 show ?thesis
      by (metis Uniq-D Uniq-is-least-false-clause prod.inject)
  next
    case (resolution  $U_{rr2}$   $U_{ef2}$   $C2$   $L2$   $D2$   $U_{rr2}$ ')
    with hyps1 have False
    by (metis Pair-inject Uniq-is-least-false-clause linorder-lit.Uniq-is-maximal-in-mset
        the1-equality')
    thus ?thesis ..
  qed
next
case hyps1: (resolution  $U_{rr1}$   $U_{ef1}$   $C1$   $L1$   $D1$   $U_{rr1}$ ')
show ?thesis
  using step2
proof (cases  $N$   $s$   $s''$  rule: ord-res-3.cases)
  case (factoring  $U_{rr2}$   $U_{ef2}$   $C2$   $L2$   $U_{ef2}$ ')
  with hyps1 have False
  by (metis Uniq-is-least-false-clause linorder-lit.Uniq-is-maximal-in-mset
      prod.inject the1-equality')
  thus ?thesis ..
next
case hyps2: (resolution  $U_{rr2}$   $U_{ef2}$   $C2$   $L2$   $D2$   $U_{rr2}$ ')

have *:  $U_{rr1} = U_{rr2}$   $U_{ef1} = U_{ef2}$ 
  using hyps1 hyps2 by simp-all

have **:  $C1 = C2$ 
  using hyps1 hyps2
  unfolding *
  by (metis Uniq-is-least-false-clause the1-equality')

have ***:  $L1 = L2$ 
  using hyps1 hyps2
  unfolding * **
  by (metis linorder-lit.Uniq-is-maximal-in-mset the1-equality')

have ****:  $D1 = D2$ 
  using hyps1 hyps2
  unfolding * ** ***
  by (metis linorder-cls.less-irrefl linorder-cls.linorder-cases
      ord-res.less-trm-iff-less-cls-if-mem-production singletonI)

show ?thesis
  using hyps1 hyps2
  unfolding * ** *** ****
  by argo
qed
qed
qed

```

**lemma** *right-unique-ord-res-3-step: right-unique ord-res-3-step*  
**proof** (rule *right-uniqueI*)  
    **fix**  $x\ y\ z$   
    **show**  $\text{ord-res-3-step } x\ y \implies \text{ord-res-3-step } x\ z \implies y = z$   
        **apply** (cases  $x$ ; cases  $y$ ; cases  $z$ )  
        **apply** (simp add: *ord-res-3-step.simps*)  
        **using** *right-unique-ord-res-3[THEN right-uniqueD]*  
        **by** *blast*  
**qed**

**lemma** *ex-ord-res-3-if-not-final:*  
    **assumes**  $\neg \text{ord-res-3-final } S$   
    **shows**  $\exists S'. \text{ord-res-3-step } S\ S'$   
**proof** –  
    **from** *assms* **obtain**  $N\ U_r\ U_{ef}$  **where**  
         $S\text{-def}: S = (N, (U_r, U_{ef}))$  **and**  
         $\{\#\} \notin N \mid \cup U_r \mid \cup U_{ef}$  **and**  
         $\text{ex-false-clause } (\text{fset } (N \mid \cup U_r \mid \cup U_{ef}))$   
        **by** (*metis ord-res-3-final.intros ord-res-final-def surj-pair*)

**obtain**  $C$  **where**  $C\text{-least-false: is-least-false-clause } (N \mid \cup U_r \mid \cup U_{ef})\ C$   
    **using**  $\langle \text{ex-false-clause } (\text{fset } (N \mid \cup U_r \mid \cup U_{ef})) \rangle$  *obtains-least-false-clause-if-ex-false-clause*  
    **by** *metis*

**hence**  $C \neq \{\#\}$   
    **using**  $\langle \{\#\} \notin N \mid \cup U_r \mid \cup U_{ef} \rangle$   
    **unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
    **by** *metis*

**then obtain**  $L$  **where**  $C\text{-max: linorder-lit.is-maximal-in-mset } C\ L$   
    **using** *linorder-lit.ex-maximal-in-mset* **by** *metis*

**show** *?thesis*  
    **proof** (cases  $L$ )  
        **case** ( $\text{Pos } A$ )  
        **thus** *?thesis*  
            **using** *ord-res-3.factoring[OF C-least-false C-max] S-def is-pos-def*  
            **by** (*metis ord-res-3-step.intros*)

**next**  
        **case** ( $\text{Neg } A$ )  
        **then obtain**  $D$  **where**  
             $D \in N \mid \cup U_r \mid \cup U_{ef}$  **and**  
             $D \prec_c C$  **and**  
             $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A)\ D$  **and**  
             $\text{ord-res.production } (\text{fset } (N \mid \cup U_r \mid \cup U_{ef}))\ D = \{A\}$   
            **using** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*  
            **using**  $C\text{-least-false } C\text{-max}$  **by** *metis*

```

moreover then obtain DC where
  full-run (ground-resolution D) C DC
  using ex-ground-resolutionI C-max Neg
  using ex1-eres-eq-full-run-ground-resolution by blast

ultimately show ?thesis
  using ord-res-3.resolution[OF C-least-false C-max]
  by (metis Neg S-def literal.disc(2) literal.sel(2) ord-res-3-step.intros)
qed
qed

corollary ord-res-3-step-safe: ord-res-3-final S ∨ (∃ S'. ord-res-3-step S S')
  using ex-ord-res-3-if-not-final by metis

end

end
theory Implicit-Exhaustive-Factorization
  imports
    Exhaustive-Factorization
    Exhaustive-Resolution
begin

```

## 18 Function for implicit full factorization

```

context simulation-SCLFOL-ground-ordered-resolution begin

```

```

definition iefac where
  iefac F C = (if C |∈| F then efac C else C)

```

```

lemma iefac-empty[simp]:
  fixes F :: 'f gclause fset
  shows iefac F {#} = {#}
  by (metis efac-empty iefac-def)

```

```

lemma fset-mset-iefac[simp]:
  fixes F :: 'f gclause fset and C :: 'f gclause
  shows fset-mset (iefac F C) = fset-mset C

```

```

proof (cases C |∈| F)
  case True
  hence iefac F C = efac C
  unfolding iefac-def by simp
  thus ?thesis
  by auto
next
  case False
  hence iefac F C = C
  unfolding iefac-def by simp
  thus ?thesis by simp

```

qed

**lemma** *atms-of-cls-iefac[simp]*:

**fixes**  $\mathcal{F} :: 'f\ gclause\ fset$  **and**  $C :: 'f\ gclause$   
**shows** *atms-of-cls (iefac  $\mathcal{F}$   $C$ ) = atms-of-cls  $C$*   
**by** (*simp add: atms-of-cls-def*)

**lemma** *iefac-le*:

**fixes**  $\mathcal{F} :: 'f\ gclause\ fset$  **and**  $C :: 'f\ gclause$   
**shows** *iefac  $\mathcal{F}$   $C \preceq_c C$*   
**by** (*metis efac-subset iefac-def reflclp-iff subset-implies-reflclp-multp*)

**lemma** *true-cls-iefac-iff[simp]*:

**fixes**  $\mathcal{I} :: 'f\ gterm\ set$  **and**  $\mathcal{F} :: 'f\ gclause\ fset$  **and**  $C :: 'f\ gclause$   
**shows**  $\mathcal{I} \models iefac\ \mathcal{F}\ C \longleftrightarrow \mathcal{I} \models C$   
**by** (*metis iefac-def true-cls-efac-iff*)

**lemma** *funion-funion-eq-funion-funion-fimage-iefac-if*:

**assumes**  *$U_{ef}$ -eq:  $U_{ef} = iefac\ \mathcal{F} \upharpoonright \{C \mid C \in N \cup U_{er}. iefac\ \mathcal{F}\ C \neq C\}$*   
**shows**  $N \cup U_{er} \cup U_{ef} = N \cup U_{er} \cup (iefac\ \mathcal{F} \upharpoonright (N \cup U_{er}))$

**proof** (*intro fsubset-antisym fsubsetI*)

**fix**  $C :: 'f\ gterm\ clause$

**assume**  $C \in N \cup U_{er} \cup U_{ef}$

**hence**  $C \in N \cup U_{er} \vee C \in U_{ef}$

**by** *simp*

**thus**  $C \in N \cup U_{er} \cup (iefac\ \mathcal{F} \upharpoonright (N \cup U_{er}))$

**proof** (*elim disjE*)

**assume**  $C \in N \cup U_{er}$

**thus** *?thesis*

**by** *simp*

**next**

**assume**  $C \in U_{ef}$

**hence**  $C \in iefac\ \mathcal{F} \upharpoonright \{C \mid C \in N \cup U_{er}. iefac\ \mathcal{F}\ C \neq C\}$

**using**  *$U_{ef}$ -eq* **by** *argo*

**then obtain**  $C_0 :: 'f\ gterm\ clause$  **where**

$C_0 \in N \cup U_{er}$  **and**  $iefac\ \mathcal{F}\ C_0 \neq C_0$  **and**  $C = iefac\ \mathcal{F}\ C_0$

**by** *auto*

**thus** *?thesis*

**by** *simp*

qed

next

**fix**  $C :: 'f\ gterm\ clause$

**assume**  $C \in N \cup U_{er} \cup (iefac\ \mathcal{F} \upharpoonright (N \cup U_{er}))$

**hence**  $C \in N \cup U_{er} \vee C \in iefac\ \mathcal{F} \upharpoonright (N \cup U_{er})$

**by** *simp*

**thus**  $C \in N \cup U_{er} \cup U_{ef}$

**proof** (*elim disjE*)

**assume**  $C \in N \cup U_{er}$

**thus** *?thesis*  
**by** *simp*  
**next**  
**assume**  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**then obtain**  $C_0 :: \text{'f gterm clause where}$   
 $C_0 \in | N \mid \cup \mid U_{er}$  **and**  $C = \text{iefac } \mathcal{F} C_0$   
**by** *auto*  
  
**show** *?thesis*  
**proof** (*cases iefac F C<sub>0</sub> = C<sub>0</sub>*)  
**case** *True*  
**hence**  $C = C_0$   
**using**  $\langle C = \text{iefac } \mathcal{F} C_0 \rangle$  **by** *argo*  
**thus** *?thesis*  
**using**  $\langle C_0 \in | N \mid \cup \mid U_{er} \rangle$  **by** *simp*  
**next**  
**case** *False*  
**hence**  $C \in | U_{ef}$   
**unfolding**  $U_{ef}\text{-eq}$   
**using**  $\langle C_0 \in | N \mid \cup \mid U_{er} \rangle$   $\langle C = \text{iefac } \mathcal{F} C_0 \rangle$  **by** *simp*  
**thus** *?thesis*  
**by** *simp*  
**qed**  
**qed**  
**qed**

**lemma** *clauses-for-iefac-are-unproductive:*  
 $\forall C \in | N \mid - \mid \text{iefac } \mathcal{F} \mid \uparrow N. \forall U. \text{ord-res.production } U C = \{\}$   
**proof** (*intro ballI allI*)  
**fix**  $C U$   
**assume**  $C \in | N \mid - \mid \text{iefac } \mathcal{F} \mid \uparrow N$   
**hence**  $C \in | N$  **and**  $C \notin | \text{iefac } \mathcal{F} \mid \uparrow N$   
**by** *simp-all*  
**hence**  $\text{iefac } \mathcal{F} C \neq C$   
**by** (*metis fimage-iff*)  
**hence**  $\text{efac } C \neq C$   
**by** (*metis iefac-def*)  
**hence**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L C$   
**using** *nex-strictly-maximal-pos-lit-if-neq-efac* **by** *metis*  
**thus**  $\text{ord-res.production } U C = \{\}$   
**using** *unproductive-if-nex-strictly-maximal-pos-lit* **by** *metis*  
**qed**

**lemma** *clauses-for-iefac-have-smaller-entailing-clause:*  
 $\forall C \in | N \mid - \mid \text{iefac } \mathcal{F} \mid \uparrow N. \exists D \in | \text{iefac } \mathcal{F} \mid \uparrow N. D \prec_c C \wedge \{D\} \Vdash_e \{C\}$   
**proof** (*intro ballI allI*)  
**fix**  $C$   
**assume**  $C \in | N \mid - \mid \text{iefac } \mathcal{F} \mid \uparrow N$



hence  $C \in N$  and  $C \notin \text{iefac } \mathcal{F} \mid \uparrow N$

by *simp-all*

hence  $\text{iefac } \mathcal{F} C \neq C$

by (*metis fimage-iff*)

hence  $\text{efac } C \neq C$

by (*metis iefac-def*)

show  $\exists D \in \text{iefac } \mathcal{F} \mid \uparrow N. D \prec_c C \wedge \{D\} \models_e \{C\}$

proof (*intro bexI conjI*)

show  $\text{efac } C \prec_c C$  and  $\{\text{efac } C\} \models_e \{C\}$

using *efac-properties-if-not-ident*[*OF*  $\langle \text{efac } C \neq C \rangle$ ] by *simp-all*

next

show  $\text{efac } C \in \text{iefac } \mathcal{F} \mid \uparrow N$

using  $\langle C \in N \rangle \langle \text{iefac } \mathcal{F} C \neq C \rangle$  by (*metis fimageI iefac-def*)

qed

qed

lemma *is-least-false-clause-with-iefac-conv*:

*is-least-false-clause*  $(N \mid \cup U_{er} \mid \cup \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) =$

*is-least-false-clause*  $(\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}))$

using *is-least-false-clause-union-cancel-right-strong*[*OF* *clauses-for-iefac-are-unproductive clauses-for-iefac-have-smaller-entailing-clause*]

by (*simp add: sup-commute*)

lemma *MAGIC4*:

fixes  $N \mathcal{F} \mathcal{F}' U_{er} U_{er}'$

defines

$N1 \equiv \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})$  and

$N2 \equiv \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup U_{er}')$

assumes

*subsets-agree*:  $\{x \in N1. x \prec_c C\} = \{x \in N2. x \prec_c C\}$  and

*is-least-false-clause*  $N1 D$  and

*is-least-false-clause*  $N2 E$  and

$C \prec_c D$

shows  $C \preceq_c E$

proof –

have  $\neg E \prec_c C$

proof (*rule notI*)

assume  $E \prec_c C$

have *is-least-false-clause*  $N2 E \longleftrightarrow \text{is-least-false-clause } \{x \in N2. x \preceq_c E\}$   
 $E$

proof (*rule is-least-false-clause-swap-swap-compatible-fsets*)

show  $\{x \in N2. (\prec_c)^{==} x E\} = \{x \in \{x \in N2. (\prec_c)^{==} x E\}. (\prec_c)^{==} x E\}$

using  $\langle E \prec_c C \rangle$  by *force*

qed

also have  $\dots \longleftrightarrow \text{is-least-false-clause } \{x \in N1. x \preceq_c E\} E$

**proof** (*rule is-least-false-clause-swap-swap-compatible-fsets*)  
**show**  $\{|x \in | \{|x \in | N2. (\prec_c)^{==} x E|\}. (\prec_c)^{==} x E|\} =$   
 $\{|x \in | \{|x \in | N1. (\prec_c)^{==} x E|\}. (\prec_c)^{==} x E|\}$   
**using** *subsets-agree*  $\langle E \prec_c C \rangle$  **by** *fastforce*  
**qed**

**also have**  $\dots \longleftrightarrow$  *is-least-false-clause*  $N1 E$   
**proof** (*rule is-least-false-clause-swap-swap-compatible-fsets*)  
**show**  $\{|x \in | \{|x \in | N1. (\prec_c)^{==} x E|\}. (\prec_c)^{==} x E|\} = \{|x \in | N1. (\prec_c)^{==}$   
 $x E|\}$   
**using**  $\langle E \prec_c C \rangle$  **by** *fastforce*  
**qed**

**finally have** *is-least-false-clause*  $N1 E$   
**using**  $\langle$ *is-least-false-clause*  $N2 E \rangle$  **by** *argo*

**hence**  $D = E$   
**using**  $\langle$ *is-least-false-clause*  $N1 D \rangle$   
**by** (*metis Uniq-is-least-false-clause the1-equality'*)

**thus** *False*  
**using**  $\langle E \prec_c C \rangle \langle C \prec_c D \rangle$  **by** *order*  
**qed**

**thus**  $C \preceq_c E$   
**by** *order*  
**qed**

**lemma** *atms-of-clss-fimage-iefac[simp]*:  
 $atms-of-clss (iefac \mathcal{F} \mid\! \mid N) = atms-of-clss N$   
**proof** –  
**have**  $atms-of-clss (iefac \mathcal{F} \mid\! \mid N) = \text{ffUnion } (atms-of-clss \mid\! \mid iefac \mathcal{F} \mid\! \mid N)$   
**unfolding** *atms-of-clss-def* ..

**also have**  $\dots = \text{ffUnion } ((atms-of-clss \circ iefac \mathcal{F}) \mid\! \mid N)$   
**by** *simp*

**also have**  $\dots = \text{ffUnion } (atms-of-clss \mid\! \mid N)$   
**unfolding** *comp-def atms-of-clss-iefac* ..

**also have**  $\dots = atms-of-clss N$   
**unfolding** *atms-of-clss-def* ..

**finally show** *?thesis* .  
**qed**

**lemma** *atm-of-in-atms-of-clssI*:  
**assumes** *L-in*:  $L \in \# C$  **and** *C-in*:  $C \in | iefac \mathcal{F} \mid\! \mid N$   
**shows** *atm-of*  $L \in | atms-of-clss N$   
**proof** –

**have**  $atm\text{-of } L \mid \in \mid atms\text{-of-cls } C$   
**unfolding**  $atms\text{-of-cls-def}$   
**using**  $L\text{-in}$  **by**  $simp$

**hence**  $atm\text{-of } L \mid \in \mid atms\text{-of-clss } (iefac \mathcal{F} \mid \uparrow \mid N)$   
**unfolding**  $atms\text{-of-clss-def}$   
**using**  $C\text{-in}$  **by**  $(metis fmember-ffUnion-iff)$

**thus**  $atm\text{-of } L \mid \in \mid atms\text{-of-clss } N$   
**by**  $simp$   
**qed**

**lemma**  $clause\text{-almost-almost-definedI}$ :  
**fixes**  $\Gamma D K$   
**assumes**  
 $D\text{-in}: D \mid \in \mid iefac \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**  
 $D\text{-max-lit}: ord\text{-res.is-maximal-lit } K D$  **and**  
 $no\text{-undef-atm}: \neg (\exists A \mid \in \mid atms\text{-of-clss } (N \mid \cup \mid U_{er}). A \prec_t atm\text{-of } K \wedge A \not\in \mid trail\text{-atms } \Gamma)$   
**shows**  $trail\text{-defined-cls } \Gamma \{ \#L \in \# D. L \neq K \wedge L \neq -K \# \}$   
**unfolding**  $trail\text{-defined-cls-def}$   
**proof**  $(intro ballI)$   
**have**  $K \in \# D$  **and**  $lit\text{-in-}D\text{-le-}K: \bigwedge L. L \in \# D \implies L \preceq_l K$   
**using**  $D\text{-max-lit}$   
**unfolding**  $atomize\text{-imp atomize-all atomize-conj linorder-lit.is-maximal-in-mset-iff}$   
**using**  $linorder\text{-lit.leI}$  **by**  $blast$

**fix**  $L :: 'f gterm literal$   
**assume**  $L \in \# \{ \#L \in \# D. L \neq K \wedge L \neq -K \# \}$

**hence**  $L \in \# D$  **and**  $L \neq K$  **and**  $L \neq -K$   
**by**  $simp\text{-all}$

**have**  $atm\text{-of } L \mid \in \mid atms\text{-of-clss } (N \mid \cup \mid U_{er})$   
**using**  $\langle L \in \# D \rangle D\text{-in}$  **by**  $(metis atm\text{-of-in-atms-of-clssI})$

**hence**  $atm\text{-of } L \prec_t atm\text{-of } K \implies atm\text{-of } L \mid \in \mid trail\text{-atms } \Gamma$   
**using**  $no\text{-undef-atm}$  **by**  $metis$

**moreover have**  $atm\text{-of } L \prec_t atm\text{-of } K$   
**using**  $lit\text{-in-}D\text{-le-}K[OF \langle L \in \# D \rangle] \langle L \neq K \rangle \langle L \neq -K \rangle$   
**by**  $(cases L; cases K) simp\text{-all}$

**ultimately show**  $trail\text{-defined-lit } \Gamma L$   
**using**  $trail\text{-defined-lit-iff-trail-defined-atm}$  **by**  $auto$   
**qed**

**lemma**  $clause\text{-almost-definedI}$ :  
**fixes**  $\Gamma D K$

**assumes**  
*D-in*:  $D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
*D-max-lit*: *ord-res.is-maximal-lit*  $K D$  **and**  
*no-undef-atm*:  $\neg (\exists A \in \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma)$  **and**  
*K-defined*: *trail-defined-lit*  $\Gamma K$   
**shows** *trail-defined-cls*  $\Gamma \{ \#Ka \in \# D. Ka \neq K \# \}$   
**using** *clause-almost-almost-definedI*[*OF D-in D-max-lit no-undef-atm*]  
**using** *K-defined*  
**unfolding** *trail-defined-cls-def trail-defined-lit-def*  
**by** (*metis (mono-tags, lifting) mem-Collect-eq set-mset-filter uminus-lit-swap*)

**lemma** *eres-not-in-known-clauses-if-trail-false-cls*:

**fixes**  
 $\mathcal{F} :: 'f \text{ gclause fset}$  **and**  
 $\Gamma :: ('f \text{ gliteral} \times 'f \text{ gclause option}) \text{ list}$   
**assumes**  
 $\Gamma$ -*consistent*: *trail-consistent*  $\Gamma$  **and**  
*clauses-lt-E-true*:  $\forall C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow \text{trail-true-cls } \Gamma C$  **and**  
*eres*  $D E \prec_c E$  **and**  
*trail-false-cls*  $\Gamma (\text{eres } D E)$   
**shows** *eres*  $D E \notin N \mid \cup \mid U_{er}$   
**proof** (*rule notI*)  
**have** *iefac*  $\mathcal{F} (\text{eres } D E) \preceq_c \text{eres } D E$   
**using** *iefac-le* **by** *metis*  
**hence** *iefac*  $\mathcal{F} (\text{eres } D E) \prec_c E$   
**using**  $\langle \text{eres } D E \prec_c E \rangle$  **by** *order*  
  
**moreover assume** *eres*  $D E \in N \mid \cup \mid U_{er}$   
  
**ultimately have** *trail-true-cls*  $\Gamma (\text{iefac } \mathcal{F} (\text{eres } D E))$   
**using** *clauses-lt-E-true*[*rule-format, of iefac*  $\mathcal{F} (\text{eres } D E)$ ]  
**by** (*simp add: iefac-def linorder-lit.is-maximal-in-mset-iff*)  
  
**hence** *trail-true-cls*  $\Gamma (\text{eres } D E)$   
**unfolding** *trail-true-cls-def*  
**by** (*metis fset-fset-mset fset-mset-iefac*)  
  
**thus** *False*  
**using**  $\Gamma$ -*consistent*  $\langle \text{trail-false-cls } \Gamma (\text{eres } D E) \rangle$   
**by** (*metis not-trail-true-cls-and-trail-false-cls*)  
**qed**

**lemma** *no-undefined-atom-le-max-lit-of-false-clause*:

**assumes**  
 $\Gamma$ -*lower-set*: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-clss*  $(N \mid \cup \mid U_{er})$ )  
**and**  
*D-in*:  $D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**

*D-false*: *trail-false-cls*  $\Gamma$  *D* **and**  
*D-max-lit*: *linorder-lit.is-maximal-in-mset* *D* *L*  
**shows**  $\neg (\exists A | \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \preceq_t \text{atm-of } L \wedge A \notin | \text{trail-atms } \Gamma)$   
**proof** –  
**have** *trail-false-lit*  $\Gamma$  *L*  
**using** *D-false* *D-max-lit*  
**unfolding** *trail-false-cls-def* *linorder-lit.is-maximal-in-mset-iff* **by** *simp*  
  
**hence** *trail-defined-lit*  $\Gamma$  *L*  
**unfolding** *trail-false-lit-def* *trail-defined-lit-def* **by** *argo*  
  
**hence** *atm-of* *L*  $| \in |$  *trail-atms*  $\Gamma$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm* .  
  
**thus** *?thesis*  
**using**  *$\Gamma$ -lower-set*  
**using** *linorder-trm.not-in-lower-setI* **by** *blast*  
**qed**

**lemma** *trail-defined-if-no-undef-atom-le-max-lit*:  
**assumes**  
*C-in*:  $C | \in | \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**  
*C-max-lit*: *linorder-lit.is-maximal-in-mset* *C* *K* **and**  
*no-undef-atom-le-K*:  
 $\neg (\exists A | \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \preceq_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma)$   
**shows** *trail-defined-cls*  $\Gamma$  *C*  
**proof** –

**have**  $\bigwedge x. x \in \# C \implies \text{atm-of } x \preceq_t \text{atm-of } K$   
**using** *C-in* *C-max-lit*  
**unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**by** (*metis* *linorder-trm.le-cases* *linorder-trm.le-less-linear* *literal.exhaust-sel*  
*ord-res.less-lit-simps(1)* *ord-res.less-lit-simps(2)* *ord-res.less-lit-simps(3)*  
*ord-res.less-lit-simps(4)*)

**hence**  $\bigwedge x. x \in \# C \implies \text{trail-defined-lit } \Gamma$  *x*  
**using** *C-in* *no-undef-atom-le-K*  
**by** (*meson* *atm-of-in-atms-of-clssI* *trail-defined-lit-iff-trail-defined-atm*)

**thus** *trail-defined-cls*  $\Gamma$  *C*  
**unfolding** *trail-defined-cls-def*  
**by** *metis*  
**qed**

**lemma** *no-undef-atom-le-max-lit-if-lt-false-clause*:  
**assumes**  
 $\Gamma$ -*lower-set*: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-clss* (*N*  $\mid \cup \mid$  *U<sub>er</sub>*))  
**and**  
*D-in*:  $D | \in | \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**

```

    D-false: trail-false-cls  $\Gamma$  D and
    D-max-lit: linorder-lit.is-maximal-in-mset D L and
    C-in:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  and
    C-max-lit: linorder-lit.is-maximal-in-mset C K and
    C-lt:  $C \prec_c D$ 
shows  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \preceq_t \text{atm-of } K \wedge A \not\in \mid \text{trail-atms } \Gamma)$ 
proof –
  have  $K \preceq_t L$ 
    using C-lt C-max-lit D-max-lit
    using linorder-cls.less-imp-not-less linorder-lit.multip-if-maximal-less-that-maximal
      linorder-lit.nle-le by blast

  hence atm-of  $K \preceq_t \text{atm-of } L$ 
    by (cases K; cases L) simp-all

  thus  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \preceq_t \text{atm-of } K \wedge A \not\in \mid \text{trail-atms } \Gamma)$ 
    using no-undefined-atom-le-max-lit-of-false-clause[OF assms(1,2,3,4)]
    by fastforce
qed

lemma bex-trail-false-cls-simp:
  fixes  $\mathcal{F} N \Gamma$ 
  shows  $fBex (\text{iefac } \mathcal{F} \mid \uparrow \mid N) (\text{trail-false-cls } \Gamma) \longleftrightarrow fBex N (\text{trail-false-cls } \Gamma)$ 
proof (rule iffI ; elim bexE)
  fix  $C :: \text{'f gclause}$ 
  assume C-in:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid N$  and C-false: trail-false-cls  $\Gamma$  C
  thus  $fBex N (\text{trail-false-cls } \Gamma)$ 
    by (smt (verit, ccfv-SIG) fimage-iff iefac-def set-mset-efac trail-false-cls-def)
next
  fix  $C :: \text{'f gclause}$ 
  assume  $C \mid \in \mid N$  and trail-false-cls  $\Gamma$  C
  thus  $fBex (\text{iefac } \mathcal{F} \mid \uparrow \mid N) (\text{trail-false-cls } \Gamma)$ 
    by (metis fimageI iefac-def set-mset-efac trail-false-cls-def)
qed

end

end

theory ORD-RES-4
  imports
    ORD-RES
    Implicit-Exhaustive-Factorization
    Exhaustive-Resolution
begin

```

## 19 ORD-RES-4 (implicit factorization)

```

context simulation-SCLFOL-ground-ordered-resolution begin

```

**inductive ord-res-4 where**

*factoring:*

$NN = \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \implies$   
 $\text{is-least-false-clause } NN \ C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C \ L \implies$   
 $\text{is-pos } L \implies$   
 $\mathcal{F}' = \text{finsert } C \ \mathcal{F} \implies$   
 $\text{ord-res-4 } N \ (U_{er}, \mathcal{F}) \ (U_{er}, \mathcal{F}') \mid$

*resolution:*

$NN = \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \implies$   
 $\text{is-least-false-clause } NN \ C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C \ L \implies$   
 $\text{is-neg } L \implies$   
 $D \mid \in \mid NN \implies$   
 $D \prec_c C \implies$   
 $\text{ord-res.production (fset } NN) \ D = \{\text{atm-of } L\} \implies$   
 $U_{er}' = \text{finsert } (\text{eres } D \ C) \ U_{er} \implies$   
 $\text{ord-res-4 } N \ (U_{er}, \mathcal{F}) \ (U_{er}', \mathcal{F})$

**inductive ord-res-4-step where**

$\text{ord-res-4 } N \ s \ s' \implies \text{ord-res-4-step } (N, s) \ (N, s')$

**inductive ord-res-4-final where**

$\text{ord-res-final } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \implies \text{ord-res-4-final } (N, U_{er}, \mathcal{F})$

**sublocale ord-res-4-semantic: semantics where**

$\text{step} = \text{ord-res-4-step}$  **and**

$\text{final} = \text{ord-res-4-final}$

**proof** *unfold-locales*

**fix**  $S_4 :: 'f \text{ gterm clause fset} \times 'f \text{ gterm clause fset} \times 'f \text{ gterm clause fset}$

**obtain**  $N \ U_{er} \ \mathcal{F} :: 'f \text{ gterm clause fset}$  **where**

$S_4\text{-def}: S_4 = (N, (U_{er}, \mathcal{F}))$

**by** (*metis prod.exhaust*)

**assume**  $\text{ord-res-4-final } S_4$

**hence**  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \vee \neg \text{ex-false-clause } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$

**by** (*simp add: S4-def ord-res-4-final.simps ord-res-final-def*)

**thus** *finished*  $\text{ord-res-4-step } S_4$

**proof** (*elim disjE*)

**assume**  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$

**hence**  $\text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \ \{\#\}$

**using**  $\text{is-least-false-clause-mempty}$  **by** *metis*

**hence**  $\nexists C \ L. \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \ C \wedge \text{linorder-lit.is-maximal-in-mset } C \ L$

**by** (*metis Uniq-D Uniq-is-least-false-clause bot-fset.rep-eq fBex-fempty linorder-lit.is-maximal-in-mset-iff set-mset-empty*)

```

hence  $\nexists x. \text{ord-res-4 } N (U_{er}, \mathcal{F}) x$ 
  by (auto simp: S4-def elim: ord-res-4.cases)
thus ?thesis
  by (simp add: S4-def finished-def ord-res-4-step.simps)
next
assume  $\neg \text{ex-false-clause } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$ 
hence  $\nexists C. \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) C$ 
  unfolding ex-false-clause-def is-least-false-clause-def
  by (metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD(1)
    linorder-cls.is-least-in-fset-ffilterD(2))
hence  $\nexists x. \text{ord-res-4 } N (U_{er}, \mathcal{F}) x$ 
  by (auto simp: S4-def elim: ord-res-4.cases)
thus ?thesis
  by (simp add: S4-def finished-def ord-res-4-step.simps)
qed
qed

lemma right-unique-ord-res-4: right-unique (ord-res-4 N)
proof (rule right-uniqueI)
fix s s' s'' :: 'f gterm clause fset  $\times$  'f gterm clause fset
assume step1: ord-res-4 N s s' and step2: ord-res-4 N s s''
show s' = s''
  using step1
proof (cases N s s' rule: ord-res-4.cases)
case hyps1: (factoring NN1  $\mathcal{F}1 U_{er}1 C1 L1 \mathcal{F}'$ )
show ?thesis
  using step2
proof (cases N s s'' rule: ord-res-4.cases)
case (factoring NN2  $\mathcal{F}2 U_{er}2 C2 L2 \mathcal{F}'$ )
with hyps1 show ?thesis
  by (metis Uniq-D Uniq-is-least-false-clause prod.inject)
next
case (resolution NN2  $\mathcal{F}2 U_{er}2 C2 L2 D2 U_{er}2'$ )
with hyps1 have False
by (metis Pair-inject Uniq-is-least-false-clause linorder-lit.Uniq-is-maximal-in-mset
  the1-equality')
thus ?thesis ..
qed
next
case hyps1: (resolution NN1  $\mathcal{F}1 U_{er}1 C1 L1 D1 U_{er}1'$ )
show ?thesis
  using step2
proof (cases N s s'' rule: ord-res-4.cases)
case (factoring NN  $\mathcal{F} U_{er} C L \mathcal{F}'$ )
with hyps1 have False
by (metis Pair-inject Uniq-is-least-false-clause linorder-lit.Uniq-is-maximal-in-mset
  the1-equality')
thus ?thesis ..
next

```



```

case (resolution NN  $\mathcal{F}$   $U_{er}$  C L D  $U_{er}'$ )
with hyps1 show ?thesis
  by (metis (mono-tags, lifting) Uniq-D Uniq-is-least-false-clause
    ord-res.less-trm-iff-less-cls-if-mem-production insertI1 linorder-cls.neq-iff
    linorder-lit.Uniq-is-maximal-in-mset prod.inject)
qed
qed
qed

```

```

lemma right-unique-ord-res-4-step: right-unique ord-res-4-step
proof (rule right-uniqueI)
  fix x y z
  show ord-res-4-step x y  $\implies$  ord-res-4-step x z  $\implies$  y = z
    using right-unique-ord-res-4[THEN right-uniqueD]
    by (elim ord-res-4-step.cases) (metis prod.inject)
qed

```

```

lemma ex-ord-res-4-if-not-final:
  assumes  $\neg$  ord-res-4-final S
  shows  $\exists S'$ . ord-res-4-step S S'
proof –

```

```

  from assms obtain N  $U_{er}$   $\mathcal{F}$  where
    S-def: S = (N, ( $U_{er}$ ,  $\mathcal{F}$ )) and
    {#} | $\notin$ | iefac  $\mathcal{F}$  | $\uparrow$ | (N | $\cup$ |  $U_{er}$ ) and
    ex-false-clause (fset (iefac  $\mathcal{F}$  | $\uparrow$ | (N | $\cup$ |  $U_{er}$ )))
  by (metis ord-res-4-final.intros ord-res-final-def surj-pair)

```

```

obtain C where C-least-false: is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$ | (N | $\cup$ |  $U_{er}$ )) C
  using  $\langle$ ex-false-clause (fset (iefac  $\mathcal{F}$  | $\uparrow$ | (N | $\cup$ |  $U_{er}$ ))) $\rangle$ 
    obtains-least-false-clause-if-ex-false-clause
  by metis

```

```

hence C  $\neq$  {#}
  using  $\langle$ {#} | $\notin$ | iefac  $\mathcal{F}$  | $\uparrow$ | (N | $\cup$ |  $U_{er}$ ) $\rangle$ 
  unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
  by metis

```

```

then obtain L where C-max: linorder-lit.is-maximal-in-mset C L
  using linorder-lit.ex-maximal-in-mset by metis

```

```

show ?thesis
proof (cases L)
  case (Pos A)
  thus ?thesis
    using ord-res-4.factoring[OF refl C-least-false C-max] S-def is-pos-def
    by (metis ord-res-4-step.intros)
next
  case (Neg A)
  then obtain D where

```

$D \models \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $D \prec_c C$  **and**  
 $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) D$  **and**  
 $\text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D = \{A\}$   
**using** *lex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*  
**using** *C-least-false C-max* **by** *metis*

**thus** *?thesis*  
**using** *ord-res-4.resolution[OF refl C-least-false C-max]*  
**by** (*metis Neg S-def literal.disc(2) literal.sel(2) ord-res-4-step.simps*)

**qed**  
**qed**

**corollary** *ord-res-4-step-safe: ord-res-4-final S  $\vee$  ( $\exists S'$ . ord-res-4-step S S')*  
**using** *ex-ord-res-4-if-not-final* **by** *metis*

**end**

**end**

**theory** *ORD-RES-5*

**imports**

*Background*

*Implicit-Exhaustive-Factorization*

*Exhaustive-Resolution*

**begin**

## 20 ORD-RES-5 (explicit model construction)

**type-synonym** *'f ord-res-5 = 'f gclause fset  $\times$  'f gclause fset  $\times$  'f gclause fset  $\times$*   
*('f gterm  $\Rightarrow$  'f gclause option)  $\times$  'f gclause option*

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-5* **where**

*skip:*

$(\text{dom } \mathcal{M}) \models C \Longrightarrow$

$C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } \{|D \models \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$

$C \prec_c D|\}) \Longrightarrow$

$\text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}, C') \mid$

*production:*

$\neg (\text{dom } \mathcal{M}) \models C \Longrightarrow$

$\text{linorder-lit.is-maximal-in-mset } C L \Longrightarrow$

$\text{is-pos } L \Longrightarrow$

$\text{linorder-lit.is-greatest-in-mset } C L \Longrightarrow$

$\mathcal{M}' = \mathcal{M}(\text{atm-of } L := \text{Some } C) \Longrightarrow$

$C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } \{|D \models \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$

$C \prec_c D|\}) \Longrightarrow$

$\text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}', C') \mid$

*factoring:*

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\text{is-pos } L \implies$   
 $\neg \text{linorder-lit.is-greatest-in-mset } C L \implies$   
 $\mathcal{F}' = \text{finsert } C \mathcal{F} \implies$   
 $\mathcal{M}' = (\lambda-. \text{None}) \implies$   
 $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}))) \implies$   
 $\text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}', \mathcal{M}', C') \mid$

*resolution:*

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\text{is-neg } L \implies$   
 $\mathcal{M} (\text{atm-of } L) = \text{Some } D \implies$   
 $U_{er}' = \text{finsert } (\text{eres } D C) U_{er} \implies$   
 $\mathcal{M}' = (\lambda-. \text{None}) \implies$   
 $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}'))) \implies$   
 $\text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}', \mathcal{F}, \mathcal{M}', C')$

**inductive** *ord-res-5-step* :: 'f ord-res-5  $\Rightarrow$  'f ord-res-5  $\Rightarrow$  bool **where**  
*ord-res-5*  $N s s' \implies \text{ord-res-5-step } (N, s) (N, s')$

**lemma** *tranclp-ord-res-5-step-if-tranclp-ord-res-5*:  
 $(\text{ord-res-5 } N)^{++} s s' \implies \text{ord-res-5-step}^{++} (N, s) (N, s')$   
**by** (*induction s' rule: tranclp-induct*)  
(*auto intro: ord-res-5-step.intros tranclp.intros*)

**inductive** *ord-res-5-final* :: 'f ord-res-5  $\Rightarrow$  bool **where**  
*model-found*:  
*ord-res-5-final*  $(N, U_{er}, \mathcal{F}, \mathcal{M}, \text{None}) \mid$

*contradiction-found*:

*ord-res-5-final*  $(N, U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } \{\#\})$

**sublocale** *ord-res-5-antics*: *antics* **where**

*step* = *ord-res-5-step* **and**

*final* = *ord-res-5-final*

**proof** *unfold-locals*

**fix** *S5* :: 'f gclause fset  $\times$  'f gclause fset  $\times$  'f gclause fset  $\times$   
( 'f gterm  $\Rightarrow$  'f gclause option )  $\times$  'f gclause option

**obtain**

$N U_{er} \mathcal{F} ::$  'f gterm clause fset **and**

$\mathcal{M} ::$  'f gterm  $\Rightarrow$  'f gclause option **and**

$\mathcal{C} ::$  'f gclause option **where**

*S5-def*:  $S5 = (N, (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}))$

**by** (*metis prod.exhaust*)

```

assume ord-res-5-final S5
hence  $C = \text{None} \vee C = \text{Some } \{\#\}$ 
  by (simp add: S5-def ord-res-5-final.simps)
hence  $\nexists x. \text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, C) x$ 
  by (auto simp: linorder-lit.is-maximal-in-mset-iff elim: ord-res-5.cases)
thus finished ord-res-5-step S5
  by (simp add: S5-def finished-def ord-res-5-step.simps)
qed

```

**lemma** *right-unique-ord-res-5: right-unique (ord-res-5 N)*

**proof** (*rule right-uniqueI*)

**fix**  $s s' s''$

**assume** *step1: ord-res-5 N s s'* **and** *step2: ord-res-5 N s s''*

**show**  $s' = s''$

**using** *step1*

**proof** (*cases N s s' rule: ord-res-5.cases*)

**case** *hyp1: (skip M1 C1 C1' F1 U1<sub>er</sub>)*

**with** *step2* **show** *?thesis*

**by** (*cases N s s'' rule: ord-res-5.cases*) *simp-all*

**next**

**case** *hyp1: (production M1 C1 L1 M1' C1' F1 U1<sub>er</sub>)*

**show** *?thesis*

**using** *step2*

**proof** (*cases N s s'' rule: ord-res-5.cases*)

**case** (*skip M2 C2 C2' F2 U2<sub>er</sub>*)

**with** *hyp1* **show** *?thesis*

**by** *simp*

**next**

**case** *hyp2: (production M2 C2 L2 M2' C2' F2 U2<sub>er</sub>)*

**have**  $C1 = C2$

**using** *hyp1 hyp2* **by** *simp*

**hence**  $L1 = L2$

**using** *hyp1 hyp2*

**by** (*metis linorder-lit.Uniq-is-maximal-in-mset Uniq-D*)

**thus** *?thesis*

**using** *hyp1 hyp2* **by** *simp*

**next**

**case** *hyp2: (factoring M2 C2 L2 F2 F2' M2' C2' U2<sub>er</sub>)*

**have**  $C1 = C2$

**using** *hyp1 hyp2* **by** *simp*

**hence**  $L1 = L2$

**using** *hyp1 hyp2*

**by** (*metis linorder-lit.Uniq-is-maximal-in-mset Uniq-D*)

**thus** *?thesis*

**using** *hyp1 hyp2* **by** *simp*

**next**

**case** *hyp2: (resolution M2 C2 L2 D2 U2<sub>er</sub>' U2<sub>er</sub> M2' C2' F2)*

**have**  $C1 = C2$

```

    using hyps1 hyps2 by simp
  hence L1 = L2
    using hyps1 hyps2
    by (metis linorder-lit.Uniq-is-maximal-in-mset Uniq-D)
  thus ?thesis
    using hyps1 hyps2 by simp
qed
next
case hyps1: (factoring M1 C1 L1 F1 F1' M1' C1' U1er)
show ?thesis
  using step2
proof (cases N s s'' rule: ord-res-5.cases)
  case (skip M2 C2 C2' F2 U2er)
  with hyps1 show ?thesis
    by simp
next
case hyps2: (production M2 C2 L2 M2' C2' F2 U2er)
have C1 = C2
  using hyps1 hyps2 by simp
hence L1 = L2
  using hyps1 hyps2
  by (metis linorder-lit.Uniq-is-maximal-in-mset Uniq-D)
thus ?thesis
  using hyps1 hyps2 by simp
next
case hyps2: (factoring M2 C2 L2 F2 F2' M2' C2' U2er)
have C1 = C2
  using hyps1 hyps2 by simp
thus ?thesis
  using hyps1 hyps2 by simp
next
case hyps2: (resolution M2 C2 L2 D2 U2er' U2er M2' C2' F2)
have C1 = C2
  using hyps1 hyps2 by simp
hence L1 = L2
  using hyps1 hyps2
  by (metis linorder-lit.Uniq-is-maximal-in-mset Uniq-D)
hence False
  using hyps1 hyps2 by argo
thus ?thesis ..
qed
next
case hyps1: (resolution M1 C1 L1 D1 U1er' U1er M1' C1' F1)
show ?thesis
  using step2
proof (cases N s s'' rule: ord-res-5.cases)
  case (skip M2 C2 C2' F2 U2er)
  with hyps1 show ?thesis
    by simp

```

```

next
  case hyps2: (production M2 C2 L2 M2' C2' F2 U2er)
  have C1 = C2
    using hyps1 hyps2 by simp
  hence L1 = L2
    using hyps1 hyps2
    by (metis linorder-lit.Uniq-is-maximal-in-mset Uniq-D)
  thus ?thesis
    using hyps1 hyps2 by simp
next
  case hyps2: (factoring M2 C2 L2 F2 F2' M2' C2' U2er)
  have C1 = C2
    using hyps1 hyps2 by simp
  hence L1 = L2
    using hyps1 hyps2
    by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset)
  hence False
    using hyps1 hyps2 by argo
  thus ?thesis ..
next
  case hyps2: (resolution M2 C2 L2 D2 U2er' U2er M2' C2' F2)
  have U1er = U2er F1 = F2 M1 = M2 C1 = C2
    using hyps1 hyps2 by simp-all
  hence L1 = L2
    using hyps1 hyps2
    by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset)
  hence D1 = D2
    using ⟨M1 (atm-of L1) = Some D1⟩ ⟨M2 (atm-of L2) = Some D2⟩ ⟨M1
= M2⟩
    by simp

  have U1er' = U2er'
    using ⟨U1er' = finsert (eres D1 C1) U1er⟩ ⟨U2er' = finsert (eres D2 C2)
U2er⟩
    ⟨D1 = D2⟩ ⟨C1 = C2⟩ ⟨U1er = U2er⟩
    by argo

  moreover have M1' = M2'
    using ⟨M1' = (λ-. None)⟩ ⟨M2' = (λ-. None)⟩
    by argo

  moreover have C1' = C2'
    using hyps1 hyps2 ⟨F1 = F2⟩ ⟨U1er' = U2er'⟩ by simp

  ultimately show ?thesis
    using ⟨s' = (U1er', F1, M1', C1')⟩ ⟨s'' = (U2er', F2, M2', C2')⟩
    ⟨F1 = F2⟩
    by argo
qed

```

qed  
qed

**lemma** *right-unique-ord-res-5-step: right-unique ord-res-5-step*

**proof** (rule *right-uniqueI*)

fix  $x y z$

show  $\text{ord-res-5-step } x y \implies \text{ord-res-5-step } x z \implies y = z$

using *right-unique-ord-res-5[THEN right-uniqueD]*

by (*elim ord-res-5-step.cases*) (*metis prod.inject*)

qed

**definition** *next-clause-in-factorized-clause* **where**

*next-clause-in-factorized-clause*  $N s \longleftrightarrow$

$(\forall U_{er} \mathcal{F} \mathcal{M} C. s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \longrightarrow C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}))$

**lemma** *next-clause-in-factorized-clause:*

**assumes** *step: ord-res-5*  $N s s'$

**shows** *next-clause-in-factorized-clause*  $N s'$

**using** *step*

**proof** (*cases N s s' rule: ord-res-5.cases*)

**case** (*skip*  $\mathcal{M} C C' \mathcal{F} U_{er}$ )

**thus** *?thesis*

**unfolding** *next-clause-in-factorized-clause-def*

**by** (*metis Pair-inject The-optional-eq-SomeD linorder-clis.is-minimal-in-fset-eq-is-least-in-fset linorder-clis.is-minimal-in-fset-filter-iff*)

**next**

**case** (*production*  $\mathcal{M} C L M' C' \mathcal{F} U_{er}$ )

**thus** *?thesis*

**unfolding** *next-clause-in-factorized-clause-def*

**by** (*metis Pair-inject The-optional-eq-SomeD linorder-clis.is-minimal-in-fset-eq-is-least-in-fset linorder-clis.is-minimal-in-fset-filter-iff*)

**next**

**case** (*factoring*  $\mathcal{M} C L \mathcal{F}' \mathcal{F} M' C' U_{er}$ )

**thus** *?thesis*

**unfolding** *next-clause-in-factorized-clause-def*

**by** (*metis Pair-inject The-optional-eq-SomeD linorder-clis.is-least-in-fset-iff*)

**next**

**case** (*resolution*  $\mathcal{M} C L D U_{er}' U_{er} M' C' \mathcal{F}$ )

**thus** *?thesis*

**unfolding** *next-clause-in-factorized-clause-def*

**by** (*metis Pair-inject The-optional-eq-SomeD linorder-clis.is-least-in-fset-iff*)

qed

**definition** *implicitly-factorized-clauses-subset* **where**

*implicitly-factorized-clauses-subset*  $N s \longleftrightarrow$

$(\forall U_{er} \mathcal{F} \mathcal{M} C. s = (U_{er}, \mathcal{F}, \mathcal{M}, C) \longrightarrow \mathcal{F} \subseteq N \cup U_{er})$

**lemma** *ord-res-5-preserves-implicitly-factorized-clauses-subset:*

**assumes**

```

    step: ord-res-5 N s s' and
    invars:
      implicitly-factorized-clauses-subset N s and
      next-clause-in-factorized-clause N s
    shows implicitly-factorized-clauses-subset N s'
    using step
  proof (cases N s s' rule: ord-res-5.cases)
    case (skip M C C' F Uer)
    thus ?thesis
      using invars
      by (simp add: implicitly-factorized-clauses-subset-def)
  next
    case (production M C L M' C' F Uer)
    thus ?thesis
      using invars
      by (simp add: implicitly-factorized-clauses-subset-def)
  next
    case (factoring M C L F' F M' C' Uer)
    thus ?thesis
      using invars
      by (smt (verit) Pair-inject assms(3) fimage-iff finsert-fsubset iefac-def
          implicitly-factorized-clauses-subset-def literal.collapse(1)
          next-clause-in-factorized-clause-def ex1-efac-eq-factoring-chain ex-ground-factoringI)
  next
    case (resolution M C L D Uer' Uer M' C' F)
    thus ?thesis
      using invars
      by (simp add: fsubset-funion-eq implicitly-factorized-clauses-subset-def)
qed

```

**lemma** *interp-eq-Interp-if-least-greater:*

```

  assumes
    C-in: C |∈| NN and
    D-least-gt-C: linorder-cls.is-least-in-fset (ffilter ((<c) C) NN) D
  shows ord-res.interp (fset NN) D = ord-res.interp (fset NN) C ∪ ord-res.production
    (fset NN) C
  proof -
    have nex-between-C-and-D: ¬ (∃ CD |∈| NN. C <c CD ∧ CD <c D)
      using D-least-gt-C
      unfolding linorder-cls.is-least-in-ffilter-iff by auto

    have ord-res.interp (fset NN) D = ord-res.interp (fset {|B |∈| NN. B ≤c C|})
      D
    proof (rule ord-res.interp-swap-clause-set)
      have NN = {|B |∈| NN. B ≤c C|} |∪| {|E |∈| NN. D ≤c E|}
        using nex-between-C-and-D by force
      show {Da. Da |∈| NN ∧ Da <c D} = {Da. Da |∈| {|B |∈| NN. (<c)== B C|}
        ∧ Da <c D}
        using ‹NN = {|B |∈| NN. (<c)== B C|} |∪| ffilter ((<c)== D) NN›

```



*linorder-cls.leD* by auto

**qed**

**also have**  $\dots = \bigcup (ord-res.production (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) \text{ ‘}$   
 $\{Da. Da | \in | \{|B | \in | NN. (\prec_c)^{==} B C|\} \wedge Da \prec_c D\})$   
**unfolding** *ord-res.interp-def* ..

**also have**  $\dots = \bigcup (ord-res.production (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) \text{ ‘}$   
 $\{Da \in fset NN. (\prec_c)^{==} Da C \wedge Da \prec_c D\})$   
**by** *auto*

**also have**  $\dots = \bigcup (ord-res.production (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) \text{ ‘}$   
 $\{Da \in fset NN. (\prec_c)^{==} Da C\})$

**proof** –

**have**  $\{|Da | \in | NN. Da \preceq_c C \wedge Da \prec_c D|\} = \{|Da | \in | NN. Da \preceq_c C|\}$

**using** *nex-between-C-and-D*

**by** (*metis (no-types, opaque-lifting) D-least-gt-C linorder-cls.is-least-in-fset-ffilterD(2)*  
*linorder-cls.le-less-trans*)

**thus** *?thesis*

**by** *fastforce*

**qed**

**also have**  $\dots = \bigcup (ord-res.production (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) \text{ ‘}$   
 $\{Da \in fset NN. Da \prec_c C\}) \cup ord-res.production (fset \{|B | \in | NN. (\prec_c)^{==} B$   
 $C|\}) C$

**proof** –

**have**  $\{Da. Da | \in | NN \wedge (\prec_c)^{==} Da C\} = insert C \{Da. Da | \in | NN \wedge Da \prec_c$   
 $C\}$

**using** *C-in* by auto

**thus** *?thesis*

**by** *blast*

**qed**

**also have**  $\dots = ord-res-Interp (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) C$

**unfolding** *ord-res.interp-def* by auto

**also have**  $\dots = ord-res-Interp (fset NN) C$

**proof** –

**have**  $ord-res.interp (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) C = ord-res.interp (fset$   
 $NN) C$

**using** *ord-res.interp-swap-clause-set*[of  $fset \{|B | \in | NN. (\prec_c)^{==} B C|\} C fset$   
 $NN]$

**by** *auto*

**moreover have**  $ord-res.production (fset \{|B | \in | NN. (\prec_c)^{==} B C|\}) C =$   
 $ord-res.production (fset NN) C$

**using** *ord-res.production-swap-clause-set*[of  $fset \{|B | \in | NN. (\prec_c)^{==} B C|\}$   
 $C fset NN]$

**by** *auto*

ultimately show *?thesis*  
 by *simp*  
 qed

finally show *?thesis* .  
 qed

**lemma** *interp-eq-empty-if-least-in-set*:  
 assumes *linorder-clc.is-least-in-set*  $N$   $C$   
 shows *ord-res.interp*  $N$   $C = \{\}$   
 using *assms*  
 unfolding *ord-res.interp-def*  
 unfolding *linorder-clc.is-least-in-set-iff*  
 by *auto*

**definition** *model-eq-interp-upto-next-clause* **where**  
 $model\text{-}eq\text{-}interp\text{-}upto\text{-}next\text{-}clause\ N\ s \longleftrightarrow$   
 $(\forall U_{er}\ \mathcal{F}\ \mathcal{M}\ C. s = (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C) \longrightarrow$   
 $dom\ \mathcal{M} = ord\text{-}res.interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow|\ (N\ |\cup|\ U_{er})))\ C)$

**lemma** *model-eq-interp-upto-next-clause*:  
 assumes *step*: *ord-res-5*  $N$   $s$   $s'$  **and**  
*invars*:  
 $model\text{-}eq\text{-}interp\text{-}upto\text{-}next\text{-}clause\ N\ s$   
 $next\text{-}clause\text{-}in\text{-}factorized\text{-}clause\ N\ s$   
 shows *model-eq-interp-upto-next-clause*  $N$   $s'$   
 using *step*

**proof** (*cases*  $N$   $s$   $s'$  *rule*: *ord-res-5.cases*)  
 case *step-hyps*: (*skip*  $\mathcal{M}$   $C$   $C'$   $\mathcal{F}$   $U_{er}$ )

**have**  $dom\ \mathcal{M} = ord\text{-}res.interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow|\ (N\ |\cup|\ U_{er})))\ D$  **if**  $C' = Some\ D$  **for**  $D$

**proof** –

**have**  $dom\ \mathcal{M} = ord\text{-}res.interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow|\ (N\ |\cup|\ U_{er})))\ C$   
 using *invars*(1)[*unfolded model-eq-interp-upto-next-clause-def*, *rule-format*,  
 $OF\ \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C) \rangle$ ].

**also have**  $\dots = ord\text{-}res\text{-}Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow|\ (N\ |\cup|\ U_{er})))\ C$

**proof** –

**have** *ord-res.production* (*fset* (*iefac*  $\mathcal{F}$   $|\uparrow|$  ( $N$   $|\cup|$   $U_{er}$ )))  $C = \{\}$   
 using  $\langle dom\ \mathcal{M} \models C \rangle$   
 unfolding *invars*(1)[*unfolded model-eq-interp-upto-next-clause-def*, *rule-format*,  
 $OF\ \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C) \rangle$ ]  
 by (*simp add*: *ord-res.production-unfold*)  
 thus *?thesis*  
 by *simp*

qed

**also have**  $\dots = \text{ord-res.interp } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$   
**proof** (*rule interp-eq-Interp-if-least-greater[symmetric]*)  
**show**  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using** *invars(2)[unfolded next-clause-in-factorized-clause-def, rule-format,*  
 $OF \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$ .  
**next**  
**show** *linorder-clis-least-in-fset (ffilter (( $\prec_c$ ) C) (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   
 $D$   
**using** *step-hyps(3-)* **that by** (*metis Some-eq-The-optionalD*)  
**qed**  
  
**finally show** *?thesis*.  
**qed**  
  
**thus** *?thesis*  
**using** *step-hyps* **by** (*simp add: model-eq-interp-upto-next-clause-def*)  
**next**  
**case** *step-hyps: (production  $\mathcal{M} C L \mathcal{M}' C' \mathcal{F} U_{er}$ )*  
  
**have**  $\text{dom } \mathcal{M}' = \text{ord-res.interp } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$  **if**  $C' = \text{Some } D$   
**for**  $D$   
**proof** –  
**have**  $\text{dom } \mathcal{M}' = \text{dom } \mathcal{M} \cup \{\text{atm-of } L\}$   
**unfolding**  $\langle \mathcal{M}' = \mathcal{M}(\text{atm-of } L \mapsto C) \rangle$  **by** *simp*  
  
**also have**  $\dots = \text{ord-res.interp } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \cup \{\text{atm-of } L\}$   
**unfolding** *invars(1)[unfolded model-eq-interp-upto-next-clause-def, rule-format,*  
 $OF \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$  **]** ..  
  
**also have**  $\dots = \text{ord-res.interp } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \cup$   
 $\text{ord-res.production } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C$   
**proof** –  
**have**  $\neg \text{ord-res.interp } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$   
**using**  $\langle \neg \text{dom } \mathcal{M} \models C \rangle$   
**unfolding** *invars(1)[unfolded model-eq-interp-upto-next-clause-def, rule-format,*  
 $OF \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$  **]**.  
**hence**  $\text{atm-of } L \in \text{ord-res.production } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C$   
**using**  $\langle \text{is-pos } L \rangle \langle \text{ord-res.is-strictly-maximal-lit } L C \rangle$   
**using** *invars(2)[unfolded next-clause-in-factorized-clause-def, rule-format,*  
 $OF \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$   
**unfolding** *ord-res.production-unfold mem-Collect-eq*  
**by** (*metis linorder-lit.is-greatest-in-mset-iff literal.collapse(1) multi-member-split*)  
**hence**  $\text{ord-res.production } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C = \{\text{atm-of } L\}$   
**by** (*metis empty-iff insertE ord-res.production-eq-empty-or-singleton*)  
**thus** *?thesis*  
**by** *argo*  
**qed**  
  
**also have**  $\dots = \text{ord-res.interp } (fset \text{ (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$*

```

proof (rule interp-eq-Interp-if-least-greater[symmetric])
  show  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ 
    using invars(2)[unfolding next-clause-in-factorized-clause-def, rule-format,
      OF  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$ ].
next
  show linorder-clis-least-in-fset (ffilter (( $\prec_c$ ) C) (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ ))
D
  using step-hyps( $\beta$ -) that by (metis Some-eq-The-optionalD)
qed

finally show ?thesis .
qed

thus ?thesis
  using step-hyps by (simp add: model-eq-interp-upto-next-clause-def)
next
case step-hyps: (factoring  $\mathcal{M} \ C \ L \ \mathcal{F}' \ \mathcal{F} \ \mathcal{M}' \ \mathcal{C}' \ U_{er}$ )

  have  $\text{dom } \mathcal{M}' = \text{ord-res.interp (fset (iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) \ D \ \text{if } \mathcal{C}' = \text{Some } D \ \text{for } D$ 
proof -
  have  $\text{dom } \mathcal{M}' = \{\}$ 
    using step-hyps( $\beta$ -) by simp
  also have  $\dots = \text{ord-res.interp (fset (iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) \ D$ 
proof (rule interp-eq-empty-if-least-in-set[symmetric])
  show linorder-clis-least-in-set (fset (iefac  $\mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$ )) D
    using step-hyps( $\beta$ -) that
    by (metis Some-eq-The-optionalD linorder-clis-least-in-fset-iff
      linorder-clis-least-in-set-iff)
qed
  finally show ?thesis .
qed

thus ?thesis
  using step-hyps by (simp add: model-eq-interp-upto-next-clause-def)
next
case step-hyps: (resolution  $\mathcal{M} \ C \ L \ D \ U_{er}' \ U_{er} \ \mathcal{M}' \ \mathcal{C}' \ \mathcal{F}$ )

  have  $\text{dom } \mathcal{M}' = \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')) \ D \ \text{if } \mathcal{C}' = \text{Some } D \ \text{for } D$ 
proof -
  have  $\text{dom } \mathcal{M}' = \{\}$ 
    using step-hyps( $\beta$ -) by simp
  also have  $\dots = \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')) \ D$ 
proof (rule interp-eq-empty-if-least-in-set[symmetric])
  show linorder-clis-least-in-set (fset (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$ )) D
    using step-hyps( $\beta$ -) that
    by (metis Some-eq-The-optionalD linorder-clis-least-in-fset-iff
      linorder-clis-least-in-set-iff)

```

**qed**  
**finally show** *?thesis* .  
**qed**

**thus** *?thesis*  
**using** *step-hyps* **by** (*simp add: model-eq-interp-upto-next-clause-def*)  
**qed**

**definition** *all-smaller-clauses-true-wrt-respective-Interp* **where**  
*all-smaller-clauses-true-wrt-respective-Interp*  $N\ s \longleftrightarrow$   
 $(\forall U_{er}\ \mathcal{F}\ \mathcal{M}\ \mathcal{C}. s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \longrightarrow$   
 $(\forall C\ |\in| iefac\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er}). (\forall D. \mathcal{C} = Some\ D \longrightarrow C \prec_c D) \longrightarrow$   
 $ord-res-Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})))\ C \models C))$

**lemma** *all-smaller-clauses-true-wrt-respective-Interp*:  
**assumes** *step: ord-res-5*  $N\ s\ s'$  **and**  
*invars*:  
*all-smaller-clauses-true-wrt-respective-Interp*  $N\ s$   
*model-eq-interp-upto-next-clause*  $N\ s$   
*next-clause-in-factorized-clause*  $N\ s$   
**shows** *all-smaller-clauses-true-wrt-respective-Interp*  $N\ s'$   
**using** *step*

**proof** (*cases*  $N\ s\ s'$  *rule: ord-res-5.cases*)  
**case** *step-hyps*: (*skip*  $\mathcal{M}\ D\ C'\ \mathcal{F}\ U_{er}$ )

**have** *D-true*: *ord-res-Interp* (*fset* (*iefac*  $\mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$ ))  $D \models D$   
**using** *invars*(2) *ord-res.production-unfold* *step-hyps*(1) *step-hyps*(3)  
**by** (*auto simp: model-eq-interp-upto-next-clause-def*)

**have** *ord-res-Interp* (*fset* (*iefac*  $\mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$ ))  $C \models C$   
**if**  $C' = Some\ E$  **and** *C-in*:  $C\ |\in| iefac\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$  **and**  $C \prec_c E$  **for**  $C\ E$

**proof** –  
**have** *linorder-cls.is-least-in-fset* (*ffilter* ( $(\prec_c)\ D$ ) (*iefac*  $\mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$ ))  $E$   
**using** *step-hyps*  $\langle C' = Some\ E \rangle$  **by** (*metis* *Some-eq-The-optionalD*)  
**hence**  $C \preceq_c D$   
**using** *C-in*  $\langle C \prec_c E \rangle$   
**by** (*metis* *asymptD* *linorder-cls.is-least-in-ffilter-iff* *linorder-cls.le-less-linear* *ord-res.asymp-less-cls*)  
**thus** *?thesis*  
**using** *D-true*  
**using** *invars*(1)[*unfolded all-smaller-clauses-true-wrt-respective-Interp-def*,  
*rule-format*,  
 $OF\ \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, Some\ D) \rangle\ C-in]$  **by** *auto*

**qed**

**moreover** **have** *ord-res-Interp* (*fset* (*iefac*  $\mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$ ))  $C \models C$   
**if**  $C' = None$  **and** *C-in*:  $C\ |\in| iefac\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$  **for**  $C$

**proof** –  
**have**  $\nexists x. linorder-cls.is-least-in-fset\ (ffilter\ ((\prec_c)\ D)\ (iefac\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})))$

$x$   
**using** *step-hyps*  $\langle C' = \text{None} \rangle$   
**by** (*metis* (*no-types*, *opaque-lifting*) *None-eq-The-optionalD linorder-cls.Uniq-is-least-in-fset the1-equality'*)  
**hence**  $\neg (\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). D \prec_c x)$   
**by** (*metis* *emptyE fmember-filter linorder-cls.ex1-least-in-fset*)  
**hence**  $C \prec_c D \vee C = D$   
**using** *C-in* **by force**  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**then show** *?thesis*  
**using** *invars(1)*  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) \rangle$  *C-in*  
**by** (*auto simp: all-smaller-clauses-true-wrt-respective-Interp-def*)  
**next**  
**assume**  $C = D$   
**then show** *?thesis*  
**using** *D-true* **by argo**  
**qed**  
**qed**  
  
**ultimately show** *?thesis*  
**using** *step-hyps*  
**by** (*smt* (*verit*, *best*) *all-smaller-clauses-true-wrt-respective-Interp-def old.prod.inject option.exhaust*)  
**next**  
**case** *step-hyps: (production M D K M' C' F U<sub>er</sub>)*  
  
**have**  $K \in \# D$   
**using** *step-hyps(3-)* **by** (*metis linorder-lit.is-greatest-in-mset-iff*)  
  
**have**  $\neg \text{ord-res.interp} (\text{fset} (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \models D$   
**using**  $\langle \neg \text{dom } \mathcal{M} \models D \rangle$   
**unfolding** *invars(2)[unfolding model-eq-interp-upto-next-clause-def, rule-format, OF  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) \rangle$ ]*.  
**hence** *atm-of*  $K \in \text{ord-res.production} (\text{fset} (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$   
**using**  $\langle \text{is-pos } K \rangle \langle \text{ord-res.is-strictly-maximal-lit } K D \rangle \langle K \in \# D \rangle$   
**using** *invars(3)[unfolding next-clause-in-factorized-clause-def, rule-format, OF  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) \rangle$ ]*  
**unfolding** *ord-res.production-unfold mem-Collect-eq*  
**by** (*metis literal.collapse(1) multi-member-split*)  
**hence** *prod-D-eq: ord-res.production* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D = \{\text{atm-of } K\}$   
**by** (*metis empty-iff insertE ord-res.production-eq-empty-or-singleton*)  
**hence** *ord-res-Interp* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \models_l K$   
**using**  $\langle \text{is-pos } K \rangle$  **by force**  
**hence** *D-true: ord-res-Interp* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \models D$   
**using**  $\langle K \in \# D \rangle$  **by auto**

**have**  $dom \mathcal{M}' = dom \mathcal{M} \cup \{atm-of K\}$   
**unfolding**  $\langle \mathcal{M}' = \mathcal{M}(atm-of K \mapsto D) \rangle$  **by** *simp*

**also have**  $\dots = ord-res.interp (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \cup \{atm-of K\}$   
**unfolding** *invars(2)[unfolding model-eq-interp-upto-next-clause-def, rule-format,*  
 $OF \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, Some D) \rangle$  **]** ..

**also have**  $\dots = ord-res.interp (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \cup$   
 $ord-res.production (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$   
**using** *prod-D-eq* **by** *argo*

**finally have** *dom-M'-eq*:  $dom \mathcal{M}' = ord-res-Interp (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$  .

**have**  $ord-res-Interp (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$   
**if**  $C' = Some E$  **and** *C-in*:  $C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $C \prec_c E$  **for**  $C E$   
**proof** –  
**have** *linorder-cls.is-least-in-fset* (*ffilter* ( $(\prec_c) D$ ) (*iefac*  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ ))  $E$   
**using** *step-hyps*  $\langle C' = Some E \rangle$  **by** (*metis Some-eq-The-optionalD*)  
**hence**  $C \preceq_c D$   
**using** *C-in*  $\langle C \prec_c E \rangle$   
**by** (*metis asympD linorder-cls.is-least-in-ffilter-iff linorder-cls.le-less-linear*  
 $ord-res.asymp-less-cls$ )  
**thus** *?thesis*  
**using** *D-true*  
**using** *invars(1)[unfolding all-smaller-clauses-true-wrt-respective-Interp-def,*  
*rule-format,*  
 $OF \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, Some D) \rangle$  *C-in*] **by** *auto*

**qed**

**moreover have**  $ord-res-Interp (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$   
**if**  $C' = None$  **and** *C-in*:  $C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **for**  $C$   
**proof** –  
**have**  $\nexists x. linorder-cls.is-least-in-fset (ffilter ((\prec_c) D) (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   
 $x$   
**using** *step-hyps*  $\langle C' = None \rangle$   
**by** (*metis (no-types, opaque-lifting) None-eq-The-optionalD linorder-cls.Uniq-is-least-in-fset*  
 $the1-equality'$ )  
**hence**  $\neg (\exists x \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). D \prec_c x)$   
**by** (*metis emptyE ffilter-filter linorder-cls.ex1-least-in-fset*)  
**hence**  $C \prec_c D \vee C = D$   
**using** *C-in* **by** *force*  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**then show** *?thesis*  
**using** *invars(1)*  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, Some D) \rangle$  *C-in*  
**by** (*auto simp: all-smaller-clauses-true-wrt-respective-Interp-def*)  
**next**

```

    assume  $C = D$ 
    thus ?thesis
      using  $D\text{-true}$  by argo
  qed
qed

ultimately show ?thesis
  unfolding  $\text{step-hyps}(2)$   $\text{all-smaller-clauses-true-wrt-respective-Interp-def}$ 
  by ( $\text{metis prod.inject option.exhaust}$ )
next
case  $\text{step-hyps}$ : ( $\text{factoring } \mathcal{M} D K \mathcal{F}' \mathcal{F} \mathcal{M}' C' U_{er}$ )
have  $D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ 
  using  $\text{invars}(2-)$   $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) \rangle$ 
  by ( $\text{metis next-clause-in-factorized-clause-def}$ )
hence  $D \in N \mid \cup \mid U_{er}$ 
  using  $\text{step-hyps}(3-)$ 
  by ( $\text{smt (verit, ccfv-threshold) fimage-iff iefac-def literal.collapse(1)}$ 
     $\text{ex1-efac-eq-factoring-chain ex-ground-factoringI}$ )
hence  $\text{iefac } \mathcal{F}' D \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$ 
  unfolding  $\langle \mathcal{F}' = \text{finsert } D \mathcal{F} \rangle$  by simp
hence  $C' \neq \text{None}$ 
  using  $\text{step-hyps}(3-)$ 
  by ( $\text{metis The-optional-eq-NoneD finsert-not-fempty linorder-cls.ex1-least-in-fset}$ 
     $\text{set-finsert}$ )
then obtain  $E$  where
   $C' = \text{Some } E$ 
  by auto

have  $\neg (\exists C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E)$ 
  using  $\langle C' = \text{Some } E \rangle$   $\text{step-hyps}(9)$ 
  by ( $\text{metis The-optional-eq-SomeD linorder-cls.is-least-in-fset-iff}$ 
     $\text{linorder-cls.less-imp-not-less}$ )

thus ?thesis
  unfolding  $\text{step-hyps}(1,2)$   $\text{all-smaller-clauses-true-wrt-respective-Interp-def}$ 
  using  $\langle C' = \text{Some } E \rangle$  by simp
next
case  $\text{step-hyps}$ : ( $\text{resolution } \mathcal{M} C L D U_{er}' U_{er} \mathcal{M}' C' \mathcal{F}$ )
hence  $\text{eres } D C \in N \mid \cup \mid U_{er}'$ 
  by simp
hence  $\text{iefac } \mathcal{F} (\text{eres } D C) \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$ 
  by simp
hence  $C' \neq \text{None}$ 
  using  $\text{step-hyps}(3-)$ 
  by ( $\text{metis The-optional-eq-NoneD finsert-not-fempty linorder-cls.ex1-least-in-fset}$ 
     $\text{set-finsert}$ )
then obtain  $E$  where
   $C' = \text{Some } E$ 
  by auto

```



**have**  $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})). C \prec_c E$   
**using**  $\langle C' = \text{Some } E \rangle \text{ step-hyps}(9)$   
**by** (*metis The-optional-eq-SomeD linorder-cls.is-least-in-fset-iff linorder-cls.less-imp-not-less*)

**thus** *?thesis*  
**unfolding** *step-hyps(1,2) all-smaller-clauses-true-wrt-respective-Interp-def*  
**using**  $\langle C' = \text{Some } E \rangle$  **by** *simp*

**qed**

**lemma** *all-smaller-clauses-true-wrt-model*:  
**assumes**  
*invars:*  
*all-smaller-clauses-true-wrt-respective-Interp N s*  
*model-eq-interp-upto-next-clause N s*  
**shows**  $\forall U_{er} \mathcal{F} \mathcal{M} D. s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{dom } \mathcal{M} \models C)$

**proof** (*intro allI impI ballI*)  
**fix**  $U_{er} \mathcal{F} \mathcal{M} D C$   
**assume**  
*s-def:  $s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D)$  and*  
*C-in:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  and*  
*C-lt:  $C \prec_c D$*

**hence** *C-true:  $\text{ord-res-Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$*   
**using** *invars(1)[unfolded all-smaller-clauses-true-wrt-respective-Interp-def s-def]*  
**by** *auto*

**moreover have**  $\text{dom } \mathcal{M} = \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$   
**using** *invars(2) s-def* **by** (*metis model-eq-interp-upto-next-clause-def*)

**ultimately show**  $\text{dom } \mathcal{M} \models C$   
**using** *ord-res.entailed-clause-stays-entailed' C-lt* **by** *metis*

**qed**

**definition** *model-eq-sublocale* **where**  
*model-eq-sublocale N s  $\longleftrightarrow$*   
 $(\forall U_{er} \mathcal{F} \mathcal{M}. s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{None}) \longrightarrow$   
 $(\text{let } NN = \text{fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \text{ in } \text{dom } \mathcal{M} = \bigcup (\text{ord-res.production } NN \text{ ' } NN)))$

**lemma** *all-smaller-clauses-true-wrt-model-strong*:  
**assumes**  
*invars:*  
*all-smaller-clauses-true-wrt-respective-Interp N s*  
*model-eq-interp-upto-next-clause N s*  
*model-eq-sublocale N s*  
**shows**  $\forall U_{er} \mathcal{F} \mathcal{M} C. s = (U_{er}, \mathcal{F}, \mathcal{M}, C) \longrightarrow$

$(\forall C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). (\forall D. \mathcal{C} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow \text{dom } \mathcal{M} \models C)$

**proof** (*intro allI impI ballI*)

**fix**  $U_{er} \mathcal{F} \mathcal{M} \mathcal{C} C$

**assume**

*s-def*:  $s = (U_{er}, \mathcal{F}, \mathcal{M}, C)$  **and**

*C-in*:  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**

*C-lt*:  $\forall D. \mathcal{C} = \text{Some } D \longrightarrow C \prec_c D$

**hence** *C-true*:  $\text{ord-res-Interp } (fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$

**using** *invars(1)* **by** (*metis all-smaller-clauses-true-wrt-respective-Interp-def*)

**show**  $\text{dom } \mathcal{M} \models C$

**proof** (*cases C*)

**case** *C-def*: *None*

**have** *let*  $NN = fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))$  *in*  $\text{dom } \mathcal{M} = \bigcup (\text{ord-res.production } NN \text{ ' } NN)$

**using** *invars(3)* *s-def* *C-def*

**by** (*metis model-eq-sublocale-def*)

**then show** *?thesis*

**using** *C-true*

**by** (*smt (verit, ccfv-SIG) C-in UN-I insertCI linorder-lit.is-greatest-in-mset-iff*

*ord-res.lift-entailment-to-Union ord-res.mem-productionE*

*ord-res.production-eq-empty-or-singleton pos-literal-in-imp-true-cls*

*sup-bot.right-neutral*)

**next**

**case** *C-def*: (*Some D*)

**have**  $C \prec_c D$

**using** *C-lt* *C-def* **by** *metis*

**moreover have**  $\text{dom } \mathcal{M} = \text{ord-res.interp } (fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$

**using** *invars(2)* *s-def* *C-def* **by** (*metis model-eq-interp-upto-next-clause-def*)

**ultimately show** *?thesis*

**using** *ord-res.entailed-clause-stays-entailed' C-true* **by** *metis*

**qed**

**qed**

**lemma** *next-clause-lt-least-false-clause*:

**assumes**

*invars*:

*all-smaller-clauses-true-wrt-respective-Interp N s*

*model-eq-interp-upto-next-clause N s*

**shows**  $\forall U_{er} \mathcal{F} \mathcal{M} C. s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \longrightarrow$

$(\forall D. \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) D \longrightarrow C \preceq_c D)$

**proof** (*intro allI impI ballI*)

**fix**  $U_{er} \mathcal{F} \mathcal{M} C D$

**assume**

*s-def*:  $s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C)$  **and**

*D-least-false*: *is-least-false-clause*  $(\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) D$

**then show**  $C \preceq_c D$

**using** *invars[unfolded model-eq-interp-upto-next-clause-def*

*all-smaller-clauses-true-wrt-respective-Interp-def, rule-format, OF s-def, simplified]*

**by** (*metis (no-types, lifting) fimage.rep-eq is-least-false-clause-def  
linorder-cls.is-least-in-fset-ffilterD(1) linorder-cls.is-least-in-fset-ffilterD(2)  
linorder-cls.le-less-linear sup-fset.rep-eq*)

**qed**

**definition** *atoms-in-model-were-produced* **where**

*atoms-in-model-were-produced*  $N\ s \longleftrightarrow$   
 $(\forall U_{er}\ \mathcal{F}\ \mathcal{M}\ \mathcal{C}. s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \longrightarrow (\forall A\ C. \mathcal{M}\ A = \text{Some}\ C \longrightarrow$   
 $A \in \text{ord-res.production (fset (iefac}\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})))\ C}))$

**lemma** *atoms-in-model-were-produced:*

**assumes** *step: ord-res-5*  $N\ s\ s'$  **and**

*invars:*

*atoms-in-model-were-produced*  $N\ s$   
*model-eq-interp-upto-next-clause*  $N\ s$   
*next-clause-in-factorized-clause*  $N\ s$

**shows** *atoms-in-model-were-produced*  $N\ s'$

**using** *step*

**proof** (*cases*  $N\ s\ s'$  *rule: ord-res-5.cases*)

**case** (*skip*  $\mathcal{M}\ C\ C'\ \mathcal{F}\ U_{er}$ )

**thus** *?thesis*

**using** *invars(1)* **by** (*simp add: atoms-in-model-were-produced-def*)

**next**

**case** (*production*  $\mathcal{M}\ C\ L\ \mathcal{M}'\ C'\ \mathcal{F}\ U_{er}$ )

**obtain**  $A$  **where**  $L = \text{Pos}\ A$

**using**  $\langle \text{is-pos}\ L \rangle$  **by** (*cases*  $L$ ) *simp-all*

**have** *atm-of*  $L \in \text{ord-res.production (fset (iefac}\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})))\ C$

**unfolding** *ord-res.production-unfold mem-Collect-eq*

**proof** (*intro exI conjI*)

**show** *atm-of*  $L = A$

**unfolding**  $\langle L = \text{Pos}\ A \rangle$  *literal.sel ..*

**next**

**show**  $C \in |\text{iefac}\ \mathcal{F}\ |\uparrow| (N\ |\cup| U_{er})$

**using** *invars(3)[unfolded next-clause-in-factorized-clause-def, rule-format,*  
 $\text{OF } \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some}\ C) \rangle$ ].

**next**

**have**  $L \in \# C$

**using**  $\langle \text{linorder-lit.is-maximal-in-mset}\ C\ L \rangle$

**unfolding** *linorder-lit.is-maximal-in-mset-iff ..*

**thus**  $C = \text{add-mset (Pos}\ A) (C - \{\#\text{Pos}\ A\#})$

**unfolding**  $\langle L = \text{Pos}\ A \rangle$  **by** *simp*

**next**

**show** *ord-res.is-strictly-maximal-lit (Pos A) C*

**using**  $\langle \text{ord-res.is-strictly-maximal-lit}\ L\ C \rangle$

**unfolding**  $\langle L = \text{Pos}\ A \rangle$  .

**next**

```

show  $\neg$  ord-res.interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  ( $N \mid \cup \mid U_{er}$ )))  $C \models C$ 
using  $\langle \neg \text{ dom } \mathcal{M} \models C \rangle$ 
unfolding invars(2)[unfolding model-eq-interp-upto-next-clause-def, rule-format,
  OF  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$ ].
qed simp-all

thus ?thesis
using invars(1)
by (simp add: atoms-in-model-were-produced-def
   $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle \langle s' = (U_{er}, \mathcal{F}, \mathcal{M}', C') \rangle \langle \mathcal{M}' = \mathcal{M}(\text{atm-of } L \mapsto C) \rangle$ )
qed (simp-all add: atoms-in-model-were-produced-def)

definition all-produced-atoms-in-model where
  all-produced-atoms-in-model  $N s \longleftrightarrow$ 
  ( $\forall U_{er} \mathcal{F} \mathcal{M} D. s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) \longrightarrow (\forall C A. C \prec_c D \longrightarrow$ 
     $A \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))) C \longrightarrow \mathcal{M} A = \text{Some } C)$ )

lemma all-produced-atoms-in-model:
assumes step: ord-res-5  $N s s'$  and
  invars:
    all-produced-atoms-in-model  $N s$ 
    model-eq-interp-upto-next-clause  $N s$ 
    next-clause-in-factorized-clause  $N s$ 
shows all-produced-atoms-in-model  $N s'$ 
using step
proof (cases  $N s s'$  rule: ord-res-5.cases)
case (skip  $\mathcal{M} C C' \mathcal{F} U_{er}$ )
have ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  ( $N \mid \cup \mid U_{er}$ )))  $C = \{\}$ 
using invars
by (metis ex-in-conv model-eq-interp-upto-next-clause-def local.skip(1) local.skip(3)
  ord-res.mem-productionE)
thus ?thesis
using invars(1)  $\langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$ 
unfolding all-produced-atoms-in-model-def
by (smt (verit, del-insts) Pair-inject The-optional-eq-SomeD empty-iff
  linorder-cls.is-least-in-filter-iff linorder-cls.not-less-iff-gr-or-eq local.skip(2)
  local.skip(4) ord-res.mem-productionE)
next
case step-hyps: (production  $\mathcal{M} C L \mathcal{M}' C' \mathcal{F} U_{er}$ )
obtain  $A$  where  $L = \text{Pos } A$ 
using  $\langle \text{is-pos } L \rangle$  by (cases  $L$ ) simp-all

have atm-of  $L \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))) C$ 
unfolding ord-res.production-unfold mem-Collect-eq
proof (intro exI conjI)
show atm-of  $L = A$ 
unfolding  $\langle L = \text{Pos } A \rangle$  literal.sel ..

```

```

next
  show  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ 
  using  $\text{invars } \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$  by (metis next-clause-in-factorized-clause-def)
next
  have  $L \in \# C$ 
  using  $\langle \text{linorder-lit.is-maximal-in-mset } C L \rangle$ 
  unfolding  $\text{linorder-lit.is-maximal-in-mset-iff ..}$ 
  thus  $C = \text{add-mset } (\text{Pos } A) (C - \{\# \text{Pos } A \# \})$ 
  unfolding  $\langle L = \text{Pos } A \rangle$  by simp
next
  show  $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) C$ 
  using  $\langle \text{ord-res.is-strictly-maximal-lit } L C \rangle$ 
  unfolding  $\langle L = \text{Pos } A \rangle$  .
next
  show  $\neg \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$ 
  using  $\langle \neg \text{dom } \mathcal{M} \models C \rangle$ 
  using  $\text{invars } \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$  by (metis model-eq-interp-upto-next-clause-def)
qed simp-all

thus ?thesis
  using  $\text{invars } \langle s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \rangle$ 
  unfolding  $\text{all-produced-atoms-in-model-def}$ 
  using  $\langle s' = (U_{er}, \mathcal{F}, \mathcal{M}', C') \rangle \langle \mathcal{M}' = \mathcal{M}(\text{atm-of } L \mapsto C) \rangle$ 
  using  $\text{prod.inject}$ 
  by (smt (verit, del-insts) Some-eq-The-optionalD asympD ord-res.mem-productionE
      linorder-cls.antisym-conv3 linorder-cls.is-least-in-ffilter-iff
      linorder-trm.not-less-iff-gr-or-eq step-hyps(8) map-upd-Some-unfold
      ord-res.asymp-less-cls ord-res.less-trm-iff-less-cls-if-mem-production)
next
  case  $\text{step-hyps: (factoring } \mathcal{M} C L \mathcal{F}' \mathcal{F} \mathcal{M}' C' U_{er})$ 
  have  $\neg (\exists C \in | \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D)$  if  $C' = \text{Some } D$  for  $D$ 
  proof (rule nbex-less-than-least-in-fset)
    show  $\text{linorder-cls.is-least-in-fset } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) D$ 
    using  $\text{step-hyps that by (metis The-optional-eq-SomeD)}$ 
  qed
  thus ?thesis
    unfolding  $\text{all-produced-atoms-in-model-def}$ 
    by (metis step-hyps(2) ord-res.mem-productionE prod.inject)
next
  case  $\text{step-hyps: (resolution } \mathcal{M} C L D U_{er}' U_{er} \mathcal{M}' C' \mathcal{F})$ 
  have  $\neg (\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}'). C \prec_c D)$  if  $C' = \text{Some } D$  for  $D$ 
  proof (rule nbex-less-than-least-in-fset)
    show  $\text{linorder-cls.is-least-in-fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')) D$ 
    using  $\text{step-hyps that by (metis The-optional-eq-SomeD)}$ 
  qed
  thus ?thesis
    unfolding  $\text{all-produced-atoms-in-model-def}$ 
    by (metis step-hyps(2) ord-res.mem-productionE prod.inject)
qed

```

**definition** *ord-res-5-invars* **where**

*ord-res-5-invars*  $N\ s \longleftrightarrow$   
*next-clause-in-factorized-clause*  $N\ s \wedge$   
*implicitly-factorized-clauses-subset*  $N\ s \wedge$   
*model-eq-interp-upto-next-clause*  $N\ s \wedge$   
*all-smaller-clauses-true-wrt-respective-Interp*  $N\ s \wedge$   
*atoms-in-model-were-produced*  $N\ s \wedge$   
*all-produced-atoms-in-model*  $N\ s$

**lemma** *ord-res-5-invars-initial-state*:

**assumes**

*F-subset*:  $\mathcal{F} \mid\subseteq\mid N \mid\cup\mid U_{er}$  **and**

*C-least*: *linorder-cls.is-least-in-fset* (*iefac*  $\mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$ )  $C$

**shows** *ord-res-5-invars*  $N (U_{er}, \mathcal{F}, \text{Map.empty}, \text{Some } C)$

**unfolding** *ord-res-5-invars-def*

**proof** (*intro conjI*)

**show** *next-clause-in-factorized-clause*  $N (U_{er}, \mathcal{F}, \lambda x. \text{None}, \text{Some } C)$

**unfolding** *next-clause-in-factorized-clause-def*

**using** *C-least*[*unfolded linorder-cls.is-least-in-fset-iff*] **by** *simp*

**next**

**show** *implicitly-factorized-clauses-subset*  $N (U_{er}, \mathcal{F}, \lambda x. \text{None}, \text{Some } C)$

**unfolding** *implicitly-factorized-clauses-subset-def*

**using** *F-subset* **by** *simp*

**next**

**have** *ord-res.interp* (*iefac*  $\mathcal{F} \text{ ' (fset } N \cup \text{fset } U_{er})$ )  $C = \{\}$

**using** *C-least*[*unfolded linorder-cls.is-least-in-fset-iff*]

**by** (*simp add: interp-eq-empty-if-least-in-set linorder-cls.is-least-in-set-iff*)

**thus** *model-eq-interp-upto-next-clause*  $N (U_{er}, \mathcal{F}, \lambda x. \text{None}, \text{Some } C)$

**unfolding** *model-eq-interp-upto-next-clause-def* **by** *simp*

**next**

**have**  $\neg(\exists Ca \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er}). Ca \prec_c C)$

**using** *C-least*[*unfolded linorder-cls.is-least-in-fset-iff*]

**by** (*metis linorder-cls.less-asm*)

**thus** *all-smaller-clauses-true-wrt-respective-Interp*  $N (U_{er}, \mathcal{F}, \lambda x. \text{None}, \text{Some } C)$

**unfolding** *all-smaller-clauses-true-wrt-respective-Interp-def* **by** *simp*

**next**

**show** *atoms-in-model-were-produced*  $N (U_{er}, \mathcal{F}, \lambda x. \text{None}, \text{Some } C)$

**unfolding** *atoms-in-model-were-produced-def* **by** *simp*

**next**

**have**  $\forall Ca. Ca \prec_c C \longrightarrow Ca \mid\notin\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$

**using** *C-least*[*unfolded linorder-cls.is-least-in-fset-iff*]

**by** (*metis linorder-cls.order-asm*)

**thus** *all-produced-atoms-in-model*  $N (U_{er}, \mathcal{F}, \lambda x. \text{None}, \text{Some } C)$

**unfolding** *all-produced-atoms-in-model-def*

**by** (*simp add: ord-res.production-unfold*)

**qed**

**lemma** *ord-res-5-preserves-invars*:  
**assumes** *step*: *ord-res-5*  $N$   $s$   $s'$  **and** *invars*: *ord-res-5-invars*  $N$   $s$   
**shows** *ord-res-5-invars*  $N$   $s'$   
**proof** –  
**obtain**  $U_{er}$   $\mathcal{F}$   $\mathcal{M}$   $\mathcal{C}$  **where** *s-def*:  $s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$   
**by** (*metis prod.exhaust*)

**then show** *?thesis*  
**unfolding** *ord-res-5-invars-def*  
**using** *invars[unfolded ord-res-5-invars-def]*  
**using** *next-clause-in-factorized-clause[OF step]*  
*ord-res-5-preserves-implicitly-factorized-clauses-subset[OF step]*  
*model-eq-interp-upto-next-clause[OF step]*  
*all-smaller-clauses-true-wrt-respective-Interp[OF step]*  
*atoms-in-model-were-produced[OF step]*  
*all-produced-atoms-in-model[OF step]*  
**by** *metis*

**qed**

**lemma** *rtranclp-ord-res-5-preserves-invars*:  
**assumes** *steps*: (*ord-res-5*  $N$ )<sup>\*\*</sup>  $s$   $s'$  **and** *invars*: *ord-res-5-invars*  $N$   $s$   
**shows** *ord-res-5-invars*  $N$   $s'$   
**using** *steps invars*  
**by** (*induction s rule: converse-rtranclp-induct*) (*auto intro: ord-res-5-preserves-invars*)

**lemma** *tranclp-ord-res-5-preserves-invars*:  
**assumes** *steps*: (*ord-res-5*  $N$ )<sup>++</sup>  $s$   $s'$  **and** *invars*: *ord-res-5-invars*  $N$   $s$   
**shows** *ord-res-5-invars*  $N$   $s'$   
**using** *rtranclp-ord-res-5-preserves-invars*  
**by** (*metis invars steps tranclp-into-rtranclp*)

**lemma** *le-least-false-clause*:  
**fixes**  $N$   $s$   $U_{er}$   $\mathcal{F}$   $\mathcal{M}$   $C$   $D$   
**assumes**  
*invars*: *ord-res-5-invars*  $N$   $s$  **and**  
*s-def*:  $s = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C)$  **and**  
*D-least-false*: *is-least-false-clause* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N$   $| \cup |$   $U_{er}$ ))  $D$   
**shows**  $C \preceq_c D$   
**proof** –  
**have** *D-in*:  $D \in |$  *iefac*  $\mathcal{F}$   $| \uparrow$  ( $N$   $| \cup |$   $U_{er}$ )  
**using** *D-least-false*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-filter-iff*  
**by** *argo*

**show**  $C \preceq_c D$   
**proof** (*rule next-clause-lt-least-false-clause[rule-format]*)  
**show** *is-least-false-clause* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N$   $| \cup |$   $U_{er}$ ))  $D$   
**using** *D-least-false* .

```

next
  show  $(U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) = (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) ..$ 
next
  show all-smaller-clauses-true-wrt-respective-Interp  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C)$ 
    using invars by (metis s-def ord-res-5-invars-def)
next
  show model-eq-interp-upto-next-clause  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C)$ 
    using invars by (metis s-def ord-res-5-invars-def)
qed
qed

lemma ex-ord-res-5-if-not-final:
  assumes
    not-final:  $\neg \text{ord-res-5-final } S$  and
    invars:  $\forall N s. S = (N, s) \longrightarrow \text{ord-res-5-invars } N s$ 
  shows  $\exists S'. \text{ord-res-5-step } S S'$ 
proof -
  from not-final obtain  $N U_{er} \mathcal{F} \mathcal{M} C$  where
    S-def:  $S = (N, (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C))$  and  $C \neq \{\#\}$ 
  unfolding ord-res-5-final.simps de-Morgan-disj not-ex
  by (metis option.exhaust surj-pair)

  note invars' = invars[unfolded ord-res-5-invars-def, rule-format, OF S-def]

  have  $\exists s'. \text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) s'$ 
  proof (cases dom M ||= C)
    case True
    thus ?thesis
      using ord-res-5.skip by metis
  next
    case C-false: False
    obtain  $L$  where L-max: linorder-lit.is-maximal-in-mset  $C L$ 
      using linorder-lit.ex-maximal-in-mset[OF C ≠ {#}] ..

    show ?thesis
    proof (cases L)
      case (Pos A)
      hence L-pos: is-pos  $L$ 
        by simp
      show ?thesis
      proof (cases ord-res.is-strictly-maximal-lit L C)
        case True
        then show ?thesis
          using ord-res-5.production[OF C-false L-max L-pos] by metis
      next
        case L-not-strictly-max: False
        thus ?thesis
          using ord-res-5.factoring[OF C-false L-max L-pos L-not-strictly-max - refl]
          by metis
    end
  end

```



```

qed
next
case (Neg A)
hence L-neg: is-neg L
  by simp

have C-least-false: is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )) C
  unfolding is-least-false-clause-def linorder-clis.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
  show C | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )
    using invars' by (metis next-clause-in-factorized-clause-def)
next
have  $\neg$  ord-res.interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C  $\models$  C
  using invars' C-false by (metis model-eq-interp-upto-next-clause-def)
moreover have ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C = {}
proof -
  have  $\nexists$  L. is-pos L  $\wedge$  ord-res.is-strictly-maximal-lit L C
    using L-max L-neg
  by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset
    linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset)
  thus ?thesis
    using unproductive-if-nex-strictly-maximal-pos-lit by metis
qed
ultimately show  $\neg$  ord-res-Interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C  $\models$  C
  by simp
next
fix D
assume D-in: D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) and D  $\neq$  C and
  C-false:  $\neg$  ord-res-Interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) D  $\models$  D
have  $\neg$  D  $\prec_c$  C
  using C-false
  using invars' D-in
  unfolding all-smaller-clauses-true-wrt-respective-Interp-def
  by auto
thus C  $\prec_c$  D
  using  $\langle D \neq C \rangle$  by order
qed
then obtain D where D| $\in$ |iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) and
  D  $\prec_c$  C and
  ord-res.is-strictly-maximal-lit (Pos A) D and
  D-prod: ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) D = {A}
  using bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit
  L-max[unfolded Neg] by metis

have  $\mathcal{M}$  (atm-of L) = Some D
  using invars'
  by (metis Neg  $\langle D \prec_c C \rangle$  all-produced-atoms-in-model-def D-prod insertI1
  literal.sel(2))

```

```

    then show ?thesis
      using ord-res-5.resolution[OF C-false L-max L-neg] by metis
    qed
  qed
  thus ?thesis
    using S-def ord-res-5-step.simps by metis
qed

lemma ord-res-5-safe-state-if-invars:
  fixes N s
  assumes invars: ord-res-5-invars N s
  shows safe-state ord-res-5-step ord-res-5-final (N, s)
proof -
  {
    fix S'
    assume ord-res-5-semantics.eval (N, s) S' and stuck: stuck-state ord-res-5-step
    S'
    then obtain s' where S' = (N, s') and (ord-res-5 N)** s s'
    proof (induction (N, s) arbitrary: N s rule: converse-rtranclp-induct)
      case base
        thus ?case by simp
      next
        case (step z)
          thus ?case
            by (smt (verit, ccfv-SIG) converse-rtranclp-into-rtranclp ord-res-5-step.cases
prod.inject)
    qed
    hence ord-res-5-invars N s'
      using invars rtranclp-ord-res-5-preserves-invars by metis
    hence  $\neg$  ord-res-5-final S'  $\implies \exists S''$ . ord-res-5-step S' S''
      using ex-ord-res-5-if-not-final[of S']  $\langle S' = (N, s') \rangle$  by blast
    hence ord-res-5-final S'
      using stuck[unfolded stuck-state-def] by argo
  }
  thus ?thesis
    unfolding safe-state-def stuck-state-def by metis
qed

lemma MAGIC1:
  assumes invars: ord-res-5-invars N (Uer,  $\mathcal{F}$ ,  $\mathcal{M}$ ,  $\mathcal{C}$ )
  shows  $\exists \mathcal{M}' \mathcal{C}'$ . (ord-res-5 N)** (Uer,  $\mathcal{F}$ ,  $\mathcal{M}$ ,  $\mathcal{C}$ ) (Uer,  $\mathcal{F}$ ,  $\mathcal{M}'$ ,  $\mathcal{C}'$ )  $\wedge$ 
    ( $\nexists \mathcal{M}'' \mathcal{C}''$ . ord-res-5 N (Uer,  $\mathcal{F}$ ,  $\mathcal{M}'$ ,  $\mathcal{C}'$ ) (Uer,  $\mathcal{F}$ ,  $\mathcal{M}''$ ,  $\mathcal{C}''$ ))
proof -
  define R where
    R =  $(\lambda(\mathcal{M}, \mathcal{C}) (\mathcal{M}', \mathcal{C}'). \text{ord-res-5 N (U}_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) (U}_{er}, \mathcal{F}, \mathcal{M}', \mathcal{C}'))$ 

  define f :: ('f gterm  $\implies$  'f gclause option)  $\times$  'f gclause option  $\implies$  'f gclause fset
  where
    f =  $(\lambda(\mathcal{M}, \mathcal{C}). \text{case } \mathcal{C} \text{ of None } \implies \{\|\} \mid \text{Some } C \implies \text{finsert } C \{\|D \mid \in\} \text{iefac } \mathcal{F})$ 

```

```

|' (N |∪| Uer). C ≼c D|})

have Wellfounded.wfp-on {x'. R** (M, C) x'} R-1-1
proof (rule wfp-on-if-convertible-to-wfp-on)
  have wfp (|C|)
  by auto
  thus Wellfounded.wfp-on (f ' {x'. R** (M, C) x'}) (|C|)
  using Wellfounded.wfp-on-subset subset-UNIV by metis
next
fix x y :: ('f gterm ⇒ 'f gclause option) × 'f gclause option

obtain Mx Cx where x-def: x = (Mx, Cx)
by force

obtain My Cy where y-def: y = (My, Cy)
by force

have rtranclp-with-constsD: (λ(y, z) (y', z'). R (x, y, z) (x, y', z'))** (y, z) (y',
z') ⇒
  R** (x, y, z) (x, y', z') for R x y z y' z'
proof (induction (y, z) arbitrary: y z rule: converse-rtranclp-induct)
  case base
  show ?case
  by simp
next
  case (step w)
  thus ?case
  by force
qed

assume x ∈ {x'. R** (M, C) x'} and y ∈ {x'. R** (M, C) x'}
hence
  (ord-res-5 N)** (Uer, F, M, C) (Uer, F, Mx, Cx) and
  (ord-res-5 N)** (Uer, F, M, C) (Uer, F, My, Cy)
  unfolding atomize-conj mem-Collect-eq R-def x-def y-def
  by (auto intro: rtranclp-with-constsD)
hence
  x-invars: ord-res-5-invars N (Uer, F, Mx, Cx) and
  y-invars: ord-res-5-invars N (Uer, F, My, Cy)
  using invars by (metis rtranclp-ord-res-5-preserves-invars)+

assume R-1-1 y x
hence ord-res-5 N (Uer, F, Mx, Cx) (Uer, F, My, Cy)
  unfolding conversep-iff R-def x-def y-def prod.case .
thus f y |C| f x
proof (cases N (Uer, F, Mx, Cx) (Uer, F, My, Cy))
  case step-hyps: (skip C)

have f y |≤| {|D |∈| iefac F |' (N |∪| Uer). C ≼c D|}

```

```

proof (cases  $C_y$ )
  case None
  thus ?thesis
    unfolding f-def y-def prod.case by simp
next
  case  $C_y$ -def: (Some D)

  have D-least: linorder-cls.is-least-in-fset (ffilter (( $\prec_c$ ) C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$  |
  Uer))) D
    using step-hyps Cy-def by (metis The-optional-eq-SomeD)
  hence f-y-eq: f y = { $|E | \in |$  iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$  | Uer).  $D \preceq_c E$ }
  unfolding f-def y-def prod.case Cy-def option.case linorder-cls.is-least-in-ffilter-iff
  by auto

  show ?thesis
    unfolding f-y-eq subset-ffilter
    using D-least
    unfolding linorder-cls.is-least-in-ffilter-iff
    by auto
qed

also have ...  $|C|$  finsert C { $|D | \in |$  iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$  | Uer).  $C \preceq_c D$ }
  by auto

also have ... = f x
  unfolding f-def x-def y-def prod.case step-hyps option.case by metis

finally show ?thesis .
next
  case step-hyps: (production C L)

  have f y  $\subseteq$  { $|D | \in |$  iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$  | Uer).  $C \prec_c D$ }
  proof (cases  $C_y$ )
    case None
    thus ?thesis
      unfolding f-def y-def prod.case by simp
  next
    case  $C_y$ -def: (Some D)

    have D-least: linorder-cls.is-least-in-fset (ffilter (( $\prec_c$ ) C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$  |
    Uer))) D
      using step-hyps Cy-def by (metis The-optional-eq-SomeD)
    hence f-y-eq: f y = { $|E | \in |$  iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$  | Uer).  $D \preceq_c E$ }
    unfolding f-def y-def prod.case Cy-def option.case linorder-cls.is-least-in-ffilter-iff
    by auto

    show ?thesis
      unfolding f-y-eq subset-ffilter
      using D-least

```

```

    unfolding linorder-cls.is-least-in-filter-iff
    by auto
qed

also have ... |C| finsert C {|D |∈| iefac F |↑| (N |∪| Uer). C ≼c D|}
  by auto

also have ... = f x
  unfolding f-def x-def y-def prod.case step-hyps option.case by metis

finally show ?thesis .
next
case step-hyps: (factoring C L)

have C |∈| iefac F |↑| (N |∪| Uer)
  using x-invars unfolding ⟨Cx = Some C⟩
  by (metis next-clause-in-factorized-clause-def ord-res-5-invars-def)
hence C |∉| F
  using step-hyps(3,4,5)
  by (smt (verit, ccfv-SIG) fimage-iff iefac-def literal.collapse(1)
      ex1-efac-eq-factoring-chain ex-ground-factoringI)
moreover have C |∈| F
  using ⟨F = finsert C F⟩ by auto
ultimately have False
  by contradiction
thus ?thesis ..
next
case step-hyps: (resolution C L D)

  have D-productive: atm-of L ∈ ord-res.production (fset (iefac F |↑| (N |∪|
Uer))) D
  using x-invars step-hyps
  by (metis ord-res-5-invars-def atoms-in-model-were-produced-def)

hence ∃ DC. ground-resolution D C DC
  unfolding ground-resolution-def
  using step-hyps
  by (metis Neg-atm-of-iff ord-res.mem-productionE linorder-cls.le-less-linear
      linorder-lit.is-maximal-in-mset-iff linorder-trm.dual-order.order-iff-strict
      linorder-trm.not-less ord-res.less-trm-if-neg ex-ground-resolutionI)

hence eres D C ≠ C
  unfolding eres-ident-iff by metis

have eres D C |∉| Uer
proof (rule notI)
  assume eres D C |∈| Uer
  hence iefac F (eres D C) |∈| iefac F |↑| (N |∪| Uer)
  by simp

```

**moreover have**  $\text{iefac } \mathcal{F} (\text{eres } D \ C) \prec_c C$   
**proof** –  
**have**  $\text{iefac } \mathcal{F} \ C \prec_c D$  **if**  $C \prec_c D$  **for**  $\mathcal{F} \ C \ D$   
**proof** (*cases*  $C \in \mathcal{F}$ )  
**case** *True*  
**hence**  $\text{iefac } \mathcal{F} \ C = \text{efac } C$   
**by** (*simp add: iefac-def*)  
**also have**  $\dots \preceq_c C$   
**by** (*metis efac-subset subset-implies-reflclp-multp*)  
**also have**  $\dots \prec_c D$   
**using** *that* .  
**finally show** *?thesis* .  
**next**  
**case** *False*  
**thus** *?thesis*  
**using** *that by (simp add: iefac-def)*  
**qed**

**moreover have**  $\text{eres } D \ C \prec_c C$   
**proof** –  
**have**  $\text{eres } D \ C \preceq_c C$   
**using** *eres-le by metis*  
**thus**  $\text{eres } D \ C \prec_c C$   
**using**  $\langle \text{eres } D \ C \neq C \rangle$  **by** *order*  
**qed**

**ultimately show** *?thesis*  
**by** *metis*  
**qed**

**ultimately have**  $\text{ord-res-Interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) (\text{iefac } \mathcal{F} (\text{eres } D \ C)) \models \text{iefac } \mathcal{F} (\text{eres } D \ C)$   
**using** *x-invars unfolding*  $\langle C_x = \text{Some } C \rangle$   
**unfolding** *ord-res-5-invars-def all-smaller-clauses-true-wrt-respective-Interp-def*  
**by** *fast*  
**hence**  $\text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C \models \text{iefac } \mathcal{F} (\text{eres } D \ C)$   
**using** *ord-res.entailed-clause-stays-entailed'*  $\langle \text{iefac } \mathcal{F} (\text{eres } D \ C) \prec_c C \rangle$  **by** *metis*  
**hence**  $\text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C \models \text{eres } D \ C$   
**proof** (*rule true-cls-iff-set-mset-eq[THEN iffD1, rotated]*)  
**show**  $\text{set-mset } (\text{iefac } \mathcal{F} (\text{eres } D \ C)) = \text{set-mset } (\text{eres } D \ C)$   
**using** *iefac-def by auto*  
**qed**  
**hence**  $\text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C \models C$   
**proof** –  
**obtain**  $m \ A \ D' \ C'$  **where**  
 $\text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) \ D$  **and**

**D-def:**  $D = \text{add-mset } (Pos\ A)\ D'$  **and**  
**C-def:**  $C = \text{replicate-mset } (Suc\ m)\ (Neg\ A) + C'$  **and**  
**Neg A**  $\notin \# C'$  **and**  
**eres-D-C-eq:**  $\text{eres } D\ C = \text{repeat-mset } (Suc\ m)\ D' + C'$   
**using**  $\langle \text{eres } D\ C \neq C' \rangle [\text{THEN } \text{eres-not-ident } D]$  **by** *metis*

**hence**  
**atm-of**  $L = A$  **and**  
**D-in:**  $D \in | \text{iefac } \mathcal{F} \ |^q (N \ | \cup \ | U_{er})$  **and**  
**D-false:**  $\neg \text{ord-res.interp } (fset\ (\text{iefac } \mathcal{F} \ |^q (N \ | \cup \ | U_{er})))\ D \models D$   
**unfolding** *atomize-conj*  
**using** *D-productive*  
**unfolding** *ord-res.production-unfold mem-Collect-eq*  
**by** *(metis linorder-lit.Uniq-is-greatest-in-mset literal.inject(1) the1-equality')*

**have**  $D'$ -false:  $\neg \text{ord-res.interp } (fset\ (\text{iefac } \mathcal{F} \ |^q (N \ | \cup \ | U_{er})))\ D \models D'$   
**using** *D-false D-def* **by** *fastforce*

**have**  $D \prec_c C$   
**using**  $\langle \exists DC. \text{ground-resolution } D\ C\ DC \rangle$  *left-premise-lt-right-premise-if-ground-resolution*  
**by** *blast*

**let**  $?I = \text{ord-res.interp } (fset\ (\text{iefac } \mathcal{F} \ |^q (N \ | \cup \ | U_{er})))\ C$

**assume**  $?I \models \text{eres } D\ C$   
**hence**  $?I \models D' \vee ?I \models C'$   
**unfolding** *eres-D-C-eq true-cls-union true-cls-repeat-mset-Suc* .

**moreover have**  $\neg ?I \models D'$   
**using**  $\langle D \prec_c C \rangle$   
**by** *(smt (verit) D-def D-productive <ord-res.is-strictly-maximal-lit (Pos*  
A)  $D \rangle$   
*diff-single-eq-union ord-res.mem-productionE linorder-lit.is-greatest-in-mset-iff*  
*ord-res.Uniq-strictly-maximal-lit-in-ground-cls*  
*ord-res.false-cls-if-productive-production ord-res-5-invars-def*  
*next-clause-in-factorized-clause-def step-hyps(1) the1-equality' x-invars)*

**ultimately show**  $?I \models C$   
**by** *(simp add: C-def)*

**qed**  
**hence**  $\text{dom } \mathcal{M}_x \models C$   
**using** *x-invars*  $\langle \mathcal{C}_x = \text{Some } C \rangle$   
**by** *(metis model-eq-interp-upto-next-clause-def ord-res-5-invars-def)*  
**thus** *False*  
**using**  $\langle \neg \text{dom } \mathcal{M}_x \models C \rangle$  **by** *contradiction*

**qed**  
**hence** *False*  
**using**  $\langle U_{er} = \text{finsert } (\text{eres } D\ C)\ U_{er} \rangle$  **by** *auto*  
**thus** *?thesis ..*

```

qed
qed

then obtain  $M' C'$  where  $R^{**} (M, C) (M', C')$  and  $\nexists z. R (M', C') z$ 
  using ex-terminating-rtranclp-strong by (metis surj-pair)

show ?thesis
proof (intro exI conjI)
  show (ord-res-5 N)** ( $U_{er}, \mathcal{F}, M, C$ ) ( $U_{er}, \mathcal{F}, M', C'$ )
    using  $\langle R^{**} (M, C) (M', C') \rangle$ 
    by (induction ( $M, C$ ) arbitrary:  $M C$  rule: converse-rtranclp-induct) (auto
simp: R-def)
  next
    show  $\nexists M'' C''. \text{ord-res-5 } N (U_{er}, \mathcal{F}, M', C') (U_{er}, \mathcal{F}, M'', C'')$ 
      using  $\langle \nexists z. R (M', C') z \rangle$ 
      by (simp add: R-def)
qed
qed

lemma MAGIC2:
  assumes invars: ord-res-5-invars  $N (U_{er}, \mathcal{F}, M, \text{Some } C)$ 
  assumes  $C \neq \{\#\}$ 
  shows  $\exists s'. \text{ord-res-5 } N (U_{er}, \mathcal{F}, M, \text{Some } C) s'$ 
proof (cases (dom  $M$ )  $\models C$ )
  case C-true: True
  thus ?thesis
    using ord-res-5.skip by metis
next
  case C-false: False
  obtain  $L$  where L-max: linorder-lit.is-maximal-in-mset  $C L$ 
    using  $\langle C \neq \{\#\} \rangle$  linorder-lit.ex-maximal-in-mset by metis

  show ?thesis
  proof (cases  $L$ )
    case (Pos  $A$ )
    hence L-pos: is-pos  $L$ 
      by simp

    show ?thesis
    proof (cases linorder-lit.is-greatest-in-mset  $C L$ )
      case L-greatest: True
      thus ?thesis
        using C-false L-max L-pos ord-res-5.production by metis
    next
      case L-not-greatest: False
      thus ?thesis
        using C-false L-max L-pos ord-res-5.factoring by metis
    qed
  qed
next

```



**case** (*Neg A*)  
**hence** *L-neg: is-neg L*  
**by** *simp*

**have** *is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$  ( $N \cup U_{er}$ )) C*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-filter-iff*  
**proof** (*intro conjI ballI impI*)  
**show** *C | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  ( $N \cup U_{er}$ )*  
**using** *invars unfolding ord-res-5-invars-def next-clause-in-factorized-clause-def*  
**by** *metis*

**next**  
**have** *dom  $\mathcal{M} = ord-res.interp (fset (iefac \mathcal{F} | \uparrow (N \cup U_{er}))) C$*   
**using** *invars unfolding ord-res-5-invars-def model-eq-interp-upto-next-clause-def*  
**by** *metis*

**moreover have** *ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  ( $N \cup U_{er}$ ))) C = {}*  
**proof** –  
**have**  $\nexists L. is-pos L \wedge ord-res.is-strictly-maximal-lit L C$   
**using** *L-max L-neg*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)  
**thus** *?thesis*  
**using** *unproductive-if-nex-strictly-maximal-pos-lit* **by** *metis*  
**qed**

**ultimately show**  $\neg ord-res.Interp (fset (iefac \mathcal{F} | \uparrow (N \cup U_{er}))) C \models C$   
**using** *C-false model-eq-interp-upto-next-clause-def* **by** *simp*

**next**  
**fix** *D*  
**assume**  
*D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  ( $N \cup U_{er}$ ) and*  
*D  $\neq$  C and*  
 $\neg ord-res.Interp (fset (iefac \mathcal{F} | \uparrow (N \cup U_{er}))) D \models D$

**moreover have**  $\forall B | \in | iefac \mathcal{F} | \uparrow (N \cup U_{er}). B \prec_c C \longrightarrow$   
 $ord-res.Interp (fset (iefac \mathcal{F} | \uparrow (N \cup U_{er}))) B \models B$   
**using** *invars*  
**unfolding** *ord-res-5-invars-def all-smaller-clauses-true-wrt-respective-Interp-def*  
**by** *simp*

**ultimately show**  $C \prec_c D$   
**by** *force*

**qed**  
**then obtain** *D* **where** *D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  ( $N \cup U_{er}$ ) and*  
*D  $\prec_c$  C and*  
*ord-res.is-strictly-maximal-lit (Pos A) D and*  
*D-prod: ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  ( $N \cup U_{er}$ ))) D = {A}*  
**using** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*  
*L-max[unfolded Neg]* **by** *metis*

```

have  $\mathcal{M}$  (atm-of  $L$ ) = Some  $D$ 
  using invars
  unfolding ord-res-5-invars-def all-produced-atoms-in-model-def
  by (metis Neg  $\langle D \prec_c C \rangle$  D-prod insertI1 literal.sel(2))

thus ?thesis
  using ord-res-5.resolution C-false L-max L-neg by metis
qed
qed

lemma MAGIC3:
  assumes invars: ord-res-5-invars  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}$ ,  $C$ ) and
    steps: (ord-res-5  $N$ )** ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}$ ,  $C$ ) ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}'$ ,  $C'$ ) and
    no-more-steps: ( $\nexists \mathcal{M}'' C''$ . ord-res-5  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}'$ ,  $C'$ ) ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}''$ ,  $C''$ ))
  shows ( $\forall C. C' = \text{Some } C \longleftrightarrow \text{is-least-false-clause (iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) C$ )
  proof -
    have invars': ord-res-5-invars  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}'$ ,  $C'$ )
      using steps invars rtranclp-ord-res-5-preserves-invars by metis

    show ?thesis
    proof (cases  $C'$ )
      case None
        moreover have  $\nexists C$ . is-least-false-clause (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup U_{er})) C$ 
          proof (rule notI, elim exE)
            fix  $C$ 

            have all-smaller-clauses-true-wrt-respective-Interp  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\mathcal{M}'$ ,  $C'$ )
              using invars' unfolding ord-res-5-invars-def by metis
            hence ( $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})$ . ord-res-Interp (fset (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup U_{er})) C \models C$ ))
              by (simp add:  $\langle C' = \text{None} \rangle$  all-smaller-clauses-true-wrt-respective-Interp-def)

            moreover assume is-least-false-clause (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup U_{er})) C$ 

            ultimately show False
              unfolding is-least-false-clause-def linorder-clis.is-least-in-ffilter-iff by metis
            qed

            ultimately show ?thesis
              by simp
          next
            case (Some  $D$ )

            moreover have is-least-false-clause (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup U_{er})) D$ 
              unfolding is-least-false-clause-def linorder-clis.is-least-in-ffilter-iff
            proof (intro conjI ballI impI)
              show  $D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})$ 

```

```

using invars'  $\langle C' = \text{Some } D \rangle$ 
unfolding ord-res-5-invars-def next-clause-in-factorized-clause-def
by metis
next
have  $D \neq \{\#\} \implies \exists s'. \text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}', \text{Some } D) s'$ 
using MAGIC2 invars'  $\langle C' = \text{Some } D \rangle$  by metis

thus  $\neg \text{ord-res-Interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \models D$ 
by (smt (verit) Pair-inject Un-empty-right Uniq-D calculation empty-iff
invars'
linorder-lit.Uniq-is-maximal-in-mset
linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset no-more-steps op-
tion.inject
ord-res-5.cases set-mset-empty model-eq-interp-upto-next-clause-def
ord-res-5-invars-def
true-cls-def unproductive-if-nex-strictly-maximal-pos-lit)

next
fix  $E$ 
assume
 $E \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  and
 $E \neq D$  and
 $\neg \text{ord-res-Interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) E \models E$ 

moreover have  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$ 
 $\text{ord-res-Interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$ 
using invars'  $\langle C' = \text{Some } D \rangle$ 
unfolding ord-res-5-invars-def all-smaller-clauses-true-wrt-respective-Interp-def
by simp

ultimately show  $D \prec_c E$ 
by force
qed

ultimately show ?thesis
by (metis Uniq-D Uniq-is-least-false-clause option.inject)
qed
qed

lemma ord-res-5-construct-model-upto-least-false-clause:
assumes invars: ord-res-5-invars  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$ 
shows  $\exists \mathcal{M}' \mathcal{C}'. (\text{ord-res-5 } N)^{**} (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) (U_{er}, \mathcal{F}, \mathcal{M}', \mathcal{C}') \wedge$ 
 $(\forall C. C' = \text{Some } C \longleftrightarrow \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) C)$ 
using MAGIC1[OF invars] MAGIC3[OF invars] by metis

end

end
theory ORD-RES-6
imports

```

ORD-RES-5  
begin

## 21 ORD-RES-6 (model backjump)

context *simulation-SCLFOL-ground-ordered-resolution* begin

inductive *ord-res-6* where

*skip*:

$(\text{dom } \mathcal{M}) \models C \implies$   
 $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } \{|D | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}).$   
 $C \prec_c D\}) \implies$   
 $\text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}, C') |$

*production*:

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\text{is-pos } L \implies$   
 $\text{linorder-lit.is-greatest-in-mset } C L \implies$   
 $\mathcal{M}' = \mathcal{M}(\text{atm-of } L := \text{Some } C) \implies$   
 $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } \{|D | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}).$   
 $C \prec_c D\}) \implies$   
 $\text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}', C') |$

*factoring*:

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\text{is-pos } L \implies$   
 $\neg \text{linorder-lit.is-greatest-in-mset } C L \implies$   
 $\mathcal{F}' = \text{finsert } C \mathcal{F} \implies$   
 $\text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } C)) |$

*resolution-bot*:

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\text{is-neg } L \implies$   
 $\mathcal{M} (\text{atm-of } L) = \text{Some } D \implies$   
 $U_{er}' = \text{finsert } (\text{eres } D C) U_{er} \implies$   
 $\text{eres } D C = \{\#\} \implies$   
 $\mathcal{M}' = (\lambda-. \text{None}) \implies$   
 $\text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } \{\#\}) |$

*resolution-pos*:

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\text{is-neg } L \implies$   
 $\mathcal{M} (\text{atm-of } L) = \text{Some } D \implies$   
 $U_{er}' = \text{finsert } (\text{eres } D C) U_{er} \implies$   
 $\text{eres } D C \neq \{\#\} \implies$

$\mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \implies$   
 $\text{linorder-lit.is-maximal-in-mset } (\text{eres } D \ C) \ K \implies$   
 $\text{is-pos } K \implies$   
 $\text{ord-res-6 } N \ (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \ (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } (\text{eres } D \ C)) \ |$

*resolution-neg:*

$\neg (\text{dom } \mathcal{M}) \models C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C \ L \implies$   
 $\text{is-neg } L \implies$   
 $\mathcal{M} (\text{atm-of } L) = \text{Some } D \implies$   
 $U_{er}' = \text{finsert } (\text{eres } D \ C) \ U_{er} \implies$   
 $\text{eres } D \ C \neq \{\#\} \implies$   
 $\mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \implies$   
 $\text{linorder-lit.is-maximal-in-mset } (\text{eres } D \ C) \ K \implies$   
 $\text{is-neg } K \implies$   
 $\mathcal{M} (\text{atm-of } K) = \text{Some } E \implies$   
 $\text{ord-res-6 } N \ (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \ (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } E)$

**inductive** *ord-res-6-step* **where**

$\text{ord-res-6 } N \ s \ s' \implies \text{ord-res-6-step } (N, s) \ (N, s')$

**lemma** *tranclp-ord-res-6-step-if-tranclp-ord-res-6:*

$(\text{ord-res-6 } N)^{++} \ s \ s' \implies \text{ord-res-6-step}^{++} \ (N, s) \ (N, s')$

**by** (*induction*  $s'$  *rule: tranclp-induct*)

(*auto intro: ord-res-6-step.intros tranclp.intros*)

**lemma** *right-unique-ord-res-6: right-unique* (*ord-res-6*  $N$ )

**proof** (*rule right-uniqueI*)

**fix**  $s \ s' \ s''$

**assume** *step1: ord-res-6*  $N \ s \ s'$  **and** *step2: ord-res-6*  $N \ s \ s''$

**thus**  $s' = s''$

**by** (*smt (verit) Pair-inject linorder-lit.Uniq-is-maximal-in-mset option.inject*  
*ord-res-6.cases*  
*the1-equality'*)

**qed**

**lemma** *right-unique-ord-res-6-step: right-unique* *ord-res-6-step*

**proof** (*rule right-uniqueI*)

**fix**  $x \ y \ z$

**show** *ord-res-6-step*  $x \ y \implies \text{ord-res-6-step } x \ z \implies y = z$

**using** *right-unique-ord-res-6[THEN right-uniqueD]*

**by** (*elim ord-res-6-step.cases*) (*metis prod.inject*)

**qed**

**inductive** *ord-res-6-final* **where**

*model-found:*

$\text{ord-res-6-final } (N, U_{er}, \mathcal{F}, \mathcal{M}, \text{None}) \ |$

*contradiction-found:*

*ord-res-6-final* ( $N, U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } \{\#\}$ )

**sublocale** *ord-res-6-semantic*: *semantic* **where**

*step* = *ord-res-6-step* **and**

*final* = *ord-res-6-final*

**proof** *unfold-locales*

**fix**  $S :: 'f \text{ gclause } fset \times 'f \text{ gclause } fset \times 'f \text{ gclause } fset \times$   
 $('f \text{ gterm } \Rightarrow 'f \text{ gclause } option) \times 'f \text{ gclause } option$

**obtain**

$N U_{er} \mathcal{F} :: 'f \text{ gterm } clause \text{ fset}$  **and**

$\mathcal{M} :: 'f \text{ gterm } \Rightarrow 'f \text{ gclause } option$  **and**

$\mathcal{C} :: 'f \text{ gclause } option$  **where**

*S-def*:  $S = (N, (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}))$

**by** (*metis prod.exhaust*)

**assume** *ord-res-6-final*  $S$

**hence**  $\mathcal{C} = \text{None} \vee \mathcal{C} = \text{Some } \{\#\}$

**by** (*simp add: S-def ord-res-6-final.simps*)

**hence**  $\nexists x. \text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) x$

**by** (*auto simp: linorder-lit.is-maximal-in-mset-iff elim: ord-res-6.cases*)

**thus** *finished ord-res-6-step*  $S$

**by** (*simp add: S-def finished-def ord-res-6-step.simps*)

**qed**

**lemma** *ord-res-6-preserves-invars*:

**assumes** *step: ord-res-6*  $N s s'$  **and** *invars: ord-res-5-invars*  $N s$

**shows** *ord-res-5-invars*  $N s'$

**using** *step*

**proof** (*cases N s s' rule: ord-res-6.cases*)

**case** (*skip*  $\mathcal{M} \mathcal{C} \mathcal{C}' \mathcal{F} U_{er}$ )

**thus** *?thesis*

**by** (*metis invars ord-res-5-preserves-invars ord-res-5.skip*)

**next**

**case** (*production*  $\mathcal{M} \mathcal{C} L \mathcal{M}' \mathcal{C}' \mathcal{F} U_{er}$ )

**thus** *?thesis*

**by** (*metis invars ord-res-5.production ord-res-5-preserves-invars*)

**next**

**case** *step-hyps*: (*factoring*  $\mathcal{M} \mathcal{C} L \mathcal{F}' \mathcal{F} U_{er}$ )

**have** *efac*  $C \neq C$

**by** (*metis ex1-efac-eq-factoring-chain is-pos-def ex-ground-factoringI step-hyps(4,5,6)*)

**moreover** **have** *efac*  $C \preceq_c C$

**by** (*metis efac-subset subset-implies-reflclp-multp*)

**ultimately** **have** *efac*  $C \prec_c C$

**by** *order*

**show** *?thesis*

**unfolding** *step-hyps(1,2) ord-res-5-invars-def*

**proof** (*intro conjI*)  
**have**  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using** *invars step-hyps*  
**by** (*metis next-clause-in-factorized-clause-def ord-res-5-invars-def*)  
**hence**  $C \in \mid N \mid \cup \mid U_{er}$   
**using**  $\langle \text{efac } C \neq C \rangle$   
**by** (*smt (verit, best) fimage-iff iefac-def ex1-efac-eq-factoring-chain factorizable-if-neq-efac*)  
**hence**  $\text{efac } C \in \mid \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$   
**using** *step-hyps(3-)*  
**using** *iefac-def* **by** *auto*  
**thus** *next-clause-in-factorized-clause*  $N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } C))$   
**unfolding** *next-clause-in-factorized-clause-def* **by** *simp*

**have**  $\mathcal{F} \subseteq \mid N \mid \cup \mid U_{er}$   
**using** *invars*  
**unfolding** *step-hyps(1,2) ord-res-5-invars-def implicitly-factorized-clauses-subset-def*  
**by** *metis*

**hence**  $\mathcal{F}' \subseteq \mid N \mid \cup \mid U_{er}$   
**using**  $\langle C \in \mid N \mid \cup \mid U_{er} \rangle \langle \mathcal{F}' = \text{finsert } C \mathcal{F} \rangle$  **by** *simp*

**thus** *implicitly-factorized-clauses-subset*  $N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } C))$   
**unfolding** *implicitly-factorized-clauses-subset-def* **by** *simp*

**have** *dom-M-eq*:  $\text{dom } \mathcal{M} = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C$   
**using** *invars step-hyps*  
**by** (*simp add: model-eq-interp-upto-next-clause-def ord-res-5-invars-def*)

**have**  $\text{efac } C \notin \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**proof** (*rule notI*)  
**assume**  $\text{efac } C \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**show** *False*  
**proof** (*cases atm-of L*  $\in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$ )  
(*efac C*)  
**assume** *atm-of L*  $\in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$  (*efac C*)  
**hence** *atm-of L*  $\in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C$   
**using**  $\langle \text{efac } C \prec_c C \rangle$  *ord-res.production-subset-if-less-cls* **by** *blast*  
**hence**  $\text{dom } \mathcal{M} \models C$   
**using** *step-hyps*  
**by** (*metis dom-M-eq linorder-lit.is-maximal-in-mset-iff literal.collapse(1) pos-literal-in-imp-true-cls*)  
**thus** *False*  
**using**  $\langle \neg \text{dom } \mathcal{M} \models C \rangle$  **by** *contradiction*

**next**  
**assume** *atm-of L*  $\notin \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$  (*efac C*)  
(*C*)  
**hence**  $\text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$  (*efac C*)  $\models \text{efac } C$

**unfolding** *ord-res.production-unfold mem-Collect-eq*  
**using**  $\langle \text{efac } C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   
**by** (*metis Pos-atm-of-iff*  $\langle \text{efac } C \neq C \rangle$  *insert-DiffM*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*  
*linorder-lit.is-maximal-in-mset-iff linorder-lit.neqE*  
*obtains-positive-greatest-lit-if-efac-not-ident set-mset-efac step-hyps(4)*)  
**hence** *ord-res.interp* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$ ))  $C \models \text{efac } C$   
**using**  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   $\langle \text{efac } C \prec_c C \rangle$   $\langle \text{efac } C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   
*ord-res.lift-interp-entails* **by** *metis*  
**hence** *ord-res.interp* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$ ))  $C \models C$   
**by** (*simp add: true-cls-def*)  
**hence** *dom*  $\mathcal{M} \models C$   
**using** *dom-M-eq* **by** *argo*  
**thus** *False*  
**using**  $\langle \neg \text{dom } \mathcal{M} \models C \rangle$  **by** *contradiction*  
**qed**  
**qed**

**have** *iefac*  $\mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}) = \text{finsert}$  (*iefac*  $\mathcal{F}' C$ ) (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$ )  
 $-\{C\}$   
**proof** (*intro fsubset-antisym fsubsetI*)  
**fix**  $x :: 'f \text{ gclause}$   
**assume**  $x \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**thus**  $x \mid \in \mid \text{finsert}$  (*iefac*  $\mathcal{F}' C$ ) (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$ )  $|- \{C\}$   
**by** (*smt (verit)*  $\langle \text{efac } C \neq C \rangle$  *factorizable-if-neq-efac fimage-iff finsert-iff*  
*fminusI*  
*fsingletonE iefac-def ex1-efac-eq-factoring-chain step-hyps(7)*)  
**next**  
**fix**  $x :: 'f \text{ gclause}$   
**assume** *x-in:*  $x \mid \in \mid \text{finsert}$  (*iefac*  $\mathcal{F}' C$ ) (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$ )  $|- \{C\}$   
**hence**  $x = \text{iefac } \mathcal{F}' C \vee x \mid \in \mid (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})) \mid - \{C\}$   
**by** *blast*  
**thus**  $x \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**proof** (*elim disjE*)  
**assume**  $x = \text{iefac } \mathcal{F}' C$   
**thus**  $x \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**using**  $\langle C \mid \in \mid N \mid \cup \mid U_{er} \rangle$  **by** *blast*  
**next**  
**assume**  $x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \mid - \{C\}$   
**hence**  $x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**  $x \neq C$   
**by** *simp-all*  
**then obtain**  $x'$  **where**  $x' \mid \in \mid N \mid \cup \mid U_{er}$  **and**  $x = \text{iefac } \mathcal{F} x'$   
**by** *auto*  
**moreover have**  $x' \neq C$   
**using**  $\langle x \neq C \rangle$   $\langle x = \text{iefac } \mathcal{F} x' \rangle$   
**by** (*metis*  $\langle \text{efac } C \notin \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   $\langle x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$  *iefac-def*)  
**ultimately show**  $x \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er})$



```

    using iefac-def step-hyps(7) by simp
  qed
qed

have ord-res.interp (fset (iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) (efac C) =
  ord-res.interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) (efac C)
proof (rule ord-res.interp-swap-clause-set)
  show {D. D | $\in$ | iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )  $\wedge$  D  $\prec_c$  efac C} =
    {D. D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )  $\wedge$  D  $\prec_c$  efac C}
    unfolding  $\langle$ iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) = finsert (iefac  $\mathcal{F}'$  C) (iefac  $\mathcal{F}$  | $\uparrow$  (N
| $\cup$ |  $U_{er}$ )) - {C| $\rangle$ 
    using  $\langle$ efac C  $\prec_c$  C $\rangle$ 
    using iefac-def by force
  qed

also have ... = ord-res.interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C
proof -
  have  $\forall x$  | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ).  $x \prec_c C \longrightarrow$ 
    ord-res-Interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )))  $x \models x$ 
    using invars[unfolded ord-res-5-invars-def step-hyps(1)
      all-smaller-clauses-true-wrt-respective-Interp-def, simplified]
    by simp
  then have ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )))  $x = \{$ 
  if  $x$  | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) and efac C  $\prec_c x$  and  $x \prec_c C$  for  $x$ 
  proof -
    have  $x \preceq_l y \wedge y \preceq_l z$ 
    if  $X \preceq_c Y$  and  $Y \preceq_c Z$  and
      linorder-lit.is-maximal-in-mset X  $x$  and
      linorder-lit.is-maximal-in-mset Y  $y$  and
      linorder-lit.is-maximal-in-mset Z  $z$ 
    for  $x y z X Y Z$ 
    using that
    unfolding linorder-lit.is-maximal-in-mset-iff
    by (metis ord-res.asymp-less-lit ord-res.transp-less-lit linorder-cls.leD
      linorder-lit.leI linorder-lit.multipHO-if-maximal-less-that-maximal
      multp-eq-multpHO
      that(3) that(4) that(5))

  hence  $y = x$ 
  if  $X \preceq_c Y$  and  $Y \preceq_c Z$  and
    linorder-lit.is-maximal-in-mset X  $x$  and
    linorder-lit.is-maximal-in-mset Y  $y$  and
    linorder-lit.is-maximal-in-mset Z  $x$ 
  for  $x y X Y Z$ 
  using that
  by (metis linorder-lit.order.ordering-axioms ordering.antisym)

  hence  $y = x$ 
  if  $X \prec_c Y$  and  $Y \prec_c Z$  and

```

*linorder-lit.is-maximal-in-mset*  $X$   $x$  **and**  
*linorder-lit.is-maximal-in-mset*  $Y$   $y$  **and**  
*linorder-lit.is-maximal-in-mset*  $Z$   $x$   
**for**  $x$   $y$   $X$   $Y$   $Z$   
**using** *that* **by** *blast*

**hence**  $K = L$   
**if** *efac*  $C \prec_c x$  **and**  $x \prec_c C$  **and**  
*linorder-lit.is-maximal-in-mset* (*efac*  $C$ )  $L$  **and**  
*linorder-lit.is-maximal-in-mset*  $x$   $K$  **and**  
*linorder-lit.is-maximal-in-mset*  $C$   $L$   
**for**  $K$   
**using** *that* **by** *metis*

**hence**  $K = L$   
**if** *linorder-lit.is-maximal-in-mset*  $x$   $K$   
**for**  $K$   
**using** *that*  
**using** *ord-res.is-maximal-lit*  $L$   $C$   
**using** *efac*  $C \prec_c x$  *x*  $\prec_c C$  *ex1-efac-eq-factoring-chain*  
*ord-res.ground-factorings-preserves-maximal-literal* **by** *blast*

**hence** *ord-res.is-maximal-lit*  $L$   $x$   
**by** (*metis linorder-cls.leD linorder-lit.ex-maximal-in-mset mempty-lesseq-cls*  
*that(2)*)

**have**  $\# A$ .  $A \in$  *ord-res.production* (*fset* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \cup U_{er}$ )))  $x$   
**proof** (*rule notI* , *elim exE*)  
**fix**  $A :: 'f$  *gterm*  
**assume**  $A \in$  *ord-res.production* (*fset* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \cup U_{er}$ )))  $x$   
**then obtain**  $x'$  **where**  
 $x \in |$  *iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \cup U_{er}$ ) **and**  
 $x =$  *add-mset* (*Pos*  $A$ )  $x'$  **and**  
*ord-res.is-strictly-maximal-lit* (*Pos*  $A$ )  $x$  **and**  
 $\neg$  *ord-res.interp* (*fset* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \cup U_{er}$ )))  $x \Vdash x$   
**by** (*metis ord-res.mem-productionE*)

**have**  $A \in$  *ord-res.interp* (*fset* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \cup U_{er}$ )))  $C$   
**using**  $\langle A \in$  *ord-res.production* (*fset* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \cup U_{er}$ )))  $x$   
*ord-res.production-subset-if-less-cls* *that(3)* **by** *fastforce*

**moreover have**  $L =$  *Pos*  $A$   
**using** *ord-res.is-maximal-lit*  $L$   $x$  *ord-res.is-strictly-maximal-lit* (*Pos*  $A$ )

$x$

**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)

**moreover have**  $L \in \#$   $C$   
**using** *step-hyps linorder-lit.is-maximal-in-mset-iff* **by** *metis*

ultimately have  $ord-res.interp (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \models C$   
 by *auto*

hence  $dom \mathcal{M} \models C$   
 using *dom-M-eq* by *argo*

thus *False*  
 using  $\langle \neg dom \mathcal{M} \models C \rangle$  by *contradiction*  
 qed

thus *?thesis*  
 by *simp*  
 qed

moreover have  $\{x. x \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge x \prec_c C\} =$   
 $\{x. x \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge x \prec_c efac C\} \cup$   
 $\{x. x \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge efac C \prec_c x \wedge x \prec_c C\}$

proof –  
 have  $\{w \in NN. w \prec_c z\} = \{w \in NN. w \prec_c x\} \cup \{y \in NN. x \prec_c y \wedge y \prec_c$   
 $z\}$

if  $x \notin NN$  and  $x \prec_c z$  for  $NN x z$

proof –  
 have  $\{w \in NN. w \prec_c z\} = \{w \in NN. w \preceq_c x \vee x \prec_c w \wedge w \prec_c z\}$   
 using *that(2)* by *auto*  
 also have  $\dots = \{w \in NN. w \prec_c x \vee x \prec_c w \wedge w \prec_c z\}$   
 using *that(1)* by *auto*  
 also have  $\dots = \{w \in NN. w \prec_c x\} \cup \{y \in NN. x \prec_c y \wedge y \prec_c z\}$   
 by *auto*  
 finally show *?thesis* .

qed  
 thus *?thesis*  
 using  $\langle efac C \prec_c C \rangle \langle efac C \notin \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \rangle$  by (*simp only*):  
 qed

ultimately show *?thesis*  
 unfolding *ord-res.interp-def* by *auto*  
 qed

finally show *model-eq-interp-upto-next-clause N (U<sub>er</sub>, F', M, Some (efac C))*  
 unfolding *model-eq-interp-upto-next-clause-def*  
 using *dom-M-eq*  
 by *simp*

have *ord-res-Interp (fset (iefac F' \mid \uparrow (N \mid \cup \mid U<sub>er</sub>))) x \models x*  
 if  $x \in \mid iefac \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$  and  $x \prec_c efac C$  for  $x$   
 proof –  
 have  $x \in \mid iefac \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$   
 using *that*

**by** (*metis*  $\langle \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}) = \text{finsert } (\text{iefac } \mathcal{F}' C) (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})) \mid - \mid \{C\} \rangle$   
*finsert-iff fminusE iefac-def linorder-cls.neq-iff*)

**moreover have**  $x \prec_c C$   
**using**  $\langle x \prec_c \text{efac } C \rangle \langle \text{efac } C \prec_c C \rangle$  **by** *order*

**ultimately have** *ord-res-Interp* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))$ )  $x \models x$   
**using** *invars*  
**unfolding** *ord-res-5-invars-def*  
**unfolding** *all-smaller-clauses-true-wrt-respective-Interp-def step-hyps(1,2)*  
**by** *blast*

**moreover have** *ord-res-Interp* (*fset* (*iefac*  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))$ )  $x =$   
*ord-res-Interp* (*fset* (*iefac*  $\mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}))$ )  $x$

**proof** (*rule ord-res.Interp-swap-clause-set*)

**show**  $\{D. D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} D x\} =$   
 $\{D. D \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} D x\}$

**unfolding**  $\langle \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}) = \text{finsert } (\text{iefac } \mathcal{F}' C) (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})) \mid - \mid \{C\} \rangle$

**using**  $\langle x \prec_c \text{efac } C \rangle \langle x \prec_c C \rangle$

**by** (*metis* (*no-types, opaque-lifting*) *finsertCI finsertE fminusE fminusI fsingleton-iff*  
*iefac-def linorder-cls.less-le-not-le*)

**qed**

**ultimately show** *?thesis*

**by** *argo*

**qed**

**thus** *all-smaller-clauses-true-wrt-respective-Interp*  $N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } C))$

**unfolding** *all-smaller-clauses-true-wrt-respective-Interp-def* **by** *blast*

**have** *linorder-lit.is-greatest-in-mset* (*efac*  $C$ )  $L$

**using**  $\langle \text{linorder-lit.is-maximal-in-mset } C L \rangle$

**by** (*metis*  $\langle \text{efac } C \neq C \rangle$  *ex1-efac-eq-factoring-chain linorder-lit.Uniq-is-maximal-in-mset*  
*linorder-lit.is-greatest-in-mset-iff-is-maximal-and-count-eq-one*  
*ord-res.ground-factorings-preserved-maximal-literal*  
*obtains-positive-greatest-lit-if-efac-not-ident the1-equality'*)

**have**  $A \in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}))) D$

**if**  $\mathcal{M} A = \text{Some } D$  **for**  $A D$

**proof** –

**have**  $A \in \text{dom } \mathcal{M}$

**using**  $\langle \mathcal{M} A = \text{Some } D \rangle$  **by** *blast*

**hence**  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))) C$

**using** *dom-M-eq* **by** *argo*

```

have  $A \in \text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$ 
  using  $\langle \mathcal{M} A = \text{Some } D \rangle$ 
  unfolding  $\text{ord-res-5-invars-def step-hyps}(1,2)$ 
  unfolding  $\text{atoms-in-model-were-produced-def}$ 
  by  $\text{simp}$ 

hence  $\text{linorder-lit.is-greatest-in-mset } D (\text{Pos } A)$ 
  by  $(\text{metis } \text{ord-res.mem-production.E})$ 

moreover have  $\text{Pos } A \prec_l L$ 
  using  $\langle A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C \rangle$ 
  by  $(\text{smt } (\text{verit}, \text{del-insts}) \text{UN-E } \langle A \in \text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \rangle$ 
     $\text{calculation dom-}\mathcal{M}\text{-eq ord-res.interp-def ord-res.less-lit-simps}(1)$ 
     $\text{ord-res.totalp-less-lit linorder-cls.less-trans}$ 
     $\text{linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset}$ 
     $\text{linorder-lit.is-maximal-in-mset-iff linorder-lit.less-irrefl}$ 
     $\text{linorder-lit.mulp-if-maximal-less-that-maximal mem-Collect-eq}$ 
     $\text{ord-res.less-trm-iff-less-cls-if-mem-production}$ 
     $\text{pos-literal-in-imp-true-cls step-hyps}(3) \text{ step-hyps}(4) \text{ totalpD})$ 

ultimately have  $D \prec_c \text{efac } C$ 
  using  $\langle \text{linorder-lit.is-greatest-in-mset } (\text{efac } C) L \rangle$ 
  by  $(\text{metis } \text{ord-res.asymp-less-lit ord-res.transp-less-lit}$ 
     $\text{linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset}$ 
     $\text{linorder-lit.mulp}_{HO}\text{-if-maximal-less-that-maximal mulp-eq-mulp}_{HO})$ 

have  $\text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D =$ 
   $\text{ord-res.production } (fset (iefac \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}))) D$ 
proof  $(\text{rule } \text{ord-res.production-swap-clause-set})$ 
  have  $D \prec_c C$ 
  using  $\langle D \prec_c \text{efac } C \rangle \langle \text{efac } C \prec_c C \rangle$  by  $\text{order}$ 
  thus  $\{Da. Da \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} Da D\} =$ 
   $\{Da. Da \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} Da D\}$ 
  using  $\langle D \prec_c \text{efac } C \rangle$ 
  unfolding  $\langle \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}) = \text{finsert } (\text{iefac } \mathcal{F}' C) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \mid - \mid \{C\} \rangle$ 
  by  $(\text{metis } (\text{no-types}, \text{opaque-lifting}) \text{finsertE finsertI2 fminus-iff fsingleton-iff}$ 
     $\text{iefac-def linorder-cls.leD})$ 
qed

thus  $?thesis$ 
  using  $\langle A \in \text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D \rangle$  by  $\text{argo}$ 
qed

thus  $\text{atoms-in-model-were-produced } N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } C))$ 
  unfolding  $\text{atoms-in-model-were-produced-def}$  by  $\text{simp}$ 

```

```

have  $\mathcal{M} A = \text{Some } x$ 
  if  $x \prec_c \text{efac } C$  and  $A \in \text{ord-res.production (fset (iefac } \mathcal{F}' \mid \uparrow (N \mid \cup U_{er})) )} x$ 
  for  $x A$ 
proof -
  have  $x \prec_c C$ 
  using  $\langle x \prec_c \text{efac } C \rangle \langle \text{efac } C \prec_c C \rangle$  by order

  moreover have  $A \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) )} x$ 
  proof -
    have  $\text{ord-res.production (fset (iefac } \mathcal{F}' \mid \uparrow (N \mid \cup U_{er})) )} x =$ 
       $\text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) )} x$ 
    proof (rule ord-res.production-swap-clause-set)
      show  $\{D. D \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup U_{er}) \wedge (\prec_c)^{==} D x\} = \{D. D \mid \in \mid \text{iefac}$ 
 $\mathcal{F} \mid \uparrow (N \mid \cup U_{er}) \wedge (\prec_c)^{==} D x\}$ 
      using  $\langle x \prec_c \text{efac } C \rangle \langle x \prec_c C \rangle$ 
      unfolding  $\langle \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup U_{er}) = \text{finsert (iefac } \mathcal{F}' C) (\text{iefac } \mathcal{F} \mid \uparrow$ 
 $(N \mid \cup U_{er})) \mid - \mid \{C\} \rangle$ 
      by (metis (no-types, opaque-lifting) finsert-iff fminus-iff fsingleton-iff
iefac-def
        linorder-cls.dual-order.strict-iff-not)
    qed

  thus ?thesis
  using  $\langle A \in \text{ord-res.production (fset (iefac } \mathcal{F}' \mid \uparrow (N \mid \cup U_{er})) )} x \rangle$  by argo
qed

  ultimately show ?thesis
  using invars
  unfolding ord-res-5-invars-def step-hyps(1,2)
  unfolding all-produced-atoms-in-model-def
  by simp
qed

  thus all-produced-atoms-in-model  $N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some (efac } C))$ 
  unfolding all-produced-atoms-in-model-def by simp
qed
next
case step-hyps: (resolution-bot  $\mathcal{M} D L C U_{er}' U_{er} \mathcal{M}' \mathcal{F}$ )
have  $\mathcal{F} \mid \subseteq \mid N \mid \cup U_{er}$ 
  using invars
  unfolding step-hyps(1,2) ord-res-5-invars-def implicitly-factorized-clauses-subset-def
  by metis

  hence  $\mathcal{F} \mid \subseteq \mid N \mid \cup U_{er}'$ 
  using step-hyps by blast

  moreover have linorder-cls.is-least-in-fset  $(\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')) \{ \# \}$ 
  using step-hyps linorder-cls.is-least-in-fset-iff mempty-lesseq-cls by fastforce

```

**ultimately show** *?thesis*  
**using** *step-hyps*  
**using** *ord-res-5-invars-initial-state*  
**by** (*metis ord-res-5-invars-initial-state*)  
**next**  
**case** *step-hyps: (resolution-pos M E L D U<sub>er</sub>' U<sub>er</sub> M' K F)*  
  
**hence**  
*L-max: ord-res.is-maximal-lit L E* **and**  
*L-neg: is-neg L*  
**using** *step-hyps* **by** *simp-all*  
  
**have** *F-subset: F | $\subseteq$ | N | $\cup$ | U<sub>er</sub>*  
**using** *invars*  
**unfolding** *step-hyps(1,2) ord-res-5-invars-def implicitly-factorized-clauses-subset-def*  
**by** *metis*  
  
**have** *eres D E  $\neq$  E*  
**using** *step-hyps* **by** (*metis linorder-lit.Uniq-is-maximal-in-mset the1-equality'*)  
  
**moreover have** *eres D E  $\preceq_c$  E*  
**using** *eres-le* .  
  
**ultimately have** *eres D E  $\prec_c$  E*  
**by** *order*  
  
**have**  $\forall F.$  *is-least-false-clause (iefac F | $\uparrow$ | (N | $\cup$ | U<sub>er</sub>)) F  $\longrightarrow$  E  $\preceq_c$  F*  
**using** *invars*  
**unfolding** *ord-res-5-invars-def step-hyps(1,2)*  
**using** *next-clause-lt-least-false-clause[of N (U<sub>er</sub>, F, M, Some E)]*  
**by** *simp*  
  
**have** *E-least-false: is-least-false-clause (iefac F | $\uparrow$ | (N | $\cup$ | U<sub>er</sub>)) E*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
**proof** (*intro conjI ballI impI*)  
**show** *E | $\in$ | iefac F | $\uparrow$ | (N | $\cup$ | U<sub>er</sub>)*  
**using** *invars*  
**unfolding** *ord-res-5-invars-def step-hyps(1,2)*  
**by** (*metis next-clause-in-factorized-clause-def*)  
**next**  
**have**  $\neg$  *ord-res.interp (fset (iefac F | $\uparrow$ | (N | $\cup$ | U<sub>er</sub>))) E  $\models$  E*  
**using** *invars*  
**unfolding** *ord-res-5-invars-def step-hyps(1,2)*  
**using**  $\langle \neg \text{dom } \mathcal{M} \models E \rangle$  **by** (*metis model-eq-interp-upto-next-clause-def*)  
  
**moreover have** *ord-res.production (fset (iefac F | $\uparrow$ | (N | $\cup$ | U<sub>er</sub>))) E = {}*  
**proof** –  
**have**  $\nexists L.$  *is-pos L  $\wedge$  ord-res.is-strictly-maximal-lit L E*  
**using**  $\langle \text{ord-res.is-maximal-lit } L E \rangle$   $\langle \text{is-neg } L \rangle$

**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)  
**thus** *?thesis*  
**using** *unproductive-if-nex-strictly-maximal-pos-lit* **by** *metis*  
**qed**

**ultimately show**  $\neg \text{ord-res-Interp } (fset \ (iefac \ \mathcal{F} \ |^{\dagger} \ (N \ | \cup \ U_{er}))) \ E \models E$   
**by** *simp*  
**next**  
**fix**  $F$   
**assume**  $F\text{-in}: F \in |iefac \ \mathcal{F} \ |^{\dagger} \ (N \ | \cup \ U_{er})$  **and**  $F \neq E$  **and**  
 $F\text{-false}: \neg \text{ord-res-Interp } (fset \ (iefac \ \mathcal{F} \ |^{\dagger} \ (N \ | \cup \ U_{er}))) \ F \models F$

**have**  $\neg F \prec_c E$   
**using** *invars*  
**unfolding** *ord-res-5-invars-def step-hyps(1,2)*  
**unfolding** *all-smaller-clauses-true-wrt-respective-Interp-def*  
**using**  $F\text{-in}$   $F\text{-false}$   
**by** (*metis option.inject*)

**thus**  $E \prec_c F$   
**using**  $\langle F \neq E \rangle$  **by** *order*  
**qed**

**have**  $L\text{-prod-by-D}: atm\text{-of } L \in \text{ord-res.production } (fset \ (iefac \ \mathcal{F} \ |^{\dagger} \ (N \ | \cup \ U_{er})))$   
 $D$   
**using** *invars*  
**unfolding** *ord-res-5-invars-def step-hyps(1,2)*  
**by** (*metis atoms-in-model-were-produced-def step-hyps(6)*)

**hence**  $D\text{-prod}: \text{ord-res.production } (fset \ (iefac \ \mathcal{F} \ |^{\dagger} \ (N \ | \cup \ U_{er}))) \ D \neq \{\}$   
**by** (*metis empty-iff*)

**have**  $\text{ord-res.is-maximal-lit } (-L) \ D$   
**using**  $L\text{-prod-by-D}$   $L\text{-neg}$   
**by** (*metis linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset literal.collapse(2)*  
*ord-res.mem-productionE uminus-Neg*)

**moreover have**  $-L \prec_l L$   
**using**  $L\text{-neg}$   
**by** (*metis Neg-atm-of-iff atm-of-uminus linorder-lit.not-less-iff-gr-or-eq*  
*linorder-trm.less-imp-not-eq literal.collapse(1) ord-res.less-lit-simps(4) umi-*  
*nus-not-id*)

**ultimately have**  $D \prec_c E$   
**using**  $L\text{-max}$  *linorder-lit.mulp-if-maximal-less-that-maximal* **by** *metis*

**have**  $eres \ D \ E \notin |iefac \ \mathcal{F} \ |^{\dagger} \ (N \ | \cup \ U_{er})$   
**proof** (*rule notI*)



**assume**  $eres\ D\ E\ |\in|\ iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})$   
**moreover have**  $\neg\ (E\ \prec_c\ eres\ D\ E)$   
**using**  $\langle eres\ D\ E\ \prec_c\ E \rangle$  **by order**  
**ultimately have**  $ord\text{-}res\text{-}Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ (eres\ D\ E)\ \models$   
 $eres\ D\ E$   
**using**  $E\text{-least-false}\ \langle eres\ D\ E\ \neq\ E \rangle$   
**unfolding**  $is\text{-}least\text{-}false\text{-}clause\text{-}def\ linorder\text{-}cls.is\text{-}least\text{-}in\text{-}ffilter\text{-}iff$   
**by metis**  
**then show**  $False$   
**by**  $(metis\ (no\text{-}types,\ lifting)\ D\text{-}prod\ E\text{-}least\text{-}false\ clause\text{-}true\text{-}if\text{-}resolved\text{-}true$   
 $ex1\text{-}eres\text{-}eq\text{-}full\text{-}run\text{-}ground\text{-}resolution\ full\text{-}run\text{-}def\ is\text{-}least\text{-}false\text{-}clause\text{-}def$   
 $linorder\text{-}cls.is\text{-}least\text{-}in\text{-}fset\text{-}ffilterD(2)\ rtranclpD)$   
**qed**

**moreover have**  $efac\ (eres\ D\ E)\ |\notin|\ iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})$   
**proof**  $(rule\ notI)$   
**have**  $efac\ (eres\ D\ E)\ \preceq_c\ eres\ D\ E$   
**by**  $(meson\ efac\text{-}subset\ subset\text{-}implies\text{-}reflclp\text{-}multp)$   
**hence**  $\neg\ (E\ \prec_c\ efac\ (eres\ D\ E))$   
**using**  $\langle eres\ D\ E\ \prec_c\ E \rangle$  **by order**  
**moreover assume**  $efac\ (eres\ D\ E)\ |\in|\ iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})$   
**moreover have**  $efac\ (eres\ D\ E)\ \neq\ E$   
**by**  $(metis\ \langle eres\ D\ E\ \prec_c\ E \rangle\ efac\text{-}properties\text{-}if\text{-}not\text{-}ident(1)\ linorder\text{-}cls.not\text{-}less\text{-}iff\text{-}gr\text{-}or\text{-}eq)$   
**ultimately have**  $ord\text{-}res\text{-}Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ (efac\ (eres\ D$   
 $E))\ \models\ efac\ (eres\ D\ E)$   
**using**  $E\text{-least-false}$   
**unfolding**  $is\text{-}least\text{-}false\text{-}clause\text{-}def\ linorder\text{-}cls.is\text{-}least\text{-}in\text{-}ffilter\text{-}iff$   
**by metis**  
**hence**  $ord\text{-}res\text{-}Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ (eres\ D\ E)\ \models\ eres\ D\ E$   
**using**  $\langle efac\ (eres\ D\ E)\ \preceq_c\ eres\ D\ E \rangle$   $ord\text{-}res.entailed\text{-}clause\text{-}stays\text{-}entailed$  **by**  
 $fastforce$   
**hence**  $ord\text{-}res\text{-}Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ E\ \models\ E$   
**using**  $clause\text{-}true\text{-}if\text{-}resolved\text{-}true$   
**by**  $(smt\ (verit)\ D\text{-}prod\ ex1\text{-}eres\text{-}eq\text{-}full\text{-}run\text{-}ground\text{-}resolution\ full\text{-}run\text{-}def\ rtran\text{-}$   
 $clpD)$   
**moreover have**  $\neg\ ord\text{-}res\text{-}Interp\ (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ E\ \models\ E$   
**using**  $E\text{-least-false}\ is\text{-}least\text{-}false\text{-}clause\text{-}def\ linorder\text{-}cls.is\text{-}least\text{-}in\text{-}fset\text{-}ffilterD(2)$   
**by blast**  
**ultimately show**  $False$   
**by contradiction**  
**qed**

**ultimately have**  $eres\ D\ E\ |\notin|\ N\ |\cup|\ U_{er}$   
**unfolding**  $iefac\text{-}def$  **by fastforce**

**hence**  $iefac\ \mathcal{F}\ (eres\ D\ E) = eres\ D\ E$   
**unfolding**  $iefac\text{-}def$   
**using**  $\mathcal{F}\text{-subset}$  **by auto**

**hence**  $iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er}') = finsert (eres D E) (iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er}))$   
**unfolding**  $\langle U_{er}' = finsert (eres D E) U_{er} \rangle$  **by** *simp*

**show** *?thesis*  
**unfolding** *ord-res-5-invars-def step-hyps(1,2)*  
**proof** (*intro conjI*)  
**have**  $eres D E \mid \in \mid iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er}')$   
**unfolding**  $\langle iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er}') = finsert (eres D E) (iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er})) \rangle$  **by** *simp*

**thus** *next-clause-in-factorized-clause*  $N (U_{er}', \mathcal{F}, \mathcal{M}', Some (eres D E))$   
**unfolding** *next-clause-in-factorized-clause-def* **by** *simp*

**have**  $\mathcal{F} \mid \subseteq \mid N \mid \cup U_{er}'$   
**unfolding**  $\langle U_{er}' = finsert (eres D E) U_{er} \rangle$   
**using**  $\langle \mathcal{F} \mid \subseteq \mid N \mid \cup U_{er} \rangle$  **by** *blast*

**thus** *implicitly-factorized-clauses-subset*  $N (U_{er}', \mathcal{F}, \mathcal{M}', Some (eres D E))$   
**unfolding** *implicitly-factorized-clauses-subset-def* **by** *simp*

**have** *dom-M-eq*:  $dom \mathcal{M} = ord-res.interp (fset (iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er}))) E$   
**using** *invars*  
**unfolding** *step-hyps(1,2) ord-res-5-invars-def model-eq-interp-upto-next-clause-def*  
**by** *metis*

**have**  $\forall x \in \# E. \neg dom \mathcal{M} \models x$   
**using**  $\langle \neg dom \mathcal{M} \models E \rangle$  **by** (*simp add: true-cls-def*)

**moreover have**  $\forall x \in \# D. x \neq -L \longrightarrow \neg dom \mathcal{M} \models x$   
**proof** –  
**have**  $\forall x \in \# D. x \neq -L \longrightarrow \neg ord-res.interp (fset (iefac \mathcal{F} \mid^{\dagger} (N \mid \cup U_{er})))$   
 $D \models x$   
**using** *L-prod-by-D* **by** (*metis ord-res.mem-productionE true-cls-def*)  
**moreover have**  $\forall x \in \# D. x \neq -L \longrightarrow atm-of x \prec_t atm-of (-L)$   
**using**  $\langle ord-res.is-maximal-lit (-L) D \rangle$  *L-neg*  
**by** (*smt (verit, best) L-prod-by-D atm-of-eq-atm-of linorder-cls.order-refl*  
*linorder-trm.antisym-conv1 ord-res.less-trm-if-neg ord-res.lesseq-trm-if-pos*)  
**ultimately have**  $\forall x \in \# D. x \neq -L \longrightarrow \neg ord-res.interp (fset (iefac \mathcal{F} \mid^{\dagger}$   
 $(N \mid \cup U_{er}))) E \models x$   
**using** *ord-res.interp-fixed-for-smaller-literals*  
 $OF \langle ord-res.is-maximal-lit (-L) D \rangle - \langle D \prec_c E \rangle$   
**by** *fastforce*

**then show** *?thesis*  
**unfolding** *dom-M-eq[symmetric]* .  
**qed**

**moreover have**  $K \in \# eres D E$   
**using**  $\langle ord-res.is-maximal-lit K (eres D E) \rangle$

**using** *linorder-lit.is-maximal-in-mset-iff* **by** *metis*

**moreover have**  $\forall x \in\# \text{eres } D \ E. x \in\# D \vee x \in\# E$   
**using** *lit-in-one-of-resolvents-if-in-eres* **by** *metis*

**moreover have**  $\forall x \in\# \text{eres } D \ E. x \neq - L$

**proof** (*intro ballI notI*)

**fix**  $x$  **assume**  $x \in\# \text{eres } D \ E \ x = - L$

**obtain**  $m \ A \ D' \ E'$  **where**

*ord-res.is-strictly-maximal-lit* (*Pos*  $A$ )  $D$  **and**

$D = \text{add-mset}$  (*Pos*  $A$ )  $D'$  **and**

$E = \text{replicate-mset}$  (*Suc*  $m$ ) (*Neg*  $A$ ) +  $E'$  **and**

*Neg*  $A \notin\# E'$  **and**

$\text{eres } D \ E = \text{repeat-mset}$  (*Suc*  $m$ )  $D' + E'$

**using**  $\langle \text{eres } D \ E \neq E' [ \text{THEN } \text{eres-not-ident} D ] \rangle$  **by** *metis*

**have**  $L = \text{Neg } A$

**using**  $\langle \text{ord-res.is-strictly-maximal-lit} \ (\text{Pos } A) \ D \rangle$

**by** (*metis* *L-neg L-prod-by-D Uniq-D ord-res.mem-productionE*  
*linorder-lit.Uniq-is-greatest-in-mset literal.collapse(2) uminus-Pos*)

**have**  $x \in\# D' \vee x \in\# E'$

**using**  $\langle x \in\# \text{eres } D \ E \rangle$

**unfolding**  $\langle \text{eres } D \ E = \text{repeat-mset} \ (\text{Suc } m) \ D' + E' \rangle$

**by** (*metis member-mset-repeat-mset-Suc union-iff*)

**thus** *False*

**proof** (*elim disjE*)

**assume**  $x \in\# D'$

**hence**  $\text{Pos } A \in\# D'$

**unfolding**  $\langle x = - L \rangle \langle L = \text{Neg } A \rangle$  **by** *simp*

**hence**  $\neg \text{ord-res.is-strictly-maximal-lit} \ (\text{Pos } A) \ D$

**using**  $\langle D = \text{add-mset} \ (\text{Pos } A) \ D' \rangle$

**using** *linorder-lit.is-greatest-in-mset-iff* **by** *auto*

**thus** *False*

**using**  $\langle \text{ord-res.is-strictly-maximal-lit} \ (\text{Pos } A) \ D \rangle$  **by** *contradiction*

**next**

**assume**  $x \in\# E'$

**hence**  $\text{Pos } A \in\# E'$

**unfolding**  $\langle x = - L \rangle \langle L = \text{Neg } A \rangle$  **by** *simp*

**hence**  $\text{Pos } A \in\# E$

**unfolding**  $\langle E = \text{replicate-mset} \ (\text{Suc } m) \ (\text{Neg } A) + E' \rangle$  **by** *simp*

**hence** *ord-res.production* (*fset* (*iefac*  $\mathcal{F}$  | $\uparrow$  ( $N \ |\cup| \ U_{er}$ )))  $D \models_l \ \text{Pos } A$

**using** *L-prod-by-D*  $\langle L = \text{Neg } A \rangle$  **by** *auto*

**hence** *ord-res.interp* (*fset* (*iefac*  $\mathcal{F}$  | $\uparrow$  ( $N \ |\cup| \ U_{er}$ )))  $E \models_l \ \text{Pos } A$

**by** (*metis*  $\langle L = \text{Neg } A \rangle$  *dom-M-eq linorder-lit.is-maximal-in-mset-iff*

*neg-literal-notin-imp-true-cls step-hyps(3) step-hyps(4) true-lit-simps(1)*)

**hence** *ord-res.interp* (*fset* (*iefac*  $\mathcal{F}$  | $\uparrow$  ( $N \ |\cup| \ U_{er}$ )))  $E \models E$

**using**  $\langle \text{Pos } A \in\# E \rangle$  **by** *blast*

**hence** *dom M*  $\models E$

```

    using dom-M-eq by argo
  thus False
    using  $\langle \neg \text{dom } \mathcal{M} \models E \rangle$  by contradiction
qed
qed

ultimately have  $\neg \text{dom } \mathcal{M} \models_l K$ 
  by metis

have  $\text{dom } \mathcal{M}' = \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')))$  (eres D E)
proof (intro subset-antisym subsetI)
  fix A :: 'f gterm
  assume  $A \in \text{dom } \mathcal{M}'$ 
  hence  $A \in \text{dom } \mathcal{M}$  and  $A \prec_t \text{atm-of } K$ 
    unfolding  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$  by simp-all

  have  $A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  E
    using  $\langle A \in \text{dom } \mathcal{M} \rangle$ 
  unfolding  $\langle \text{dom } \mathcal{M} = \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  E  $\rangle$  .
  hence  $A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  (eres D E)
    using  $\text{ord-res.interp-fixed-for-smaller-literals } \langle \text{ord-res.is-maximal-lit } K$  (eres
D E)  $\rangle$ 
     $\langle A \prec_t \text{atm-of } K \rangle \langle \text{eres } D \ E \prec_c E \rangle$ 
  by metis
  thus  $A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')))$  (eres D E)
    unfolding  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') = \text{finsert } (eres D E) (iefac \mathcal{F} \mid \uparrow (N \mid \cup
U_{er})) \rangle$ 
    using  $\text{ord-res.interp-insert-greater-clause-strong}$  by simp
  next
  fix A :: 'f gterm
  assume  $A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')))$  (eres D E)
  hence  $A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  (eres D E)
    unfolding  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') = \text{finsert } (eres D E) (iefac \mathcal{F} \mid \uparrow (N \mid \cup
U_{er})) \rangle$ 
    using  $\text{ord-res.interp-insert-greater-clause-strong}$  by simp
  hence  $A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  E
    using  $\langle \text{eres } D \ E \prec_c E \rangle$   $\text{ord-res.interp-subset-if-less-cl}$  by blast
  hence  $A \in \text{dom } \mathcal{M}$ 
    unfolding  $\langle \text{dom } \mathcal{M} = \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  E  $\rangle$  .

moreover have  $A \prec_t \text{atm-of } K$ 
proof -
  obtain C where
    C  $\mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})$  and
    C  $\prec_c \text{eres } D \ E$  and
    A-prod-by-C:  $A \in \text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  C
  using  $\langle A \in \text{ord-res.interp } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er})))$  (eres D E)  $\rangle$ 
  unfolding  $\text{ord-res.interp-def}$  by blast

```

**have** *ord-res.is-maximal-lit* (*Pos A*) *C*  
**using** *A-prod-by-C ord-res.mem-productionE* **by** *blast*

**hence**  $A \preceq_t \text{atm-of } K$   
**using** *ord-res.is-maximal-lit K (eres D E)*  $\langle C \prec_c \text{eres } D E \rangle$   
**by** (*metis linorder-cls.dual-order.asym linorder-lit.multip-if-maximal-less-that-maximal*  
*linorder-trm.not-le-imp-less literal.collapse(1) ord-res.less-lit-simps(1)*  
*step-hyps(11)*)

**moreover have**  $A \neq \text{atm-of } K$   
**using**  $\langle A \in \text{dom } \mathcal{M} \rangle \langle \neg \text{dom } \mathcal{M} \models_l K \rangle \langle \text{is-pos } K \rangle$  **by** *force*

**ultimately show** *?thesis*  
**by** *order*  
**qed**

**ultimately show**  $A \in \text{dom } \mathcal{M}'$   
**unfolding**  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$  **by** *simp*  
**qed**

**thus** *model-eq-interp-upto-next-clause*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } (\text{eres } D E))$   
**unfolding** *model-eq-interp-upto-next-clause-def* **by** *simp*

**have**  $\forall C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c \text{eres } D E \longrightarrow$   
*ord-res-Interp (iefac } \mathcal{F} ' (fset } N \cup fset } U\_{er}') C \models C*  
**by** (*smt (verit, ccfv-threshold) E-least-false Uniq-def Uniq-is-least-false-clause*  
 $\langle \text{eres } D E \prec_c E \rangle \langle \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F}$   
 $| \uparrow (N \cup U_{er})) \rangle$   
*finite-fset finsert.rep-eq finsertE finsert-absorb*  
*fset.set-map ord-res.transp-less-cls*  
*is-least-false-clause-finsert-smaller-false-clause linorder-cls.max.strict-order-iff*  
*ord-res.interp-insert-greater-clause ord-res.production-insert-greater-clause*  
*transpE*  
*true-cls-iefac-iff union-fset*)

**hence**  $\forall C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}'). C \prec_c \text{eres } D E \longrightarrow$   
*ord-res-Interp (iefac } \mathcal{F} ' (fset } N \cup fset } U\_{er}') C \models C*  
**unfolding**  $\langle \text{eres } D E \prec_c E \rangle \langle \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}') = \text{finsert } (\text{eres } D E)$   
 $(\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er})) \rangle$   
**by** *simp*

**thus** *all-smaller-clauses-true-wrt-respective-Interp*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } (\text{eres } D E))$   
**unfolding** *all-smaller-clauses-true-wrt-respective-Interp-def* **by** *simp*

**have**  $A \in \text{ord-res.production } (fset (\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}')))$  *C*  
**if**  $\mathcal{M}' A = \text{Some } C$  **for**  $A C$   
**proof** –  
**have**  $\mathcal{M} A = \text{Some } C$  **and**  $A \prec_t \text{atm-of } K$

**unfolding** *atomize-conj*  
**using**  $\langle \mathcal{M}' A = \text{Some } C \rangle \langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$   
**by** (*metis Int-iff domI dom-restrict mem-Collect-eq restrict-in*)

**hence** *A-prod-by-C*:  $A \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) } C$   
**using** *invars*  
**unfolding** *step-hyps(1,2) ord-res-5-invars-def atoms-in-model-were-produced-def*  
**by** *metis*

**moreover have** *ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U\_{er}))) } C =*  
*ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U\_{er}')) ) C*  
**proof** (*rule ord-res.production-swap-clause-set*)  
**have** *ord-res.is-strictly-maximal-lit (Pos A) C*  
**using** *A-prod-by-C ord-res.mem-productionE* **by** *metis*  
**moreover have** *Pos A \prec\_l K*  
**using**  $\langle A \prec_t \text{atm-of } K \rangle$   
**by** (*metis Pos-atm-of-iff ord-res.less-lit-simps(1) step-hyps(11)*)  
**ultimately have**  $C \prec_c \text{eres } D E$   
**using** *linorder-lit.mulp-if-maximal-less-that-maximal step-hyps(10)* **by**

*blast*

**thus**  $\{D. D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} D C\} =$   
 $\{D. D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') \wedge (\prec_c)^{==} D C\}$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert (eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N$   
 $\mid \cup \mid U_{er})) \rangle$   
**by** *auto*  
**qed**

**ultimately show** *?thesis*  
**by** *argo*  
**qed**

**thus** *atoms-in-model-were-produced N (U\_{er}', \mathcal{F}, \mathcal{M}', \text{Some (eres } D E))*  
**unfolding** *atoms-in-model-were-produced-def* **by** *simp*

**have**  $\mathcal{M}' A = \text{Some } C$   
**if**  $C \prec_c \text{eres } D E$  **and** *A-in*:  $A \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid$   
 $U_{er}')) ) C$   
**for**  $C A$   
**proof** –  
**have**  $C \prec_c E$   
**using**  $\langle C \prec_c \text{eres } D E \rangle \langle \text{eres } D E \prec_c E \rangle$  **by** *order*

**moreover have** *A-prod-by-C*:  $A \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid$   
 $U_{er})) ) C$   
**proof** –  
**have** *ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U\_{er})) ) C =*  
*ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U\_{er}')) ) C*  
**proof** (*rule ord-res.production-swap-clause-set*)  
**show**  $\{D. D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} D C\} =$

```

      {D. D ∈| iefac F |' (N |∪| Uer' ∧ (≺c)== D C}
unfolding ⟨iefac F |' (N |∪| Uer') = finsert (eres D E) (iefac F |' (N
|∪| Uer'))⟩
  using ⟨C ≺c eres D E⟩
  by (metis (no-types, opaque-lifting) finsert-iff linorder-cls.less-le-not-le)
qed

thus ?thesis
  using A-in by argo
qed

ultimately have M A = Some C
  using invars
unfolding step-hyps(1,2) ord-res-5-invars-def all-produced-atoms-in-model-def
by metis

moreover have A ≺t atm-of K
proof –
  have A ∈ dom M
  using ⟨M A = Some C⟩ by auto

  have ord-res.is-maximal-lit (Pos A) C
  using A-prod-by-C ord-res.mem-productionE by blast

  hence A ≼t atm-of K
  using ⟨ord-res.is-maximal-lit K (eres D E)⟩ ⟨C ≺c eres D E⟩
by (metis linorder-cls.dual-order.asym linorder-lit.multip-if-maximal-less-that-maximal
linorder-trm.not-le-imp-less literal.collapse(1) ord-res.less-lit-simps(1)
step-hyps(11))

moreover have A ≠ atm-of K
  using ⟨A ∈ dom M⟩ ⟨¬ dom M  $\models$  l K⟩ ⟨is-pos K⟩ by force

ultimately show ?thesis
  by order
qed

ultimately show M' A = Some C
  unfolding ⟨M' = restrict-map M {A. A ≺t atm-of K}⟩ by simp
qed

thus all-produced-atoms-in-model N (Uer', F, M', Some (eres D E))
  unfolding all-produced-atoms-in-model-def by simp
qed
next
case step-hyps: (resolution-neg M E L D Uer' Uer M' K C F)

obtain AL where L-def: L = Neg AL
  using ⟨is-neg L⟩ by (cases L) simp-all

```

**have**  $A_L \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) ) D$   
**using**  $\langle \mathcal{M} (\text{atm-of } L) = \text{Some } D \rangle$   
**unfolding**  $\text{step-hyps}(1,2)$   $\text{ord-res-5-invars-def atoms-in-model-were-produced-def}$   
**unfolding**  $L\text{-def literal.sel}$   
**by**  $\text{metis}$

**hence**  $D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $\text{ord-res.is-strictly-maximal-lit (Pos } A_L) D$  **and**  
 $D\text{-false: } \neg \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) ) D \Vdash D$   
**unfolding**  $\text{atomize-conj}$  **by**  $(\text{metis ord-res.mem-productionE})$

**obtain**  $A_K$  **where**  $K\text{-def: } K = \text{Neg } A_K$   
**using**  $\langle \text{is-neg } K \rangle$  **by**  $(\text{cases } K) \text{ simp-all}$

**have**  $A_K \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) ) C$   
**using**  $\langle \mathcal{M} (\text{atm-of } K) = \text{Some } C \rangle$   
**unfolding**  $\text{step-hyps}(1,2)$   $\text{ord-res-5-invars-def atoms-in-model-were-produced-def}$   
**unfolding**  $K\text{-def literal.sel}$   
**by**  $\text{metis}$

**hence**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $\text{ord-res.is-strictly-maximal-lit (Pos } A_K) C$  **and**  
 $C\text{-false: } \neg \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) ) C \Vdash C$   
**unfolding**  $\text{atomize-conj}$  **by**  $(\text{metis ord-res.mem-productionE})$

**have**  $D \prec_c E$   
**using**  $\langle \text{ord-res.is-strictly-maximal-lit (Pos } A_L) D \rangle \langle \text{ord-res.is-maximal-lit } L$   
 $E \rangle [\text{unfolded } L\text{-def}]$   
**using**  $\text{linorder-lit.mulp-if-maximal-less-that-maximal ord-res.less-lit-simps}(2)$   
**by**  $\text{blast}$

**have**  $\text{eres } D E \neq E$   
**using**  $\langle \text{ord-res.is-strictly-maximal-lit (Pos } A_L) D \rangle \langle \text{ord-res.is-maximal-lit } L$   
 $E \rangle [\text{unfolded } L\text{-def}]$   
**by**  $(\text{metis } L\text{-def eres-ident-iff ex-ground-resolutionI is-pos-def}$   
 $\text{linorder-lit.is-greatest-in-mset-iff-is-maximal-and-count-eq-one}$   
 $\text{linorder-lit.mulp-if-maximal-less-that-maximal linorder-lit.neq-iff}$   
 $\text{linorder-trm.dual-order.asym ord-res.less-lit-simps}(4) \text{ ground-resolution-def}$   
 $\text{step-hyps}(5))$

**moreover have**  $\text{eres } D E \preceq_c E$   
**using**  $\text{eres-le}$  .

**ultimately have**  $\text{eres } D E \prec_c E$   
**by**  $\text{order}$

**have**  $\text{iefac } \mathcal{F} (\text{eres } D E) = \text{eres } D E$   
**by**  $(\text{metis (mono-tags, lifting) Uniq-D efac-spec iefac-def is-pos-def})$



*linorder-lit.Uniq-is-maximal-in-mset step-hyps(10) step-hyps(11)*)

**hence**  $\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}))$   
**unfolding**  $\langle U_{er}' = \text{finsert } (\text{eres } D E) U_{er} \rangle$  **by simp**

**hence**  $\{ \mid C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') . C \prec_c \text{eres } D E \mid \} =$   
 $\{ \mid C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}) . C \prec_c \text{eres } D E \mid \}$   
**by auto**

**show** *?thesis*

**unfolding** *step-hyps(1,2) ord-res-5-invars-def*

**proof** (*intro conjI*)

**have**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')$

**using**  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}) \rangle$

**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) \rangle$

**by simp**

**thus** *next-clause-in-factorized-clause*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$

**unfolding** *next-clause-in-factorized-clause-def* **by simp**

**have**  $\mathcal{F} \mid \subseteq \mid N \mid \cup U_{er}$

**using** *invars*

**unfolding** *step-hyps(1,2) ord-res-5-invars-def implicitly-factorized-clauses-subset-def*

**by metis**

**hence**  $\mathcal{F} \mid \subseteq \mid N \mid \cup U_{er}'$

**unfolding**  $\langle U_{er}' = \text{finsert } (\text{eres } D E) U_{er} \rangle$  **by blast**

**thus** *implicitly-factorized-clauses-subset*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$

**unfolding** *implicitly-factorized-clauses-subset-def* **by simp**

**have**  $\text{Pos } A_K \prec_l \text{Neg } A_K$

**by simp**

**hence**  $C \prec_c \text{eres } D E$

**using**  $\langle \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A_K) C \rangle$

$\langle \text{ord-res.is-maximal-lit } K (\text{eres } D E) \rangle$  *[unfolded K-def]*

**using** *linorder-lit.mulp-if-maximal-less-than-maximal* **by blast**

**have**  $C \prec_c E$

**using**  $\langle C \prec_c \text{eres } D E \rangle \langle \text{eres } D E \prec_c E \rangle$  **by order**

**have**  $\{ \mid x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') . x \prec_c C \mid \} = \{ \mid x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}) . x \prec_c C \mid \}$

**using**  $\langle C \prec_c \text{eres } D E \rangle$

**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) \rangle$

**by** (*metis ffilter-eq-ffilter-minus-singleton finsert-absorb fminus-finsert-absorb*)

*linorder-cls.less-asymp*)

**hence** *ord-res.interp* (fset (iefac  $\mathcal{F}$  |<sup>q</sup> (N | $\cup$ |  $U_{er}$ ')))  $C =$   
*ord-res.interp* (fset (iefac  $\mathcal{F}$  |<sup>q</sup> (N | $\cup$ |  $U_{er}$ )))  $C$   
**using**  $\langle C \prec_c \text{eres } D \ E \rangle$   
**by** (*simp add*:  $\langle \text{iefac } \mathcal{F} \text{ |}^q (N \text{ |} \cup \text{ | } U_{er}') = \text{finsert} (\text{eres } D \ E) (\text{iefac } \mathcal{F} \text{ |}^q (N \text{ |} \cup \text{ | } U_{er})) \rangle$   
*ord-res.interp-insert-greater-clause*)

**have** *dom-M-eq*:  $\text{dom } \mathcal{M} = \text{ord-res.interp} (\text{fset} (\text{iefac } \mathcal{F} \text{ |}^q (N \text{ |} \cup \text{ | } U_{er}))) \ E$   
**using** *invars*  
**unfolding** *step-hyps(1,2)* *ord-res-5-invars-def* *model-eq-interp-upto-next-clause-def*  
**by** *metis*

**have**  $\forall x \in \# \ E. \neg \text{dom } \mathcal{M} \models_l x$   
**using**  $\langle \neg \text{dom } \mathcal{M} \models E \rangle$  **by** (*simp add*: *true-cls-def*)

**moreover have**  $\forall x \in \# \ D. x \neq -L \longrightarrow \neg \text{dom } \mathcal{M} \models_l x$   
**proof** –  
**have**  $\forall x \in \# \ D. x \neq -L \longrightarrow \neg \text{ord-res.interp} (\text{fset} (\text{iefac } \mathcal{F} \text{ |}^q (N \text{ |} \cup \text{ | } U_{er})))$   
 $D \models_l x$   
**using** *D-false* **by** *blast*  
**moreover have**  $\forall x \in \# \ D. x \neq -L \longrightarrow \text{atm-of } x \prec_t \text{atm-of } (-L)$   
**unfolding** *L-def* *uminus-Neg* *literal.sel*  
**using**  $\langle \text{ord-res.is-strictly-maximal-lit} (\text{Pos } A_L) \ D \rangle$   
**by** (*metis* *Pos-atm-of-iff*  $\langle A_L \in \text{ord-res.production} (\text{fset} (\text{iefac } \mathcal{F} \text{ |}^q (N \text{ |} \cup \text{ | } U_{er}))) \ D \rangle$   
*ord-res.less-trm-if-neg* *ord-res.lesseq-trm-if-pos* *reflclp-iff*)  
**ultimately have**  $\forall x \in \# \ D. x \neq -L \longrightarrow \neg \text{ord-res.interp} (\text{fset} (\text{iefac } \mathcal{F} \text{ |}^q (N \text{ |} \cup \text{ | } U_{er}))) \ E \models_l x$   
**using** *ord-res.interp-fixed-for-smaller-literals*  
**using**  $\langle \text{ord-res.is-strictly-maximal-lit} (\text{Pos } A_L) \ D \rangle \langle D \prec_c E \rangle$   
**using** *L-def* **by** *fastforce*  
**thus** *?thesis*  
**unfolding** *dom-M-eq[symmetric]* .

**qed**

**moreover have**  $K \in \# \ \text{eres } D \ E$   
**using**  $\langle \text{ord-res.is-maximal-lit } K (\text{eres } D \ E) \rangle$   
**using** *linorder-lit.is-maximal-in-mset-iff* **by** *metis*

**moreover have**  $\forall x \in \# \ \text{eres } D \ E. x \in \# \ D \vee x \in \# \ E$   
**using** *lit-in-one-of-resolvents-if-in-eres* **by** *metis*

**moreover have**  $\forall x \in \# \ \text{eres } D \ E. x \neq -L$   
**proof** (*intro* *ball* *notI*)  
**fix**  $x$  **assume**  $x \in \# \ \text{eres } D \ E \ x = -L$   
**obtain**  $m \ A \ D' \ E'$  **where**  
*ord-res.is-strictly-maximal-lit* (*Pos*  $A$ )  $D$  **and**

$D = \text{add-mset } (Pos\ A)\ D'$  **and**  
 $E = \text{replicate-mset } (Suc\ m)\ (Neg\ A) + E'$  **and**  
 $Neg\ A \notin\# E'$  **and**  
 $\text{eres } D\ E = \text{repeat-mset } (Suc\ m)\ D' + E'$   
**using**  $\langle \text{eres } D\ E \neq E' [THEN\ \text{eres-not-ident}D] \text{ by } \text{metis}$

**have**  $L = Neg\ A$   
**using**  $\langle \text{ord-res.is-strictly-maximal-lit } (Pos\ A)\ D \rangle$   
**by**  $(\text{metis } L\text{-def } Uniq\text{-}D\ \langle \text{ord-res.is-strictly-maximal-lit } (Pos\ A_L)\ D \rangle$   
 $\text{linorder-lit.Uniq-is-greatest-in-mset literal.inject}(1))$

**have**  $x \in\# D' \vee x \in\# E'$   
**using**  $\langle x \in\# \text{eres } D\ E \rangle$   
**unfolding**  $\langle \text{eres } D\ E = \text{repeat-mset } (Suc\ m)\ D' + E' \rangle$   
**by**  $(\text{metis } \text{member-mset-repeat-mset-Suc union-iff})$   
**thus**  $False$   
**proof**  $(\text{elim } \text{disj}E)$   
**assume**  $x \in\# D'$   
**hence**  $Pos\ A \in\# D'$   
**unfolding**  $\langle x = -\ L \rangle \langle L = Neg\ A \rangle$  **by**  $\text{simp}$   
**hence**  $\neg \text{ord-res.is-strictly-maximal-lit } (Pos\ A)\ D$   
**using**  $\langle D = \text{add-mset } (Pos\ A)\ D' \rangle$   
**using**  $\text{linorder-lit.is-greatest-in-mset-iff}$  **by**  $\text{auto}$   
**thus**  $False$   
**using**  $\langle \text{ord-res.is-strictly-maximal-lit } (Pos\ A)\ D \rangle$  **by**  $\text{contradiction}$

**next**  
**assume**  $x \in\# E'$   
**hence**  $Pos\ A \in\# E'$   
**unfolding**  $\langle x = -\ L \rangle \langle L = Neg\ A \rangle$  **by**  $\text{simp}$   
**hence**  $Pos\ A \in\# E$   
**unfolding**  $\langle E = \text{replicate-mset } (Suc\ m)\ (Neg\ A) + E' \rangle$  **by**  $\text{simp}$   
**hence**  $\text{ord-res.production } (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ D \models_l Pos\ A$   
**using**  $L\text{-def } \langle A_L \in \text{ord-res.production } (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ D \rangle$   
 $\langle L = Neg\ A \rangle$   
**by**  $\text{blast}$   
**hence**  $\text{ord-res.interp } (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ E \models_l Pos\ A$   
**by**  $(\text{metis } \langle L = Neg\ A \rangle\ \text{dom-}\mathcal{M}\text{-eq linorder-lit.is-maximal-in-mset-iff}$   
 $\text{neg-literal-notin-imp-true-cls step-hyps}(3)\ \text{step-hyps}(4)\ \text{true-lit-simps}(1))$   
**hence**  $\text{ord-res.interp } (fset\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup|\ U_{er})))\ E \models E$   
**using**  $\langle Pos\ A \in\# E \rangle$  **by**  $\text{blast}$   
**hence**  $\text{dom } \mathcal{M} \models E$   
**using**  $\text{dom-}\mathcal{M}\text{-eq}$  **by**  $\text{argo}$   
**thus**  $False$   
**using**  $\langle \neg \text{dom } \mathcal{M} \models E \rangle$  **by**  $\text{contradiction}$

**qed**  
**qed**

**ultimately have**  $\neg \text{dom } \mathcal{M} \models_l K$   
**by**  $\text{metis}$

```

have  $\text{dom } \mathcal{M}' = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')))$   $C$ 
proof (intro subset-antisym subsetI)
  fix  $A :: 'f \text{ gterm}$ 
  assume  $A \in \text{dom } \mathcal{M}'$ 
  hence  $A \in \text{dom } \mathcal{M}$  and  $A \prec_t \text{ atm-of } K$ 
    unfolding  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{ atm-of } K\} \rangle$  by simp-all

  have  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $E$ 
    using  $\langle A \in \text{dom } \mathcal{M} \rangle$ 
    unfolding  $\langle \text{dom } \mathcal{M} = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $E \rangle$  .
  hence  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $C$ 
    using ord-res.interp-fixed-for-smaller-literals  $\langle \text{ord-res.is-maximal-lit } K$  (eres
   $D E) \rangle$ 
     $\langle A \prec_t \text{ atm-of } K \rangle$   $\langle C \prec_c E \rangle$ 
    by (smt (verit, del-insts) K-def
       $\langle A_K \in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $C \rangle$   $\langle \text{eres } D E$ 
 $\prec_c E \rangle$ 
      literal.sel(2) ord-res.lesser-atoms-in-previous-interp-are-in-final-interp
ord-res.lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-productive)
    thus  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')))$   $C$ 
    unfolding  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup$ 
 $U_{er})) \rangle$ 
    using ord-res.interp-insert-greater-clause-strong
    by (simp add:  $\langle C \prec_c \text{eres } D E \rangle$ )
  next
  fix  $A :: 'f \text{ gterm}$ 
  assume  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')))$   $C$ 
  hence  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $C$ 
    unfolding  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup$ 
 $U_{er})) \rangle$ 
    using ord-res.interp-insert-greater-clause-strong by (simp add:  $\langle C \prec_c \text{eres}$ 
 $D E) \rangle$ 
  hence  $A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $E$ 
    using  $\langle C \prec_c E \rangle$  ord-res.interp-subset-if-less-cls by blast
  hence  $A \in \text{dom } \mathcal{M}$ 
    unfolding  $\langle \text{dom } \mathcal{M} = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $E \rangle$  .

moreover have  $A \prec_t \text{ atm-of } K$ 
proof –
  obtain  $B$  where
     $B \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  and
     $B \prec_c C$  and
    A-prod-by-B:  $A \in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $B$ 
    using  $\langle A \in \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   $C \rangle$ 
    unfolding ord-res.interp-def by blast

  have ord-res.is-maximal-lit (Pos  $A$ )  $B$ 
    using A-prod-by-B ord-res.mem-productionE by blast

```

**hence**  $A \preceq_t \text{atm-of } K$   
**using**  $\langle \text{ord-res.is-maximal-lit } K \text{ (eres } D \ E) \rangle \langle C \prec_c \text{eres } D \ E \rangle$   
**by**  $(\text{metis } K\text{-def } \langle B \prec_c C \rangle \text{ asymp} D$   
 $\text{linorder-cls.less-trans linorder-lit.multip-if-maximal-less-that-maximal}$   
 $\text{linorder-trm.le-less-linear literal.sel}(2)$   
 $\text{ord-res.asymp-less-cls ord-res.less-lit-simps}(4))$

**moreover have**  $A \neq \text{atm-of } K$   
**using**  $\langle A \in \text{dom } \mathcal{M} \rangle \langle \neg \text{dom } \mathcal{M} \models l \ K \rangle$   
**unfolding**  $K\text{-def}$   
**by**  $(\text{metis } \langle A \in \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C \rangle$   
 $\langle A_K \in \text{ord-res.production (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C \rangle \text{atm-of-uminus}$   
 $\text{linorder-lit.is-greatest-in-mset-iff literal.sel}(1) \text{ord-res.mem-production} E$   
 $\text{pos-literal-in-imp-true-cls uminus-Neg})$

**ultimately show**  $?thesis$   
**by**  $order$

qed

**ultimately show**  $A \in \text{dom } \mathcal{M}'$   
**unfolding**  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$  **by**  $\text{simp}$   
 qed

**thus**  $\text{model-eq-interp-upto-next-clause } N \ (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$   
**unfolding**  $\text{model-eq-interp-upto-next-clause-def}$  **by**  $\text{simp}$

**have**  $\forall x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). x \prec_c E \longrightarrow$   
 $\text{ord-res-Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ x \models x$   
**using**  $\text{invars}$   
**unfolding**  $\text{step-hyps}(1,2) \text{ord-res-5-invars-def all-smaller-clauses-true-wrt-respective-Interp-def}$   
**by**  $\text{simp}$

**hence**  $\forall x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). x \prec_c C \longrightarrow$   
 $\text{ord-res-Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ x \models x$   
**using**  $\langle C \prec_c E \rangle$  **by**  $\text{simp}$

**moreover have**  $\forall x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). x \prec_c C \longrightarrow$   
 $\text{ord-res-Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ x =$   
 $\text{ord-res-Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')) \ x$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert (eres } D \ E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid$   
 $U_{er})) \rangle$   
**by**  $(\text{metis (no-types, lifting) } \langle C \prec_c \text{eres } D \ E \rangle \text{finite-fset finsert.rep-eq}$   
 $\text{linorder-cls.dual-order.strict-trans2 ord-res.Interp-insert-greater-clause}$   
 $\text{sup2CI})$

**ultimately have**  $\forall x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). x \prec_c C \longrightarrow$   
 $\text{ord-res-Interp (iefac } \mathcal{F} \mid \uparrow (\text{fset } N \cup \text{fset } U_{er}')) \ x \models x$   
**by**  $\text{simp}$

**hence**  $\forall x \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') . x \prec_c C \longrightarrow$   
 $\text{ord-res-Interp } (\text{iefac } \mathcal{F} \mid \uparrow (fset N \cup fset U_{er}')) x \models x$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert } (eres D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \rangle$   
**using**  $\langle C \prec_c eres D E \rangle$  **by** *auto*

**thus** *all-smaller-clauses-true-wrt-respective-Interp*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$   
**unfolding** *all-smaller-clauses-true-wrt-respective-Interp-def*  
**by** *simp*

**have**  $A \in \text{ord-res.production } (fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')) C$   
**if**  $\mathcal{M}' A = \text{Some } C$  **for**  $A C$   
**proof** –  
**have**  $\mathcal{M} A = \text{Some } C$  **and**  $A \prec_t \text{atm-of } K$   
**unfolding** *atomize-conj*  
**using**  $\langle \mathcal{M}' A = \text{Some } C \rangle \langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$   
**by** (*metis Int-iff domI dom-restrict mem-Collect-eq restrict-in*)

**hence** *A-prod-by-C*:  $A \in \text{ord-res.production } (fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C$   
**using** *invars*  
**unfolding** *step-hyps(1,2) ord-res-5-invars-def atoms-in-model-were-produced-def*  
**by** *metis*

**moreover have**  $\text{ord-res.production } (fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C =$   
 $\text{ord-res.production } (fset (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')) C$   
**proof** (*rule ord-res.production-swap-clause-set*)  
**have** *ord-res.is-strictly-maximal-lit*  $(Pos A) C$   
**using** *A-prod-by-C ord-res.mem-productionE* **by** *metis*  
**moreover have**  $Pos A \prec_t K$   
**using**  $\langle A \prec_t \text{atm-of } K \rangle$   
**by** (*simp add: K-def*)  
**ultimately have**  $C \prec_c eres D E$   
**using** *linorder-lit.mulp-if-maximal-less-that-maximal step-hyps(10)* **by**

*blast*

**thus**  $\{D. D \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \wedge (\prec_c)^{==} D C\} =$   
 $\{D. D \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') \wedge (\prec_c)^{==} D C\}$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert } (eres D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \rangle$   
**by** *auto*  
**qed**

**ultimately show** *?thesis*  
**by** *argo*  
**qed**

**thus** *atoms-in-model-were-produced*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$   
**unfolding** *atoms-in-model-were-produced-def* **by** *simp*

```

have  $\mathcal{M}' A = \text{Some } B$ 
  if  $B \prec_c C$  and  $A\text{-in}: A \in \text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')))$ 
B
  for  $B A$ 
  proof –
    have  $B \prec_c \text{eres } D E$ 
      using  $\langle B \prec_c C \rangle \langle C \prec_c \text{eres } D E \rangle$  by order
    hence  $B \prec_c E$ 
      using  $\langle \text{eres } D E \prec_c E \rangle$  by order

  moreover have  $A\text{-prod-by-}B: A \in \text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}))) B$ 
  proof –
    have  $\text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}))) B =$ 
       $\text{ord-res.production } (fset (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}')) B$ 
    proof (rule ord-res.production-swap-clause-set)
      show  $\{D. D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}) \wedge (\prec_c)^{==} D B\} =$ 
         $\{D. D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') \wedge (\prec_c)^{==} D B\}$ 
      unfolding  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup U_{er})) \rangle$ 
        using  $\langle B \prec_c \text{eres } D E \rangle$ 
        by (metis (no-types, opaque-lifting) finsert-iff linorder-cls.less-le-not-le)
    qed

  thus ?thesis
    using  $A\text{-in}$  by argo
  qed

ultimately have  $\mathcal{M} A = \text{Some } B$ 
  using invars
  unfolding step-hyps(1,2) ord-res-5-invars-def all-produced-atoms-in-model-def
  by metis

moreover have  $A \prec_t \text{atm-of } K$ 
  proof –
    have  $A \in \text{dom } \mathcal{M}$ 
      using  $\langle \mathcal{M} A = \text{Some } B \rangle$  by auto

    have ord-res.is-maximal-lit (Pos  $A$ )  $B$ 
      using  $A\text{-prod-by-}B$  ord-res.mem-productionE by blast

    hence  $A \preceq_t \text{atm-of } K$ 
      using  $\langle \text{ord-res.is-maximal-lit } K (\text{eres } D E) \rangle \langle B \prec_c \text{eres } D E \rangle$ 
      by (metis K-def asympD linorder-lit.mulp-if-maximal-less-that-maximal
        linorder-trm.le-less-linear literal.sel(2) ord-res.asymp-less-cls
        ord-res.less-lit-simps(4))

  moreover have  $A \neq \text{atm-of } K$ 
    using  $\langle A \in \text{dom } \mathcal{M} \rangle \langle \neg \text{dom } \mathcal{M} \models l K \rangle$ 

```

**using**  $\langle \mathcal{M} A = \text{Some } B \rangle$  *step-hyps(12)* *that(1)* **by force**

**ultimately show** *?thesis*  
**by order**  
**qed**

**ultimately show**  $\mathcal{M}' A = \text{Some } B$   
**unfolding**  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$  **by simp**  
**qed**

**thus** *all-produced-atoms-in-model*  $N (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$   
**unfolding** *all-produced-atoms-in-model-def* **by simp**  
**qed**  
**qed**

**lemma** *rtranclp-ord-res-6-preserves-invars*:  
**assumes** *steps*:  $(\text{ord-res-6 } N)^{**} s s'$  **and** *invars*: *ord-res-5-invars*  $N s$   
**shows** *ord-res-5-invars*  $N s'$   
**using** *steps invars*  
**by** (*induction s rule: converse-rtranclp-induct*) (*auto intro: ord-res-6-preserves-invars*)

**lemma** *ex-ord-res-6-if-not-final*:  
**assumes**  
*not-final*:  $\neg \text{ord-res-6-final } S$  **and**  
*invars*:  $\forall N s. S = (N, s) \longrightarrow \text{ord-res-5-invars } N s$   
**shows**  $\exists S'. \text{ord-res-6-step } S S'$   
**proof** –  
**from** *not-final* **obtain**  $N U_{er} \mathcal{F} \mathcal{M} C$  **where**  
*S-def*:  $S = (N, (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C))$  **and**  $C \neq \{\#\}$   
**unfolding** *ord-res-6-final.simps de-Morgan-disj not-ex*  
**by** (*metis option.exhaust surj-pair*)

**note**  $\text{invars}' = \text{invars}[\text{unfolded } \text{ord-res-5-invars-def}, \text{rule-format}, \text{OF } S\text{-def}]$

**have**  $\exists s'. \text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) s'$   
**proof** (*cases dom*  $\mathcal{M} \models C$ )  
**case** *True*  
**thus** *?thesis*  
**using** *ord-res-6.skip* **by metis**

**next**  
**case** *C-false*: *False*  
**obtain**  $L$  **where** *L-max*: *linorder-lit.is-maximal-in-mset*  $C L$   
**using** *linorder-lit.ex-maximal-in-mset*[*OF*  $\langle C \neq \{\#\} \rangle$ ] **..**

**show** *?thesis*  
**proof** (*cases*  $L$ )  
**case** (*Pos*  $A$ )  
**hence** *L-pos*: *is-pos*  $L$   
**by simp**



```

show ?thesis
proof (cases ord-res.is-strictly-maximal-lit L C)
  case True
  then show ?thesis
    using ord-res-6.production[OF C-false L-max L-pos] by metis
  next
  case L-not-strictly-max: False
  thus ?thesis
    using ord-res-6.factoring[OF C-false L-max L-pos L-not-strictly-max refl]
by metis
qed
next
case (Neg A)
hence L-neg: is-neg L
  by simp

have C-least-false: is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )) C
  unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
  show C | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )
    using invars' by (metis next-clause-in-factorized-clause-def)
  next
  have  $\neg$  ord-res.interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C  $\models$  C
    using invars' C-false by (metis model-eq-interp-upto-next-clause-def)
  moreover have ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C = {}
  proof -
    have  $\nexists L. is-pos L \wedge ord-res.is-strictly-maximal-lit L C$ 
      using L-max L-neg
      by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset
        linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset)
    thus ?thesis
      using unproductive-if-nex-strictly-maximal-pos-lit by metis
  qed
  ultimately show  $\neg$  ord-res.Interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) C  $\models$  C
    by simp
  next
  fix D
  assume D-in: D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) and D  $\neq$  C and
    C-false:  $\neg$  ord-res.Interp (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) D  $\models$  D
  have  $\neg D \prec_c C$ 
    using C-false
    using invars' D-in
    unfolding all-smaller-clauses-true-wrt-respective-Interp-def
    by auto
  thus C  $\prec_c$  D
    using  $\langle D \neq C \rangle$  by order
  qed
then obtain D where D| $\in$ |iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) and
  D  $\prec_c$  C and

```

```

ord-res.is-strictly-maximal-lit (Pos A) D and
D-prod: ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) D = {A}
using bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit
  L-max[unfolded Neg] by metis

hence  $\exists DC$ . ground-resolution D C DC
unfolding ground-resolution-def
using L-max Neg ex-ground-resolutionI by blast

hence eres D C  $\neq$  C
unfolding eres-ident-iff by metis

hence eres D C  $\prec_c$  C
using eres-le[of D C] by order

have  $\mathcal{M}$  (atm-of L) = Some D
using invars'
by (metis Neg  $\langle D \prec_c C \rangle$  all-produced-atoms-in-model-def D-prod insertII
literal.sel(2))

show ?thesis
proof (cases eres D C = {#})
case True
then show ?thesis
using ord-res-6.resolution-bot[OF C-false L-max L-neg  $\langle \mathcal{M}$  (atm-of L) =
Some D  $\rangle$ ] by metis
next
case False
then obtain K where K-max: ord-res.is-maximal-lit K (eres D C)
using linorder-lit.ex-maximal-in-mset by metis
show ?thesis
proof (cases K)
case K-def: (Pos AK)
hence is-pos K
by simp
thus ?thesis
using ord-res-6.resolution-pos
using C-false L-max L-neg  $\langle \mathcal{M}$  (atm-of L) = Some D  $\rangle$   $\langle$ eres D C  $\neq$  {# $\rangle$ 
K-max by metis
next
case K-def: (Neg AK)
hence K-neg: is-neg K
by simp

have  $\neg$  ord-res-Interp (fset (finsert (eres D C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))))
(eres D C)  $\models$  eres D C
by (smt (verit) C-least-false D-prod Interp-insert-unproductive K-max
K-neg Uniq-D
 $\langle$ eres D C  $\neq$  C $\rangle$  clause-true-if-resolved-true ex1-eres-eq-full-run-ground-resolution)

```

*finite-fset fset-simps(2) full-run-def insert-not-empty is-least-false-clause-def  
linorder-cls.is-least-in-fset-filterD(2) linorder-lit.Uniq-is-maximal-in-mset  
linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset rtranclpD  
unproductive-if-nex-strictly-maximal-pos-lit)*

**hence** *eres-least:*

*is-least-false-clause (finsert (eres D C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) (eres*  
*D C)*

**using** *C-least-false*  $\langle$ eres D C  $\prec_c$  C $\rangle$   
**by** (*metis is-least-false-clause-finsert-smaller-false-clause*)

**then obtain** *E* where *E* | $\in$ | *finsert (eres D C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))*

**and**

*E*  $\prec_c$  *eres D C* **and**  
*ord-res.is-strictly-maximal-lit (Pos  $A_K$ ) E* **and**

*E-prod: ord-res.production (fset (finsert (eres D C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  
 $U_{er}$ )))) E = { $A_K$ }*

**using** *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*  
*K-max K-def*  
**by** *metis*

**have** *ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) E =*  
*ord-res.production (fset (finsert (eres D C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )))) E*  
**proof** (*rule ord-res.production-swap-clause-set*)

**have** *eres D C* | $\notin$ | *iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )*

**proof** (*rule notI*)

**assume** *eres D C* | $\in$ | *iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )*

*hence finsert (eres D C) (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )) = iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  
 $U_{er}$ )*

**by** *blast*

**hence** *is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )) (eres D C)*

**using** *eres-least by argo*

**thus** *False*

**using** *C-least-false*  $\langle$ eres D C  $\neq$  C $\rangle$

**by** (*metis Uniq-D Uniq-is-least-false-clause*)

**qed**

**thus**  $\{D. D | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \wedge (\prec_c)^{==} D E\} =$

$\{Da. Da | \in | \text{finsert (eres D C) (iefac } \mathcal{F} | \uparrow (N | \cup | U_{er})) \wedge (\prec_c)^{==} Da$   
*E}*

**by** (*metis (mono-tags, lifting)  $\langle$ E  $\prec_c$  eres D C $\rangle$  finsert-iff linorder-cls.leD*)

**qed**

**also have**  $\dots = \{A_K\}$

**using** *E-prod* .

**finally have** *ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ))) E = { $A_K$ }*

.

```

    hence  $\mathcal{M}$  (atm-of  $K$ ) = Some  $E$ 
      using invars'
    unfolding ord-res-5-invars-def all-produced-atoms-in-model-def
  by (metis K-def  $\langle E \prec_c \text{eres } D \ C \rangle$  eres-le insertI1 linorder-cls.dual-order.strict-trans1
      literal.sel(2))

    thus ?thesis
      using ord-res-6.resolution-neg
      using C-false L-max L-neg  $\langle \mathcal{M}$  (atm-of  $L$ ) = Some  $D \rangle$   $\langle \text{eres } D \ C \neq \{\#\} \rangle$ 
K-max K-neg by metis
    qed
  qed
  qed
  qed
  thus ?thesis
    using S-def ord-res-6-step.simps by metis
qed

lemma ord-res-6-safe-state-if-invars:
  safe-state ord-res-6-step ord-res-6-final (N, s) if invars: ord-res-5-invars N s for
  N s
proof -
  {
    fix S'
    assume ord-res-6-semantics.eval (N, s) S' and stuck: stuck-state ord-res-6-step
  S'
    then obtain s' where S' = (N, s') and (ord-res-6 N)** s s'
    proof (induction (N, s) arbitrary: N s rule: converse-rtranclp-induct)
      case base
        thus ?case by simp
      next
        case (step z)
          thus ?case
            by (smt (verit, ccfv-SIG) converse-rtranclp-into-rtranclp ord-res-6-step.cases
prod.inject)
    qed
    hence ord-res-5-invars N s'
      using invars rtranclp-ord-res-6-preserves-invars by metis
    hence  $\neg$  ord-res-6-final S'  $\implies \exists S''$ . ord-res-6-step S' S''
      using ex-ord-res-6-if-not-final[of S']  $\langle S' = (N, s') \rangle$  by blast
    hence ord-res-6-final S'
      using stuck[unfolded stuck-state-def] by argo
  }
  thus ?thesis
    unfolding safe-state-def stuck-state-def by metis
qed

lemma ex-model-build-from-least-clause-to-any-less-than-least-false:
  assumes

```

$\mathcal{F}$ -subset:  $\mathcal{F} \mid\subseteq\mid N \mid\cup\mid U_{er}$  **and**  
*C-least*: *linorder-cls.is-least-in-fset* (*iefac*  $\mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$ ) *C* **and**  
*D-in*:  $D \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$  **and**  
*D-lt-least-false*:  $\forall E. \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})) E \longrightarrow D \preceq_c$   
*E* **and**  
 $C \preceq_c D$   
**shows**  $\exists \mathcal{M}. (\text{ord-res-5 } N)^{**} (U_{er}, \mathcal{F}, \text{Map.empty}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D)$   
**using** *ord-res.wfP-less-cls D-in*  $\langle C \preceq_c D \rangle$  *D-lt-least-false*  
**proof** (*induction D rule: wfp-induct-rule*)  
**case** (*less D*)

**have** *invars: ord-res-5-invars N* ( $U_{er}, \mathcal{F}, \text{Map.empty}, \text{Some } C$ )  
**using** *ord-res-5-invars-initial-state*  $\mathcal{F}$ -subset *C-least* **by** *metis*

**define** *clauses-lt-D* :: '*f gclause fset where*  
*clauses-lt-D* =  $\{|C \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er}). C \prec_c D|\}$

**show** *?case*  
**proof** (*cases clauses-lt-D* =  $\{|\}\}$ )  
**case** *True*  
**hence**  $C = D$   
**unfolding** *clauses-lt-D-def*  
**using** *C-least*  $\langle C \preceq_c D \rangle$   
**by** (*metis fempty-iff fmember-filter linorder-cls.antisym-conv3*  
*linorder-cls.is-least-in-fset-iff linorder-cls.less-le-not-le*)  
**thus** *?thesis*  
**by** *blast*

**next**  
**case** *False*

**obtain** *x where x-greatest: linorder-cls.is-greatest-in-fset clauses-lt-D x*  
**using** *False linorder-cls.ex-greatest-in-fset* **by** *metis*

**have**  $x \prec_c D$  **and**  $x \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$   
**using** *x-greatest* **by** (*simp-all add: clauses-lt-D-def linorder-cls.is-greatest-in-fset-iff*)

**moreover have**  $C \preceq_c x$   
**using** *x-greatest C-least*  
**by** (*metis clauses-lt-D-def fmember-filter linorder-cls.is-greatest-in-fset-iff*  
*linorder-cls.not-less nbex-less-than-least-in-fset*)

**moreover have**  $\bigwedge E. \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})) E \implies x \prec_c$   
*E*

**using**  $\langle x \prec_c D \rangle$  *less.premis* **by** *force*

**ultimately obtain**  $\mathcal{M}$  **where**  
*IH: (ord-res-5 N)\*\** ( $U_{er}, \mathcal{F}, \text{Map.empty}, \text{Some } C$ ) ( $U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } x$ )  
**using** *less.IH* **by** *blast*

**moreover have**  $\exists \mathcal{M}'. \text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } x) (U_{er}, \mathcal{F}, \mathcal{M}', \text{Some } D)$

**proof** –

**have**  $\text{linorder-cls.is-least-in-fset } (\text{ffilter } ((\prec_c) x) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D$   
**using**  $x\text{-greatest}[\text{unfolded clauses-lt-}D\text{-def}]$   
**by**  $(\text{smt } (\text{verit}) \text{ffmember-filter less.prem } (I) \text{linorder-cls.is-greatest-in-fset-iff } \text{linorder-cls.is-least-in-ffilter-iff } \text{linorder-cls.not-less-iff-gr-or-eq})$   
**hence**  $\text{next-clause-eq: The-optional } (\text{linorder-cls.is-least-in-fset } (\text{ffilter } ((\prec_c) x) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))) = \text{Some } D$   
**by**  $(\text{metis } \text{linorder-cls.Uniq-is-least-in-fset The-optional-eq-SomeI})$

**have**  $x\text{-true: ord-res-Interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) x \models x$   
**using**  $\text{less.prem}$   
**unfolding**  $\text{is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff}$

**by**  $(\text{metis } \langle \bigwedge E. \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) E \implies x \prec_c E \rangle \langle x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \rangle \text{ex-false-clause-iff finsert-absorb is-least-false-clause-finsert-smaller-false-clause linorder-cls.order.irrefl ex-false-clause-def})$

**show**  $?thesis$

**proof**  $(\text{cases } \text{dom } \mathcal{M} \models x)$   
**case**  $\text{True}$   
**thus**  $?thesis$   
**using**  $\text{ord-res-5.skip next-clause-eq by metis}$

**next**  
**case**  $\text{False}$   
**hence**  $\neg \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) x \models x$   
**using**  $\text{rtranclp-ord-res-5-preserves-invars}[OF IH invars, \text{unfolded ord-res-5-invars-def, simplified}]$   
**by**  $(\text{simp add: model-eq-interp-upto-next-clause-def})$

**thus**  $?thesis$   
**using**  $\text{ord-res-5.production}[OF \text{False}] \text{next-clause-eq}$   
**using**  $x\text{-true}$   
**by**  $(\text{metis } \text{Un-empty-right linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset unproductive-if-nex-strictly-maximal-pos-lit})$

**qed**  
**qed**

**ultimately show**  $?thesis$   
**by**  $(\text{smt } (\text{verit}) \text{rtranclp.rtrancl-into-rtrancl})$

**qed**  
**qed**

**lemma**  $\text{full-rtranclp-ord-res-5-run-upto:}$   
**assumes**  
 $\text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E) (U_{er}', \mathcal{F}', \mathcal{M}', \text{Some } D)$  **and**

*invars*: *ord-res-5-invars*  $N (U_{er'}, \mathcal{F}', \mathcal{M}', \text{Some } D)$  **and**  
*M'-def*:  $\mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. \exists K. \text{linorder-lit.is-maximal-in-mset } D K \wedge$   
 $A \prec_t \text{atm-of } K\}$  **and**  
*C-least*: *linorder-cls.is-least-in-fset* (*iefac*  $\mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er'})$ )  $C$   
**shows** (*ord-res-5 N*)<sup>\*\*</sup> ( $U_{er'}, \mathcal{F}', \text{Map.empty}, \text{Some } C$ ) ( $U_{er'}, \mathcal{F}', \mathcal{M}', \text{Some } D$ )  
**proof** –  
**have** *D-in*:  $D \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er'})$   
**using** *invars*  
**by** (*metis next-clause-in-factorized-clause-def ord-res-5-invars-def*)  
  
**have**  $\mathcal{F}' \mid \subseteq \mid N \mid \cup \mid U_{er}'$   
**using** *invars*  
**by** (*metis implicitly-factorized-clauses-subset-def ord-res-5-invars-def*)  
  
**moreover have**  $C \preceq_c D$   
**using** *C-least D-in*  
**by** (*metis linorder-cls.dual-order.strict-iff-order linorder-cls.is-least-in-fset-iff*  
*linorder-cls.le-cases*)  
  
**moreover have**  $\forall F. \text{is-least-false-clause } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er'})) F \longrightarrow D \preceq_c$   
 $F$   
**using** *invars le-least-false-clause by metis*  
  
**ultimately obtain**  $\mathcal{M}''$  **where**  
*steps*: (*ord-res-5 N*)<sup>\*\*</sup> ( $U_{er'}, \mathcal{F}', \text{Map.empty}, \text{Some } C$ ) ( $U_{er'}, \mathcal{F}', \mathcal{M}'', \text{Some } D$ )  
**using** *C-least D-in*  
**by** (*metis ex-model-build-from-least-clause-to-any-less-than-least-false*)  
  
**have** *ord-res-5-invars*  $N (U_{er'}, \mathcal{F}', \text{Map.empty}, \text{Some } C)$   
**using**  $\langle \mathcal{F}' \mid \subseteq \mid N \mid \cup \mid U_{er}' \rangle$  *C-least ord-res-5-invars-initial-state* **by** *metis*  
  
**hence** *ord-res-5-invars*  $N (U_{er'}, \mathcal{F}', \mathcal{M}'', \text{Some } D)$   
**using**  $\langle (\text{ord-res-5 } N)^{**} (U_{er'}, \mathcal{F}', \lambda x. \text{None}, \text{Some } C) (U_{er'}, \mathcal{F}', \mathcal{M}'', \text{Some } D) \rangle$   
*rtranclp-ord-res-5-preserves-invars* **by** *metis*  
  
**hence**  $\mathcal{M}''$ -*spec*:  
 $\text{dom } \mathcal{M}'' = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er'}))) D$   
 $\forall A C. \mathcal{M}'' A = \text{Some } C \longrightarrow A \in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er'}))) C$   
 $\forall C A. C \prec_c D \longrightarrow A \in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er'}))) C$   
 $\longrightarrow \mathcal{M}'' A = \text{Some } C$   
**unfolding** *ord-res-5-invars-def*  
**unfolding** *model-eq-interp-upto-next-clause-def atoms-in-model-were-produced-def*  
*all-produced-atoms-in-model-def*  
**by** *metis+*  
  
**have**  $\mathcal{M}' = \mathcal{M}''$

```

proof (cases D = {#})
  case True
  have M' = Map.empty
  proof -
    have  $\nexists K. \text{ord-res.is-maximal-lit } K D \wedge A \prec_t \text{atm-of } K$  for A
      unfolding ⟨D = {#}⟩
      by (simp add: linorder-lit.is-maximal-in-mset-iff)
    hence {A.  $\exists K. \text{ord-res.is-maximal-lit } K D \wedge A \prec_t \text{atm-of } K$ } = {}
      by simp
    thus ?thesis
      unfolding M'-def by auto
  qed

also have Map.empty = M''
proof -
  have ord-res.interp (fset (iefac F' |' (N | $\cup$ | Uer')))) D = {}
    unfolding ⟨D = {#}⟩ by simp
  thus ?thesis
    using M''-spec(1) by simp
qed

finally show ?thesis .
next
  case False
  then obtain K where ord-res.is-maximal-lit K D
    using linorder-lit.ex-maximal-in-mset by metis
  hence {A.  $\exists K. \text{ord-res.is-maximal-lit } K D \wedge A \prec_t \text{atm-of } K$ } = {A. A  $\prec_t$ 
atm-of K}
  by (metis (no-types, lifting) linorder-lit.Uniq-is-maximal-in-mset the1-equality')
  hence M'-def': M' = restrict-map M {A. A  $\prec_t$  atm-of K}
    unfolding M'-def by argo

show ?thesis
proof (intro ext)
  fix x
  have M'-spec:
    dom M' = ord-res.interp (fset (iefac F' |' (N | $\cup$ | Uer')))) D
     $\forall A C. M' A = \text{Some } C \longrightarrow A \in \text{ord-res.production (fset (iefac F' |' (N$ 
| $\cup$ | Uer')))) C
     $\forall C A. C \prec_c D \longrightarrow A \in \text{ord-res.production (fset (iefac F' |' (N | $\cup$ | Uer'))))$ 
C  $\longrightarrow M' A = \text{Some } C$ 
    using invars
    unfolding ord-res-5-invars-def
  unfolding model-eq-interp-upto-next-clause-def atoms-in-model-were-produced-def
    all-produced-atoms-in-model-def
  by metis+

have dom M' = dom M''
  using M'-spec(1) M''-spec(1) by argo

```



```

moreover have  $\forall A C. \mathcal{M}' A = \text{Some } C \longleftrightarrow \mathcal{M}'' A = \text{Some } C$ 
using  $\mathcal{M}'\text{-spec}(2)$   $\mathcal{M}''\text{-spec}(2)$ 
by (smt (verit, del-insts) calculation domD domI linorder-clc.less-irrefl
linorder-clc.neqE
ord-res.less-trm-iff-less-clc-if-mem-production)

ultimately show  $\mathcal{M}' x = \mathcal{M}'' x$ 
by (metis domD domIff)
qed
qed

thus ?thesis
using  $\langle(\text{ord-res-5 } N)^{**} (U_{er}', \mathcal{F}', \text{Map.empty}, \text{Some } C) (U_{er}', \mathcal{F}', \mathcal{M}'', \text{Some } D)\rangle$  by argo
qed

end

end

theory ORD-RES-7
imports
  Background
  Implicit-Exhaustive-Factorization
  Exhaustive-Resolution
begin

```

## 22 ORD-RES-7 (clause-guided literal trail construction)

```

context simulation-SCLFOL-ground-ordered-resolution begin

```

```

inductive ord-res-7 where

```

```

  decide-neg:

```

```

     $\neg \text{trail-false-clc } \Gamma C \implies$ 
     $\text{linorder-lit.is-maximal-in-mset } C L \implies$ 
     $\text{linorder-trm.is-least-in-fset } \{|A| \in | \text{atms-of-clss } (N \cup U_{er})\}.$ 
     $A \prec_t \text{atm-of } L \wedge A \notin \text{trail-atms } \Gamma \implies$ 
     $\Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \implies$ 
     $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}, \Gamma', \text{Some } C) |$ 

```

```

  skip-defined:

```

```

     $\neg \text{trail-false-clc } \Gamma C \implies$ 
     $\text{linorder-lit.is-maximal-in-mset } C L \implies$ 
     $\neg(\exists A \in | \text{atms-of-clss } (N \cup U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \text{trail-atms } \Gamma) \implies$ 
     $\text{trail-defined-lit } \Gamma L \implies$ 
     $C' = \text{The-optional } (\text{linorder-clc.is-least-in-fset } \{|D| \in | \text{iefac } \mathcal{F} \} \uparrow (N \cup U_{er}).$ 
     $C \prec_c D \implies$ 

```

*ord-res-7*  $N (U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}, \Gamma, C') \mid$

*skip-undefined-neg:*

$\neg \text{trail-false-cls } \Gamma C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\neg(\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \mid \text{trail-atms } \Gamma) \implies$   
 $\neg \text{trail-defined-lit } \Gamma L \implies$   
 $\text{is-neg } L \implies$   
 $\Gamma' = (L, \text{None}) \# \Gamma \implies$   
 $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } \{|D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$   
 $C \prec_c D|\}) \implies$   
 $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}, \Gamma', C') \mid$

*skip-undefined-pos:*

$\neg \text{trail-false-cls } \Gamma C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\neg(\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \mid \text{trail-atms } \Gamma) \implies$   
 $\neg \text{trail-defined-lit } \Gamma L \implies$   
 $\text{is-pos } L \implies$   
 $\neg \text{trail-false-cls } \Gamma \{\#K \in \# C. K \neq L\# \} \implies$   
 $\text{linorder-cls.is-least-in-fset } \{|D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D|\} D \implies$   
 $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \mid$

*skip-undefined-pos-ultimate:*

$\neg \text{trail-false-cls } \Gamma C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\neg(\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \mid \text{trail-atms } \Gamma) \implies$   
 $\neg \text{trail-defined-lit } \Gamma L \implies$   
 $\text{is-pos } L \implies$   
 $\neg \text{trail-false-cls } \Gamma \{\#K \in \# C. K \neq L\# \} \implies$   
 $\Gamma' = (-L, \text{None}) \# \Gamma \implies$   
 $\neg(\exists D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D) \implies$   
 $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}, \Gamma', \text{None}) \mid$

*production:*

$\neg \text{trail-false-cls } \Gamma C \implies$   
 $\text{linorder-lit.is-maximal-in-mset } C L \implies$   
 $\neg(\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \mid \text{trail-atms } \Gamma) \implies$   
 $\neg \text{trail-defined-lit } \Gamma L \implies$   
 $\text{is-pos } L \implies$   
 $\text{trail-false-cls } \Gamma \{\#K \in \# C. K \neq L\# \} \implies$   
 $\text{linorder-lit.is-greatest-in-mset } C L \implies$   
 $\Gamma' = (L, \text{Some } C) \# \Gamma \implies$   
 $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset } \{|D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$   
 $C \prec_c D|\}) \implies$   
 $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}, \Gamma', C') \mid$

*factoring:*

$\neg \text{trail-false-cls } \Gamma C \implies$

$linorder-lit.is-maximal-in-mset\ C\ L \implies$   
 $\neg (\exists A\ |\in|\ atm\text{-of-clss}\ (N\ |\cup|\ U_{er}).\ A\ \prec_t\ atm\text{-of}\ L \wedge A\ |\notin|\ trail\text{-atms}\ \Gamma) \implies$   
 $\neg\ trail\text{-defined-lit}\ \Gamma\ L \implies$   
 $is\text{-pos}\ L \implies$   
 $trail\text{-false-cl}\ \Gamma\ \{\#K \in\# C.\ K \neq L\#\} \implies$   
 $\neg\ linorder-lit.is-greatest-in-mset\ C\ L \implies$   
 $\mathcal{F}' = finsert\ C\ \mathcal{F} \implies$   
 $ord\text{-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ Some\ C)\ (U_{er},\ \mathcal{F}',\ \Gamma,\ Some\ (efac\ C))\ |$

*resolution-bot:*

$trail\text{-false-cl}\ \Gamma\ E \implies$   
 $linorder-lit.is-maximal-in-mset\ E\ L \implies$   
 $is\text{-neg}\ L \implies$   
 $map\text{-of}\ \Gamma\ (-\ L) = Some\ (Some\ D) \implies$   
 $U_{er}' = finsert\ (eres\ D\ E)\ U_{er} \implies$   
 $eres\ D\ E = \{\#\} \implies$   
 $\Gamma' = [] \implies$   
 $ord\text{-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ Some\ E)\ (U_{er}',\ \mathcal{F},\ \Gamma',\ Some\ \{\#\})\ |$

*resolution-pos:*

$trail\text{-false-cl}\ \Gamma\ E \implies$   
 $linorder-lit.is-maximal-in-mset\ E\ L \implies$   
 $is\text{-neg}\ L \implies$   
 $map\text{-of}\ \Gamma\ (-\ L) = Some\ (Some\ D) \implies$   
 $U_{er}' = finsert\ (eres\ D\ E)\ U_{er} \implies$   
 $eres\ D\ E \neq \{\#\} \implies$   
 $\Gamma' = dropWhile\ (\lambda Ln.\ atm\text{-of}\ K\ \preceq_t\ atm\text{-of}\ (fst\ Ln))\ \Gamma \implies$   
 $linorder-lit.is-maximal-in-mset\ (eres\ D\ E)\ K \implies$   
 $is\text{-pos}\ K \implies$   
 $ord\text{-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ Some\ E)\ (U_{er}',\ \mathcal{F},\ \Gamma',\ Some\ (eres\ D\ E))\ |$

*resolution-neg:*

$trail\text{-false-cl}\ \Gamma\ E \implies$   
 $linorder-lit.is-maximal-in-mset\ E\ L \implies$   
 $is\text{-neg}\ L \implies$   
 $map\text{-of}\ \Gamma\ (-\ L) = Some\ (Some\ D) \implies$   
 $U_{er}' = finsert\ (eres\ D\ E)\ U_{er} \implies$   
 $eres\ D\ E \neq \{\#\} \implies$   
 $\Gamma' = dropWhile\ (\lambda Ln.\ atm\text{-of}\ K\ \preceq_t\ atm\text{-of}\ (fst\ Ln))\ \Gamma \implies$   
 $linorder-lit.is-maximal-in-mset\ (eres\ D\ E)\ K \implies$   
 $is\text{-neg}\ K \implies$   
 $map\text{-of}\ \Gamma\ (-\ K) = Some\ (Some\ C) \implies$   
 $ord\text{-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ Some\ E)\ (U_{er}',\ \mathcal{F},\ \Gamma',\ Some\ C)$

**lemma** *right-unique-ord-res-7:*

**fixes**  $N :: 'f\ gclause\ fset$

**shows** *right-unique* ( $ord\text{-res-7}\ N$ )

**proof** (*rule right-uniqueI*)

```

fix  $x\ y\ z$ 
assume  $step1: ord-res-7\ N\ x\ y$  and  $step2: ord-res-7\ N\ x\ z$ 
show  $y = z$ 
  using  $step1$ 
proof ( $cases\ N\ x\ y\ rule: ord-res-7.cases$ )
  case  $step-hyps1: (decide-neg\ \Gamma\ C\ L\ U_{er}\ A\ \Gamma'\ \mathcal{F})$ 
  show  $?thesis$ 
    using  $step2$ 
    unfolding  $step-hyps1(1)$ 
proof ( $cases\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ Some\ C)\ z\ rule: ord-res-7.cases$ )
  case  $step-hyps2: (decide-neg\ L2\ A2\ \Gamma2')$ 
  have  $L2 = L$ 
    using  $step-hyps1\ step-hyps2$ 
    using  $linorder-lit.Uniq-is-maximal-in-mset[of\ C,\ THEN\ Uniq-D]$  by  $metis$ 
  have  $A2 = A$ 
    using  $step-hyps1\ step-hyps2$ 
    unfolding  $\langle L2 = L \rangle$ 
    using  $linorder-trm.Uniq-is-least-in-fset[THEN\ Uniq-D]$  by  $presburger$ 
  have  $\Gamma2' = \Gamma'$ 
    using  $step-hyps1\ step-hyps2$ 
    unfolding  $\langle L2 = L \rangle\ \langle A2 = A \rangle$  by  $metis$ 
  show  $?thesis$ 
    unfolding  $step-hyps1(2)\ step-hyps2(1)$ 
    unfolding  $\langle \Gamma2' = \Gamma' \rangle\ ..$ 
next
  case  $step-hyps2: (skip-defined\ L2\ C2')$ 
  have  $L2 = L$ 
    using  $step-hyps1\ step-hyps2$ 
    using  $linorder-lit.Uniq-is-maximal-in-mset[THEN\ Uniq-D]$  by  $metis$ 
  have  $False$ 
    using  $step-hyps1\ step-hyps2$ 
    unfolding  $\langle L2 = L \rangle$ 
    unfolding  $linorder-trm.is-least-in-filter-iff$  by  $metis$ 
  then show  $?thesis\ ..$ 
next
  case  $step-hyps2: (skip-undefined-neg\ L2\ \Gamma2'\ C2')$ 
  have  $L2 = L$ 
    using  $step-hyps1\ step-hyps2$ 
    using  $linorder-lit.Uniq-is-maximal-in-mset[THEN\ Uniq-D]$  by  $metis$ 
  have  $False$ 
    using  $step-hyps1\ step-hyps2$ 
    unfolding  $\langle L2 = L \rangle$ 
    unfolding  $linorder-trm.is-least-in-filter-iff$  by  $metis$ 
  then show  $?thesis\ ..$ 
next
  case  $step-hyps2: (skip-undefined-pos\ L2\ D2)$ 
  have  $L2 = L$ 
    using  $step-hyps1\ step-hyps2$ 
    using  $linorder-lit.Uniq-is-maximal-in-mset[THEN\ Uniq-D]$  by  $metis$ 

```

```

have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-filter-iff by metis
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2 Γ2′)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-filter-iff by metis
then show ?thesis ..
next
case step-hyps2: (production L2 Γ2′ C2′)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-filter-iff by metis
then show ?thesis ..
next
case step-hyps2: (factoring L2 F2′)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-filter-iff by metis
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2 Uer2′ Γ2′)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2 Uer2′ Γ2′ K2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-neg L2 D2 Uer2′ Γ2′ K2 C2)
have False
  using step-hyps1 step-hyps2 by argo

```

```

    then show ?thesis ..
qed
next
case step-hyps1: (skip-defined  $\Gamma$  C L  $U_{er}$  C'  $\mathcal{F}$ )
show ?thesis
  using step2
  unfolding step-hyps1(1)
proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , Some C) z rule: ord-res-7.cases)
case step-hyps2: (decide-neg L2 A2  $\Gamma 2'$ )
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$ 
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..
next
case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have C2' = C'
  using step-hyps1 step-hyps2 by argo
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding  $\langle C2' = C' \rangle$  ..
next
case step-hyps2: (skip-undefined-neg L2  $\Gamma 2'$  C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$ 
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$ 
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2  $\Gamma 2'$ )

```

```

have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-filter-iff by metis
then show ?thesis ..
next
case step-hyps2: (production L2  $\Gamma 2'$  C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (factoring L2  $\mathcal{F} 2'$ )
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2  $U_{er} 2'$   $\Gamma 2'$ )
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2  $U_{er} 2'$   $\Gamma 2'$  K2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-neg L2 D2  $U_{er} 2'$   $\Gamma 2'$  K2 C2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
qed
next
case step-hyps1: (skip-undefined-neg  $\Gamma$  C L  $U_{er}$   $\Gamma'$  C'  $\mathcal{F}$ )
show ?thesis
  using step2 unfolding step-hyps1(1)
proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , Some C) z rule: ord-res-7.cases)
  case step-hyps2: (decide-neg L2 A2  $\Gamma 2'$ )
  have L2 = L

```

```

    using step-hyps1 step-hyps2
    using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
  have False
    using step-hyps1 step-hyps2
    unfolding ⟨L2 = L⟩
    unfolding linorder-trm.is-least-in-ffilter-iff by metis
  then show ?thesis ..
next
case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-neg L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have Γ2' = Γ'
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
have C2' = C'
  using step-hyps1 step-hyps2 by argo
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding ⟨C2' = C'⟩ ⟨Γ2' = Γ'⟩ ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2 Γ2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..

```



```

next
  case step-hyps2: (production L2 Γ2' C2')
  have  $L2 = L$ 
    using step-hyps1 step-hyps2
    using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
  have False
    using step-hyps1 step-hyps2
    unfolding  $\langle L2 = L \rangle$  by argo
  then show ?thesis ..
next
  case step-hyps2: (factoring L2 F2')
  have  $L2 = L$ 
    using step-hyps1 step-hyps2
    using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
  have False
    using step-hyps1 step-hyps2
    unfolding  $\langle L2 = L \rangle$  by argo
  then show ?thesis ..
next
  case step-hyps2: (resolution-bot L2 D2 Uer2' Γ2')
  have False
    using step-hyps1 step-hyps2 by argo
  then show ?thesis ..
next
  case step-hyps2: (resolution-pos L2 D2 Uer2' Γ2' K2)
  have False
    using step-hyps1 step-hyps2 by argo
  then show ?thesis ..
next
  case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
  have False
    using step-hyps1 step-hyps2 by argo
  then show ?thesis ..
qed
next
  case step-hyps1: (skip-undefined-pos Γ C L Uer F D)
  show ?thesis
    using step2 unfolding step-hyps1(1)
  proof (cases N (Uer, F, Γ, Some C) z rule: ord-res-7.cases)
  case step-hyps2: (decide-neg L2 A2 Γ2')
  have  $L2 = L$ 
    using step-hyps1 step-hyps2
    using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
  have False
    using step-hyps1 step-hyps2
    unfolding  $\langle L2 = L \rangle$ 
    unfolding linorder-trm.is-least-in-ffilter-iff by metis
  then show ?thesis ..
next

```

```

case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-neg L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have D2 = D
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  by (meson Uniq-D linorder-cls.Uniq-is-least-in-fset)
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding ⟨D2 = D⟩ ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2 Γ')
have False
  using step-hyps1 step-hyps2
  by (metis linorder-cls.is-least-in-ffilter-iff)
thus ?thesis ..
next
case step-hyps2: (production L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (factoring L2 F2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis

```

```

have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2 Uer2' Γ2')
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2 Uer2' Γ2' K2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
qed
next
case step-hyps1: (skip-undefined-pos-ultimate Γ C L Uer Γ' F)
show ?thesis
  using step2 unfolding step-hyps1(1)
proof (cases N (Uer, F, Γ, Some C) z rule: ord-res-7.cases)
case step-hyps2: (decide-neg L2 A2 Γ2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..
next
case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-neg L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False

```

```

    using step-hyps1 step-hyps2
    unfolding ⟨L2 = L⟩ by argo
  then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have False
  using step-hyps1 step-hyps2
  by (metis linorder-cls.is-least-in-filter-iff)
thus ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2 Γ2′)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have Γ2′ = Γ′
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding ⟨Γ2′ = Γ′⟩ ..
next
case step-hyps2: (production L2 Γ2′ C2′)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (factoring L2 F2′)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2 Uer2′ Γ2′)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2 Uer2′ Γ2′ K2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next

```

```

    case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
    have False
      using step-hyps1 step-hyps2 by argo
    then show ?thesis ..
qed
next
case step-hyps1: (production Γ C L Uer Γ' C' F)
show ?thesis
  using step2
  unfolding step-hyps1(1)
proof (cases N (Uer, F, Γ, Some C) z rule: ord-res-7.cases)
  case step-hyps2: (decide-neg L2 A2 Γ2')
  have L2 = L
    using step-hyps1 step-hyps2
    using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
  have False
    using step-hyps1 step-hyps2
    unfolding ⟨L2 = L⟩
    unfolding linorder-trm.is-least-in-filter-iff by metis
  then show ?thesis ..
next
case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-neg L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2 Γ2')

```

```

have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (production L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have Γ2' = Γ'
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
have C2' = C'
  using step-hyps1 step-hyps2 by argo
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding ⟨Γ2' = Γ'⟩ ⟨C2' = C'⟩ ..
next
case step-hyps2: (factoring L2 F2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  unfolding linorder-trm.is-least-in-ffilter-iff by metis
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2 Uer2' Γ2')
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2 Uer2' Γ2' K2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
qed
next
case step-hyps1: (factoring Γ C L Uer F' F)
show ?thesis

```

```

    using step2
    unfolding step-hyps1(1)
  proof (cases N (Uer, F, Γ, Some C) z rule: ord-res-7.cases)
    case step-hyps2: (decide-neg L2 A2 Γ2')
      have L2 = L
        using step-hyps1 step-hyps2
        using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
      have False
        using step-hyps1 step-hyps2
        unfolding ⟨L2 = L⟩
        unfolding linorder-trm.is-least-in-filter-iff by metis
      then show ?thesis ..
    next
      case step-hyps2: (skip-defined L2 C2')
        have L2 = L
          using step-hyps1 step-hyps2
          using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
        have False
          using step-hyps1 step-hyps2
          unfolding ⟨L2 = L⟩ by argo
        then show ?thesis ..
    next
      case step-hyps2: (skip-undefined-neg L2 Γ2' C2')
        have L2 = L
          using step-hyps1 step-hyps2
          using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
        have False
          using step-hyps1 step-hyps2
          unfolding ⟨L2 = L⟩ by argo
        then show ?thesis ..
    next
      case step-hyps2: (skip-undefined-pos L2 D2)
        have L2 = L
          using step-hyps1 step-hyps2
          using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
        have False
          using step-hyps1 step-hyps2
          unfolding ⟨L2 = L⟩ by argo
        then show ?thesis ..
    next
      case step-hyps2: (skip-undefined-pos-ultimate L2 Γ2')
        have L2 = L
          using step-hyps1 step-hyps2
          using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
        have False
          using step-hyps1 step-hyps2
          unfolding ⟨L2 = L⟩ by argo
        then show ?thesis ..
  next

```

```

case step-hyps2: (production L2  $\Gamma 2'$  C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of C, THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$ 
  unfolding linorder-trm.is-least-in-filter-iff by metis
then show ?thesis ..
next
case step-hyps2: (factoring L2  $\mathcal{F} 2'$ )
have  $\mathcal{F} 2' = \mathcal{F}'$ 
  using step-hyps1 step-hyps2
  by argo
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding  $\langle \mathcal{F} 2' = \mathcal{F}' \rangle$  ..
next
case step-hyps2: (resolution-bot L2 D2  $U_{er} 2' \Gamma 2'$ )
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2  $U_{er} 2' \Gamma 2' K2$ )
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-neg L2 D2  $U_{er} 2' \Gamma 2' K2 C2$ )
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
qed
next
case step-hyps1: (resolution-bot  $\Gamma E L D U_{er}' U_{er} \Gamma' \mathcal{F}$ )
show ?thesis
  using step2
  unfolding step-hyps1(1)
proof (cases N ( $U_{er}, \mathcal{F}, \Gamma, \text{Some } E$ ) z rule: ord-res-7.cases)
case step-hyps2: (decide-neg L2 A2  $\Gamma 2'$ )
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False

```



```

    using step-hyps1 step-hyps2
    unfolding ⟨L2 = L⟩ by argo
  then show ?thesis ..
next
case step-hyps2: (skip-undefined-neg L2 Γ2' C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2 Γ2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo
then show ?thesis ..
next
case step-hyps2: (production L2 Γ2' C2')
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (factoring L2 F2')
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2 Uer2' Γ2')
have Uer2' = Uer'
  using step-hyps1 step-hyps2
  by argo
have Γ2' = Γ'
  using step-hyps1 step-hyps2
  by argo
show ?thesis

```

```

    unfolding step-hyps1(2) step-hyps2(1)
    unfolding ⟨Uer2' = Uer'⟩ ⟨Γ2' = Γ'⟩ ..
next
case step-hyps2: (resolution-pos L2 D2 Uer2' Γ2' K2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
have D2 = D
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  by (metis option.inject)
have False
  using step-hyps1 step-hyps2
  unfolding ⟨D2 = D⟩
  by argo
then show ?thesis ..
next
case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
have D2 = D
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  by (metis option.inject)
have False
  using step-hyps1 step-hyps2
  unfolding ⟨D2 = D⟩
  by argo
then show ?thesis ..
qed
next
case step-hyps1: (resolution-pos Γ E L D Uer' Uer Γ' K F)
show ?thesis
  using step2
  unfolding step-hyps1(1)
proof (cases N (Uer, F, Γ, Some E) z rule: ord-res-7.cases)
case step-hyps2: (decide-neg L2 A2 Γ2')
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (skip-defined L2 C2')
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩ by argo

```

```

    then show ?thesis ..
  next
  case step-hyps2: (skip-undefined-neg L2  $\Gamma 2'$  C2')
  have L2 = L
    using step-hyps1 step-hyps2
    using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
  have False
    using step-hyps1 step-hyps2
    unfolding  $\langle L2 = L \rangle$  by argo
  then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos L2 D2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$  by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate L2  $\Gamma 2'$ )
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$  by argo
then show ?thesis ..
next
case step-hyps2: (production L2  $\Gamma 2'$  C2')
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (factoring L2  $\mathcal{F} 2'$ )
have False
  using step-hyps1 step-hyps2 by argo
then show ?thesis ..
next
case step-hyps2: (resolution-bot L2 D2  $U_{er} 2'$   $\Gamma 2'$ )
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
have D2 = D
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$ 
  by (metis option.inject)
have False
  using step-hyps1 step-hyps2

```

```

    unfolding ⟨D2 = D⟩
    by argo
  then show ?thesis ..
next
case step-hyps2: (resolution-pos L2 D2 Uer2' Γ2' K2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
have D2 = D
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  by (metis option.inject)
have K2 = K
  using step-hyps1 step-hyps2
  unfolding ⟨D2 = D⟩
  using linorder-lit.Uniq-is-maximal-in-mset[of eres D E, THEN Uniq-D] by
metis
have Uer2' = Uer'
  using step-hyps1 step-hyps2
  unfolding ⟨D2 = D⟩
  by argo
have Γ2' = Γ'
  using step-hyps1 step-hyps2
  unfolding ⟨K2 = K⟩
  by argo
show ?thesis
  unfolding step-hyps1(2) step-hyps2(1)
  unfolding ⟨Uer2' = Uer'⟩ ⟨Γ2' = Γ'⟩ ⟨D2 = D⟩ ..
next
case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
have L2 = L
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
have D2 = D
  using step-hyps1 step-hyps2
  unfolding ⟨L2 = L⟩
  by (metis option.inject)
have K2 = K
  using step-hyps1 step-hyps2
  unfolding ⟨D2 = D⟩
  using linorder-lit.Uniq-is-maximal-in-mset[of eres D E, THEN Uniq-D] by
metis
have False
  using step-hyps1 step-hyps2
  unfolding ⟨K2 = K⟩
  by argo
then show ?thesis ..
qed
next

```

```

case step-hyps1: (resolution-neg  $\Gamma$   $E$   $L$   $D$   $U_{er}$   $U_{er}$   $\Gamma'$   $K$   $C$   $\mathcal{F}$ )
show ?thesis
  using step2
  unfolding step-hyps1(1)
proof (cases  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , Some  $E$ ) z rule: ord-res-7.cases)
  case step-hyps2: (decide-neg  $L2$   $A2$   $\Gamma2'$ )
  have False
    using step-hyps1 step-hyps2 by argo
  then show ?thesis ..
next
case step-hyps2: (skip-defined  $L2$   $C2'$ )
have  $L2 = L$ 
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$  by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-neg  $L2$   $\Gamma2'$   $C2'$ )
have  $L2 = L$ 
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$  by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos  $L2$   $D2$ )
have  $L2 = L$ 
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$  by argo
then show ?thesis ..
next
case step-hyps2: (skip-undefined-pos-ultimate  $L2$   $\Gamma2'$ )
have  $L2 = L$ 
  using step-hyps1 step-hyps2
  using linorder-lit.Uniq-is-maximal-in-mset[THEN Uniq-D] by metis
have False
  using step-hyps1 step-hyps2
  unfolding  $\langle L2 = L \rangle$  by argo
then show ?thesis ..
next
case step-hyps2: (production  $L2$   $\Gamma2'$   $C2'$ )
have False
  using step-hyps1 step-hyps2 by argo

```

```

    then show ?thesis ..
  next
    case step-hyps2: (factoring L2 F2')
    have False
      using step-hyps1 step-hyps2 by argo
    then show ?thesis ..
  next
    case step-hyps2: (resolution-bot L2 D2 Uer2' Γ2')
    have L2 = L
      using step-hyps1 step-hyps2
      using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
    have D2 = D
      using step-hyps1 step-hyps2
      unfolding ⟨L2 = L⟩
      by (metis option.inject)
    have False
      using step-hyps1 step-hyps2
      unfolding ⟨D2 = D⟩
      by argo
    then show ?thesis ..
  next
    case step-hyps2: (resolution-pos L2 D2 Uer2' Γ2' K2)
    have L2 = L
      using step-hyps1 step-hyps2
      using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
    have D2 = D
      using step-hyps1 step-hyps2
      unfolding ⟨L2 = L⟩
      by (metis option.inject)
    have K2 = K
      using step-hyps1 step-hyps2
      unfolding ⟨D2 = D⟩
      using linorder-lit.Uniq-is-maximal-in-mset[of eres D E, THEN Uniq-D] by
metis
    have False
      using step-hyps1 step-hyps2
      unfolding ⟨K2 = K⟩
      by argo
    then show ?thesis ..
  next
    case step-hyps2: (resolution-neg L2 D2 Uer2' Γ2' K2 C2)
    have L2 = L
      using step-hyps1 step-hyps2
      using linorder-lit.Uniq-is-maximal-in-mset[of E, THEN Uniq-D] by metis
    have D2 = D
      using step-hyps1 step-hyps2
      unfolding ⟨L2 = L⟩
      by (metis option.inject)
    have K2 = K

```

```

using step-hyps1 step-hyps2
unfolding  $\langle D2 = D \rangle$ 
using linorder-lit.Uniq-is-maximal-in-mset[of eres D E, THEN Uniq-D] by
metis
have  $U_{er}2' = U_{er}'$ 
using step-hyps1 step-hyps2
unfolding  $\langle D2 = D \rangle$ 
by argo
have  $\Gamma2' = \Gamma'$ 
using step-hyps1 step-hyps2
unfolding  $\langle K2 = K \rangle$ 
by argo
have  $C2 = C$ 
using step-hyps1 step-hyps2
unfolding  $\langle K2 = K \rangle$ 
by (metis option.inject)
show ?thesis
unfolding step-hyps1(2) step-hyps2(1)
unfolding  $\langle U_{er}2' = U_{er}' \rangle \langle \Gamma2' = \Gamma' \rangle \langle C2 = C \rangle ..$ 
qed
qed
qed

```

**inductive** *ord-res-7-final* **where**

*model-found:*

*ord-res-7-final* ( $N, U_{er}, \mathcal{F}, \Gamma, None$ ) |

*contradiction-found:*

*ord-res-7-final* ( $N, U_{er}, \mathcal{F}, \Gamma, Some \{\#\}$ )

**sublocale** *ord-res-7-antics: semantics* **where**

*step = constant-context ord-res-7* **and**

*final = ord-res-7-final*

**proof** *unfold-locales*

**fix**  $S :: 'f\ gclause\ fset \times 'f\ gclause\ fset \times 'f\ gclause\ fset \times$   
 $('f\ gterm\ literal \times 'f\ gterm\ literal\ multiset\ option)\ list \times$   
 $'f\ gterm\ literal\ multiset\ option$

**obtain**

$N\ U_{er}\ \mathcal{F} :: 'f\ gterm\ clause\ fset$  **and**

$\Gamma :: ('f\ gterm\ literal \times 'f\ gterm\ literal\ multiset\ option)\ list$  **and**

$\mathcal{C} :: 'f\ gclause\ option$

**where**

*S-def:*  $S = (N, U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$

**by** (*metis prod.exhaust*)

**assume** *ord-res-7-final S*

**hence**  $\nexists x. \text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})\ x$

**unfolding** *S-def*  
**proof** (*cases* ( $N, U_{er}, \mathcal{F}, \Gamma, \mathcal{C}$ ) *rule: ord-res-7-final.cases*)  
**case** *model-found*  
**thus** *?thesis*  
**by** (*auto elim: ord-res-7.cases*)  
**next**  
**case** *contradiction-found*  
**thus** *?thesis*  
**by** (*auto simp: linorder-lit.is-maximal-in-mset-iff elim: ord-res-7.cases*)  
**qed**

**thus** *finished* (*constant-context ord-res-7*) *S*  
**by** (*simp add: S-def finished-def constant-context.simps*)  
**qed**

**inductive** *ord-res-7-invars* **for**  $N$  **where**  
*ord-res-7-invars*  $N$  ( $U_{er}, \mathcal{F}, \Gamma, \mathcal{C}$ ) **if**  
 $\mathcal{F} \mid\subseteq N \mid\cup U_{er}$  **and**  
 $(\forall C. \mathcal{C} = \text{Some } C \longrightarrow C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}))$  **and**  
 $(\forall D. \mathcal{C} = \text{Some } D \longrightarrow$   
 $(\forall C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in\# C. K \neq L_C\#})$   
 $\wedge \text{is-pos } L_C)))$  **and**  
 $(\forall C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}). (\forall D. \mathcal{C} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $\text{trail-true-cls } \Gamma C)$  **and**  
 $(\forall C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}). (\forall D. \mathcal{C} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \mid\in \text{atms-of-clss } (N \mid\cup U_{er}). A \prec_t \text{atm-of } K \wedge A \not\in \text{trail-atms}$   
 $\Gamma)))$  **and**  
 $\text{sorted-wrt } (\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid\cup U_{er}))$  **and**  
 $(\mathcal{C} = \text{None} \longrightarrow \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid\cup U_{er}))$  **and**  
 $(\forall C. \mathcal{C} = \text{Some } C \longrightarrow$   
 $(\forall A \mid\in \text{trail-atms } \Gamma. \exists L. \text{linorder-lit.is-maximal-in-mset } C L \wedge A \preceq_t \text{atm-of}$   
 $L))$  **and**  
 $(\forall Ln \in \text{set } \Gamma. \text{is-neg } (fst Ln) \longleftrightarrow \text{snd } Ln = \text{None})$  **and**  
 $(\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}). C \prec_c \{\#fst Ln\#} \longrightarrow \text{trail-true-cls } \Gamma C))$

**and**  
 $(\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
 $(fst Ln))$  **and**  
 $(\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}))$  **and**  
 $(\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{\#K \in\#$   
 $C. K \neq L\#})$  **and**  
 $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid\in \text{iefac } \mathcal{F} \mid\uparrow (N \mid\cup U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$

**lemma** *clause-almost-defined-if-lt-next-clause:*



```

assumes ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma, C)$ 
shows  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). (\forall D. C = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$ 
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C \ K \longrightarrow \text{trail-defined-clt } \Gamma \ \{\#L \in \# \ C. L$ 
 $\neq K\})$ 
proof (intro ballI impI allI)
  fix  $C :: 'f \text{ gclause}$  and  $K :: 'f \text{ gliteral}$ 
  assume
     $C\text{-in}: C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  and
     $C\text{-lt}: \forall D. C = \text{Some } D \longrightarrow C \prec_c D$  and
     $C\text{-max-lit}: \text{ord-res.is-maximal-lit } K \ C$ 

  show  $\text{trail-defined-clt } \Gamma \ \{\#L \in \# \ C. L \neq K\}$ 
  using assms
  proof (cases  $N (U_{er}, \mathcal{F}, \Gamma, C)$  rule: ord-res-7-invars.cases)
  case invars: 1

    have  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \mid \notin \mid \text{trail-atms } \Gamma)$ 
    using invars C-in C-lt C-max-lit by metis

    hence  $C\text{-almost-almost-defined}: \text{trail-defined-clt } \Gamma \ \{\#L \in \# \ C. L \neq K \wedge L \neq$ 
     $- K\}$ 
    using clause-almost-almost-definedI[OF C-in C-max-lit] by blast

  show ?thesis
  proof (cases  $\text{trail-defined-lit } \Gamma \ K$ )
  case True
    hence  $\text{trail-defined-lit } \Gamma \ (- K)$ 
    unfolding trail-defined-lit-def uminus-of-uminus-id by argo
    then show ?thesis
    using C-almost-almost-defined
    unfolding trail-defined-clt-def
    by auto
  next
  case False
  show ?thesis
  proof (cases  $C$ )
  case None
    hence  $\text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er})$ 
    using invars by argo
    then show ?thesis
  by (metis C-in C-max-lit False atm-of-in-atms-of-clssI linorder-lit.is-maximal-in-mset-iff
trail-defined-lit-iff-trail-defined-atm)
  next
  case (Some D)
  hence  $C \prec_c D$ 
  using C-lt by simp
  then show ?thesis
  using invars
  by (smt (verit, ccfv-SIG) C-almost-almost-defined C-in C-max-lit False

```

*Some*  
 $\langle \neg (\exists A | \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms}$   
 $\Gamma) \rangle$   
*atm-of-in-atms-of-clss* *I* *atm-of-uminus* *filter-mset-cong0*  
*linorder-lit.is-greatest-in-set-iff* *linorder-lit.is-maximal-in-mset-iff*  
*linorder-lit.is-maximal-in-set-eq-is-greatest-in-set*  
*linorder-lit.is-maximal-in-set-iff* *literal.collapse(1)* *literal.exhaust-sel*  
*ord-res.less-lit-simps(4)* *trail-defined-lit-iff-trail-defined-atm*  
**qed**  
**qed**  
**qed**  
**qed**

**lemma** *ord-res-7-invars-def*:

*ord-res-7-invars*  $N \ s \longleftrightarrow$   
 $(\forall U_{er} \ \mathcal{F} \ \Gamma \ \mathcal{C}. \ s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \longrightarrow$   
 $\mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er} \wedge$   
 $(\forall C. \ \mathcal{C} = \text{Some } C \longrightarrow C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})) \wedge$   
 $(\forall D. \ \mathcal{C} = \text{Some } D \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \ C \prec_c D \longrightarrow$   
 $(\forall L_C. \ \text{linorder-lit.is-maximal-in-mset } C \ L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma \ L_C \longrightarrow (\text{trail-true-cls } \Gamma \ \{\#K \in \# C. \ K \neq L_C \#\}$   
 $\wedge \text{is-pos } L_C)))) \wedge$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \ (\forall D. \ \mathcal{C} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $\text{trail-true-cls } \Gamma \ C) \wedge$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \ (\forall D. \ \mathcal{C} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \ \text{linorder-lit.is-maximal-in-mset } C \ K \longrightarrow$   
 $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms}$   
 $\Gamma))) \wedge$   
 $\text{sorted-wrt } (\lambda x \ y. \ \text{atm-of } (fst \ y) \prec_t \text{atm-of } (fst \ x)) \ \Gamma \wedge$   
 $\text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) \ (\text{atms-of-clss } (N \mid \cup \mid U_{er})) \wedge$   
 $(\mathcal{C} = \text{None} \longrightarrow \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er})) \wedge$   
 $(\forall C. \ \mathcal{C} = \text{Some } C \longrightarrow$   
 $(\forall A \mid \in \mid \text{trail-atms } \Gamma. \ \exists L. \ \text{linorder-lit.is-maximal-in-mset } C \ L \wedge A \preceq_t \text{atm-of}$   
 $L)) \wedge$   
 $(\forall Ln \in \text{set } \Gamma. \ \text{is-neg } (fst \ Ln) \longleftrightarrow \text{snd } Ln = \text{None}) \wedge$   
 $(\forall Ln \in \text{set } \Gamma. \ \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \ C \prec_c \{\#fst \ Ln \# \} \longrightarrow \text{trail-true-cls } \Gamma \ C))$   
 $\wedge$   
 $(\forall Ln \in \text{set } \Gamma. \ \forall C. \ \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
 $(fst \ Ln)) \wedge$   
 $(\forall Ln \in \text{set } \Gamma. \ \forall C. \ \text{snd } Ln = \text{Some } C \longrightarrow C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})) \wedge$   
 $(\forall \Gamma_1 \ L \ C \ \Gamma_0. \ \Gamma = \Gamma_1 \ @ \ (L, \text{Some } C) \ \# \ \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \ \{\#K \in \#$   
 $C. \ K \neq L \#\}) \wedge$   
 $(\forall \Gamma_1 \ L \ D \ \Gamma_0. \ \Gamma = \Gamma_1 \ @ \ (L, \text{Some } D) \ \# \ \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \ C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 \ C)))$   
**(is ?NICE**  $N \ s \longleftrightarrow ?UGLY \ N \ s)$   
**proof** (*rule iffI*)  
**show**  $?NICE \ N \ s \implies ?UGLY \ N \ s$

**apply** (*intro allI impI*)  
**subgoal premises** *prems* **for**  $U_{er} \mathcal{F} \Gamma \mathcal{C}$   
**using** *prems(1)* **unfolding** *prems(2)*  
**by** (*cases N (U<sub>er</sub>,  $\mathcal{F}$ ,  $\Gamma$ ,  $\mathcal{C}$ ) rule: ord-res- $\gamma$ -invars.cases*) (*simp only:*)  
**done**  
**next**  
**assume** *?UGLY N s*  
**obtain**  $U_{er} \mathcal{F} \Gamma \mathcal{C}$  **where** *s-def: s = (U<sub>er</sub>,  $\mathcal{F}$ ,  $\Gamma$ ,  $\mathcal{C}$ )*  
**by** (*metis prod.exhaust*)  
**show** *?NICE N s*  
**using**  $\langle ?UGLY N s \rangle$  [*rule-format, OF s-def*]  
**unfolding** *s-def*  
**by** (*intro ord-res- $\gamma$ -invars.intros*) *simp-all*  
**qed**

**lemma** *ord-res- $\gamma$ -invars-implies-trail-consistent:*  
**assumes** *ord-res- $\gamma$ -invars N (U<sub>er</sub>,  $\mathcal{F}$ ,  $\Gamma$ ,  $\mathcal{C}$ )*  
**shows** *trail-consistent  $\Gamma$*   
**using** *assms* **unfolding** *ord-res- $\gamma$ -invars-def*  
**by** (*metis trail-consistent-if-sorted-wrt-atoms*)

**lemma** *ord-res- $\gamma$ -invars-implies-propagated-clause-almost-false:*  
**assumes** *invars: ord-res- $\gamma$ -invars N (U<sub>er</sub>,  $\mathcal{F}$ ,  $\Gamma$ ,  $\mathcal{C}$ )* **and**  $(L, \text{Some } C) \in \text{set } \Gamma$   
**shows** *trail-false-cls  $\Gamma$   $\{\#K \in \# C. K \neq L\#$*   
**proof** –  
**obtain**  $\Gamma_1 \Gamma_0$  **where**  *$\Gamma$ -eq:  $\Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0$*   
**using**  $\langle (L, \text{Some } C) \in \text{set } \Gamma \rangle$  **by** (*metis split-list*)  
  
**hence** *trail-false-cls  $\Gamma_0$   $\{\#K \in \# C. K \neq L\#$*   
**using** *invars* **by** (*simp-all add: ord-res- $\gamma$ -invars-def*)  
  
**moreover have** *suffix  $\Gamma_0 \Gamma$*   
**using**  *$\Gamma$ -eq* **by** (*simp add: suffix-def*)  
  
**ultimately show** *?thesis*  
**by** (*metis trail-false-cls-if-trail-false-suffix*)  
**qed**

**lemma** *ord-res- $\gamma$ -preserves-invars:*  
**assumes** *step: ord-res- $\gamma$  N s s'* **and** *invar: ord-res- $\gamma$ -invars N s*  
**shows** *ord-res- $\gamma$ -invars N s'*  
**using** *step*  
**proof** (*cases N s s' rule: ord-res- $\gamma$ .cases*)  
**case** *step-hyps: (decide-neg  $\Gamma$  D L U<sub>er</sub> A  $\Gamma'$   $\mathcal{F}$ )*  
  
**note** *D-max-lit =  $\langle \text{ord-res.is-maximal-lit } L D \rangle$*   
  
**have**  
 $A \in | \text{atms-of-cls } (N \cup U_{er})$  **and**

$A \prec_t \text{atm-of } L$  **and**  
 $A \notin \text{trail-atms } \Gamma$   
**using** *step-hyps* **unfolding** *atomize-conj linorder-trm.is-least-in-filter-iff* **by**  
*argo*

**have** *suffix*  $\Gamma \Gamma'$   
**using** *step-hyps* **unfolding** *suffix-def* **by** *simp*

**have**

*F*-subset:  $\mathcal{F} \subseteq N \cup U_{er}$  **and**

*D*-in:  $D \in \text{iefac } \mathcal{F} \mid (N \cup U_{er})$  **and**

*clauses-lt-D-seldomly-have-undef-max-lit*:

$\forall C \in \text{iefac } \mathcal{F} \mid (N \cup U_{er}). C \prec_c D \longrightarrow$

$(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$

$\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in \# C. K \neq L_C\} \wedge$   
*is-pos } L\_C)) **and***

*clauses-lt-D-true*:  $\forall C \in \text{iefac } \mathcal{F} \mid (N \cup U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma$

**and**

*no-undef-atm-lt-max-lit-if-lt-D*:  $\forall C \in \text{iefac } \mathcal{F} \mid (N \cup U_{er}). C \prec_c D \longrightarrow$

$(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$

$\neg (\exists A \in \text{atms-of-clss } (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma))$

**and**

$\Gamma$ -sorted: *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**

$\Gamma$ -lower: *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \cup U_{er}))$

**and**

*trail-atms-le0*:  $\forall A \in \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L D \wedge (A \preceq_t$   
*atm-of } L) **and***

$\Gamma$ -deci-iff-neg:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$  **and**

$\Gamma$ -deci-ball-lt-true:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$

$(\forall C \in \text{iefac } \mathcal{F} \mid (N \cup U_{er}). C \prec_c \{\#fst Ln\} \longrightarrow \text{trail-true-cls } \Gamma C)$

**and**

$\Gamma$ -prop-in:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in \text{iefac } \mathcal{F} \mid (N \cup$   
*U<sub>er</sub>)* **and**

$\Gamma$ -prop-greatest:

$\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$

*(fst Ln)* **and**

$\Gamma$ -prop-almost-false:

$\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{\#K \in \# C.$   
*K \neq L\}* **and**

$\Gamma$ -prop-ball-lt-true:  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$

$(\forall C \in \text{iefac } \mathcal{F} \mid (N \cup U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$

**using** *invar* **by**  $(\text{simp-all add: } \langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle \text{ord-res-7-invars-def})$

**have** *trail-atms-le*:  $\forall A \in \text{trail-atms } \Gamma. A \preceq_t \text{atm-of } L$

**using** *trail-atms-le0*  $\langle \text{ord-res.is-maximal-lit } L D \rangle$

**by**  $(\text{metis } \text{Uniq-D linorder-lit.Uniq-is-maximal-in-mset})$

**have** *clauses-lt-D-almost-defined*:  $\forall C \in \text{iefac } \mathcal{F} \mid (N \cup U_{er}). C \prec_c D \longrightarrow$

$(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{\#L \in \# C. L$

$\neq K\#\}$ )  
**using** *invar*[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*

**have**  $\mathcal{F} \mid\subseteq\mid N \mid\cup\mid U_{er}$   
**using**  *$\mathcal{F}$ -subset* .

**moreover have**  $\forall C'. \text{Some } D = \text{Some } C' \longrightarrow C' \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$   
**using** *D-in* **by** *simp*

**moreover have** *trail-true-cls*  $\Gamma' \{\#K \in\# C. K \neq L_C\# \} \wedge \text{is-pos } L_C$   
**if** *Some*  $D = \text{Some } E$  **and**  
*C-in*:  $C \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$  **and**  
*C-lt*:  $C \prec_c E$  **and**  
*C-max-lit*: *linorder-lit.is-maximal-in-mset*  $C L_C$  **and**  
*L<sub>C</sub>-undef*:  $\neg \text{trail-defined-lit } \Gamma' L_C$   
**for**  $E C L_C$

**proof** –  
**have**  $E = D$   
**using** *that* **by** *simp*  
**hence**  $C \prec_c D$   
**using** *C-lt* **by** *argo*

**moreover have**  $\neg \text{trail-defined-lit } \Gamma L_C$   
**using** *L<sub>C</sub>-undef*  $\langle \text{suffix } \Gamma \Gamma' \rangle$   
**using** *trail-defined-lit-if-trail-defined-suffix* **by** *blast*

**ultimately have** *trail-true-cls*  $\Gamma \{\#K \in\# C. K \neq L_C\# \} \wedge \text{is-pos } L_C$   
**using** *clauses-lt-D-seldomly-have-undef-max-lit*[*rule-format, OF C-in - C-max-lit*]  
**by** *metis*

**thus** *?thesis*  
**using**  $\langle \text{suffix } \Gamma \Gamma' \rangle$  *trail-true-cls-if-trail-true-suffix* **by** *blast*  
**qed**

**moreover have**  
*trail-true-cls*  $\Gamma' C$   
 $\bigwedge K. \text{linorder-lit.is-maximal-in-mset } C K \implies$   
 $\neg (\exists A \mid\in\mid \text{atms-of-cls } (N \mid\cup\mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma')$   
**if** *C-lt*:  $\bigwedge D'. \text{Some } D = \text{Some } D' \implies C \prec_c D'$  **and**  $C \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$  **for**  $C$

**proof** –  
**have**  $C \prec_c D$   
**using** *C-lt* **by** *metis*

**hence** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true*  $\langle C \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er}) \rangle$  **by** *metis*

**thus** *trail-true-cls*  $\Gamma' C$

**using**  $\langle \text{suffix } \Gamma \Gamma' \rangle$   
**by** (*metis trail-true-cls-if-trail-true-suffix*)

**fix**  $K$   
**assume**  $C\text{-max-lit: linorder-lit.is-maximal-in-mset } C K$

**have**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \not\in \mid \text{trail-atms } \Gamma)$   
**using** *no-undef-atm-lt-max-lit-if-lt-D*  
**using**  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle \langle C \prec_c D \rangle C\text{-max-lit}$  **by** *metis*

**thus**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \not\in \mid \text{trail-atms } \Gamma')$   
**using** *step-hyps(6)* **by** *auto*  
**qed**

**moreover have** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma'$   
**proof** –  
**have**  $\forall x \mid \in \mid \text{trail-atms } \Gamma. x \prec_t A$   
**using** *step-hyps(5)[unfolded linorder-trm.is-least-in-filter-iff, simplified]*  
**by** (*metis*  $\Gamma\text{-lower linorder-trm.antisym-conv3 linorder-trm.is-lower-set-iff}$ )

**hence**  $\forall x \in \text{set } \Gamma. \text{atm-of } (fst x) \prec_t A$   
**by** (*simp add: fset-trail-atms*)

**thus** *?thesis*  
**using**  $\Gamma\text{-sorted}$  **by** (*simp add:*  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$ )  
**qed**

**moreover have** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma') (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**proof** –  
**have** *linorder-trm.is-lower-set*  $(\text{insert } A (\text{fset } (\text{trail-atms } \Gamma)))$   
 $(\text{fset } (\text{atms-of-clss } (N \mid \cup \mid U_{er})))$   
**proof** (*rule linorder-trm.is-lower-set-insertI*)  
**show**  $A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using** *step-hyps(5)[unfolded linorder-trm.is-least-in-filter-iff, simplified]*  
**by** *argo*  
**next**  
**show**  $\forall w \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). w \prec_t A \longrightarrow w \mid \in \mid \text{trail-atms } \Gamma$   
**using** *step-hyps(5)[unfolded linorder-trm.is-least-in-filter-iff, simplified]*  
**by** *fastforce*  
**next**  
**show** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**using**  $\Gamma\text{-lower}$  .  
**qed**

**moreover have** *trail-atms*  $\Gamma' = \text{finsert } A (\text{trail-atms } \Gamma)$   
**by** (*simp add:*  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$ )

**ultimately show** *?thesis*  
**by** *simp*  
**qed**

**moreover have**  $\forall A \mid \in \mid \text{trail-atms } \Gamma'. \exists L. \text{ord-res.is-maximal-lit } L \ D \wedge (A \preceq_t \text{atm-of } L)$   
**proof** (*intro ballI exI conjI*)  
**show** *ord-res.is-maximal-lit*  $L \ D$   
**using**  $\langle \text{ord-res.is-maximal-lit } L \ D \rangle$  .  
**next**  
**fix**  $x$  **assume**  $x \mid \in \mid \text{trail-atms } \Gamma'$

**hence**  $x = A \vee x \mid \in \mid \text{trail-atms } \Gamma$   
**unfolding**  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$  **by** *simp*

**thus**  $x \preceq_t \text{atm-of } L$   
**proof** (*elim disjE*)  
**assume**  $x = A$   
**thus**  $x \preceq_t \text{atm-of } L$   
**using** *step-hyps(5)* **by** (*simp add: linorder-trm.is-least-in-filter-iff*)  
**next**  
**assume**  $x \mid \in \mid \text{trail-atms } \Gamma$   
**thus**  $x \preceq_t \text{atm-of } L$   
**using** *trail-atms-le* **by** *metis*  
**qed**  
**qed**

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$   
**unfolding**  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$   
**using**  *$\Gamma$ -deci-iff-neg* **by** *simp*

**moreover have** *trail-true-cls*  $\Gamma' \ C$   
**if**  $Ln \in \text{set } \Gamma'$  **and**  $\text{snd } Ln = \text{None}$  **and**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $C \prec_c \{\# \text{fst } Ln \# \}$   
**for**  $Ln \ C$   
**proof** –  
**have**  $Ln = (\text{Neg } A, \text{None}) \vee Ln \in \text{set } \Gamma$   
**using**  $\langle Ln \in \text{set } \Gamma' \rangle$  **unfolding**  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$  **by** *simp*

**hence** *trail-true-cls*  $\Gamma \ C$   
**proof** (*elim disjE*)  
**assume**  $Ln = (\text{Neg } A, \text{None})$

**hence**  $\text{fst } Ln \prec_l L$   
**by** (*metis*  $\langle A \prec_t \text{atm-of } L \rangle$  *fst-conv literal.exhaust-sel ord-res.less-lit-simps(3)* *ord-res.less-lit-simps(4)*)

**moreover have** *linorder-lit.is-maximal-in-mset*  $\{\# \text{fst } Ln \# \}$   $(\text{fst } Ln)$

**unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *simp*

**ultimately have**  $\{\#fst Ln\} \prec_c D$   
**using** *D-max-lit*  
**using** *linorder-lit.multip-if-maximal-less-that-maximal* **by** *metis*

**hence**  $C \prec_c D$   
**using**  $\langle C \prec_c \{\#fst Ln\} \rangle$  **by** *order*

**thus** *trail-true-cls*  $\Gamma C$   
**using**  $\langle C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$   
**using** *clauses-lt-D-true* **by** *blast*

**next**  
**assume**  $Ln \in set \Gamma$

**thus** *trail-true-cls*  $\Gamma C$   
**using**  $\Gamma-deci-ball-lt-true \langle snd Ln = None \rangle \langle C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$   
 $\langle C \prec_c \{\#fst Ln\} \rangle$   
**by** *metis*  
**qed**

**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle suffix \Gamma \Gamma' \rangle$  **by** (*metis trail-true-cls-if-trail-true-suffix*)  
**qed**

**moreover have**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$   
**unfolding**  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  **using**  $\Gamma-prop-in$  **by** *simp*

**moreover have**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow linorder-lit.is-greatest-in-mset C (fst Ln)$   
**unfolding**  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  **using**  $\Gamma-prop-greatest$  **by** *simp*

**moreover have**  $\forall \Gamma_1 L C \Gamma_0. \Gamma' = \Gamma_1 @ (L, Some C) \# \Gamma_0 \longrightarrow trail-false-cls \Gamma_0 \{\#K \in \# C. K \neq L\}$   
**unfolding**  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  **using**  $\Gamma-prop-almost-false$   
**by** (*metis (no-types, lifting) append-eq-Cons-conv list.inject option.discI prod.inject*)

**moreover have**  $(\forall \Gamma_1 L D \Gamma_0. \Gamma' = \Gamma_1 @ (L, Some D) \# \Gamma_0 \longrightarrow (\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow trail-true-cls \Gamma_0 C))$   
**using**  $\Gamma-prop-ball-lt-true$   
**unfolding**  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$   
**by** (*smt (verit, best) append-eq-Cons-conv list.inject option.discI snd-conv*)

**ultimately show** *?thesis*  
**by** (*auto simp add: \langle s' = (U\_{er}, \mathcal{F}, \Gamma', Some D) \rangle ord-res-7-invars-def*)

**next**  
**case** *step-hyps: (skip-defined  $\Gamma D K U_{er} C' \mathcal{F}$ )*



**note**  $D\text{-max-lit} = \langle \text{ord-res.is-maximal-lit } K D \rangle$

**have**

$\mathcal{F}\text{-subset}: \mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er}$  **and**

$D\text{-in}: D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**

$\text{clauses-lt-}D\text{-seldomly-have-undef-max-lit}:$

$\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$

$(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$

$\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in \# C. K \neq L_C\} \wedge$

$\text{is-pos } L_C))$  **and**

$\text{clauses-lt-}D\text{-true}: \forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma$

$C$  **and**

$\text{no-undef-atm-lt-max-lit-if-lt-}D: \forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$

$(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$

$\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \not\in \mid \text{trail-atms } \Gamma))$

**and**

$\Gamma\text{-sorted}: \text{sorted-wrt } (\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**

$\Gamma\text{-lower}: \text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$

**and**

$\text{trail-atms-le0}: \forall A \mid \in \mid \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L D \wedge (A \preceq_t \text{atm-of } L)$  **and**

$\Gamma\text{-deci-iff-neg}: \forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$  **and**

$\Gamma\text{-deci-ball-lt-true}: \forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$

$(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). C \prec_c \{\#fst Ln\} \longrightarrow \text{trail-true-cls } \Gamma C)$

**and**

$\Gamma\text{-prop-in}: \forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**

$\Gamma\text{-prop-greatest}:$

$\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$

$(fst Ln)$  **and**

$\Gamma\text{-prop-almost-false}:$

$\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{\#K \in \# C.$

$K \neq L\#\}$  **and**

$\Gamma\text{-prop-ball-lt-true}: (\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$

$(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$

**using invar by** ( $\text{simp-all add}: \langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle \text{ord-res-}\gamma\text{-invars-def}$ )

**have**  $\text{clauses-lt-}D\text{-almost-defined}: \forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$

$(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{\#L \in \# C. L$

$\neq K\#\})$

**using invar**[ $\text{unfolded } \langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$ ]  $\text{clause-almost-defined-if-lt-next-clause}$

**by simp**

**have**  $\Gamma\text{-consistent}: \text{trail-consistent } \Gamma$

**using**  $\text{trail-consistent-if-sorted-wrt-atoms } \Gamma\text{-sorted by metis}$

**have**  $K \in \# D$  **and**  $\text{lit-in-}D\text{-le-}K: \bigwedge L. L \in \# D \implies L \preceq_l K$

**using**  $\langle \text{ord-res.is-maximal-lit } K D \rangle$

**unfolding**  $\text{atomize-imp atomize-all atomize-conj linorder-lit.is-maximal-in-mset-iff}$

**using** *linorder-lit.leI* **by** *blast*

**hence** *atm-of K |∈| atms-of-cls (N |∪| U<sub>er</sub>)*  
**using** *D-in atm-of-in-atms-of-clsI* **by** *metis*

**have** *trail-atms-le: ∀ A |∈| trail-atms Γ. A ≼<sub>t</sub> atm-of K*  
**using** *trail-atms-le0 ‹ord-res.is-maximal-lit K D›*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

**have** *trail-defined-cls Γ {#Ka ∈# D. Ka ≠ K#}*  
**using** *step-hyps(4,5,6) D-in* **by** (*metis clause-almost-definedI*)

**show** *?thesis*  
**unfolding** *‹s' = (U<sub>er</sub>, F, Γ, C')›*  
**proof** (*intro ord-res-7-invars.intros*)  
**show** *F |⊆| N |∪| U<sub>er</sub>*  
**using** *F-subset* .

**show** *∀ C'. C' = Some C' → C' |∈| iefac F |↑| (N |∪| U<sub>er</sub>)*  
**using** *step-hyps(7)* **by** (*metis linorder-cls.is-least-in-ffilter-iff Some-eq-The-optionalD*)

**have** *trail-true-cls Γ {#K ∈# C. K ≠ L<sub>C</sub>#} ∧ is-pos L<sub>C</sub>*  
**if** *C' = Some E* **and**  
*C-in: C |∈| iefac F |↑| (N |∪| U<sub>er</sub>)* **and**  
*C-lt: C ≺<sub>c</sub> E* **and**  
*C-max-lit: linorder-lit.is-maximal-in-mset C L<sub>C</sub>* **and**  
*L<sub>C</sub>-undef: ¬ trail-defined-lit Γ L<sub>C</sub>*  
**for** *E C L<sub>C</sub>*

**proof** –  
**have** *linorder-cls.is-least-in-fset (ffilter ((≺<sub>c</sub>) D) (iefac F |↑| (N |∪| U<sub>er</sub>))) E*  
**using** *‹C' = Some E› step-hyps* **by** (*metis Some-eq-The-optionalD*)  
**hence** *C ≺<sub>c</sub> D ∨ C = D*  
**unfolding** *linorder-cls.is-least-in-ffilter-iff*  
**using** *C-lt* **by** (*metis C-in linorder-cls.not-less-iff-gr-or-eq*)  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume** *C ≺<sub>c</sub> D*  
**thus** *?thesis*  
**using** *clauses-lt-D-seldomly-have-undef-max-lit[rule-format, OF C-in -*  
*C-max-lit L<sub>C</sub>-undef]*  
**by** *argo*

**next**  
**assume** *C = D*  
**hence** *L<sub>C</sub> = K*  
**using** *C-max-lit D-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)  
**hence** *False*  
**using** *L<sub>C</sub>-undef ‹trail-defined-lit Γ K›* **by** *argo*  
**thus** *?thesis ..*

**qed**  
**qed**

**thus**  $\forall D. C' = \text{Some } D \longrightarrow (\forall C \in |\text{iefac } \mathcal{F} \mid^\dagger (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. \text{ord-res.is-maximal-lit } L_C \ C \longrightarrow \neg \text{trail-defined-lit } \Gamma \ L_C \longrightarrow$   
 $\text{trail-true-cls } \Gamma \ \{\#K \in \# \ C. K \neq L_C \#\} \wedge \text{is-pos } L_C))$   
**by** *metis*

**have**  
 $\text{trail-true-cls } \Gamma \ C$   
 $\wedge K. \text{linorder-lit.is-maximal-in-mset } C \ K \implies$   
 $\neg (\exists A \in |\text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin |\text{trail-atms } \Gamma)$   
**if** *C-lt*:  $\wedge E. C' = \text{Some } E \implies C \prec_c E$  **and** *C-in*:  $C \in |\text{iefac } \mathcal{F} \mid^\dagger (N \mid \cup \mid$   
 $U_{er})$  **for** *C*  
**proof** –  
**have**  $C \preceq_c D$   
**using** *step-hyps that by (metis clause-le-if-lt-least-greater)*

**hence**  $C \prec_c D \vee C = D$   
**by** *simp*

**thus**  $\text{trail-true-cls } \Gamma \ C$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**thus**  $\text{trail-true-cls } \Gamma \ C$   
**using** *clauses-lt-D-true*  $\langle C \in |\text{iefac } \mathcal{F} \mid^\dagger (N \mid \cup \mid U_{er}) \rangle$  **by** *metis*

**next**  
**assume**  $C = D$

**have**  $\text{trail-defined-cls } \Gamma \ D$   
**using**  $\langle \text{trail-defined-lit } \Gamma \ K \rangle \langle \text{trail-defined-cls } \Gamma \ \{\#Ka \in \# \ D. Ka \neq K \#\} \rangle$   
**unfolding** *trail-defined-cls-def* **by** *auto*

**hence**  $\text{trail-true-cls } \Gamma \ D$   
**using**  $\Gamma$ -*consistent*  $\langle \neg \text{trail-false-cls } \Gamma \ D \rangle$   
**by** (*metis trail-true-or-false-cls-if-defined*)

**thus**  $\text{trail-true-cls } \Gamma \ C$   
**using**  $\langle C = D \rangle$  **by** *simp*

**qed**

**fix**  $K_c$   
**assume** *C-max-lit*:  $\text{linorder-lit.is-maximal-in-mset } C \ K_c$   
**thus**  $\neg (\exists A \in |\text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_c \wedge A \notin |\text{trail-atms}$   
 $\Gamma)$   
**using**  $\langle C \prec_c D \vee C = D \rangle$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**thus** *?thesis*

**using** *no-undef-atm-lt-max-lit-if-lt-D*  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   
*C-max-lit* **by** *metis*  
**next**  
**assume**  $C = D$   
**thus** *?thesis*  
**by** (*metis C-max-lit D-max-lit Uniq-D linorder-lit.Uniq-is-maximal-in-mset*  
*step-hyps(5)*)  
**qed**  
**qed**

**thus**  
 $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). (\forall D. C' = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow \text{trail-true-cls}$   
 $\Gamma \ C$   
 $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). (\forall D. C' = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K \ C \longrightarrow$   
 $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma))$   
**unfolding** *atomize-conj* **by** *metis*

**show** *sorted-wrt*  $(\lambda x \ y. \text{atm-of } (fst \ y) \prec_t \text{atm-of } (fst \ x)) \ \Gamma$   
**using**  $\Gamma$ -*sorted* .

**show** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) \ (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**using**  $\Gamma$ -*lower* .

**have**  $\text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er})$  **if**  $C' = \text{None}$   
**proof** (*intro fsubset-antisym*)  
**show**  $\text{trail-atms } \Gamma \mid \subseteq \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using**  $\Gamma$ -*lower* **unfolding** *linorder-trm.is-lower-set-iff* **by** *blast*  
**next**  
**have** *nbex-gt-D*:  $\neg (\exists E \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). D \prec_c E)$   
**using** *step-hyps*  $\langle C' = \text{None} \rangle$   
**by** (*metis clause-le-if-lt-least-greater linorder-clss.leD option.simps(3)*)

**have**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \text{atm-of } K \prec_t A)$   
**proof** (*intro notI , elim bexE*)  
**fix**  $A :: 'f \ \text{gterm}$   
**assume**  $A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$  **and**  $\text{atm-of } K \prec_t A$

**hence**  $A \mid \in \mid \text{atms-of-clss } (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))$   
**by** *simp*

**then obtain**  $E \ L$  **where** *E-in*:  $E \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**  $L \in \# \ E$   
**and**  $A = \text{atm-of } L$   
**unfolding** *atms-of-clss-def atm-of-clss-def*  
**by** (*smt (verit, del-insts) fimage-iff fmember-ffUnion-iff fset-fset-mset*)

**have**  $K \prec_l L$   
**using**  $\langle \text{atm-of } K \prec_t A \rangle \ \langle A = \text{atm-of } L \rangle$   
**by** (*cases K; cases L*) *simp-all*

**hence**  $D \prec_c E$   
**using**  $D\text{-max-lit } \langle L \in\# E \rangle$   
**by** (*metis empty-iff linorder-lit.ex-maximal-in-mset linorder-lit.is-maximal-in-mset-iff*  
*linorder-lit.less-linear linorder-lit.less-trans*  
*linorder-lit.mulp-if-maximal-less-that-maximal set-mset-empty*)

**thus**  $False$   
**using**  $E\text{-in nbex-gt-D by metis}$   
**qed**

**hence**  $\neg (\exists A | \in | \text{atms-of-clss } (N \cup | U_{er}). A | \notin | \text{trail-atms } \Gamma)$   
**using**  $\text{step-hyps}(5) \text{ step-hyps}(6)$   
**by** (*metis linorder-trm.linorder-cases*  
*trail-defined-lit-iff-trail-defined-atm*)

**then show**  $\text{atms-of-clss } (N \cup | U_{er}) | \subseteq | \text{trail-atms } \Gamma$   
**by**  $\text{blast}$   
**qed**

**thus**  $C' = None \longrightarrow \text{trail-atms } \Gamma = \text{atms-of-clss } (N \cup | U_{er})$   
**by**  $\text{metis}$

**show**  $\forall D. C' = \text{Some } D \longrightarrow (\forall A | \in | \text{trail-atms } \Gamma.$   
 $\exists L. \text{ord-res.is-maximal-lit } L D \wedge A \preceq_t \text{atm-of } L)$   
**proof** (*intro allI impI ballI*)  
**fix**  $E :: 'f \text{ gterm literal multiset and } A :: 'f \text{ gterm}$   
**assume**  $C' = \text{Some } E \text{ and } A | \in | \text{trail-atms } \Gamma$

**have**  $\text{linorder-cls.is-least-in-fset } (\text{ffilter } ((\prec_c) D) (\text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}))) E$   
**using**  $\text{step-hyps}(7) \langle C' = \text{Some } E \rangle \text{ by } (\text{metis } \text{Some-eq-The-optionalD})$

**hence**  $D \prec_c E$   
**unfolding**  $\text{linorder-cls.is-least-in-ffilter-iff by argo}$

**obtain**  $L \text{ where } \text{linorder-lit.is-maximal-in-mset } E L$   
**by** (*metis*  $\langle D \prec_c E \rangle \text{linorder-cls.leD linorder-lit.ex-maximal-in-mset mempty-lesseq-cls}$ )

**show**  $\exists L. \text{ord-res.is-maximal-lit } L E \wedge A \preceq_t \text{atm-of } L$   
**proof** (*intro exI conjI*)  
**show**  $\text{ord-res.is-maximal-lit } L E$   
**using**  $\langle \text{ord-res.is-maximal-lit } L E \rangle .$   
**next**  
**have**  $K \preceq_l L$   
**using**  $\text{step-hyps}(4) \langle \text{ord-res.is-maximal-lit } L E \rangle$   
**by** (*metis*  $\langle D \prec_c E \rangle \text{linorder-cls.less-asym linorder-lit.leI}$   
*linorder-lit.mulp-if-maximal-less-that-maximal*)

**hence**  $\text{atm-of } K \preceq_t \text{atm-of } L$

**by** (*cases K; cases L*) *simp-all*  
**moreover have**  $A \preceq_t \text{atm-of } K$   
**using**  $\langle A \mid \in \mid \text{trail-atms } \Gamma \rangle \text{trail-atms-le}$  **by** *blast*  
  
**ultimately show**  $A \preceq_t \text{atm-of } L$   
**by** *order*  
**qed**  
**qed**

**show**  $\forall Ln \in \text{set } \Gamma. \text{is-neg } (\text{fst } Ln) = (\text{snd } Ln = \text{None})$   
**using**  $\Gamma\text{-deci-iff-neg}$  **by** *metis*

**show**  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \{\#\text{fst } Ln\#\} \longrightarrow \text{trail-true-cls } \Gamma C)$   
**using**  $\Gamma\text{-deci-ball-lt-true}$  .

**show**  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\Gamma\text{-prop-in}$  **by** *simp*

**show**  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset}$   
 $C (\text{fst } Ln)$   
**using**  $\Gamma\text{-prop-greatest}$  **by** *simp*

**show**  $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{\#K$   
 $\in \# C. K \neq L\#\}$   
**using**  $\Gamma\text{-prop-almost-false}$  .

**show**  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$   
**using**  $\Gamma\text{-prop-ball-lt-true}$  .

**qed**  
**next**

**case** *step-hyps*: (*skip-undefined-neg*  $\Gamma D K U_{er} \Gamma' C' \mathcal{F}$ )

**note**  $D\text{-max-lit} = \langle \text{ord-res.is-maximal-lit } K D \rangle$

**have** *suffix*  $\Gamma \Gamma'$   
**using** *step-hyps unfolding suffix-def* **by** *simp*

**have**  
 $\mathcal{F}\text{-subset}$ :  $\mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er}$  **and**  
 $D\text{-in}$ :  $D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $\text{clauses-lt-D-seldomly-have-undef-max-lit}$ :  
 $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in \# C. K \neq L_C\#\} \wedge$   
 $\text{is-pos } L_C))$  **and**  
 $\text{clauses-lt-D-true}$ :  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma$

**$C$  and**  
*no-undef-atm-lt-max-lit-if-lt-D*:  $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma))$   
**and**  
 $\Gamma$ -sorted: *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\Gamma$ -lower: *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**and**  
*trail-atms-le0*:  $\forall A \in | \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L D \wedge (A \preceq_t$   
*atm-of*  $L)$  **and**  
 $\Gamma$ -deci-iff-neg:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$  **and**  
 $\Gamma$ -deci-ball-lt-true:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \{\#fst Ln\} \longrightarrow \text{trail-true-clc } \Gamma C)$   
**and**  
 $\Gamma$ -prop-in:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid$   
 $U_{er})$  **and**  
 $\Gamma$ -prop-greatest:  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
 $(fst Ln)$  **and**  
 $\Gamma$ -prop-almost-false:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-clc } \Gamma_0 \{\#K \in \# C.$   
 $K \neq L\# \}$  **and**  
 $\Gamma$ -prop-ball-lt-true:  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-clc } \Gamma_0 C))$   
**using invar by**  $(\text{simp-all add: } \langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle \text{ord-res-}\gamma\text{-invars-def})$   
  
**have** *clauses-lt-D-almost-defined*:  $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-clc } \Gamma \{\#L \in \# C. L$   
 $\neq K\# \})$   
**using invar**  $[\text{unfolded } \langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle]$  *clause-almost-defined-if-lt-next-clause*  
**by** *simp*  
  
**have**  $\Gamma$ -consistent: *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms*  $\Gamma$ -sorted **by** *metis*  
  
**have**  $K \in \# D$   
**using**  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *argo*  
  
**hence** *atm-of*  $K \in | \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using** *D-in atm-of-in-atms-of-clssI* **by** *metis*  
  
**have** *trail-atms-le*:  $\forall A \in | \text{trail-atms } \Gamma. A \preceq_t \text{atm-of } K$   
**using** *trail-atms-le0*  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**by**  $(\text{metis } \text{Uniq-D } \text{linorder-lit.Uniq-is-maximal-in-mset})$   
  
**have**  $\mathcal{F} \subseteq | N \mid \cup \mid U_{er}$   
**using** *F-subset* .

**moreover have**  $\forall C'. C' = \text{Some } C' \longrightarrow C' \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using** *step-hyps(9)* **by** (*metis linorder-cls.is-least-in-ffilter-iff Some-eq-The-optionalD*)

**moreover have** *trail-true-cls*  $\Gamma' \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$   
**if**  $C' = \text{Some } E$  **and**  
 $C\text{-in: } C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $C\text{-lt: } C \prec_c E$  **and**  
 $C\text{-max-lit: linorder-lit.is-maximal-in-mset } C L_C$  **and**  
 $L_C\text{-undef: } \neg \text{trail-defined-lit } \Gamma' L_C$   
**for**  $E C L_C$   
**proof** –  
**have** *linorder-cls.is-least-in-fset* (*ffilter* ( $((\prec_c) D)$  (*iefac*  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ )))  $E$   
**using**  $\langle C' = \text{Some } E \rangle$  *step-hyps* **by** (*metis Some-eq-The-optionalD*)  
**hence**  $C \prec_c D \vee C = D$   
**unfolding** *linorder-cls.is-least-in-ffilter-iff*  
**using**  $C\text{-lt}$  **by** (*metis C-in linorder-cls.not-less-iff-gr-or-eq*)  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$

**moreover have**  $\neg \text{trail-defined-lit } \Gamma L_C$   
**using**  $L_C\text{-undef}$   $\langle \text{suffix } \Gamma \Gamma' \rangle$   
**using** *trail-defined-lit-if-trail-defined-suffix* **by** *blast*

**ultimately have** *trail-true-cls*  $\Gamma \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$   
**using** *clauses-lt-D-seldomly-have-undef-max-lit*[*rule-format, OF C-in - C-max-lit*] **by** *metis*

**thus** *?thesis*  
**using**  $\langle \text{suffix } \Gamma \Gamma' \rangle$  *trail-true-cls-if-trail-true-suffix* **by** *blast*  
**next**  
**assume**  $C = D$   
**hence**  $L_C = K$   
**using**  $C\text{-max-lit } D\text{-max-lit}$   
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)  
**moreover have** *trail-defined-lit*  $\Gamma' K$   
**by** (*simp add: step-hyps(8) trail-defined-lit-def*)  
**ultimately have** *False*  
**using**  $L_C\text{-undef}$  **by** *argo*  
**thus** *?thesis ..*  
**qed**  
**qed**

**moreover have** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma'$   
**proof** –  
**have** *atm-of*  $K \notin | \text{trail-atms } \Gamma$   
**using**  $\langle \neg \text{trail-defined-lit } \Gamma K \rangle$   
**by** (*simp add: trail-defined-lit-iff-trail-defined-atm*)



**have**  $x \prec_t \text{atm-of } K$  **if**  $x\text{-in: } x \in | \text{trail-atms } \Gamma$  **for**  $x$   
**proof** –  
**have**  $x \preceq_t \text{atm-of } K$   
**using**  $x\text{-in trail-atms-le}$  **by** *metis*  
  
**moreover have**  $x \neq \text{atm-of } K$   
**using**  $x\text{-in } \langle \text{atm-of } K \notin | \text{trail-atms } \Gamma \rangle$  **by** *metis*  
  
**ultimately show** *?thesis*  
**by** *order*  
**qed**

**hence**  $\forall x \in \text{set } \Gamma. \text{atm-of } (fst\ x) \prec_t \text{atm-of } K$   
**by** (*simp add: fset-trail-atms*)

**thus** *?thesis*  
**using**  $\Gamma\text{-sorted } \langle \Gamma' = (K, None) \# \Gamma \rangle$  **by** *simp*  
**qed**

**moreover have**  $\Gamma'\text{-lower: linorder-trm.is-lower-fset } (trail\text{-atms } \Gamma') (atms\text{-of-clss } (N \cup | U_{er}))$   
**proof** –  
**have**  $linorder\text{-trm.is-lower-set } (insert\ (atm\text{-of } K) (fset\ (trail\text{-atms } \Gamma)))$   
 $(fset\ (atms\text{-of-clss } (N \cup | U_{er})))$   
**proof** (*rule linorder-trm.is-lower-set-insertI*)  
**show**  $atm\text{-of } K \in | atms\text{-of-clss } (N \cup | U_{er})$   
**using**  $\langle atm\text{-of } K \in | atms\text{-of-clss } (N \cup | U_{er}) \rangle$  .  
**next**  
**show**  $\forall w \in | atms\text{-of-clss } (N \cup | U_{er}). w \prec_t (atm\text{-of } K) \longrightarrow w \in | trail\text{-atms } \Gamma$   
**using** *step-hyps(5)[unfolded linorder-trm.is-least-in-filter-iff, simplified]*  
**by** *fastforce*  
**next**  
**show**  $linorder\text{-trm.is-lower-fset } (trail\text{-atms } \Gamma) (atms\text{-of-clss } (N \cup | U_{er}))$   
**using**  $\Gamma\text{-lower}$  .  
**qed**

**moreover have**  $trail\text{-atms } \Gamma' = finset\ (atm\text{-of } K) (trail\text{-atms } \Gamma)$   
**by** (*simp add: } \Gamma' = (K, None) \# \Gamma*)

**ultimately show** *?thesis*  
**by** *simp*  
**qed**

**moreover have**  $trail\text{-atms } \Gamma' = atms\text{-of-clss } (N \cup | U_{er})$  **if**  $C' = None$   
**proof** (*intro fsubset-antisym*)  
**show**  $trail\text{-atms } \Gamma' \subseteq | atms\text{-of-clss } (N \cup | U_{er})$   
**using**  $\Gamma'\text{-lower unfolding linorder-trm.is-lower-set-iff}$  **by** *blast*  
**next**

**have**  $nbex\text{-}gt\text{-}D: \neg (\exists E \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). D \prec_c E)$   
**using**  $step\text{-}hyps \langle \mathcal{C}' = None \rangle$   
**by** ( $metis \text{ clause-le-if-lt-least-greater linorder-cls.leD option.simps(3) }$ )

**have**  $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). \text{atm-of } K \prec_t A)$   
**proof** ( $intro \text{ notI } , \text{ elim } \text{ bexE}$ )  
**fix**  $A :: 'f \text{ gterm}$   
**assume**  $A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er})$  **and**  $\text{atm-of } K \prec_t A$

**hence**  $A \in | \text{atms-of-clss } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))$   
**by**  $simp$

**then obtain**  $E \ L$  **where**  $E\text{-in}: E \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $L \in \# \ E$   
**and**  $A = \text{atm-of } L$   
**unfolding**  $\text{atms-of-clss-def atms-of-cl-def}$   
**by** ( $smt (\text{verit}, \text{del-insts}) \text{ fimage-iff fmember-ffUnion-iff fset-fset-mset}$ )

**have**  $K \prec_l L$   
**using**  $\langle \text{atm-of } K \prec_t A \rangle \langle A = \text{atm-of } L \rangle$   
**by** ( $\text{cases } K; \text{cases } L \rangle \text{ simp-all}$ )

**hence**  $D \prec_c E$   
**using**  $D\text{-max-lit } \langle L \in \# \ E \rangle$   
**by** ( $metis \text{ empty-iff linorder-lit.ex-maximal-in-mset linorder-lit.is-maximal-in-mset-iff linorder-lit.less-linear linorder-lit.less-trans linorder-lit.mulp-if-maximal-less-that-maximal set-mset-empty}$ )

**thus**  $False$   
**using**  $E\text{-in } nbex\text{-}gt\text{-}D$  **by**  $metis$   
**qed**

**hence**  $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \notin | \text{trail-atms } \Gamma')$   
**using**  $step\text{-}hyps$   
**by** ( $metis \text{ finsert-iff linorder-trm.antisym-conv3 prod.sel(1) trail-atms.simps(2) }$ )

**then show**  $\text{atms-of-clss } (N \mid \cup \mid U_{er}) \subseteq | \text{trail-atms } \Gamma'$   
**by**  $blast$   
**qed**

**moreover have**  $\forall D. \mathcal{C}' = \text{Some } D \longrightarrow (\forall A \in | \text{trail-atms } \Gamma'. \exists L. \text{ord-res.is-maximal-lit } L \ D \wedge A \preceq_t \text{atm-of } L)$   
**proof** ( $intro \text{ allI impI ballI}$ )  
**fix**  $E :: 'f \text{ gterm literal multiset}$  **and**  $A :: 'f \text{ gterm}$   
**assume**  $\mathcal{C}' = \text{Some } E$  **and**  $A \in | \text{trail-atms } \Gamma'$

**have**  $\text{linorder-cls.is-least-in-fset } (\text{ffilter } ((\prec_c) \ D) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ E$   
**using**  $step\text{-}hyps(9) \langle \mathcal{C}' = \text{Some } E \rangle$  **by** ( $metis \text{ Some-eq-The-optionalD}$ )

**hence**  $D \prec_c E$

**unfolding** *linorder-cls.is-least-in-filter-iff* **by** *argo*

**obtain**  $L$  **where** *linorder-lit.is-maximal-in-mset*  $E$   $L$   
**by** (*metis*  $\langle D \prec_c E \rangle$  *linorder-cls.leD* *linorder-lit.ex-maximal-in-mset* *mempty-lesseq-cls*)

**show**  $\exists L. \text{ord-res.is-maximal-lit } L \ E \wedge A \preceq_t \text{atm-of } L$

**proof** (*intro exI conjI*)

**show** *ord-res.is-maximal-lit*  $L$   $E$

**using**  $\langle \text{ord-res.is-maximal-lit } L \ E \rangle$  .

**next**

**have**  $K \preceq_l L$

**using** *step-hyps*(4)  $\langle \text{ord-res.is-maximal-lit } L \ E \rangle$

**by** (*metis*  $\langle D \prec_c E \rangle$  *linorder-cls.less-asym* *linorder-lit.leI*  
*linorder-lit.mulp-if-maximal-less-that-maximal*)

**hence** *atm-of*  $K \preceq_t \text{atm-of } L$

**by** (*cases*  $K$ ; *cases*  $L$ ) *simp-all*

**moreover have**  $A \preceq_t \text{atm-of } K$

**proof** –

**have**  $A = \text{atm-of } K \vee A \mid \in \mid \text{trail-atms } \Gamma$

**using**  $\langle A \mid \in \mid \text{trail-atms } \Gamma \rangle$

**unfolding**  $\langle \Gamma' = (K, \text{None}) \# \Gamma \rangle$

**by** (*metis* (*mono-tags*, *lifting*) *fininsertE* *prod.sel*(1) *trail-atms.simps*(2))

**thus**  $A \preceq_t \text{atm-of } K$

**using** *trail-atms-le* **by** *blast*

**qed**

**ultimately show**  $A \preceq_t \text{atm-of } L$

**by** *order*

**qed**

**qed**

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$

**unfolding**  $\langle \Gamma' = (K, \text{None}) \# \Gamma \rangle$

**using**  $\Gamma\text{-deci-iff-neg}$   $\langle \text{is-neg } K \rangle$  **by** *simp*

**moreover have** *trail-true-cls*  $\Gamma' \ C$

**if**  $Ln \in \text{set } \Gamma'$  **and**  $\text{snd } Ln = \text{None}$  **and**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **and**  $C \prec_c \{ \# \text{fst } Ln \# \}$

**for**  $Ln \ C$

**proof** –

**have**  $Ln = (K, \text{None}) \vee Ln \in \text{set } \Gamma$

**using**  $\langle Ln \in \text{set } \Gamma' \rangle$  **unfolding**  $\langle \Gamma' = (K, \text{None}) \# \Gamma \rangle$  **by** *simp*

**hence** *trail-true-cls*  $\Gamma \ C$

**proof** (*elim disjE*)

**assume**  $Ln = (K, \text{None})$

**hence**  $\forall x \in \# C. x \prec_l K$   
**using**  $\langle C \prec_c \{\#fst Ln\# \} \rangle$   
**unfolding** *multp-singleton-right*[*OF linorder-lit.transp-on-less*]  
**by** *simp*

**hence**  $C \prec_c D$   
**using** *D-max-lit*  
**by** (*metis*  $\langle K \in \# D \rangle$  *empty-iff ord-res.multp-if-all-left-smaller set-mset-empty*)

**thus** *trail-true-cls*  $\Gamma C$   
**using**  $\langle C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$   
**using** *clauses-lt-D-true* **by** *blast*

**next**  
**assume**  $Ln \in set \Gamma$

**thus** *trail-true-cls*  $\Gamma C$   
**using**  $\Gamma$ -*deci-ball-lt-true*  $\langle snd Ln = None \rangle \langle C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$   
 $\langle C \prec_c \{\#fst Ln\# \} \rangle$   
**by** *metis*  
**qed**

**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle suffix \Gamma \Gamma' \rangle$  **by** (*metis* *trail-true-cls-if-trail-true-suffix*)  
**qed**

**moreover have**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$   
**using**  $\Gamma$ -*prop-in*  $\langle \Gamma' = (K, None) \# \Gamma \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow linorder-lit.is-greatest-in-mset C (fst Ln)$   
**using**  $\Gamma$ -*prop-greatest*  $\langle \Gamma' = (K, None) \# \Gamma \rangle$  **by** *simp*

**moreover have**  $\forall \Gamma_1 L C \Gamma_0. \Gamma' = \Gamma_1 @ (L, Some C) \# \Gamma_0 \longrightarrow trail-false-cls \Gamma_0 \{\#K \in \# C. K \neq L\# \}$   
**unfolding**  $\langle \Gamma' = (K, None) \# \Gamma \rangle$  **using**  $\Gamma$ -*prop-almost-false*  
**by** (*metis* (*no-types, lifting*) *append-eq-Cons-conv list.inject option.discI prod.inject*)

**moreover have**  $(\forall \Gamma_1 L D \Gamma_0. \Gamma' = \Gamma_1 @ (L, Some D) \# \Gamma_0 \longrightarrow (\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow trail-true-cls \Gamma_0 C))$   
**unfolding**  $\langle \Gamma' = (K, None) \# \Gamma \rangle$  **using**  $\Gamma$ -*prop-ball-lt-true*  
**by** (*smt* (*verit, ccfv-SIG*) *append-eq-Cons-conv list.inject option.discI prod.inject*)

**ultimately show** *?thesis*  
**unfolding**  $\langle s' = (U_{er}, \mathcal{F}, \Gamma', C') \rangle$   
**proof** (*intro ord-res-7-invars.intros*)  
**have** *trail-true-cls*  $\Gamma' C$   
 $\wedge K_C. linorder-lit.is-maximal-in-mset C K_C \implies$

$\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \mid \notin \mid \text{trail-atms } \Gamma')$   
**if**  $C\text{-lt}: \bigwedge E. C' = \text{Some } E \longrightarrow C \prec_c E$  **and**  $C\text{-in}: C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **for**  $C$   
**proof** –  
**have**  $C \preceq_c D$   
**using** *step-hyps that by (metis clause-le-if-lt-least-greater)*  
  
**hence**  $C \prec_c D \vee C = D$   
**by** *simp*  
  
**thus** *trail-true-cls*  $\Gamma' C$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
  
**hence** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true*  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$  **by** *metis*  
  
**thus** *trail-true-cls*  $\Gamma' C$   
**using** *suffix*  $\Gamma \Gamma'$  **by** (*metis trail-true-cls-if-trail-true-suffix*)  
**next**  
**assume**  $C = D$   
  
**have** *trail-true-lit*  $\Gamma' K$   
**unfolding**  $\langle \Gamma' = (K, \text{None}) \# \Gamma \rangle$  *trail-true-lit-def* **by** *simp*  
  
**thus** *trail-true-cls*  $\Gamma' C$   
**unfolding**  $\langle C = D \rangle$  *trail-true-cls-def*  
**using**  $\langle K \in \# D \rangle$  **by** *metis*  
**qed**  
  
**fix**  $K_C$   
**assume**  $C\text{-max-lit}: \text{linorder-lit.is-maximal-in-mset } C K_C$   
**show**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \mid \notin \mid \text{trail-atms } \Gamma')$   
**using**  $\langle C \prec_c D \vee C = D \rangle$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
  
**hence**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \mid \notin \mid \text{trail-atms } \Gamma)$   
**using** *no-undef-atm-lt-max-lit-if-lt-D*  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   
 $C\text{-max-lit}$  **by** *metis*  
  
**thus**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \mid \notin \mid \text{trail-atms } \Gamma')$   
**by** (*metis finsert-iff step-hyps(8) trail-atms.simps(2)*)  
**next**  
**assume**  $C = D$   
**thus**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \mid \notin \mid \text{trail-atms } \Gamma')$

$\Gamma'$   
**by** (*metis* (*mono-tags*, *lifting*) *C-max-lit D-max-lit Uniq-D  $\Gamma'$ -lower finsert-iff linorder-lit.Uniq-is-maximal-in-mset linorder-trm.not-in-lower-setI prod.sel(1)* *step-hyps(8) trail-atms.simps(2)*)  
**qed**  
**qed**  
  
**thus**  
 $\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). (\forall D. C' = Some D \longrightarrow C \prec_c D) \longrightarrow trail\text{-}true\text{-}cls$   
 $\Gamma' C$   
 $\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). (\forall D. C' = Some D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. ord\text{-}res.is\text{-}maximal\text{-}lit K C \longrightarrow$   
 $\neg (\exists A \in |atms\text{-}of\text{-}clss (N \cup U_{er}). A \prec_t atm\text{-}of K \wedge A \notin |trail\text{-}atms \Gamma'))$   
**unfolding** *atomize-conj* **by** *metis*  
**qed** *simp-all*  
**next**  
**case** *step-hyps: (skip-undefined-pos  $\Gamma D K U_{er} \mathcal{F} E$ )*  
  
**note** *D-max-lit =  $\langle ord\text{-}res.is\text{-}maximal\text{-}lit K D \rangle$*   
**note** *E-least =  $\langle linorder\text{-}cls.is\text{-}least\text{-}in\text{-}fset (ffilter ((\prec_c) D) (iefac \mathcal{F} | \uparrow (N \cup U_{er}))) E \rangle$*   
  
**have**  $D \prec_c E$   
**using** *E-least* **unfolding** *linorder-cls.is-least-in-ffilter-iff* **by** *argo*  
  
**have**  
 *$\mathcal{F}$ -subset:  $\mathcal{F} \subseteq | N \cup U_{er}$  and*  
 *$D$ -in:  $D \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$  and*  
*clauses-lt-D-seldomly-have-undef-max-lit:*  
 $\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset C L_C \longrightarrow$   
 $\neg trail\text{-}defined\text{-}lit \Gamma L_C \longrightarrow (trail\text{-}true\text{-}cls \Gamma \{\#K \in \# C. K \neq L_C\# \} \wedge$   
*is-pos  $L_C$ )) and*  
*clauses-lt-D-true:  $\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow trail\text{-}true\text{-}cls \Gamma$*   
*C and*  
*no-undef-atm-lt-max-lit-if-lt-D:  $\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow$*   
 $(\forall K. linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset C K \longrightarrow$   
 $\neg (\exists A \in |atms\text{-}of\text{-}clss (N \cup U_{er}). A \prec_t atm\text{-}of K \wedge A \notin |trail\text{-}atms \Gamma))$   
**and**  
 $\Gamma$ -sorted: *sorted-wrt  $(\lambda x y. atm\text{-}of (fst y) \prec_t atm\text{-}of (fst x)) \Gamma$  and*  
 $\Gamma$ -lower: *linorder-trm.is-lower-fset  $(trail\text{-}atms \Gamma) (atms\text{-}of\text{-}clss (N \cup U_{er}))$*   
**and**  
*trail-atms-le0:  $\forall A \in |trail\text{-}atms \Gamma. \exists L. ord\text{-}res.is\text{-}maximal\text{-}lit L D \wedge (A \preceq_t$*   
*atm-of L) and*  
 $\Gamma$ -deci-iff-neg:  *$\forall Ln \in set \Gamma. snd Ln = None \longleftrightarrow is\text{-}neg (fst Ln)$  and*  
 $\Gamma$ -deci-ball-lt-true:  *$\forall Ln \in set \Gamma. snd Ln = None \longrightarrow$*   
 $(\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c \{\#fst Ln\# \} \longrightarrow trail\text{-}true\text{-}cls \Gamma C)$   
**and**

$\Gamma$ -prop-in:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \mid^{\uparrow} (N \mid \cup \mid U_{er})$  **and**  
 $\Gamma$ -prop-greatest:  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
(fst Ln) **and**  
 $\Gamma$ -prop-almost-false:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{ \#K \in \# C. K \neq L \# \}$  **and**  
 $\Gamma$ -prop-ball-lt-true:  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \in | \text{iefac } \mathcal{F} \mid^{\uparrow} (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$   
**using invar by** (*simp-all add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle \text{ord-res-}\gamma\text{-invars-def}$ )  
  
**have** *clauses-lt-D-almost-defined*:  $\forall C \in | \text{iefac } \mathcal{F} \mid^{\uparrow} (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{ \#L \in \# C. L \neq K \# \})$   
**using invar**[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*  
  
**have**  $\Gamma$ -consistent: *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms*  $\Gamma$ -sorted **by** *metis*  
  
**have**  $K \in \# D$   
**using**  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *argo*  
  
**hence** *atm-of*  $K \in | \text{atms-of-cls } (N \mid \cup \mid U_{er})$   
**using** *D-in atm-of-in-atms-of-clsI* **by** *metis*  
  
**have** *trail-defined-cls*  $\Gamma \{ \#L \in \# D. L \neq K \wedge L \neq - K \# \}$   
**using** *clause-almost-almost-definedI*[*OF D-in step-hyps(4,5)*] .  
  
**moreover have**  $- K \notin \# D$   
**using**  $\langle \text{is-pos } K \rangle$  *D-max-lit*  
**by** (*metis* (*no-types*, *opaque-lifting*) *is-pos-def linorder-lit.antisym-conv3*  
*linorder-lit.is-maximal-in-mset-iff linorder-trm.less-imp-not-eq ord-res.less-lit-simps(4)*  
*uminus-Pos uminus-not-id*)  
  
**ultimately have** *D-almost-defined*: *trail-defined-cls*  $\Gamma \{ \#L \in \# D. L \neq K \# \}$   
**unfolding** *trail-defined-cls-def* **by** *auto*  
  
**hence** *trail-true-cls*  $\Gamma \{ \#L \in \# D. L \neq K \# \}$   
**using**  $\langle \neg \text{trail-false-cls } \Gamma \{ \#L \in \# D. L \neq K \# \} \rangle$   
**using** *trail-true-or-false-cls-if-defined* **by** *metis*  
  
**hence** *D-true*: *trail-true-cls*  $\Gamma D$   
**unfolding** *trail-true-cls-def* **by** *auto*  
  
**have** *trail-atms-le*:  $\forall A \in | \text{trail-atms } \Gamma. A \preceq_t \text{atm-of } K$   
**using** *trail-atms-le0 D-max-lit*

by (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

**obtain**  $L$  **where**  $E$ -*max-lit*: *linorder-lit.is-maximal-in-mset*  $E$   $L$   
**by** (*metis*  $\langle D \prec_c E \rangle$  *linorder-cls.leD linorder-lit.ex-maximal-in-mset mempty-lesseq-cls*)

**have**  $\mathcal{F} \mid\subseteq\mid N \mid\cup\mid U_{er}$   
**using**  *$\mathcal{F}$ -subset* .

**moreover have**  $\forall C'$ . *Some*  $E = \text{Some } C' \longrightarrow C' \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$   
**using** *step-hyps unfolding linorder-cls.is-least-in-ffilter-iff* **by** *simp*

**moreover have** *trail-true-cl*s  $\Gamma \{ \#K \in\# C. K \neq L_C\# \} \wedge \text{is-pos } L_C$   
**if** *Some*  $E = \text{Some } E'$  **and**  
 $C$ -*in*:  $C \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$  **and**  
 $C$ -*lt*:  $C \prec_c E'$  **and**  
 $C$ -*max-lit*: *linorder-lit.is-maximal-in-mset*  $C$   $L_C$  **and**  
 $L_C$ -*undef*:  $\neg$  *trail-defined-lit*  $\Gamma L_C$   
**for**  $E' C L_C$

**proof** –  
**have**  $E' = E$   
**using** *that by simp*  
**hence** *linorder-cls.is-least-in-fset* (*ffilter* ( $(\prec_c) D$ ) (*iefac*  $\mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er})$ ))  $E$   
**using** *step-hyps by metis*  
**hence**  $C \prec_c D \vee C = D$   
**unfolding** *linorder-cls.is-least-in-ffilter-iff*  
**using**  $C$ -*lt*  $\langle E' = E \rangle$  **by** (*metis C-in linorder-cls.not-less-iff-gr-or-eq*)  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**thus** *trail-true-cl*s  $\Gamma \{ \#K \in\# C. K \neq L_C\# \} \wedge \text{is-pos } L_C$   
**using** *clauses-lt-D-seldomly-have-undef-max-lit*[*rule-format*, *OF C-in - C-max-lit L\_C-undef*]  
**by** *metis*  
**next**  
**assume**  $C = D$   
**hence**  $L_C = K$   
**using**  $C$ -*max-lit D-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)  
**thus** *?thesis*  
**using**  $\langle C = D \rangle \langle \text{trail-true-cl} \text{ } \Gamma \{ \#L \in\# D. L \neq K\# \} \rangle \langle \text{is-pos } K \rangle$  **by** *metis*  
**qed**  
**qed**

**moreover have** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$   
**using**  $\Gamma$ -*sorted* .

**moreover have** *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-clss*  $(N \mid\cup\mid U_{er})$ )  
**using**  $\Gamma$ -*lower* .



**moreover have**  $\forall D. \text{Some } E = \text{Some } D \longrightarrow (\forall A \mid \in \mid \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L \ D \wedge A \preceq_t \text{atm-of } L)$   
**proof** –  
**have**  $\forall A \mid \in \mid \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L \ E \wedge A \preceq_t \text{atm-of } L$   
**proof** (*intro ballI*)  
**fix**  $A :: 'f \text{ gterm}$   
**assume**  $A \mid \in \mid \text{trail-atms } \Gamma$   
**show**  $\exists L. \text{ord-res.is-maximal-lit } L \ E \wedge A \preceq_t \text{atm-of } L$   
**proof** (*intro exI conjI*)  
**show**  $\text{ord-res.is-maximal-lit } L \ E$   
**using**  $E\text{-max-lit}$  .  
**next**  
**have**  $K \preceq_l L$   
**using**  $D\text{-max-lit } E\text{-max-lit}$   
**by** (*metis*  $\langle D \prec_c E \rangle \text{linorder-cls.less-asm linorder-lit.leI linorder-lit.mulp-if-maximal-less-that-maximal}$ )  
  
**hence**  $\text{atm-of } K \preceq_t \text{atm-of } L$   
**by** (*cases*  $K$ ; *cases*  $L$ ) *simp-all*  
  
**moreover have**  $A \preceq_t \text{atm-of } K$   
**using**  $\langle A \mid \in \mid \text{trail-atms } \Gamma \rangle \text{trail-atms-le}$  **by** *metis*  
  
**ultimately show**  $A \preceq_t \text{atm-of } L$   
**by** *order*  
**qed**  
**qed**  
  
**thus** *?thesis*  
**by** *simp*  
**qed**  
  
**moreover have**  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$   
**using**  $\Gamma\text{-deci-iff-neg}$  .  
  
**moreover have**  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow (\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \{\#\text{fst } Ln\# \} \longrightarrow \text{trail-true-cls } \Gamma \ C)$   
**using**  $\Gamma\text{-deci-ball-lt-true}$  .  
  
**moreover have**  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\Gamma\text{-prop-in}$  .  
  
**moreover have**  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C \ (\text{fst } Ln)$   
**using**  $\Gamma\text{-prop-greatest}$  .  
  
**moreover have**  $\forall \Gamma_1 \ L \ C \ \Gamma_0. \Gamma = \Gamma_1 \ @ \ (L, \text{Some } C) \ # \ \Gamma_0 \longrightarrow \text{trail-false-cls}$

$\Gamma_0 \{ \#K \in \# C. K \neq L \# \}$   
**using**  $\Gamma$ -prop-almost-false .

**moreover have**  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$   
**using**  $\Gamma$ -prop-ball-lt-true .

**ultimately show** *?thesis*  
**unfolding**  $\langle s' = (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) \rangle$   
**proof** (*intro ord-res-7-invars.intros*)  
**have** *trail-true-cls*  $\Gamma C$   
 $\wedge K_C. \text{linorder-lit.is-maximal-in-mset } C K_C \implies$   
 $\neg (\exists A | \in | \text{atms-of-clss } (N | \cup | U_{er}). A \prec_t \text{atm-of } K_C \wedge A | \notin | \text{trail-atms } \Gamma)$   
**if** *C-lt*:  $\wedge E'. \text{Some } E = \text{Some } E' \implies C \prec_c E'$  **and** *C-in*:  $C | \in | \text{iefac } \mathcal{F} | \uparrow$   
 $(N | \cup | U_{er})$  **for**  $C$   
**proof** –  
**have**  $C \prec_c E$   
**using** *C-lt* **by** *simp*

**hence**  $C \prec_c D \vee C = D$   
**using** *E-least C-in*  
**by** (*metis linorder-clss.is-least-in-ffilter-iff linorder-clss.less-imp-triv*  
*linorder-clss.linorder-cases*)

**thus** *trail-true-cls*  $\Gamma C$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**thus** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true*  $\langle C | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \rangle$  **by** *metis*

**next**  
**assume**  $C = D$   
**thus** *trail-true-cls*  $\Gamma C$   
**using** *D-true* **by** *argo*

**qed**

**fix**  $K_C$   
**assume** *C-max-lit*: *linorder-lit.is-maximal-in-mset*  $C K_C$   
**show**  $\neg (\exists A | \in | \text{atms-of-clss } (N | \cup | U_{er}). A \prec_t \text{atm-of } K_C \wedge A | \notin | \text{trail-atms}$   
 $\Gamma)$   
**using**  $\langle C \prec_c D \vee C = D \rangle$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**thus**  $\neg (\exists A | \in | \text{atms-of-clss } (N | \cup | U_{er}). A \prec_t \text{atm-of } K_C \wedge A | \notin | \text{trail-atms}$   
 $\Gamma)$   
**using** *no-undef-atm-lt-max-lit-if-lt-D*  $\langle C | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \rangle$   
*C-max-lit* **by** *metis*

**next**  
**assume**  $C = D$   
**thus**  $\neg (\exists A | \in | \text{atms-of-clss } (N | \cup | U_{er}). A \prec_t \text{atm-of } K_C \wedge A | \notin | \text{trail-atms}$

$\Gamma$ )  
**by** (*metis C-max-lit D-max-lit Uniq-D linorder-lit. Uniq-is-maximal-in-mset step-hyps(5)*)  
**qed**  
**qed**

**thus**  
 $\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). (\forall D. \text{Some } E = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
*trail-true-cls*  $\Gamma C$   
 $\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). (\forall D. \text{Some } E = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K C \longrightarrow$   
 $\neg (\exists A \in |atms-of-cls (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin |trail-atms \Gamma))$   
**unfolding** *atomize-conj* **by** *metis*  
**qed** *simp-all*

**next**  
**case** *step-hyps: (skip-undefined-pos-ultimate*  $\Gamma D K U_{er} \Gamma' \mathcal{F}$ *)*

**note** *D-max-lit = <ord-res.is-maximal-lit K D>*

**have** *suffix*  $\Gamma \Gamma'$   
**using**  $\langle \Gamma' = (- K, None) \# \Gamma \rangle$  **by** (*simp add: suffix-def*)

**have**  
*F-subset:  $\mathcal{F} \subseteq | N \cup U_{er}$  and*  
*D-in:  $D \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$  and*  
*clauses-lt-D-seldomly-have-undef-max-lit:*  
 $\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in \# C. K \neq L_C\} \wedge$   
*is-pos*  $L_C))$  **and**  
*clauses-lt-D-true:  $\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma C$  and*  
*no-undef-atm-lt-max-lit-if-lt-D:  $\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c D \longrightarrow$*   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \in |atms-of-cls (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin |trail-atms \Gamma))$   
**and**  
 $\Gamma$ -*sorted: sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\Gamma$ -*lower: linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-cls } (N \cup U_{er}))$   
**and**  
*trail-atms-le0:  $\forall A \in |trail-atms \Gamma. \exists L. \text{ord-res.is-maximal-lit } L D \wedge (A \preceq_t$*   
*atm-of*  $L)$  **and**  
 $\Gamma$ -*deci-iff-neg:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = None \longleftrightarrow \text{is-neg } (fst Ln)$  and*  
 $\Gamma$ -*deci-ball-lt-true:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = None \longrightarrow$*   
 $(\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c \{\#fst Ln\} \longrightarrow \text{trail-true-cls } \Gamma C)$   
**and**  
 $\Gamma$ -*prop-in:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$  and*  
 $\Gamma$ -*prop-greatest:*  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$

(*fst Ln*) **and**  
 $\Gamma$ -prop-almost-false:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{ \#K \in \# C. K \neq L \# \}$  **and**  
 $\Gamma$ -prop-ball-lt-true: ( $\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C)$ )  
**using invar by** (*simp-all add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$  *ord-res-7-invars-def*)

**have** *clauses-lt-D-almost-defined*:  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{ \#L \in \# C. L \neq K \# \})$   
**using invar**[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*

**have**  $\Gamma$ -consistent: *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms*  $\Gamma$ -sorted **by** *metis*

**have**  $K \in \# D$   
**using**  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *argo*

**hence** *atm-of*  $K \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er})$   
**using** *D-in atm-of-in-atms-of-clsI* **by** *metis*

**have** *trail-defined-cls*  $\Gamma \{ \#L \in \# D. L \neq K \wedge L \neq - K \# \}$   
**using** *clause-almost-almost-definedI*[*OF D-in step-hyps(4,5)*] .

**moreover have**  $- K \notin \# D$   
**using**  $\langle \text{is-pos } K \rangle$  *D-max-lit*  
**by** (*metis* (*no-types*, *opaque-lifting*) *is-pos-def linorder-lit.antisym-conv3*  
*linorder-lit.is-maximal-in-mset-iff linorder-trm.less-imp-not-eq ord-res.less-lit-simps(4)*  
*uminus-Pos uminus-not-id*)

**ultimately have** *D-almost-defined*: *trail-defined-cls*  $\Gamma \{ \#L \in \# D. L \neq K \# \}$   
**unfolding** *trail-defined-cls-def* **by** *auto*

**hence** *trail-true-cls*  $\Gamma \{ \#L \in \# D. L \neq K \# \}$   
**using**  $\langle \neg \text{trail-false-cls } \Gamma \{ \#L \in \# D. L \neq K \# \} \rangle$   
**using** *trail-true-or-false-cls-if-defined* **by** *metis*

**hence** *D-true*: *trail-true-cls*  $\Gamma D$   
**unfolding** *trail-true-cls-def* **by** *auto*

**have** *trail-atms-le*:  $\forall A \mid \in \mid \text{trail-atms } \Gamma. A \preceq_t \text{atm-of } K$   
**using** *trail-atms-le0 D-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

**have**  $\mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er}$   
**using** *F-subset* .

**moreover have**  $\forall C'. \text{None} = \text{Some } C' \longrightarrow C' \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**by** *simp*

**moreover have** *trail-true-cls*  $\Gamma' \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$   
**if**  $\text{None} = \text{Some } E'$  **and**  
*C-in*:  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
*C-lt*:  $C \prec_c E'$  **and**  
*C-max-lit*: *linorder-lit.is-maximal-in-mset*  $C L_C$  **and**  
*L<sub>C</sub>-undef*:  $\neg \text{trail-defined-lit } \Gamma' L_C$   
**for**  $E' C L_C$   
**using** *that by simp*

**moreover have** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma'$   
**proof** –

**have**  $\forall x \in \text{set } \Gamma. \text{atm-of } (fst x) \prec_t \text{atm-of } K$

**by** (*metis image-eqI linorder-trm.less-linear linorder-trm.not-le step-hyps(6)*)

*trail-atms-le*

*trail-defined-lit-def trail-defined-lit-iff-trail-defined-atm*)

**thus** *?thesis*

**unfolding**  $\langle \Gamma' = (- K, \text{None}) \# \Gamma \rangle$

**using**  $\Gamma$ -*sorted by simp*

**qed**

**moreover have**  $\Gamma'$ -*lower*: *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma') (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$

**proof** –

**have** *atm-of*  $K \in | \text{atms-of-clss } (N \mid \cup \mid U_{er})$

**using**  $\langle \text{atm-of } K \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$ .

**moreover have**  $\forall w \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). w \prec_t \text{atm-of } K \longrightarrow w \in |$   
*trail-atms*  $\Gamma$

**using** *step-hyps(5) by blast*

**ultimately show** *?thesis*

**using**  $\Gamma$ -*lower by* (*simp add*:  $\langle \Gamma' = (- K, \text{None}) \# \Gamma \rangle$  *linorder-trm.is-lower-set-insertI*)

**qed**

**moreover have** *trail-atms*  $\Gamma' = \text{atms-of-clss } (N \mid \cup \mid U_{er})$

**proof** (*intro fsubset-antisym*)

**show** *trail-atms*  $\Gamma' \subseteq | \text{atms-of-clss } (N \mid \cup \mid U_{er})$

**using**  $\Gamma'$ -*lower unfolding linorder-trm.is-lower-set-iff by blast*

**next**

**have** *nbex-gt-D*:  $\neg (\exists E \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). D \prec_c E)$

**using** *step-hyps by argo*

**have**  $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). \text{atm-of } K \prec_t A)$

**proof** (*intro notI , elim bexE*)

**fix**  $A :: 'f \text{ gterm}$   
**assume**  $A \in | \text{atms-of-clss } (N \cup | U_{er})$  **and**  $\text{atm-of } K \prec_t A$

**hence**  $A \in | \text{atms-of-clss } (\text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}))$   
**by** *simp*

**then obtain**  $E \ L$  **where**  $E\text{-in}: E \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er})$  **and**  $L \in \# E$   
**and**  $A = \text{atm-of } L$   
**unfolding** *atms-of-clss-def atms-of-cl-def*  
**by** (*smt (verit, del-insts) fimage-iff fmember-ffUnion-iff fset-fset-mset*)

**have**  $K \prec_l L$   
**using**  $\langle \text{atm-of } K \prec_t A \rangle \langle A = \text{atm-of } L \rangle$   
**by** (*cases K; cases L*) *simp-all*

**hence**  $D \prec_c E$   
**using** *D-max-lit*  $\langle L \in \# E \rangle$   
**by** (*metis empty-iff linorder-lit.ex-maximal-in-mset linorder-lit.is-maximal-in-mset-iff linorder-lit.less-linear linorder-lit.less-trans linorder-lit.multip-if-maximal-less-that-maximal set-mset-empty*)

**thus** *False*  
**using** *E-in nbex-gt-D* **by** *metis*  
**qed**

**hence**  $\neg (\exists A \in | \text{atms-of-clss } (N \cup | U_{er}). A \notin | \text{trail-atms } \Gamma')$   
**using** *step-hyps*  
**by** (*metis atm-of-uminus finsert-iff fst-conv linorder-trm.antisym-conv3 trail-atms.simps(2)*)

**then show**  $\text{atms-of-clss } (N \cup | U_{er}) \subseteq | \text{trail-atms } \Gamma'$   
**by** *blast*  
**qed**

**moreover have**  $\forall D. \text{None} = \text{Some } D \longrightarrow (\forall A \in | \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L \ D \wedge A \preceq_t \text{atm-of } L)$   
**by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$   
**using**  $\Gamma\text{-deci-iff-neg } \langle \text{is-pos } K \rangle$   
**by** (*simp add:*  $\langle \Gamma' = (- K, \text{None}) \# \Gamma \rangle$  *is-pos-neg-not-is-pos*)

**moreover have** *trail-true-cls*  $\Gamma' \ C$   
**if**  $Ln \in \text{set } \Gamma'$  **and**  $\text{snd } Ln = \text{None}$  **and**  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er})$  **and**  $C \prec_c \{\# \text{fst } Ln \# \}$   
**for**  $Ln \ C$   
**proof** –  
**have**  $Ln = (- K, \text{None}) \vee Ln \in \text{set } \Gamma$   
**using**  $\langle Ln \in \text{set } \Gamma' \rangle$  **unfolding**  $\langle \Gamma' = (- K, \text{None}) \# \Gamma \rangle$  **by** *simp*

**hence** *trail-true-cls*  $\Gamma$   $C$   
**proof** (*elim disjE*)  
**assume**  $Ln = (- K, None)$

**hence**  $\forall x \in \# C. x \prec_l - K$   
**using**  $\langle C \prec_c \{\#fst Ln\} \rangle$   
**unfolding** *multp-singleton-right*[*OF linorder-lit.transp-on-less*]  
**by** *simp*

**hence**  $C \preceq_c D$   
**using** *D-max-lit step-hyps*  
**using** *linorder-cls.leI that(3)* **by** *blast*

**thus** *trail-true-cls*  $\Gamma$   $C$   
**using**  $\langle C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$  *D-true*  
**using** *clauses-lt-D-true* **by** *blast*

**next**  
**assume**  $Ln \in set \Gamma$

**thus** *trail-true-cls*  $\Gamma$   $C$   
**using**  $\Gamma$ -*deci-ball-lt-true*  $\langle snd Ln = None \rangle$   $\langle C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$   
 $\langle C \prec_c \{\#fst Ln\} \rangle$   
**by** *metis*  
**qed**

**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle suffix \Gamma \Gamma' \rangle$  **by** (*metis trail-true-cls-if-trail-true-suffix*)  
**qed**

**moreover have**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$   
**using**  $\Gamma$ -*prop-in* **by** (*simp add:  $\langle \Gamma' = (- K, None) \# \Gamma \rangle$* )

**moreover have**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow linorder-lit.is-greatest-in-mset C (fst Ln)$   
**using**  $\Gamma$ -*prop-greatest* **by** (*simp add:  $\langle \Gamma' = (- K, None) \# \Gamma \rangle$* )

**moreover have**  $\forall \Gamma_1 L C \Gamma_0. \Gamma' = \Gamma_1 @ (L, Some C) \# \Gamma_0 \longrightarrow trail-false-cls \Gamma_0 \{\#K \in \# C. K \neq L\}$   
**using**  $\Gamma$ -*prop-almost-false*  
**unfolding**  $\langle \Gamma' = (- K, None) \# \Gamma \rangle$   
**by** (*metis (no-types, lifting) append-eq-Cons-conv list.inject option.discI prod.inject*)

**moreover have**  $(\forall \Gamma_1 L D \Gamma_0. \Gamma' = \Gamma_1 @ (L, Some D) \# \Gamma_0 \longrightarrow (\forall C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow trail-true-cls \Gamma_0 C))$   
**using**  $\Gamma$ -*prop-ball-lt-true*  
**unfolding**  $\langle \Gamma' = (- K, None) \# \Gamma \rangle$   
**by** (*metis D-true clauses-lt-D-true linorder-cls.neq-iff list.inject step-hyps(10) suffix-Cons*)

*suffix-def*)

**ultimately show** *?thesis*  
**unfolding**  $\langle s' = (U_{er}, \mathcal{F}, \Gamma', None) \rangle$   
**proof** (*intro ord-res-7-invars.intros*)  
**have** *trail-true-cls*  $\Gamma' C$   
 $\wedge K_C. \text{linorder-lit.is-maximal-in-mset } C K_C \implies$   
 $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin \mid \text{trail-atms } \Gamma')$   
**if** *C-lt*:  $\wedge E. None = Some E \implies C \prec_c E$  **and** *C-in*:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **for**  $C$   
**proof** –  
**have**  $None = \text{The-optional } (\text{linorder-cls.is-least-in-fset } (\text{ffilter } ((\prec_c) D) (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}))))$   
**using** *step-hyps*  
**by** (*metis (no-types, opaque-lifting) Some-eq-The-optionalD linorder-cls.is-least-in-ffilter-iff not-Some-eq*)

**hence**  $C \preceq_c D$   
**using** *step-hyps that by (metis clause-le-if-lt-least-greater)*

**hence**  $C \prec_c D \vee C = D$   
**by** *simp*

**hence** *trail-true-cls*  $\Gamma C$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**thus** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true*  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$  **by** *metis*

**next**  
**assume**  $C = D$   
**thus** *trail-true-cls*  $\Gamma C$   
**using** *D-true by argo*

**qed**

**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle \text{suffix } \Gamma \Gamma' \rangle$  **by** (*metis trail-true-cls-if-trail-true-suffix*)

**fix**  $K_C$   
**assume** *C-max-lit*: *linorder-lit.is-maximal-in-mset*  $C K_C$   
**thus**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin \mid \text{trail-atms } \Gamma')$   
**using**  $\langle C \prec_c D \vee C = D \rangle$

**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**hence**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin \mid \text{trail-atms } \Gamma')$   
**using** *no-undef-atm-lt-max-lit-if-lt-D C-in C-max-lit* **by** *force*

**thus** *?thesis*  
**using** *step-hyps(9) by force*



**next**  
**assume**  $C = D$   
**thus** *?thesis*  
**by** (*metis C-max-lit D-max-lit finsert-iff linorder-cls.order.irrefl*  
*linorder-lit.antisym-conv3 linorder-lit.multip-if-maximal-less-that-maximal*  
*step-hyps(5) step-hyps(9) trail-atms.simps(2)*)  
**qed**  
**qed**

**thus**  
 $\forall C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). (\forall D. \text{None} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
*trail-true-cls*  $\Gamma' C$   
 $\forall C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). (\forall D. \text{None} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K C \longrightarrow$   
 $\neg (\exists A | \in | \text{atms-of-clss } (N \ | \cup \ | U_{er}). A \prec_t \text{atm-of } K \wedge A | \notin | \text{trail-atms } \Gamma'))$   
**unfolding** *atomize-conj* **by** *metis*  
**qed** *simp-all*  
**next**  
**case** *step-hyps: (production*  $\Gamma D K U_{er} \Gamma' C' \mathcal{F})$

**note** *D-max-lit* =  $\langle \text{ord-res.is-maximal-lit } K D \rangle$

**have** *suffix*  $\Gamma \Gamma'$   
**using** *step-hyps* **by** (*simp add: suffix-def*)

**have**  
*F-subset:  $\mathcal{F} | \subseteq | N \ | \cup \ | U_{er}$  and*  
*D-in:  $D | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er})$  and*  
*clauses-lt-D-seldomly-have-undef-max-lit:*  
 $\forall C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{ \#K \in \# C. K \neq L_C \# \} \wedge$   
*is-pos*  $L_C))$  **and**  
*clauses-lt-D-true:  $\forall C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma$*   
*C and*  
*no-undef-atm-lt-max-lit-if-lt-D:  $\forall C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). C \prec_c D \longrightarrow$*   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A | \in | \text{atms-of-clss } (N \ | \cup \ | U_{er}). A \prec_t \text{atm-of } K \wedge A | \notin | \text{trail-atms } \Gamma))$   
**and**  
 $\Gamma$ -*sorted: sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\Gamma$ -*lower: linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \ | \cup \ | U_{er}))$   
**and**  
*trail-atms-le0:  $\forall A | \in | \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L D \wedge (A \preceq_t$*   
*atm-of*  $L)$  **and**  
 $\Gamma$ -*deci-iff-neg:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$  and*  
 $\Gamma$ -*deci-ball-lt-true:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$*   
 $(\forall C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). C \prec_c \{ \#fst Ln \# \} \longrightarrow \text{trail-true-cls } \Gamma C)$   
**and**  
 $\Gamma$ -*prop-in:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ |$*

$U_{er}$ ) **and**  
 $\Gamma$ -prop-greatest:  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
(fst  $Ln$ ) **and**  
 $\Gamma$ -prop-almost-false:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{ \#K \in \# C. K \neq L \# \}$  **and**  
 $\Gamma$ -prop-ball-lt-true:  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$   
**using invar by** (*simp-all add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle \text{ord-res-}\gamma\text{-invars-def}$ )  
  
**have** *clauses-lt-D-almost-defined*:  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{ \#L \in \# C. L \neq K \# \})$   
**using invar**[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*  
  
**have**  $K \in \# D$   
**using**  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *argo*  
  
**hence** *atm-of*  $K \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er})$   
**using** *D-in atm-of-in-atms-of-clsI* **by** *metis*  
  
**have** *atm-of*  $K \mid \notin \mid \text{trail-atms } \Gamma$   
**using**  $\langle \neg \text{trail-defined-lit } \Gamma K \rangle$   
**by** (*metis trail-defined-lit-iff-trail-defined-atm*)  
  
**hence** *trail-atms-lt*:  $\forall A \mid \in \mid \text{trail-atms } \Gamma. A \prec_t \text{atm-of } K$   
**using** *trail-atms-le0*  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset linorder-trm.antisym-conv1*)  
  
**have**  $\mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er}$   
**using** *F-subset* .  
  
**moreover have**  $\forall C'. C' = \text{Some } C' \longrightarrow C' \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using** *step-hyps(11)* **by** (*metis linorder-cls.is-least-in-ffilter-iff Some-eq-The-optionalD*)  
  
**moreover have** *trail-true-cls*  $\Gamma' \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$   
**if**  $C' = \text{Some } E$  **and**  
 $C$ -in:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $C$ -lt:  $C \prec_c E$  **and**  
 $C$ -max-lit: *linorder-lit.is-maximal-in-mset*  $C L_C$  **and**  
 $L_C$ -undef:  $\neg \text{trail-defined-lit } \Gamma' L_C$   
**for**  $E C L_C$   
**proof** –  
**have** *linorder-cls.is-least-in-fset* (*ffilter*  $((\prec_c) D) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))$ )  $E$   
**using**  $\langle C' = \text{Some } E \rangle$  *step-hyps* **by** (*metis Some-eq-The-optionalD*)  
**hence**  $C \prec_c D \vee C = D$

**unfolding** *linorder-cls.is-least-in-filter-iff*  
**using** *C-lt* **by** (*metis C-in linorder-cls.not-less-iff-gr-or-eq*)  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$

**moreover have**  $\neg \text{trail-defined-lit } \Gamma \ L_C$   
**using** *L<sub>C</sub>-undef*  $\langle \text{suffix } \Gamma \ \Gamma' \rangle$   
**using** *trail-defined-lit-if-trail-defined-suffix* **by** *blast*

**ultimately have** *trail-true-cl*  $\Gamma \ \{\#K \in\# \ C. \ K \neq L_C\# \} \wedge \text{is-pos } L_C$   
**using** *clauses-lt-D-seldomly-have-undef-max-lit* [*rule-format, OF C-in - C-max-lit*] **by** *metis*

**thus** *?thesis*  
**using**  $\langle \text{suffix } \Gamma \ \Gamma' \rangle$  *trail-true-cl-if-trail-true-suffix* **by** *blast*  
**next**  
**assume**  $C = D$   
**hence**  $L_C = K$   
**using** *C-max-lit D-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)  
**moreover have** *trail-defined-lit*  $\Gamma' \ K$   
**by** (*simp add: step-hyps trail-defined-lit-def*)  
**ultimately have** *False*  
**using** *L<sub>C</sub>-undef* **by** *argo*  
**thus** *?thesis ..*  
**qed**  
**qed**

**moreover have** *sorted-wrt*  $(\lambda x \ y. \ \text{atm-of } (fst \ y) \prec_t \ \text{atm-of } (fst \ x)) \ \Gamma'$   
**proof** –  
**have**  $x \prec_t \ \text{atm-of } K$  **if**  $x \text{-in: } x \in | \ \text{trail-atms } \Gamma$  **for**  $x$   
**using** *x-in trail-atms-lt* **by** *metis*

**hence**  $\forall x \in \text{set } \Gamma. \ \text{atm-of } (fst \ x) \prec_t \ \text{atm-of } K$   
**by** (*simp add: fset-trail-atms*)

**thus** *?thesis*  
**using**  $\Gamma$ -*sorted*  
**by** (*simp add:  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle$* )  
**qed**

**moreover have**  $\Gamma'$ -*lower: linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma'$ ) (*atms-of-cls*  $(N \cup U_{er})$ )  
**proof** –  
**have** *linorder-trm.is-lower-set* (*insert* (*atm-of*  $K$ ) (*fset* (*trail-atms*  $\Gamma$ )))  
(*fset* (*atms-of-cls*  $(N \cup U_{er})$ ))  
**proof** (*rule linorder-trm.is-lower-set-insertI*)  
**show** *atm-of*  $K \in | \ \text{atms-of-cls } (N \cup U_{er})$

```

    using ⟨atm-of K |∈| atms-of-clss (N |∪| Uer)⟩ .
  next
  show ∀ w|∈|atms-of-clss (N |∪| Uer). w ≺t (atm-of K) ⟶ w |∈| trail-atms
Γ
    using step-hyps(5)[unfolded linorder-trm.is-least-in-filter-iff, simplified]
    by fastforce
  next
  show linorder-trm.is-lower-fset (trail-atms Γ) (atms-of-clss (N |∪| Uer))
    using Γ-lower .
  qed

  moreover have trail-atms Γ' = finsert (atm-of K) (trail-atms Γ)
    by (simp add: ⟨Γ' = (K, Some D) # Γ⟩)

  ultimately show ?thesis
    by simp
  qed

  moreover have trail-atms Γ' = atms-of-clss (N |∪| Uer) if C' = None
  proof (intro fsubset-antisym)
    show trail-atms Γ' |⊆| atms-of-clss (N |∪| Uer)
      using Γ'-lower unfolding linorder-trm.is-lower-set-iff by blast
  next
  have nbex-gt-D: ¬ (∃ E |∈| iefac F |' (N |∪| Uer). D ≺c E)
    using step-hyps ⟨C' = None⟩
    by (metis clause-le-if-lt-least-greater linorder-clss.leD option.simps(3))

  have ¬ (∃ A|∈|atms-of-clss (N |∪| Uer). atm-of K ≺t A)
  proof (intro notI , elim bexE)
    fix A :: 'f gterm
    assume A |∈| atms-of-clss (N |∪| Uer) and atm-of K ≺t A

    hence A |∈| atms-of-clss (iefac F |' (N |∪| Uer))
      by simp

    then obtain E L where E-in: E |∈| iefac F |' (N |∪| Uer) and L ∈# E
  and A = atm-of L
    unfolding atms-of-clss-def atms-of-clss-def
    by (smt (verit, del-insts) fimage-iff fmember-ffUnion-iff fset-fset-mset)

    have K ≺l L
      using ⟨atm-of K ≺t A⟩ ⟨A = atm-of L⟩
      by (cases K; cases L) simp-all

    hence D ≺c E
      using D-max-lit ⟨L ∈# E⟩
    by (metis empty-iff linorder-lit.ex-maximal-in-mset linorder-lit.is-maximal-in-mset-iff
        linorder-lit.less-linear linorder-lit.less-trans
        linorder-lit.multip-if-maximal-less-than-maximal set-mset-empty)
  
```

**thus** *False*  
**using** *E-in nbex-gt-D by metis*  
**qed**

**hence**  $\neg (\exists A | \in | \text{atms-of-clss } (N \cup | U_{er}). A | \notin | \text{trail-atms } \Gamma')$   
**using** *step-hyps*  
**by** (*metis finsert-iff fst-conv linorder-trm.antisym-conv3 trail-atms.simps(2)*)

**then show**  $\text{atms-of-clss } (N \cup | U_{er}) | \subseteq | \text{trail-atms } \Gamma'$   
**by** *blast*  
**qed**

**moreover have**  $\forall D. C' = \text{Some } D \longrightarrow (\forall A | \in | \text{trail-atms } \Gamma').$   
 $\exists L. \text{ord-res.is-maximal-lit } L D \wedge A \preceq_t \text{atm-of } L$

**proof** (*intro allI impI ballI*)

**fix**  $E :: 'f \text{ gterm literal multiset}$  **and**  $A :: 'f \text{ gterm}$   
**assume**  $C' = \text{Some } E$  **and**  $A | \in | \text{trail-atms } \Gamma'$

**have**  $\text{linorder-cls.is-least-in-fset } (\text{ffilter } ((\prec_c) D) (\text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}))) E$   
**using** *step-hyps(11) <C' = Some E> by (metis Some-eq-The-optionalD)*

**hence**  $D \prec_c E$   
**unfolding** *linorder-cls.is-least-in-ffilter-iff* **by** *argo*

**obtain**  $L$  **where**  $\text{linorder-lit.is-maximal-in-mset } E L$   
**by** (*metis <D <\_c E> linorder-cls.leD linorder-lit.ex-maximal-in-mset mempty-lesseq-cls*)

**show**  $\exists L. \text{ord-res.is-maximal-lit } L E \wedge A \preceq_t \text{atm-of } L$

**proof** (*intro exI conjI*)

**show**  $\text{ord-res.is-maximal-lit } L E$

**using**  $\langle \text{ord-res.is-maximal-lit } L E \rangle .$

**next**

**have**  $K \preceq_l L$

**using** *step-hyps(4) <ord-res.is-maximal-lit L E>*

**by** (*metis <D <\_c E> linorder-cls.less-asym linorder-lit.leI linorder-lit.mulp-if-maximal-less-that-maximal*)

**hence**  $\text{atm-of } K \preceq_t \text{atm-of } L$

**by** (*cases K; cases L simp-all*)

**moreover have**  $A \preceq_t \text{atm-of } K$

**proof** –

**have**  $A = \text{atm-of } K \vee A | \in | \text{trail-atms } \Gamma$

**using**  $\langle A | \in | \text{trail-atms } \Gamma' \rangle$

**unfolding**  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle$

**by** *simp*

**thus**  $A \preceq_t \text{atm-of } K$

**using** *trail-atms-lt* **by** *blast*  
**qed**

**ultimately show**  $A \preceq_t \text{atm-of } L$   
**by** *order*  
**qed**  
**qed**

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$   
**unfolding**  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle$   
**using**  $\Gamma\text{-deci-iff-neg } \langle \text{is-pos } K \rangle$  **by** *simp*

**moreover have** *trail-true-cls*  $\Gamma' C$   
**if**  $Ln \in \text{set } \Gamma'$  **and**  $\text{snd } Ln = \text{None}$  **and**  $C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er})$  **and**  $C \prec_c \{ \# \text{fst } Ln \# \}$   
**for**  $Ln C$   
**proof** –  
**have**  $Ln = (K, \text{Some } D) \vee Ln \in \text{set } \Gamma$   
**using**  $\langle Ln \in \text{set } \Gamma' \rangle$  **unfolding**  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle$  **by** *simp*

**hence** *trail-true-cls*  $\Gamma C$   
**proof** (*elim disjE*)  
**assume**  $Ln = (K, \text{Some } D)$   
**hence** *False*  
**using**  $\langle \text{snd } Ln = \text{None} \rangle$  **by** *simp*  
**thus** *?thesis ..*

**next**  
**assume**  $Ln \in \text{set } \Gamma$

**thus** *trail-true-cls*  $\Gamma C$   
**using**  $\Gamma\text{-deci-ball-lt-true } \langle \text{snd } Ln = \text{None} \rangle \langle C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$   
 $\langle C \prec_c \{ \# \text{fst } Ln \# \} \rangle$   
**by** *metis*  
**qed**

**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle \text{suffix } \Gamma \Gamma' \rangle$  **by** (*metis trail-true-cls-if-trail-true-suffix*)  
**qed**

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er})$   
**using**  $\Gamma\text{-prop-in step-hyps}(10) \langle D \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}) \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C (\text{fst } Ln)$   
**using**  $\Gamma\text{-prop-greatest step-hyps}(8,9,10)$  **by** *simp*

**moreover have**  $\forall \Gamma_1 L C \Gamma_0. \Gamma' = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{ \# K \in \# C. K \neq L \# \}$

**unfolding**  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle$   
**using**  $\Gamma$ -prop-almost-false  $\langle \text{trail-false-cls } \Gamma \{ \#x \in \# D. x \neq K \# \} \rangle$   
**by** (*metis* (*no-types*, *lifting*) *append-eq-Cons-conv list.inject option.inject prod.inject*)

**moreover have**  $(\forall \Gamma_1 L D \Gamma_0. \Gamma' = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$

**proof** –  
**have**  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma C$   
**proof** (*intro ballI impI*)  
**fix**  $C :: 'f \text{ gterm literal multiset}$   
**assume**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $C \prec_c D$   
**thus** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true* **by** *metis*  
**qed**

**thus** *?thesis*  
**unfolding**  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle$   
**by** (*smt* (*verit*, *ccfv-SIG*)  $\Gamma$ -prop-ball-lt-true *append-eq-Cons-conv list.inject option.inject prod.inject*)  
**qed**

**ultimately show** *?thesis*  
**unfolding**  $\langle s' = (U_{er}, \mathcal{F}, \Gamma', C') \rangle$   
**proof** (*intro ord-res-7-invars.intros*)  
**have** *trail-true-cls*  $\Gamma' C$   
 $\wedge K_C. \text{linorder-lit.is-maximal-in-mset } C K_C \implies$   
 $\neg (\exists A \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin \text{trail-atms } \Gamma')$   
**if** *C-lt*:  $\wedge E. C' = \text{Some } E \implies C \prec_c E$  **and** *C-in*:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **for**  $C$   
**proof** –  
**have**  $C \preceq_c D$   
**using** *step-hyps* that **by** (*metis clause-le-if-lt-least-greater*)

**hence**  $C \prec_c D \vee C = D$   
**by** *simp*

**thus** *trail-true-cls*  $\Gamma' C$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$

**hence** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true*  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \rangle$  **by** *metis*

**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle \text{suffix } \Gamma \Gamma' \rangle$  **by** (*metis trail-true-cls-if-trail-true-suffix*)

**next**  
**assume**  $C = D$

**have** *trail-true-lit*  $\Gamma' K$   
**using**  $\langle \Gamma' = (K, \text{Some } D) \# \Gamma \rangle \langle \text{is-pos } K \rangle$   
**unfolding** *trail-true-lit-def* **by** (*cases*  $K$ ) *simp-all*

**thus** *trail-true-cls*  $\Gamma' C$   
**unfolding**  $\langle C = D \rangle$  *trail-true-cls-def*  
**using**  $\langle K \in \# D \rangle$  **by** *metis*

**qed**

**fix**  $K_C$   
**assume** *C-max-lit: linorder-lit.is-maximal-in-mset*  $C K_C$   
**show**  $\neg (\exists A \in | \text{atms-of-cls} (N \cup U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin | \text{trail-atms}$

$\Gamma')$

**using**  $\langle C \prec_c D \vee C = D \rangle$   
**proof** (*elim disjE*)  
**assume**  $C \prec_c D$   
**hence**  $\neg (\exists A \in | \text{atms-of-cls} (N \cup U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin | \text{trail-atms}$

$\Gamma)$

**using** *no-undef-atm-lt-max-lit-if-lt-D C-in C-max-lit* **by** *force*  
**thus** *?thesis*  
**using** *step-hyps* **by** *force*

**next**

**assume**  $C = D$   
**thus** *?thesis*  
**by** (*metis C-max-lit D-max-lit*  $\langle \text{suffix } \Gamma \Gamma' \rangle$  *linorder-lit.Uniq-is-maximal-in-mset literal.sel(2) step-hyps(5) the1-equality'* *trail-defined-lit-if-trail-defined-suffix trail-defined-lit-iff-trail-defined-atm*)

**qed**

**qed**

**thus**  
 $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup U_{er}). (\forall D. C' = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow \text{trail-true-cls}$

$\Gamma' C$

$\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup U_{er}). (\forall D. C' = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K C \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-cls} (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma'))$   
**unfolding** *atomize-conj* **by** *metis*

**qed** *simp-all*

**next**

**case** *step-hyps: (factoring*  $\Gamma D K U_{er} \mathcal{F}' \mathcal{F})$

**note** *D-max-lit* =  $\langle \text{ord-res.is-maximal-lit } K D \rangle$

**have**  
 $\mathcal{F}$ -*subset*:  $\mathcal{F} \mid \subseteq \mid N \cup U_{er}$  **and**  
 $D$ -*in*:  $D \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup U_{er})$  **and**  
*clauses-lt-D-seldomly-have-undef-max-lit*:  
 $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup U_{er}). C \prec_c D \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$



$\neg$  *trail-defined-lit*  $\Gamma$   $L_C \longrightarrow$  (*trail-true-cls*  $\Gamma$   $\{\#K \in\# C. K \neq L_C\# \} \wedge$   
*is-pos*  $L_C$ ) **and**  
*clauses-lt-D-true*:  $\forall C \mid \in \mid$  *iefac*  $\mathcal{F} \mid \uparrow \mid$  ( $N \mid \cup \mid U_{er}$ ).  $C \prec_c D \longrightarrow$  *trail-true-cls*  $\Gamma$   
 $C$  **and**  
*no-undef-atm-lt-max-lit-if-lt-D*:  $\forall C \mid \in \mid$  *iefac*  $\mathcal{F} \mid \uparrow \mid$  ( $N \mid \cup \mid U_{er}$ ).  $C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma))$   
**and**  
 $\Gamma$ -*sorted*: *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\Gamma$ -*lower*: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-clss* ( $N \mid \cup \mid U_{er}$ ))  
**and**  
*trail-atms-le0*:  $\forall A \mid \in \mid$  *trail-atms*  $\Gamma. \exists L. \text{ord-res.is-maximal-lit } L D \wedge (A \preceq_t$   
*atm-of*  $L)$  **and**  
 $\Gamma$ -*deci-iff-neg*:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$  **and**  
 $\Gamma$ -*deci-ball-lt-true*:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \mid \in \mid$  *iefac*  $\mathcal{F} \mid \uparrow \mid$  ( $N \mid \cup \mid U_{er}$ ).  $C \prec_c \{\#fst Ln\# \} \longrightarrow$  *trail-true-cls*  $\Gamma C$ )  
**and**  
 $\Gamma$ -*prop-in*:  $\forall Ln \in \text{set } \Gamma. \forall D. \text{snd } Ln = \text{Some } D \longrightarrow D \mid \in \mid$  *iefac*  $\mathcal{F} \mid \uparrow \mid$  ( $N \mid \cup \mid$   
 $U_{er}$ ) **and**  
 $\Gamma$ -*prop-greatest*:  
 $\forall Ln \in \text{set } \Gamma. \forall D. \text{snd } Ln = \text{Some } D \longrightarrow \text{linorder-lit.is-greatest-in-mset } D$   
 $(fst Ln)$  **and**  
 $\Gamma$ -*prop-almost-false*:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow$  *trail-false-cls*  $\Gamma_0 \{\#K \in\# C.$   
 $K \neq L\#\}$  **and**  
 $\Gamma$ -*prop-ball-lt-true*:  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid$  *iefac*  $\mathcal{F} \mid \uparrow \mid$  ( $N \mid \cup \mid U_{er}$ ).  $C \prec_c D \longrightarrow$  *trail-true-cls*  $\Gamma_0 C)$ )  
**using invar by** (*simp-all add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$  *ord-res-7-invars-def*)  
  
**have** *clauses-lt-D-almost-defined*:  $\forall C \mid \in \mid$  *iefac*  $\mathcal{F} \mid \uparrow \mid$  ( $N \mid \cup \mid U_{er}$ ).  $C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{\#L \in\# C. L$   
 $\neq K\#\})$   
**using invar**[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*  
  
**have** *atm-of*  $K \notin \text{trail-atms } \Gamma$   
**using**  $\langle \neg \text{trail-defined-lit } \Gamma K \rangle$   
**by** (*metis trail-defined-lit-iff-trail-defined-atm*)  
  
**hence** *trail-atms-lt*:  $\forall A \mid \in \mid$  *trail-atms*  $\Gamma. A \prec_t \text{atm-of } K$   
**using** *trail-atms-le0*  $\langle \text{ord-res.is-maximal-lit } K D \rangle$   
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset linorder-trm.antisym-conv1*)  
  
**have** *efac*  $D \neq D$   
**using**  $\langle \neg \text{ord-res.is-strictly-maximal-lit } K D \rangle$  *D-max-lit*  $\langle \text{is-pos } K \rangle$   
**by** (*metis ex1-efac-eq-factoring-chain ex-ground-factoringI is-pos-def*)  
  
**hence** *efac*  $D \prec_c D$   
**by** (*metis efac-properties-if-not-ident*(1))

hence *D-in-strong*:  $D \in N \cup U_{er}$  and  $D \notin \mathcal{F}$   
 using *D-in*  $\langle \text{efac } D \neq D \rangle$   
 unfolding *atomize-conj* *iefac-def*  
 by (*smt* (*verit*) *factorizable-if-neq-efac* *fimage-iff* *iefac-def* *ex1-efac-eq-factoring-chain*)

have  $\mathcal{F}' \subseteq N \cup U_{er}$   
 unfolding  $\langle \mathcal{F}' = \text{finsert } D \mathcal{F} \rangle$   
 using *F-subset* *D-in-strong* by *simp*

moreover have  $\forall C'. \text{Some } (\text{efac } D) = \text{Some } C' \longrightarrow C' \in \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er})$

**proof** –

have  $\text{efac } D \in \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er})$   
 using *D-in-strong* by (*simp* *add*: *iefac-def*  $\langle \mathcal{F}' = \text{finsert } D \mathcal{F} \rangle$ )

thus *?thesis*

by *simp*

**qed**

moreover have *trail-true-cls*  $\Gamma \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$

if  $\text{Some } (\text{efac } D) = \text{Some } E$  and

*C-in*:  $C \in \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er})$  and

*C-lt*:  $C \prec_c E$  and

*C-max-lit*: *linorder-lit.is-maximal-in-mset*  $C L_C$  and

*L<sub>C</sub>-undef*:  $\neg \text{trail-defined-lit } \Gamma L_C$

for  $E C L_C$

**proof** –

have  $E = \text{efac } D$

using *that* by *simp*

hence  $C \prec_c \text{efac } D$

using *C-lt* by *order*

hence  $C \neq \text{efac } D$

by *order*

hence  $C \in \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er})$

using *C-in* *iefac-def* *step-hyps(10)* by *auto*

moreover have  $C \prec_c D$

using  $\langle C \prec_c \text{efac } D \rangle \langle \text{efac } D \prec_c D \rangle$  by *order*

ultimately show *trail-true-cls*  $\Gamma \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$

using *clauses-lt-D-seldomly-have-undef-max-lit* *C-max-lit* *L<sub>C</sub>-undef* by *metis*

**qed**

moreover have *trail-true-cls*  $\Gamma C$

$\bigwedge K_C. \text{linorder-lit.is-maximal-in-mset } C K_C \implies \text{trail-defined-cls } \Gamma \{ \#L \in \# C. L \neq K_C \# \}$

if *C-lt*:  $\bigwedge E. \text{Some } (\text{efac } D) = \text{Some } E \implies C \prec_c E$  and *C-in*:  $C \in \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er})$  for  $C$

**proof** –

have  $C \prec_c \text{efac } D$

**using**  $C$ -lt **by** *metis*

**hence**  $C \neq \text{efac } D$   
**by** *order*

**hence**  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $C$ -in **by** (*auto simp:  $\langle \mathcal{F}' = \text{finsert } D \ \mathcal{F} \rangle$  iefac-def*)

**moreover have**  $C \prec_c D$   
**using**  $\langle C \prec_c \text{efac } D \rangle \langle \text{efac } D \prec_c D \rangle$  **by** *order*

**ultimately show** *trail-true-cls*  $\Gamma \ C$   
**using** *clauses-lt-D-true* **by** *metis*

**fix**  $K_C$   
**assume**  $C$ -max-lit: *linorder-lit.is-maximal-in-mset*  $C \ K_C$   
**show** *trail-defined-cls*  $\Gamma \ \{\#L \in \# \ C. \ L \neq K_C\# \}$   
**using** *clauses-lt-D-almost-defined*  $\langle C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \rangle \langle C \prec_c D \rangle$   
 $C$ -max-lit  
**by** *metis*  
**qed**

**moreover have** *sorted-wrt*  $(\lambda x \ y. \ \text{atm-of } (fst \ y) \prec_t \ \text{atm-of } (fst \ x)) \ \Gamma$   
**using**  $\Gamma$ -sorted .

**moreover have** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) \ (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**using**  $\Gamma$ -lower .

**moreover have**  $\forall D'. \ \text{Some } (\text{efac } D) = \text{Some } D' \longrightarrow (\forall A \in \text{trail-atms } \Gamma. \ \exists L. \ \text{ord-res.is-maximal-lit } L \ D' \wedge A \preceq_t \ \text{atm-of } L)$   
**proof** –  
**have**  $\forall A \in \text{trail-atms } \Gamma. \ \exists L. \ \text{ord-res.is-maximal-lit } L \ (\text{efac } D) \wedge A \preceq_t \ \text{atm-of } L$   
 $L$   
**using** *trail-atms-lt*  
**using** *ex1-efac-eq-factoring-chain step-hyps(4)*  
*ord-res.ground-factorings-preserves-maximal-literal* **by** *blast*

**thus** *?thesis*  
**by** *simp*  
**qed**

**moreover have**  $\forall Ln \in \text{set } \Gamma. \ \text{snd } Ln = \text{None} \iff \text{is-neg } (fst \ Ln)$   
**using**  $\Gamma$ -deci-iff-neg .

**moreover have** *trail-true-cls*  $\Gamma \ C$   
**if**  $Ln \in \text{set } \Gamma$  **and**  $\text{snd } Ln = \text{None}$  **and**  $C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $C \prec_c \{\#fst \ Ln\# \}$   
**for**  $Ln \ C$

**proof** –  
**have**  $C = \text{efac } D \vee C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**using**  $\langle C \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$  **by** (*auto simp: iefac-def*  $\langle \mathcal{F}' = \text{finsert } D \mathcal{F} \rangle$ )

**thus** *trail-true-cls*  $\Gamma C$   
**proof** (*elim disjE*)  
**assume**  $C = \text{efac } D$

**hence** *linorder-lit.is-greatest-in-mset*  $C K$   
**using** *D-max-lit*  $\langle \text{is-pos } K \rangle$   
**by** (*metis greatest-literal-in-efacI*)

**hence**  $K \prec_l \text{fst } Ln$   
**using**  $\langle C \prec_c \{\#\text{fst } Ln\# \} \rangle$   
**by** (*simp add: linorder-lit.is-greatest-in-mset-iff*)

**hence** *atm-of*  $K \preceq_t \text{atm-of } (\text{fst } Ln)$   
**by** (*cases K; cases fst Ln*) *simp-all*

**moreover have** *atm-of*  $(\text{fst } Ln) \mid \in \mid \text{trail-atms } \Gamma$   
**using**  $\langle Ln \in \text{set } \Gamma \rangle$  **by** (*simp add: fset-trail-atms*)

**moreover have** *atm-of*  $K \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er})$   
**by** (*meson D-in D-max-lit atm-of-in-atms-of-clsI linorder-lit.is-maximal-in-mset-iff*)

**ultimately have** *atm-of*  $K \mid \in \mid \text{trail-atms } \Gamma$   
**using**  *$\Gamma$ -lower*  
**unfolding** *linorder-trm.is-lower-set-iff*  
**by** *fastforce*

**hence** *False*  
**using**  $\langle \text{atm-of } K \mid \notin \mid \text{trail-atms } \Gamma \rangle$  **by** *contradiction*

**thus** *trail-true-cls*  $\Gamma C \dots$

**next**  
**assume**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**thus** *trail-true-cls*  $\Gamma C$   
**using**  *$\Gamma$ -deci-ball-lt-true*  $\langle Ln \in \text{set } \Gamma \rangle \langle \text{snd } Ln = \text{None} \rangle \langle C \prec_c \{\#\text{fst } Ln\# \} \rangle$   
**by** *metis*

**qed**  
**qed**

**moreover have**  $C \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er})$  **if**  $Ln \in \text{set } \Gamma$  **and**  $\text{snd } Ln = \text{Some } C$  **for**  $Ln C$   
**proof** –  
**have** *atm-of*  $(\text{fst } Ln) \prec_t \text{atm-of } K$   
**using** *trail-atms-lt[unfolded fset-trail-atms, simplified]*  $\langle Ln \in \text{set } \Gamma \rangle$  **by** *metis*

hence  $atm\text{-of } (fst Ln) \neq atm\text{-of } K$   
 by *order*

hence  $fst Ln \neq K$   
 by (*cases fst Ln; cases K*) *simp-all*

moreover have  $ord\text{-res.is-maximal-lit } (fst Ln) C$   
 using  $\Gamma\text{-prop-greatest } \langle Ln \in set \Gamma \rangle \langle snd Ln = Some C \rangle$  by *blast*

ultimately have  $C \neq D$   
 using  $\langle ord\text{-res.is-maximal-lit } K D \rangle$  by (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

moreover have  $C \in |iefac \mathcal{F}| (N \cup U_{er})$   
 using  $\Gamma\text{-prop-in } \langle Ln \in set \Gamma \rangle \langle snd Ln = Some C \rangle$  by *metis*

ultimately show *?thesis*  
 by (*auto simp: \mathcal{F}' = finsert D \mathcal{F}*) *iefac-def*)

qed

moreover have  $\forall Ln \in set \Gamma. \forall D. snd Ln = Some D \longrightarrow linorder\text{-lit.is-greatest-in-mset } D (fst Ln)$   
 using  $\Gamma\text{-prop-greatest}$  by *simp*

moreover have  $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, Some C) \# \Gamma_0 \longrightarrow trail\text{-false-cls } \Gamma_0 \{ \#K \in \# C. K \neq L \# \}$   
 using  $\Gamma\text{-prop-almost-false}$  .

moreover have  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, Some D) \# \Gamma_0 \longrightarrow (\forall C \in |iefac \mathcal{F}'| (N \cup U_{er}). C \prec_c D \longrightarrow trail\text{-true-cls } \Gamma_0 C))$

*proof* (*intro allI impI ballI*)

*fix*

$\Gamma_1 \Gamma_0 :: ('f gliteral \times 'f gclause option) list$  and  
 $L :: 'f gliteral$  and  
 $C_0 C_1 :: 'f gclause$

*assume*

$\Gamma\text{-eq}: \Gamma = \Gamma_1 @ (L, Some C_1) \# \Gamma_0$  and  
 $C_0\text{-in}: C_0 \in |iefac \mathcal{F}'| (N \cup U_{er})$  and  
 $C_0 \prec_c C_1$

have  $C_0 = efac D \vee C_0 \in |iefac \mathcal{F}'| (N \cup U_{er})$   
 using  $C_0\text{-in}$  by (*auto simp: iefac-def \mathcal{F}' = finsert D \mathcal{F}*)

thus  $trail\text{-true-cls } \Gamma_0 C_0$

*proof* (*elim disjE*)

*assume*  $C_0 = efac D$

have  $atm\text{-of } L \in |trail\text{-atms } \Gamma$   
 using  $\Gamma\text{-eq unfolding fset-trail-atms}$  by *simp*

hence  $atm\text{-of } L \prec_t atm\text{-of } K$   
 using *trail-atms-lt* by *metis*

hence  $L \prec_l K$   
 by (*cases L*; *cases K*) *simp-all*

moreover have  $linorder\text{-lit.is-greatest-in-mset } C_1 L$   
 using  $\Gamma\text{-eq } \Gamma\text{-prop-greatest}$  by *simp*

moreover have  $linorder\text{-lit.is-greatest-in-mset } (efac D) K$   
 using  $\langle is\text{-pos } K \rangle D\text{-max-lit}$  by (*metis greatest-literal-in-efacI*)

ultimately have  $C_1 \prec_c efac D$   
 by (*metis linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*  
*linorder-lit.mulp-if-maximal-less-that-maximal*)

hence *False*  
 using  $\langle C_0 = efac D \rangle \langle C_0 \prec_c C_1 \rangle$  by *order*

thus *?thesis ..*

next

assume  $C_0 \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$

thus *?thesis*

using  $\Gamma\text{-prop-ball-lt-true } \Gamma\text{-eq } \langle C_0 \prec_c C_1 \rangle$  by *metis*

qed

qed

ultimately show *?thesis*

unfolding  $\langle s' = (U_{er}, \mathcal{F}', \Gamma, \text{Some } (efac D)) \rangle$

proof (*intro ord-res-7-invars.intros*)

have *trail-true-cls*  $\Gamma C$

$\bigwedge K_C. linorder\text{-lit.is-maximal-in-mset } C K_C \implies$

$\neg (\exists A \in |atms\text{-of-cls } (N \cup U_{er}). A \prec_t atm\text{-of } K_C \wedge A \notin |trail\text{-atms } \Gamma)$

if *C-lt*:  $\bigwedge E. \text{Some } (efac D) = \text{Some } E \implies C \prec_c E$  and *C-in*:  $C \in |iefac \mathcal{F}'$   
 $| \uparrow (N \cup U_{er})$

for *C*

proof –

have  $C \prec_c efac D$

using *C-lt* by *metis*

hence  $C \neq efac D$

by *order*

hence  $C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$

using *C-in* by (*auto simp: \langle \mathcal{F}' = finsert D \mathcal{F} \rangle iefac-def*)

moreover have  $C \prec_c D$

using  $\langle C \prec_c efac D \rangle \langle efac D \prec_c D \rangle$  by *order*

**ultimately show** *trail-true-cls*  $\Gamma C$   
**using** *clauses-lt-D-true* **by** *metis*

**fix**  $K_C$   
**assume** *C-max-lit: linorder-lit.is-maximal-in-mset*  $C K_C$   
**thus**  $\neg (\exists A \in | \text{atms-of-clss } (N \cup U_{er}). A \prec_t \text{atm-of } K_C \wedge A \notin | \text{trail-atms } \Gamma)$

**using** *no-undef-atm-lt-max-lit-if-lt-D*  $\langle C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}) \rangle \langle C \prec_c D \rangle$  **by** *metis*  
**qed**

**thus**  
 $\forall C \in | \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er}). (\forall Da. \text{Some } (efac D) = \text{Some } Da \longrightarrow C \prec_c Da) \longrightarrow$   
*trail-true-cls*  $\Gamma C$   
 $\forall C \in | \text{iefac } \mathcal{F}' \uparrow (N \cup U_{er}). (\forall Da. \text{Some } (efac D) = \text{Some } Da \longrightarrow C \prec_c Da) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K C \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma))$   
**unfolding** *atomize-conj* **by** *metis*  
**qed** *simp-all*

**next**  
**case** *step-hyps: (resolution-bot*  $\Gamma E K D U_{er}' U_{er} \Gamma' \mathcal{F})$

**have**  
*F-subset:  $\mathcal{F} \subseteq | N \cup U_{er}$*  **and**  
*E-in:  $E \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er})$*  **and**  
*clauses-lt-E-seldomly-have-undef-max-lit:*  
 $\forall C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}). C \prec_c E \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in \# C. K \neq L_C\} \wedge$   
*is-pos*  $L_C))$  **and**  
*clauses-lt-E-true:  $\forall C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}). C \prec_c E \longrightarrow \text{trail-true-cls } \Gamma C$*  **and**  
*no-undef-atm-lt-max-lit-if-lt-E:  $\forall C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}). C \prec_c E \longrightarrow$*   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma))$

**and**  
 $\Gamma$ -*sorted: sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\Gamma$ -*lower: linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \cup U_{er}))$

**and**  
*trail-atms-le0:  $\forall A \in | \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L E \wedge (A \preceq_t \text{atm-of } L)$*  **and**  
 $\Gamma$ -*deci-iff-neg:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$*  **and**  
 $\Gamma$ -*deci-ball-lt-true:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$*   
 $(\forall C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}). C \prec_c \{\#fst Ln\} \longrightarrow \text{trail-true-cls } \Gamma C)$

**and**  
 $\Gamma$ -*prop-in:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er})$*  **and**

$\Gamma$ -prop-greatest:  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
 (fst Ln) **and**  
 $\Gamma$ -prop-almost-false:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{ \#K \in \# C. K \neq L \# \}$  **and**  
 $\Gamma$ -prop-ball-lt-true:  $\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C)$   
**using invar by** (simp-all add:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) \rangle \text{ord-res-}\gamma\text{-invars-def}$ )

**have clauses-lt-E-almost-defined:**  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{ \#L \in \# C. L \neq K \# \})$   
 $L \neq K \# \}$   
**using invar**[unfolded  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by simp**

**have**  $\mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er}'$   
**unfolding**  $\langle U_{er}' = \text{finsert } (\text{eres } D E) U_{er} \rangle$   
**using**  $\mathcal{F}$ -subset **by blast**

**moreover have**  $\forall C'. \text{Some } \{ \# \} = \text{Some } C' \longrightarrow C' \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$   
**by** (simp add:  $\langle \text{eres } D E = \{ \# \} \rangle \langle U_{er}' = \text{finsert } (\text{eres } D E) U_{er} \rangle$ )

**moreover have**  $\text{trail-true-cls } \Gamma' \{ \#K \in \# C. K \neq L_C \# \} \wedge \text{is-pos } L_C$   
**if**  $\text{Some } \{ \# \} = \text{Some } E$  **and**  
 $C$ -in:  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$  **and**  
 $C$ -lt:  $C \prec_c E$  **and**  
 $C$ -max-lit:  $\text{linorder-lit.is-maximal-in-mset } C L_C$  **and**  
 $L_C$ -undef:  $\neg \text{trail-defined-lit } \Gamma' L_C$   
**for**  $E C L_C$

**proof** –  
**have**  $E = \{ \# \}$   
**using that by simp**  
**hence** *False*  
**using**  $C$ -lt  $\text{linorder-cls.leD mempty-lesseq-cls}$  **by blast**  
**thus** *?thesis ..*  
**qed**

**moreover have**  $\text{trail-defined-cls } \Gamma' \{ \#L \in \# x. L \neq K_x \# \}$   
**if**  $\text{Some } \{ \# \} = \text{Some } y$  **and**  $x$ -in:  $x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$  **and**  $x \prec_c y$   
**and**  
 $x$ -max-lit:  $\text{linorder-lit.is-maximal-in-mset } x K_x$  **for**  $x y K_x$

**proof** –  
**have** *False*  
**using**  $\text{linorder-cls.leD mempty-lesseq-cls that(1) that(3)}$  **by blast**  
**thus** *?thesis ..*  
**qed**

**moreover have**  $\text{sorted-wrt } (\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma'$



**unfolding**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have** *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma'$ ) (*atms-of-clss* ( $N \mid \cup U_{er}'$ ))

**unfolding**  $\langle \Gamma' = [] \rangle$   
**by** (*simp add: linorder-trm.is-lower-set-iff*)

**moreover have**  $\forall D. \text{Some } \{\#\} = \text{Some } D \longrightarrow (\forall A \mid \in \mid \text{trail-atms } \Gamma')$   
 $\exists L. \text{ord-res.is-maximal-lit } L \ D \wedge A \preceq_t \text{atm-of } L$

**unfolding**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$

**unfolding**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}'). C \prec_c \{\#\text{fst } Ln\# \} \longrightarrow \text{trail-true-cls } \Gamma' \ C)$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longrightarrow$   
 $\neg(\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}'). \text{linorder-lit.is-maximal-in-mset } C \ (- \ (\text{fst } Ln)))$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall D. \text{snd } Ln = \text{Some } D \longrightarrow$   
 $\neg(\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}'). C \prec_c D \wedge \text{fst } Ln \in \# \ C)$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C \ (\text{fst } Ln)$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall \Gamma_1 \ L \ C \ \Gamma_0. \Gamma' = \Gamma_1 \ @ \ (L, \text{Some } C) \ \# \ \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \ \{\#K \in \# \ C. K \neq L\# \}$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**moreover have**  $\forall \Gamma_1 \ L \ D \ \Gamma_0. \Gamma' = \Gamma_1 \ @ \ (L, \text{Some } D) \ \# \ \Gamma_0 \longrightarrow$   
 $(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}'). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 \ C)$

**using**  $\langle \Gamma' = [] \rangle$  **by** *simp*

**ultimately show** *?thesis*

**unfolding**  $\langle s' = (U_{er}', \mathcal{F}, \Gamma', \text{Some } \{\#\}) \rangle$

**proof** (*intro ord-res-7-invars.intros*)

**have** *trail-true-cls*  $\Gamma' \ x$   
 $\bigwedge K_x. \text{linorder-lit.is-maximal-in-mset } x \ K_x \implies$   
 $\neg(\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}'). A \prec_t \text{atm-of } K_x \wedge A \mid \notin \mid \text{trail-atms } \Gamma')$

**if**  $C\text{-lt}: \bigwedge E. \text{Some } \{\#\} = \text{Some } E \implies x \prec_c E$  **and**  $C\text{-in}: x \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er}')$   
**for**  $x$   
**proof** –  
**have**  $x \prec_c \{\#\}$   
**using**  $C\text{-lt}$  **by** *metis*  
**hence** *False*  
**using** *linorder-cls.leD mempty-lesseq-cls* **by** *blast*  
**thus**  
*trail-true-cls*  $\Gamma' x$   
 $\bigwedge K_x. \text{linorder-lit.is-maximal-in-mset } x K_x \implies$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}'). A \prec_t \text{atm-of } K_x \wedge A \notin | \text{trail-atms } \Gamma')$   
**by** *argo+*  
**qed**

**thus**  
 $\forall C \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er}'). (\forall D. \text{Some } \{\#\} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
*trail-true-cls*  $\Gamma' C$   
 $\forall C \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er}'). (\forall D. \text{Some } \{\#\} = \text{Some } D \longrightarrow C \prec_c D) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K C \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}'). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma'))$   
**unfolding** *atomize-conj* **by** *metis*  
**qed** *simp-all*

**next**  
**case** *step-hyps: (resolution-pos*  $\Gamma E L D U_{er}' U_{er} \Gamma' K \mathcal{F})$

**note**  $E\text{-max-lit} = \langle \text{ord-res.is-maximal-lit } L E \rangle$   
**note**  $\text{eres-max-lit} = \langle \text{ord-res.is-maximal-lit } K (\text{eres } D E) \rangle$

**have**  
 $\mathcal{F}\text{-subset}: \mathcal{F} \subseteq | N \mid \cup \mid U_{er}$  **and**  
 $E\text{-in}: E \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er})$  **and**  
 $\text{clauses-lt-}D\text{-seldomly-have-undef-max-lit:}$   
 $\forall C \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{\#K \in \# C. K \neq L_C \#\} \wedge$   
 $\text{is-pos } L_C))$  **and**  
 $\text{clauses-lt-}E\text{-true: } \forall C \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow \text{trail-true-cls } \Gamma C$  **and**  
 $\text{no-undef-atm-lt-max-lit-if-lt-}E: \forall C \in | \text{iefac } \mathcal{F} \uparrow | (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma))$

**and**  
 $\Gamma\text{-sorted: sorted-wrt } (\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
 $\Gamma\text{-lower: linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$

**and**  
 $\text{trail-atms-le0: } \forall A \in | \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L E \wedge (A \preceq_t \text{atm-of } L)$  **and**

$\Gamma$ -*deci-iff-neg*:  $\forall Ln \in set \Gamma. snd Ln = None \longleftrightarrow is-neg (fst Ln)$  **and**  
 $\Gamma$ -*deci-ball-lt-true*:  $\forall Ln \in set \Gamma. snd Ln = None \longrightarrow$   
 $(\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c \{\#fst Ln\}) \longrightarrow trail-true-cls \Gamma C$   
**and**  
 $\Gamma$ -*prop-in*:  $\forall Ln \in set \Gamma. \forall C. snd Ln = Some C \longrightarrow C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$  **and**  
 $\Gamma$ -*prop-greatest*:  
 $\forall Ln \in set \Gamma. \forall C. snd Ln = Some C \longrightarrow linorder-lit.is-greatest-in-mset C$   
 $(fst Ln)$  **and**  
 $\Gamma$ -*prop-almost-false*:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, Some C) \# \Gamma_0 \longrightarrow trail-false-cls \Gamma_0 \{\#K \in \# C. K \neq L\}$  **and**  
 $\Gamma$ -*prop-ball-lt-true*:  $\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, Some D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c D \longrightarrow trail-true-cls \Gamma_0 C)$   
**using invar by** (*simp-all add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, Some E) \rangle$  *ord-res-7-invars-def*)  
  
**have** *clauses-lt-E-almost-defined*:  $\forall C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}). C \prec_c E \longrightarrow$   
 $(\forall K. linorder-lit.is-maximal-in-mset C K \longrightarrow trail-defined-cls \Gamma \{\#L \in \# C. L \neq K\})$   
**using invar**[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, Some E) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*  
  
**have**  $\Gamma$ -*consistent*: *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms*  $\Gamma$ -*sorted* **by** *metis*  
  
**have** *trail-atms-le*:  $\forall A \in |trail-atms \Gamma. A \preceq_t atm-of L$   
**using** *trail-atms-le0 E-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)  
  
**have**  $(- L, Some D) \in set \Gamma$   
**using**  $\langle map-of \Gamma (- L) = Some (Some D) \rangle$  **by** (*metis map-of-SomeD*)  
  
**hence** *D-in*:  $D \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$   
**using**  $\Gamma$ -*prop-in* **by** *simp*  
  
**have** *D-max-lit*: *linorder-lit.is-greatest-in-mset*  $D (- L)$   
**using**  $\Gamma$ -*prop-greatest*  $\langle (- L, Some D) \in set \Gamma \rangle$  **by** *fastforce*  
  
**have** *suffix*  $\Gamma' \Gamma$   
**using** *step-hyps(9) suffix-dropWhile* **by** *metis*  
  
**hence** *atms-of-cls* (*eres*  $D E$ )  $\subseteq |atms-of-cls D \cup |atms-of-cls E$   
**using** *lit-in-one-of-resolvents-if-in-eres*  
**unfolding** *atms-of-cls-def* **by** *fastforce*  
  
**moreover have** *atms-of-cls*  $D \subseteq |atms-of-clss (N \cup U_{er})$   
**using**  $\langle D \in |iefac \mathcal{F} |^\dagger (N \cup U_{er}) \rangle$   
**by** (*metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb funion-upper1*)

**moreover have**  $atms\text{-of}\text{-cls } E \mid\subseteq\mid atms\text{-of}\text{-clss } (N \mid\cup\mid U_{er})$   
**using**  $\langle E \mid\in\mid iefac \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er}) \rangle$   
**by** (*metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb funion-upper1*)

**ultimately have**  $atms\text{-of}\text{-clss } (N \mid\cup\mid U_{er}) = atms\text{-of}\text{-clss } (N \mid\cup\mid U_{er}')$   
**unfolding**  $\langle U_{er}' = finsert (eres D E) U_{er} \rangle$  **atms-of-clss-def** **by** *auto*

**obtain**  $A_L$  **where**  $L\text{-def}: L = Neg A_L$   
**using**  $\langle is\text{-neg } L \rangle$  **by** (*cases L*) *simp-all*

**have**  $D \prec_c E$   
**using** *clause-lt-clause-if-max-lit-comp*  
**using**  $E\text{-max-lit } \langle is\text{-neg } L \rangle D\text{-max-lit}$   
**by** (*metis linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)

**have**  $eres D E \prec_c E$   
**using** *eres-lt-if*  
**using**  $E\text{-max-lit } \langle is\text{-neg } L \rangle D\text{-max-lit}$  **by** *metis*

**hence**  $eres D E \neq E$   
**by** *order*

**have**  $L \in\# E$   
**using**  $E\text{-max-lit}$  **unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *metis*

**hence**  $-L \notin\# E$   
**using** *not-both-lit-and-comp-lit-in-false-clause-if-trail-consistent*  
**using**  $\Gamma\text{-consistent } \langle trail\text{-false-cl } \Gamma E \rangle$  **by** *metis*

**hence**  $\forall K \in\# eres D E. atm\text{-of } K \prec_t atm\text{-of } L$   
**using** *lit-in-eres-lt-greatest-lit-in-grestest-resolvant*[*OF*  $\langle eres D E \neq E \rangle E\text{-max-lit}$ ]  
**by** *metis*

**hence**  $\forall K \in\# eres D E. K \neq L \wedge K \neq -L$   
**by** *fastforce*

**moreover have**  $\forall L \in\# eres D E. L \in\# D \vee L \in\# E$   
**using** *lit-in-one-of-resolvents-if-in-eres* **by** *metis*

**moreover have**  $D\text{-almost-false}: trail\text{-false-cl } \Gamma \{\#K \in\# D. K \neq -L\# \}$   
**using** *ord-res-7-invars-implies-propagated-clause-almost-false*  
**using**  $\langle (-L, Some D) \in set \Gamma \rangle$  *invar*[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, Some E) \rangle$ ]  
**by** *metis*

**ultimately have**  $trail\text{-false-cl } \Gamma (eres D E)$   
**using**  $\langle trail\text{-false-cl } \Gamma E \rangle$  **unfolding** *trail-false-cl-def* **by** *fastforce*

**hence**  $trail\text{-false-lit } \Gamma K$   
**using**  $eres\text{-max-lit}$  **unfolding** *linorder-lit.is-maximal-in-mset-iff* *trail-false-cl-def*

by *metis*

**have**  $eres\ D\ E\ |\notin\ N\ |\cup\ U_{er}$   
**using** *eres-not-in-known-clauses-if-trail-false-cls*  
**using**  $\Gamma$ -consistent clauses-*lt-E-true*  $\langle eres\ D\ E\ \prec_c\ E \rangle$  *trail-false-cls*  $\Gamma$   $(eres\ D\ E)$  by *metis*

**hence**  $eres\ D\ E\ |\notin\ \mathcal{F}$   
**using**  $\mathcal{F}$ -subset by *blast*

**hence**  $iefac\ \mathcal{F}\ (eres\ D\ E) = eres\ D\ E$   
**by** (*simp add: iefac-def*)

**hence**  $iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}') = finsert\ (eres\ D\ E)\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}))$   
**unfolding**  $\langle U_{er}' = finsert\ (eres\ D\ E)\ U_{er} \rangle$  by *simp*

**have** *trail-false-lit*  $\Gamma\ K$   
**by** (*meson*  $\langle trail-false-cls\ \Gamma\ (eres\ D\ E) \rangle$  *linorder-lit.is-maximal-in-mset-iff* *step-hyps(10)* *trail-false-cls-def*)

**have** *mem-set- $\Gamma'$ -iff*:  $(Ln \in set\ \Gamma') = (\neg\ atm-of\ K\ \preceq_t\ atm-of\ (fst\ Ln)) \wedge Ln \in set\ \Gamma)$  **for**  $Ln$

**unfolding**  $\langle \Gamma' = dropWhile\ (\lambda Ln.\ atm-of\ K\ \preceq_t\ atm-of\ (fst\ Ln))\ \Gamma \rangle$

**proof** (*rule mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone*)

**show** *sorted-wrt*  $(\lambda x\ y.\ atm-of\ (fst\ y)\ \prec_t\ atm-of\ (fst\ x))\ \Gamma$

**using**  $\Gamma$ -sorted .

**next**

**show** *monotone-on*  $(set\ \Gamma)\ (\lambda x\ y.\ atm-of\ (fst\ y)\ \prec_t\ atm-of\ (fst\ x))\ (\lambda x\ y.\ y \leq x)$

$(\lambda Ln.\ atm-of\ K\ \preceq_t\ atm-of\ (fst\ Ln))$

**by** (*rule monotone-onI*) *auto*

**qed**

**hence** *atms-in- $\Gamma'$ -lt-atm-K*:  $\forall x\ |\in\ trail-atms\ \Gamma'.\ x\ \prec_t\ atm-of\ K$   
**by** (*auto simp add: fset-trail-atms*)

**have**  $\mathcal{F}\ |\subseteq\ N\ |\cup\ U_{er}'$   
**unfolding**  $\langle U_{er}' = finsert\ (eres\ D\ E)\ U_{er} \rangle$   
**using**  $\mathcal{F}$ -subset by *blast*

**moreover have**  $\forall C'.\ Some\ (eres\ D\ E) = Some\ C' \longrightarrow C' |\in\ iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}')$

**proof** –

**have**  $eres\ D\ E\ |\in\ iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}')$

**using**  $\langle iefac\ \mathcal{F}\ (eres\ D\ E) = eres\ D\ E \rangle$

**by** (*simp add:  $\langle U_{er}' = finsert\ (eres\ D\ E)\ U_{er} \rangle$* )

**thus** *?thesis*

by *simp*  
 qed

**moreover have** *sorted-wrt* ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma'$   
**using**  $\Gamma$ -*sorted step-hyps*(9) **by** (*metis sorted-wrt-dropWhile*)

**moreover have**  $\Gamma'$ -*lower*: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma'$ ) (*atms-of-clss* ( $N \mid \cup \mid U_{er}'$ ))  
**proof** –  
**have** *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma'$ ) (*trail-atms*  $\Gamma$ )  
**unfolding** *linorder-trm.is-lower-set-iff*  
**proof** (*intro conjI ballI impI*)  
**show** *fset* (*trail-atms*  $\Gamma'$ )  $\subseteq$  *fset* (*trail-atms*  $\Gamma$ )  
**unfolding** *fset-trail-atms* **using**  $\langle \text{suffix } \Gamma' \Gamma \rangle$  **by** (*metis image-mono set-mono-suffix*)  
**next**  
**obtain**  $\Gamma''$  **where**  $\Gamma = \Gamma'' @ \Gamma'$   
**using**  $\langle \text{suffix } \Gamma' \Gamma \rangle$  **unfolding** *suffix-def* **by** *metis*

**fix**  $l x$   
**assume**  $l \mid \in \mid \text{trail-atms } \Gamma'$  **and**  $x \mid \in \mid \text{trail-atms } \Gamma$  **and**  $x \prec_t l$

**have**  $x \mid \in \mid \text{trail-atms } \Gamma'' \vee x \mid \in \mid \text{trail-atms } \Gamma'$   
**using**  $\langle x \mid \in \mid \text{trail-atms } \Gamma \rangle$  **unfolding**  $\langle \Gamma = \Gamma'' @ \Gamma' \rangle$  *fset-trail-atms* **by** *auto*

**thus**  $x \mid \in \mid \text{trail-atms } \Gamma'$   
**proof** (*elim disjE*)  
**assume**  $x \mid \in \mid \text{trail-atms } \Gamma''$

**hence**  $l \prec_t x$   
**using**  $\Gamma$ -*sorted*  $\langle l \mid \in \mid \text{trail-atms } \Gamma' \rangle$   
**unfolding**  $\langle \Gamma = \Gamma'' @ \Gamma' \rangle$  *sorted-wrt-append fset-trail-atms* **by** *blast*

**hence** *False*  
**using**  $\langle x \prec_t l \rangle$  **by** *order*

**thus**  $x \mid \in \mid \text{trail-atms } \Gamma' ..$   
**next**  
**assume**  $x \mid \in \mid \text{trail-atms } \Gamma'$   
**thus**  $x \mid \in \mid \text{trail-atms } \Gamma'$ .  
 qed  
 qed

**thus** *?thesis*  
**using**  $\Gamma$ -*lower*  
**unfolding**  $\langle \text{atms-of-clss } (N \mid \cup \mid U_{er}) = \text{atms-of-clss } (N \mid \cup \mid U_{er}') \rangle$   
**by** *order*  
 qed

**moreover have**  $\forall DE. \text{Some } (\text{eres } D \ E) = \text{Some } DE \longrightarrow (\forall A \mid \in \mid \text{trail-atms } \Gamma'. \exists L. \text{ord-res.is-maximal-lit } L \ DE \wedge A \preceq_t \text{atm-of } L)$   
**using**  $\text{atms-in-}\Gamma'\text{-lt-atm-}K \ \text{eres-max-lit}$  **by**  $\text{blast}$

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$   
**using**  $\Gamma\text{-deci-iff-neg } \langle \text{suffix } \Gamma' \ \Gamma \rangle$   
**by**  $(\text{metis } (\text{no-types}, \text{opaque-lifting}) \text{in-set-conv-decomp } \text{suffixE } \text{suffix-appendD})$

**moreover have**  $\text{trail-true-cls } \Gamma' \ C$   
**if**  $Ln \in \text{set } \Gamma'$  **and**  $\text{snd } Ln = \text{None}$  **and**  $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')$  **and**  $C \prec_c \{\#\text{fst } Ln\#\}$   
**for**  $Ln \ C$   
**proof**  $-$   
**have**  $Ln \in \text{set } \Gamma$   
**using**  $\langle Ln \in \text{set } \Gamma' \rangle \langle \text{suffix } \Gamma' \ \Gamma \rangle \text{set-mono-suffix}$  **by**  $\text{blast}$

**have**  $C = \text{eres } D \ E \vee C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') \rangle$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert } (\text{eres } D \ E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \rangle$   
**by**  $\text{simp}$

**thus**  $\text{trail-true-cls } \Gamma' \ C$   
**proof**  $(\text{elim } \text{disjE})$   
**assume**  $C = \text{eres } D \ E$

**hence**  $K \prec_l \text{fst } Ln$   
**using**  $\langle C \prec_c \{\#\text{fst } Ln\#\} \rangle \text{eres-max-lit}$   
**by**  $(\text{simp add: linorder-lit.is-maximal-in-mset-iff})$

**hence**  $\text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)$   
**by**  $(\text{cases } K; \text{cases } \text{fst } Ln) \text{simp-all}$

**moreover have**  $\text{atm-of } (\text{fst } Ln) \mid \in \mid \text{trail-atms } \Gamma'$   
**using**  $\langle Ln \in \text{set } \Gamma' \rangle$  **by**  $(\text{simp add: fset-trail-atms})$

**moreover have**  $\text{atm-of } K \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}')$   
**by**  $(\text{metis } \langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = \text{finsert } (\text{eres } D \ E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \rangle$   
 $\text{atm-of-in-atms-of-clsI } \text{eres-max-lit } \text{finsert-iff } \text{linorder-lit.is-maximal-in-mset-iff})$

**ultimately have**  $\text{atm-of } K \mid \in \mid \text{trail-atms } \Gamma'$   
**using**  $\Gamma'\text{-lower}$   
**unfolding**  $\text{linorder-trm.is-lower-set-iff}$   
**by**  $\text{fastforce}$

**moreover have**  $\text{atm-of } K \mid \notin \mid \text{trail-atms } \Gamma'$   
**using**  $\text{atms-in-}\Gamma'\text{-lt-atm-}K$  **by**  $\text{blast}$

**ultimately show** *trail-true-cls*  $\Gamma' C$   
**by contradiction**  
**next**  
**assume**  $C \in | \text{iefac } \mathcal{F} \text{ |}^\dagger (N \cup U_{er})$   
  
**hence** *trail-true-cls*  $\Gamma C$   
**using**  $\langle \Gamma\text{-deci-ball-lt-true } \langle Ln \in \text{set } \Gamma \rangle \langle \text{snd } Ln = \text{None} \rangle \langle C \prec_c \{\#fst Ln\# \} \rangle$   
**by metis**  
  
**then obtain**  $L_C$  **where**  $L_C \in \# C$  **and** *trail-true-lit*  $\Gamma L_C$   
**unfolding** *trail-true-cls-def* **by auto**  
  
**hence**  $\forall x \in \# C. x \prec_l fst Ln$   
**using**  $\langle C \prec_c \{\#fst Ln\# \} \rangle$   
**unfolding** *multp-singleton-right*[*OF linorder-lit.transp-on-less*]  
**by simp**  
  
**hence**  $L_C \prec_l fst Ln$   
**using**  $\langle L_C \in \# C \rangle$  **by metis**  
  
**hence** *atm-of*  $L_C \preceq_t \text{atm-of } (fst Ln)$   
**by** (*cases*  $L_C$ ; *cases*  $fst Ln$ ) *simp-all*  
  
**moreover have** *atm-of*  $(fst Ln) \prec_t \text{atm-of } K$   
**using** *atms-in- $\Gamma'$ -lt-atm-K*  
**by** (*simp add: fset-trail-atms that(1)*)  
  
**ultimately have** *atm-of*  $L_C \prec_t \text{atm-of } K$   
**by order**  
  
**have**  $L_C \in \text{fst } ' \text{set } \Gamma$   
**using**  $\langle \text{trail-true-lit } \Gamma L_C \rangle$   
**unfolding** *trail-true-lit-def* .  
  
**hence**  $L_C \in \text{fst } ' \text{set } \Gamma'$   
**using** *mem-set- $\Gamma'$ -iff*  
**using**  $\langle \text{atm-of } L_C \prec_t \text{atm-of } K \rangle$  *linorder-trm.not-le* **by auto**  
  
**hence** *trail-true-lit*  $\Gamma' L_C$   
**unfolding** *trail-true-lit-def* .  
  
**thus** *trail-true-cls*  $\Gamma' C$   
**using**  $\langle L_C \in \# C \rangle$   
**unfolding** *trail-true-cls-def* **by auto**  
  
**qed**  
**qed**  
  
**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \text{ |}^\dagger (N \cup U_{er})$



**using**  $\Gamma$ -prop-in  $\langle \text{suffix } \Gamma' \Gamma \rangle$  set-mono-suffix  $\langle U_{er}' = \text{finsert } (\text{eres } D \ E) \ U_{er} \rangle$   
**by** *blast*

**moreover have**  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C \ (\text{fst } Ln)$   
**using**  $\Gamma$ -prop-greatest  $\langle \text{suffix } \Gamma' \Gamma \rangle$  set-mono-suffix **by** *blast*

**moreover have**  $\forall \Gamma_1 \ L \ C \ \Gamma_0. \Gamma' = \Gamma_1 \ @ \ (L, \text{Some } C) \ # \ \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \ \{\#K \in \# \ C. \ K \neq L\# \}$   
**using**  $\Gamma$ -prop-almost-false  $\langle \text{suffix } \Gamma' \Gamma \rangle$   
**by** (*metis* (*no-types*, *lifting*) *append.assoc suffix-def*)

**moreover have**  $\forall \Gamma_1 \ L \ D \ \Gamma_0. \Gamma' = \Gamma_1 \ @ \ (L, \text{Some } D) \ # \ \Gamma_0 \longrightarrow$   
 $(\forall C \ |\in| \ \text{iefac } \mathcal{F} \ |\uparrow| \ (N \ |\cup| \ U_{er}'). \ C \prec_c \ D \longrightarrow \text{trail-true-cls } \Gamma_0 \ C)$   
**proof** (*intro allI impI ballI*)

**fix**  
 $\Gamma_1 \ \Gamma_0 :: ('f \ \text{gliteral} \times 'f \ \text{gclause option}) \ \text{list} \ \mathbf{and}$   
 $L :: 'f \ \text{gliteral} \ \mathbf{and}$   
 $C_0 \ C_1 :: 'f \ \text{gclause}$

**assume**  
 $\Gamma'$ -eq:  $\Gamma' = \Gamma_1 \ @ \ (L, \text{Some } C_1) \ # \ \Gamma_0 \ \mathbf{and}$   
 $C_0$ -in:  $C_0 \ |\in| \ \text{iefac } \mathcal{F} \ |\uparrow| \ (N \ |\cup| \ U_{er}') \ \mathbf{and}$   
 $C_0 \prec_c \ C_1$

**have**  $C_0 = \text{eres } D \ E \ \vee \ C_0 \ |\in| \ \text{iefac } \mathcal{F} \ |\uparrow| \ (N \ |\cup| \ U_{er})$   
**using**  $C_0$ -in  
**unfolding**  $\langle \text{iefac } \mathcal{F} \ |\uparrow| \ (N \ |\cup| \ U_{er}') = \text{finsert } (\text{eres } D \ E) \ (\text{iefac } \mathcal{F} \ |\uparrow| \ (N \ |\cup| \ U_{er})) \rangle$   
**by** *simp*

**thus** *trail-true-cls*  $\Gamma_0 \ C_0$   
**proof** (*elim disjE*)  
**assume**  $C_0 = \text{eres } D \ E$

**have**  $\neg \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } (L, \text{Some } C_1)) \ \mathbf{and} \ (L, \text{Some } C_1) \in \text{set } \Gamma$   
**unfolding** *atomize-conj*  
**using**  $\Gamma'$ -eq  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \ \Gamma \rangle$   
**using** *mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone[OF  $\Gamma$ -sorted mono-atms-lt]*  
**by** (*metis in-set-conv-decomp*)

**then have**  $\neg \text{atm-of } K \preceq_t \text{atm-of } L$   
**by** *simp*

**hence** *atm-of*  $L \prec_t \text{atm-of } K$   
**by** *order*

**moreover have** *linorder-lit.is-greatest-in-mset*  $C_1 \ L$   
**using**  $\Gamma$ -prop-greatest  $\langle (L, \text{Some } C_1) \in \text{set } \Gamma \rangle$  **by** *fastforce*

```

ultimately have False
  using ⟨C0 <c C1⟩
  by (metis Neg-atm-of-iff ⟨C0 = eres D E⟩ asympD eres-max-lit
      linorder-lit.is-greatest-in-mset-iff-is-maximal-and-count-eq-one
      linorder-lit.mulp-if-maximal-less-than-maximal literal.collapse(1)
      ord-res.asymp-less-cls ord-res.less-lit-simps(1) ord-res.less-lit-simps(4)
      step-hyps(11))

thus trail-true-cls Γ0 C0 ..
next
assume C0 |∈| iefac ℱ |↑| (N |∪| Uer)
thus trail-true-cls Γ0 C0
  using Γ-prop-ball-lt-true Γ'-eq ⟨C0 <c C1⟩
  by (metis (no-types, opaque-lifting) ⟨suffix Γ' Γ⟩ append-assoc suffixE)
qed
qed

ultimately show ?thesis
  unfolding ⟨s' = (Uer' , ℱ , Γ' , Some (eres D E))⟩
  proof (intro ord-res-7-invars.intros)
    have clause-true-in-Γ'-if: trail-true-cls Γ' x and
      ∧Kx. linorder-lit.is-maximal-in-mset x Kx ⇒
      ¬ (∃ A |∈| atms-of-cls (N |∪| Uer'). A <t atm-of Kx ∧ A |∉| trail-atms Γ')
    if x-lt: ∧DE. Some (eres D E) = Some DE → x <c DE and x-in: x |∈|
iefac ℱ |↑| (N |∪| Uer')
    for x
  proof -
    have x <c eres D E
      using x-lt by metis

    hence x ≠ eres D E
      by order

    hence x-in': x |∈| iefac ℱ |↑| (N |∪| Uer)
      using x-in ⟨iefac ℱ (eres D E) = eres D E⟩
      by (simp add: ⟨Uer' = finsert (eres D E) Uer⟩)

    have x <c E
      using ⟨x <c eres D E⟩ ⟨eres D E <c E⟩ by order

    have x-true: trail-true-cls Γ x
      using clauses-lt-E-true x-in' ⟨x <c E⟩ by metis

    have (− K, None) ∈ set Γ
      using ⟨trail-false-lit Γ K⟩
      using ⟨is-pos K⟩
      using Γ-deci-iff-neg
    by (metis is-pos-neg-not-is-pos map-of-SomeD map-of-eq-None-iff not-Some-eq)

```

*prod.collapse*  
*prod.inject trail-false-lit-def)*

**obtain**  $L_x$  **where**  $L_x \in\# x$  **and**  $L_x\text{-true}$ : *trail-true-lit*  $\Gamma L_x$   
**using**  $x\text{-true}$  **unfolding** *trail-true-cls-def* **by** *metis*

**moreover have**  $L_x \neq K$   
**using**  $\Gamma\text{-consistent}$   $\langle \text{trail-false-cls } \Gamma (eres D E) \rangle L_x\text{-true}$  *eres-max-lit*  
**by** (*metis linorder-lit.is-maximal-in-mset-iff not-trail-true-cls-and-trail-false-cls trail-true-cls-def*)

**moreover have**  $L_x \neq -K$   
**using** *eres-max-lit*  $\langle x \prec_c eres D E \rangle \langle L_x \neq K \rangle \langle L_x \in\# x \rangle$   
**by** (*smt (verit, del-insts) empty-iff linorder-cls.less-not-sym linorder-lit.ex-maximal-in-mset linorder-lit.is-maximal-in-mset-iff linorder-lit.less-trans linorder-lit.multip-if-maximal-less-that-maximal linorder-lit.neqE linorder-trm.not-less-iff-gr-or-eq literal.collapse(1) ord-res.less-lit-simps(4) set-mset-empty step-hyps(11) uminus-literal-def*)

**ultimately have**  $\text{atm-of } L_x \neq \text{atm-of } K$   
**by** (*simp add: atm-of-eq-atm-of*)

**moreover have**  $L_x \preceq_l K$   
**proof** (*rule linorder-lit.less-than-maximal-if-multip<sub>HO</sub>[OF eres-max-lit -  $\langle L_x \in\# x \rangle]$* )  
**show** *multip<sub>HO</sub>*  $(\prec_l) x (eres D E)$   
**using**  $\langle x \prec_c eres D E \rangle$   
**by** (*simp add: multip-imp-multip<sub>HO</sub>*)  
**qed**

**ultimately have**  $\text{atm-of } L_x \prec_t \text{atm-of } K$   
**by** (*cases L<sub>x</sub>; cases K*) *simp-all*

**hence** *trail-true-lit*  $\Gamma' L_x$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle$   
**using**  $L_x\text{-true}$   
**unfolding** *trail-true-lit-def*  
**using** *mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone[OF  $\Gamma\text{-sorted mono-atms-lt}$ ]*  
**by** (*metis (no-types, lifting) image-iff linorder-trm.not-le*)

**thus** *trail-true-cls*  $\Gamma' x$   
**using**  $\langle L_x \in\# x \rangle$  **unfolding** *trail-true-cls-def* **by** *metis*

**fix**  $K_x$   
**assume**  $x\text{-max-lit}$ : *ord-res.is-maximal-lit*  $K_x x$

**hence**  $K_x \preceq_l K$   
**using**  $\langle x \prec_c \text{eres } D \ E \rangle \text{eres-max-lit}$   
**using** *linorder-lit.multip-if-maximal-less-that-maximal* **by** *fastforce*

**hence**  $\text{atm-of } K_x \preceq_t \text{atm-of } K$   
**by** *(cases } K\_x; \text{cases } K) \text{simp-all}*

**have**  $A \in | \text{trail-atms } \Gamma'$   
**if**  $A\text{-lt: } A \prec_t \text{atm-of } K_x$  **and**  $A\text{-in: } A \in | \text{atms-of-clss } (N \cup | U_{er})$  **for**  $A$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle$   
**proof** *(rule in-trail-atms-dropWhileI)*  
**show**  $\text{sorted-wrt } (\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$   
**using**  $\Gamma\text{-sorted}$  .  
**next**  
**show**  $\text{monotone-on } (\text{set } \Gamma) (\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) (\lambda x y. y \leq x) (\lambda x. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } x))$   
**using** *mono-atms-lt* .  
**next**  
**show**  $\neg \text{atm-of } K \preceq_t A$   
**using**  $A\text{-lt } \langle \text{atm-of } K_x \preceq_t \text{atm-of } K \rangle$  **by** *order*  
**next**  
**have**  $\neg (\exists A \in | \text{atms-of-clss } (N \cup | U_{er}). A \prec_t \text{atm-of } K_x \wedge A \notin | \text{trail-atms } \Gamma)$   
**using**  $\text{no-undef-atm-lt-max-lit-if-lt-E } \langle x \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}) \rangle \langle x \prec_c E \rangle \text{x-max-lit}$   
**by** *metis*

**thus**  $A \in | \text{trail-atms } \Gamma$   
**using**  $A\text{-in } A\text{-lt}$  **by** *metis*  
**qed**

**thus**  $\neg (\exists A \in | \text{atms-of-clss } (N \cup | U_{er}'). A \prec_t \text{atm-of } K_x \wedge A \notin | \text{trail-atms } \Gamma')$   
**using**  $\langle \text{atms-of-clss } (N \cup | U_{er}) = \text{atms-of-clss } (N \cup | U_{er}') \rangle$  **by** *metis*  
**qed**

**thus**  
 $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}'). (\forall DE. \text{Some } (\text{eres } D \ E) = \text{Some } DE \longrightarrow C \prec_c DE) \longrightarrow$   
 $\text{trail-true-cls } \Gamma' \ C$   
 $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}'). (\forall DE. \text{Some } (\text{eres } D \ E) = \text{Some } DE \longrightarrow C \prec_c DE) \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K \ C \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \cup | U_{er}'). A \prec_t \text{atm-of } K \wedge A \notin | \text{trail-atms } \Gamma'))$   
**unfolding** *atomize-conj* **by** *metis*

**have**  $\text{trail-true-cls } \Gamma' \ \{\#K \in \# \ C. K \neq L_C \#\} \wedge \text{is-pos } L_C$   
**if**  $\text{Some } (\text{eres } D \ E) = \text{Some } DE$  **and**  
 $C\text{-in: } C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}') \text{ and}$

*C-lt*:  $C \prec_c DE$  **and**  
*C-max-lit*: *linorder-lit.is-maximal-in-mset*  $C L_C$  **and**  
*L<sub>C</sub>-undef*:  $\neg$  *trail-defined-lit*  $\Gamma' L_C$   
**for**  $DE C L_C$   
**proof** –  
**have**  $DE = \text{eres } D E$   
**using** *that by simp*  
**hence**  $C \prec_c \text{eres } D E$   
**using** *C-lt by order*  
**hence**  $C \neq \text{eres } D E$   
**by order**  
**hence**  $C \in | \text{iefac } \mathcal{F} |^\dagger (N \cup U_{er})$   
**using**  $C \text{-in } \langle \text{iefac } \mathcal{F} |^\dagger (N \cup U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} |^\dagger (N \cup U_{er})) \rangle$   
**by simp**  
**moreover have**  $C \prec_c E$   
**using**  $\langle C \prec_c \text{eres } D E \rangle \langle \text{eres } D E \prec_c E \rangle$  **by order**  
  
**have**  $L_C \preceq_l K$   
**using**  $\langle C \prec_c \text{eres } D E \rangle$  *C-max-lit eres-max-lit*  
**by** (*meson linorder-cls.dual-order.asym linorder-lit.leI linorder-lit.mulp-if-maximal-less-that-maximal*)  
**hence**  $L_C \prec_l K \vee L_C = K$   
**by simp**  
  
**thus** *?thesis*  
**proof** (*elim disjE*)  
**assume**  $L_C \prec_l K$   
**hence** *atm-of*  $L_C \prec_t \text{atm-of } K$   
**using** *is-pos K*  
**by** (*cases L<sub>C</sub>; cases K*) *simp-all*  
  
**have** *trail-defined-lit*  $\Gamma' L_C$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle$   
**proof** (*intro trail-defined-lit-dropWhileI ballI*)  
**show** *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$   
**using**  $\Gamma$ -*sorted* .  
**next**  
**show** *monotone-on* (*set*  $\Gamma$ )  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) (\lambda x y. y \leq x)$   
 $(\lambda x. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } x))$   
**using** *mono-atms-lt* .  
**next**  
**have**  $\neg \text{atm-of } K \preceq_t \text{atm-of } L_C$   
**using**  $\langle \text{atm-of } L_C \prec_t \text{atm-of } K \rangle$  **by order**  
**thus**  $\neg \text{atm-of } K \preceq_t \text{atm-of } L_C \wedge \neg \text{atm-of } K \preceq_t \text{atm-of } (\neg L_C)$   
**by simp**  
**next**  
**have** *trail-defined-lit*  $\Gamma K$

**using**  $\langle \text{trail-false-lit } \Gamma \ K \rangle$   
**using**  $\text{trail-defined-lit-iff-true-or-false}$  **by**  $\text{blast}$   
**moreover have**  $\text{atm-of } K \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**by**  $(\text{metis } \langle \text{atms-of-clss } (N \mid \cup \mid U_{er}) = \text{atms-of-clss } (N \mid \cup \mid U_{er}') \rangle$   
 $\langle \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}') = \text{finsert } (\text{eres } D \ E) \ (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid$   
 $U_{er})) \rangle$   
 $\text{atm-of-in-atms-of-clssI } \text{eres-max-lit } \text{finsertI1 } \text{linorder-lit.is-maximal-in-mset-iff})$   
**moreover have**  $\text{atm-of } L_C \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using**  $\langle C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \rangle$   $C\text{-max-lit } \text{atm-of-in-atms-of-clssI}$   
**unfolding**  $\text{linorder-lit.is-maximal-in-mset-iff}$   
**by**  $\text{metis}$   
**ultimately show**  $\text{trail-defined-lit } \Gamma \ L_C$   
**using**  $\langle \text{atm-of } L_C \prec_t \text{atm-of } K \rangle$   $\Gamma\text{-lower}$   
**unfolding**  $\text{trail-defined-lit-iff-trail-defined-atm}$   
**by**  $(\text{meson } \text{linorder-trm.is-lower-set-iff})$   
**qed**

**hence**  $\text{False}$   
**using**  $L_C\text{-undef}$  **by**  $\text{contradiction}$

**thus**  $?thesis ..$

**next**

**have**  $\text{trail-true-cls } \Gamma' \ C$

**using**  $\text{clause-true-in-}\Gamma'\text{-if } C\text{-in } \langle C \prec_c \text{eres } D \ E \rangle$  **by**  $\text{blast}$

**hence**  $\text{trail-true-cls } \Gamma' \ \{\#K \in \# \ C. \ K \neq L_C \#\}$

**using**  $L_C\text{-undef}$

**by**  $(\text{smt } (\text{verit}, \text{best}) \ \text{mem-Collect-eq } \text{set-mset-filter } \text{trail-defined-lit-iff-true-or-false}$   
 $\text{trail-true-cls-def})$

**moreover assume**  $L_C = K$

**ultimately show**  $\text{trail-true-cls } \Gamma' \ \{\#K \in \# \ C. \ K \neq L_C \#\} \wedge \text{is-pos } L_C$

**using**  $\langle \text{is-pos } K \rangle$  **by**  $\text{metis}$

**qed**

**qed**

**thus**  $\forall Da. \ \text{Some } (\text{eres } D \ E) = \text{Some } Da \longrightarrow$

$(\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}'). \ C \prec_c \ Da \longrightarrow (\forall L_C. \ \text{ord-res.is-maximal-lit}$   
 $L_C \ C \longrightarrow$

$\neg \text{trail-defined-lit } \Gamma' \ L_C \longrightarrow \text{trail-true-cls } \Gamma' \ \{\#K \in \# \ C. \ K \neq L_C \#\} \wedge$   
 $\text{is-pos } L_C))$

**by**  $\text{metis}$

**qed**  $\text{simp-all}$

**next**

**case**  $\text{step-hyps: } (\text{resolution-neg } \Gamma \ E \ L \ D \ U_{er}' \ U_{er} \ \Gamma' \ K \ C \ \mathcal{F})$

**note**  $E\text{-max-lit} = \langle \text{ord-res.is-maximal-lit } L \ E \rangle$

**note**  $\text{eres-max-lit} = \langle \text{ord-res.is-maximal-lit } K \ (\text{eres } D \ E) \rangle$

**have**

$\mathcal{F}\text{-subset: } \mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er}$  **and**

***E-in***:  $E \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
***clauses-lt-D-seldomly-have-undef-max-lit***:  
 $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow$   
 $(\forall L_C. \text{linorder-lit.is-maximal-in-mset } C L_C \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma L_C \longrightarrow (\text{trail-true-cls } \Gamma \{ \#K \in \# C. K \neq L_C \# \} \wedge$   
***is-pos } L\_C)) **and**  
***clauses-lt-E-true***:  $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow \text{trail-true-cls } \Gamma$   
***C*** **and**  
***no-undef-atm-lt-max-lit-if-lt-E***:  $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg (\exists A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma))$   
**and**  
***Γ-sorted***:  $\text{sorted-wrt } (\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$  **and**  
***Γ-lower***:  $\text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**and**  
***trail-atms-le0***:  $\forall A \in | \text{trail-atms } \Gamma. \exists L. \text{ord-res.is-maximal-lit } L E \wedge (A \preceq_t$   
***atm-of } L) **and**  
***Γ-deci-iff-neg***:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longleftrightarrow \text{is-neg } (fst Ln)$  **and**  
***Γ-deci-ball-lt-true***:  $\forall Ln \in \text{set } \Gamma. \text{snd } Ln = \text{None} \longrightarrow$   
 $(\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \{ \#fst Ln \# \} \longrightarrow \text{trail-true-cls } \Gamma C)$   
**and**  
***Γ-prop-in***:  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid$   
***U<sub>er</sub>)*** **and**  
***Γ-prop-greatest***:  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
***(fst Ln)*** **and**  
***Γ-prop-almost-false***:  
 $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{ \#K \in \# C.$   
***K \neq L \# \}*** **and**  
***Γ-prop-ball-lt-true***:  $(\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C))$  **and**  
***Γ-prop-ball-lt-true***:  $\forall \Gamma_1 L D \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C)$   
**using invar by** (*simp-all add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) \rangle \text{ord-res-}\gamma\text{-invars-def}$ )  
  
**have** *clauses-lt-E-almost-defined*:  $\forall C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow \text{trail-defined-cls } \Gamma \{ \#L \in \# C.$   
***L \neq K \# \}*)  
**using invar**[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) \rangle$ ] *clause-almost-defined-if-lt-next-clause*  
**by** *simp*  
  
**have**  $\Gamma$ -*consistent*: *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms*  $\Gamma$ -*sorted* **by** *metis*  
  
**have** *trail-atms-le*:  $\forall A \in | \text{trail-atms } \Gamma. A \preceq_t \text{atm-of } L$   
**using** *trail-atms-le0 E-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)  
  
**have**  $(- L, \text{Some } D) \in \text{set } \Gamma$********

**using**  $\langle \text{map-of } \Gamma (- L) = \text{Some } (\text{Some } D) \rangle$  **by**  $(\text{metis map-of-SomeD})$

**hence**  $D\text{-in: } D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\Gamma\text{-prop-in}$  **by**  $\text{simp}$

**have**  $D\text{-max-lit: linorder-lit.is-greatest-in-mset } D (- L)$   
**using**  $\Gamma\text{-prop-greatest } \langle (- L, \text{Some } D) \in \text{set } \Gamma \rangle$  **by**  $\text{fastforce}$

**have**  $D \prec_c E$   
**using**  $\text{clause-lt-clause-if-max-lit-comp}$   
**using**  $E\text{-max-lit } \langle \text{is-neg } L \rangle D\text{-max-lit}$   
**by**  $(\text{metis linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset})$

**have**  $\text{eres } D E \prec_c E$   
**using**  $\text{eres-lt-if}$   
**using**  $E\text{-max-lit } \langle \text{is-neg } L \rangle D\text{-max-lit}$  **by**  $\text{metis}$

**hence**  $\text{eres } D E \neq E$   
**by**  $\text{order}$

**have**  $(- K, \text{Some } C) \in \text{set } \Gamma$   
**using**  $\langle \text{map-of } \Gamma (- K) = \text{Some } (\text{Some } C) \rangle$  **by**  $(\text{metis map-of-SomeD})$

**hence**  $C\text{-in: } C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\Gamma\text{-prop-in}$  **by**  $\text{simp}$

**have**  $C\text{-max-lit: linorder-lit.is-greatest-in-mset } C (- K)$   
**using**  $\Gamma\text{-prop-greatest } \langle (- K, \text{Some } C) \in \text{set } \Gamma \rangle$  **by**  $\text{fastforce}$

**hence**  $C \prec_c \text{eres } D E$   
**using**  $\langle \text{ord-res.is-maximal-lit } K (\text{eres } D E) \rangle \langle \text{is-neg } L \rangle$   
**by**  $(\text{metis Neg-atm-of-iff Pos-atm-of-iff linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset}$   
 $\text{linorder-lit.mulp-if-maximal-less-that-maximal linorder-lit.not-less-iff-gr-or-eq}$   
 $\text{linorder-trm.less-irrefl ord-res.less-lit-simps}(4) \text{ step-hyps}(11) \text{ uminus-Neg})$

**have**  $\text{suffix } \Gamma' \Gamma$   
**using**  $\text{step-hyps}(9) \text{ suffix-dropWhile}$  **by**  $\text{metis}$

**hence**  $\text{atms-of-cls } (\text{eres } D E) \mid \subseteq \mid \text{atms-of-cls } D \mid \cup \mid \text{atms-of-cls } E$   
**using**  $\text{lit-in-one-of-resolvents-if-in-eres}$   
**unfolding**  $\text{atms-of-cls-def}$  **by**  $\text{fastforce}$

**moreover have**  $\text{atms-of-cls } D \mid \subseteq \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using**  $D\text{-in}$   
**by**  $(\text{metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb funion-upper1})$

**moreover have**  $\text{atms-of-cls } E \mid \subseteq \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using**  $E\text{-in}$   
**by**  $(\text{metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb funion-upper1})$



**ultimately have**  $atms\text{-of-clss } (N \mid \cup \mid U_{er}) = atms\text{-of-clss } (N \mid \cup \mid U_{er}')$   
**unfolding**  $\langle U_{er}' = finsert (eres D E) U_{er} \rangle$  *atms-of-clss-def* **by** *auto*

**have**  $L \in \# E$   
**using** *E-max-lit* **unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *metis*

**hence**  $- L \notin \# E$   
**using** *not-both-lit-and-comp-lit-in-false-clause-if-trail-consistent*  
**using**  $\Gamma$ -*consistent*  $\langle trail\text{-false-cls } \Gamma E \rangle$  **by** *metis*

**hence**  $\forall K \in \# eres D E. atm\text{-of } K \prec_t atm\text{-of } L$   
**using** *lit-in-eres-lt-greatest-lit-in-greatest-resolvent* [*OF*  $\langle eres D E \neq E \rangle$  *E-max-lit*]  
**by** *metis*

**hence**  $\forall K \in \# eres D E. K \neq L \wedge K \neq - L$   
**by** *fastforce*

**moreover have**  $\forall L \in \# eres D E. L \in \# D \vee L \in \# E$   
**using** *lit-in-one-of-resolvents-if-in-eres* **by** *metis*

**moreover have** *D-almost-false: trail-false-cls*  $\Gamma \{ \#K \in \# D. K \neq - L \# \}$   
**using** *ord-res-7-invars-implies-propagated-clause-almost-false*  
**using**  $\langle (- L, Some D) \in set \Gamma \rangle$  *invar*[*unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, Some E) \rangle$ ]  
**by** *metis*

**ultimately have** *trail-false-cls*  $\Gamma (eres D E)$   
**using**  $\langle trail\text{-false-cls } \Gamma E \rangle$  **unfolding** *trail-false-cls-def* **by** *fastforce*

**have**  $eres D E \not\subseteq N \mid \cup \mid U_{er}$   
**using** *eres-not-in-known-clauses-if-trail-false-cls*  
**using**  $\Gamma$ -*consistent* *clauses-lt-E-true*  $\langle eres D E \prec_c E \rangle$   $\langle trail\text{-false-cls } \Gamma (eres D E) \rangle$  **by** *metis*

**hence**  $eres D E \not\subseteq \mathcal{F}$   
**using** *F-subset* **by** *blast*

**hence**  $iefac \mathcal{F} (eres D E) = eres D E$   
**by** (*simp add: iefac-def*)

**hence**  $iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}') = finsert (eres D E) (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))$   
**by** (*simp add:*  $\langle U_{er}' = finsert (eres D E) U_{er} \rangle$ )

**have** *mem-set- $\Gamma'$ -iff*:  $(Ln \in set \Gamma') = (\neg atm\text{-of } K \preceq_t atm\text{-of } (fst Ln) \wedge Ln \in set \Gamma)$  **for**  $Ln$   
**unfolding**  $\langle \Gamma' = dropWhile (\lambda Ln. atm\text{-of } K \preceq_t atm\text{-of } (fst Ln)) \Gamma \rangle$   
**proof** (*rule mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone*)  
**show** *sorted-wrt*  $(\lambda x y. atm\text{-of } (fst y) \prec_t atm\text{-of } (fst x)) \Gamma$   
**using**  $\Gamma$ -*sorted* .

**next**  
**show** *monotone-on* (*set*  $\Gamma$ ) ( $\lambda x y. \text{atm-of } (fst\ y) \prec_t \text{atm-of } (fst\ x)$ ) ( $\lambda x y. y \leq x$ )  
 $(\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (fst\ Ln))$   
**by** (*rule monotone-onI*) *auto*  
**qed**

**have** *atms-in- $\Gamma'$ -lt-atm-K*:  $\forall A \in | \text{trail-atms } \Gamma'. A \prec_t \text{atm-of } K$   
**proof** –  
**have**  $\exists L. \text{ord-res.is-maximal-lit } L\ C \wedge x \prec_t \text{atm-of } L$  **if**  $x \in | \text{trail-atms } \Gamma'$  **for**  $x$   
**proof** (*intro exI conjI*)  
**show** *ord-res.is-maximal-lit* ( $- K$ )  $C$   
**using** *C-max-lit* **by** *blast*  
**next**  
**show**  $x \prec_t \text{atm-of } (- K)$   
**using**  $\langle x \in | \text{trail-atms } \Gamma' \rangle \text{mem-set-}\Gamma'\text{-iff}$  **unfolding** *fset-trail-atms* **by** *fastforce*  
**qed**

**hence**  $\forall A \in | \text{trail-atms } \Gamma'. A \prec_t \text{atm-of } (- K)$   
**using** *C-max-lit*  
**by** (*metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)

**thus** *?thesis*  
**by** (*metis atm-of-uminus*)  
**qed**

**have**  $\mathcal{F} \subseteq | N \cup | U_{er}'$   
**unfolding**  $\langle U_{er}' = \text{finsert } (eres\ D\ E)\ U_{er} \rangle$   
**using** *F-subset* **by** *blast*

**moreover have**  $\forall C'. \text{Some } C = \text{Some } C' \longrightarrow C' \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}')$   
**using** *C-in* **by** (*simp add:  $\langle U_{er}' = \text{finsert } (eres\ D\ E)\ U_{er} \rangle$* )

**moreover have**  
 $\bigwedge K_x. \text{linorder-lit.is-maximal-in-mset } x\ K_x \implies \text{trail-defined-cls } \Gamma' \{ \#L \in \# x. L \neq K_x \# \}$  **and**  
*clause-true-in- $\Gamma'$ -if: trail-true-cls  $\Gamma'$   $x$*   
**if** *x-lt*:  $\bigwedge y. \text{Some } C = \text{Some } y \implies x \prec_c y$  **and** *x-in*:  $x \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup | U_{er}')$  **for**  $x$   
**proof** –  
**have**  $x \prec_c C$   
**using** *x-lt* **by** *metis*

**hence**  $x \prec_c eres\ D\ E$   
**using**  $\langle C \prec_c eres\ D\ E \rangle$  **by** *order*

**hence**  $x \neq \text{eres } D \ E$   
**by** *order*

**have**  $x \prec_c \ E$   
**using**  $\langle x \prec_c \ \text{eres } D \ E \rangle \langle \text{eres } D \ E \prec_c \ E \rangle$  **by** *order*

**moreover have**  $x\text{-in}' : x \in | \text{iefac } \mathcal{F} \ |^{\dagger} (N \ | \cup \ | \ U_{er})$   
**using**  $x\text{-in} \ \langle x \neq \text{eres } D \ E \rangle$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \ |^{\dagger} (N \ | \cup \ | \ U_{er}') = \text{finsert} (\text{eres } D \ E) (\text{iefac } \mathcal{F} \ |^{\dagger} (N \ | \cup \ | \ U_{er})) \rangle$   
**by** *simp*

**ultimately have**  $x\text{-true} : \text{trail-true-cls } \Gamma \ x$   
**using** *clauses-lt-E-true* **by** *metis*

**then obtain**  $L_x$  **where**  $L_x \in \# \ x$  **and**  $\text{trail-true-lit } \Gamma \ L_x$   
**unfolding** *trail-true-cls-def* **by** *metis*

**have**  $L_x \preceq_l \ - \ K$   
**using** *C-max-lit*  $\langle x \prec_c \ C \rangle \langle L_x \in \# \ x \rangle$   
**by** (*smt* (*verit*, *ccfv-threshold*) *asymptD empty-not-add-mset ord-res.transp-less-lit insert-DiffM*  
*linorder-lit.is-greatest-in-set-iff linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*  
*linorder-lit.is-maximal-in-mset-iff linorder-lit.is-maximal-in-set-eq-is-greatest-in-set*  
*linorder-lit.is-maximal-in-set-iff linorder-lit.leI ord-res.asymp-less-cls*  
*ord-res.multip-if-all-left-smaller transpE*)

**have**  $\text{mono-atms-lt} : \text{monotone-on } (\text{set } \Gamma) (\lambda x \ y. \ \text{atm-of } (\text{fst } y) \prec_t \ \text{atm-of } (\text{fst } x)) (\lambda x \ y. \ y \leq x)$   
 $(\lambda x. \ \text{atm-of } K \preceq_t \ \text{atm-of } (\text{fst } x))$   
**proof** (*intro monotone-onI*, *unfold le-bool-def*, *intro impI*)  
**fix**  $x \ y$   
**assume**  $\text{atm-of } (\text{fst } y) \prec_t \ \text{atm-of } (\text{fst } x)$  **and**  $\text{atm-of } K \preceq_t \ \text{atm-of } (\text{fst } y)$   
**thus**  $\text{atm-of } K \preceq_t \ \text{atm-of } (\text{fst } x)$   
**by** *order*

**qed**

**obtain**  $\Gamma_1 \ \Gamma_0$  **where**  $\Gamma\text{-eq} : \Gamma = \Gamma_1 \ @ \ (- \ K, \ \text{Some } C) \ \# \ \Gamma_0$   
**using**  $\langle (- \ K, \ \text{Some } C) \in \text{set } \Gamma \rangle$  **by** (*metis split-list*)

**hence**  $\text{trail-true-cls } \Gamma_0 \ x$   
**using**  $\Gamma\text{-prop-ball-lt-true } x\text{-in}' \ \langle x \prec_c \ C \rangle$  **by** *metis*

**then obtain**  $L_x$  **where**  $L_x \in \# \ x$  **and**  $L_x\text{-true} : \text{trail-true-lit } \Gamma_0 \ L_x$   
**unfolding** *trail-true-cls-def* **by** *auto*

**moreover have**  $\Gamma' = \Gamma_0$   
**proof** –  
**have**  $\Gamma' = \text{dropWhile } (\lambda Ln. \ \text{atm-of } K \preceq_t \ \text{atm-of } (\text{fst } Ln)) ((\Gamma_1 \ @ \ [(- \ K,$

*Some C*)) @  $\Gamma_0$ )  
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle \Gamma\text{-eq}$   
**by** *simp*

**also have**  $\dots = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma_0$   
**proof** (*rule dropWhile-append2*)  
**fix**  $Ln :: 'f \text{ gterm literal} \times 'f \text{ gclause option}$   
**assume**  $Ln \in \text{set } (\Gamma_1 @ [(- K, \text{Some } C)])$

**moreover have**  $\forall x \in \text{set } \Gamma_1. \text{atm-of } K \prec_t \text{atm-of } (\text{fst } x)$   
**using**  $\Gamma\text{-sorted}$  **by** (*simp add: \Gamma-eq sorted-wrt-append*)

**ultimately show**  $\text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)$   
**using**  $\langle \text{is-neg } K \rangle$  **by** *auto*  
**qed**

**also have**  $\dots = \Gamma_0$   
**proof** (*cases \Gamma\_0*)  
**case** *Nil*  
**thus** *?thesis*  
**by** (*simp add: dropWhile-eq-self-iff*)  
**next**  
**case** (*Cons Ln \Gamma\_0'*)

**hence**  $\text{atm-of } (\text{fst } Ln) \prec_t \text{atm-of } K$   
**using**  $\Gamma\text{-sorted}$  **by** (*simp add: \Gamma-eq sorted-wrt-append*)

**hence**  $\neg \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)$   
**by** *order*

**hence**  $\neg \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } (\text{hd } \Gamma_0))$   
**by** (*simp add: \langle \Gamma\_0 = Ln \# \Gamma\_0' \rangle*)

**thus** *?thesis*  
**by** (*simp add: dropWhile-eq-self-iff*)  
**qed**

**finally show** *?thesis* .  
**qed**

**ultimately have** *trail-true-lit*  $\Gamma' L_x$   
**by** *argo*

**thus** *trail-true-cls*  $\Gamma' x$   
**using**  $\langle L_x \in \# x \rangle$  **unfolding** *trail-true-cls-def* **by** *metis*

**fix**  $K_x$  **assume** *x-max-lit: ord-res.is-maximal-lit*  $K_x x$   
**show** *trail-defined-cls*  $\Gamma' \{\#L \in \# x. L \neq K_x \#\}$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle$

**proof** (*intro trail-defined-cls-dropWhileI ballI*)  
**show** *sorted-wrt* ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma$   
**using**  $\Gamma$ -sorted .  
**next**  
**show** *monotone-on* (*set*  $\Gamma$ ) ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ ) ( $\lambda x y. y$   
 $\leq x$ )  
( $\lambda x. \text{atm-of } K \preceq_t \text{atm-of } (fst x)$ )  
**using** *mono-atms-lt* .  
**next**  
**fix**  $L_x$   
**assume**  $L_x \in\# \{\#L \in\# x. L \neq K_x\#$   
  
**hence**  $L_x \in\# x$  **and**  $L_x \neq K_x$   
**by** *simp-all*  
  
**hence**  $L_x \prec_l K_x$   
**using**  $x$ -max-lit  $\langle L_x \in\# x \rangle \langle L_x \neq K_x \rangle$  **unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**by** *fastforce*  
  
**moreover have**  $K_x \preceq_l K$   
**using**  $\langle x \prec_c \text{eres } D E \rangle$   
**using** *linorder-lit.mulp-if-maximal-less-that-maximal*[*OF eres-max-lit x-max-lit*]  
**by** *fastforce*  
  
**ultimately have** *atm-of*  $L_x \prec_t \text{atm-of } K$   
**using**  $\langle \text{is-neg } K \rangle$   
**by** (*metis C-max-lit Pos-atm-of-iff*  $\langle x \prec_c C \rangle$  *linorder-cls.dual-order.asym*  
*linorder-lit.dual-order.strict-trans1*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset* *linorder-lit.less-le-not-le*  
*linorder-lit.mulp-if-maximal-less-that-maximal* *linorder-trm.linorder-cases*  
*literal.collapse(2)* *ord-res.less-lit-simps(3)* *ord-res.less-lit-simps(4)* *umi-*  
*nus-Neg*  
*x-max-lit*)  
  
**hence**  $\neg \text{atm-of } K \preceq_t \text{atm-of } L_x$   
**by** *order*  
  
**thus**  $\neg \text{atm-of } K \preceq_t \text{atm-of } L_x \wedge \neg \text{atm-of } K \preceq_t \text{atm-of } (- L_x)$   
**unfolding** *atm-of-uminus conj-absorb* .  
**next**  
**show** *trail-defined-cls*  $\Gamma \{\#L \in\# x. L \neq K_x\#$   
**using** *clauses-lt-E-almost-defined x-in'*  $\langle x \prec_c E \rangle$   $x$ -max-lit **by** *metis*  
**qed**  
**qed**  
  
**moreover have** *sorted-wrt* ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma'$   
**using**  $\Gamma$ -sorted *step-hyps(9)* **by** (*metis sorted-wrt-dropWhile*)  
  
**moreover have**  $\Gamma'$ -lower: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma'$ ) (*atms-of-cls*

```

( $N \mid \cup \mid U_{er}$ )
proof -
  have linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (trail-atms  $\Gamma$ )
    unfolding linorder-trm.is-lower-set-iff
  proof (intro conjI ballI impI)
    show fset (trail-atms  $\Gamma'$ )  $\subseteq$  fset (trail-atms  $\Gamma$ )
      unfolding fset-trail-atms using  $\langle$ suffix  $\Gamma'$   $\Gamma$  $\rangle$  by (metis image-mono
set-mono-suffix)
  next
    obtain  $\Gamma''$  where  $\Gamma = \Gamma'' @ \Gamma'$ 
      using  $\langle$ suffix  $\Gamma'$   $\Gamma$  $\rangle$  unfolding suffix-def by metis

  fix  $l\ x$ 
  assume  $l \mid \in \mid$  trail-atms  $\Gamma'$  and  $x \mid \in \mid$  trail-atms  $\Gamma$  and  $x \prec_t l$ 

  have  $x \mid \in \mid$  trail-atms  $\Gamma'' \vee x \mid \in \mid$  trail-atms  $\Gamma'$ 
    using  $\langle$  $x \mid \in \mid$  trail-atms  $\Gamma$  $\rangle$  unfolding  $\langle$  $\Gamma = \Gamma'' @ \Gamma'$  $\rangle$  fset-trail-atms by auto

  thus  $x \mid \in \mid$  trail-atms  $\Gamma'$ 
  proof (elim disjE)
    assume  $x \mid \in \mid$  trail-atms  $\Gamma''$ 

    hence  $l \prec_t x$ 
      using  $\Gamma$ -sorted  $\langle$  $l \mid \in \mid$  trail-atms  $\Gamma'$  $\rangle$ 
      unfolding  $\langle$  $\Gamma = \Gamma'' @ \Gamma'$  $\rangle$  sorted-wrt-append fset-trail-atms by blast

    hence False
      using  $\langle$  $x \prec_t l$  $\rangle$  by order

    thus  $x \mid \in \mid$  trail-atms  $\Gamma'$  ..
  next
    assume  $x \mid \in \mid$  trail-atms  $\Gamma'$ 
    thus  $x \mid \in \mid$  trail-atms  $\Gamma'$  .
  qed
qed

  thus ?thesis
    using  $\Gamma$ -lower
    unfolding  $\langle$ atms-of-clss ( $N \mid \cup \mid U_{er}$ ) = atms-of-clss ( $N \mid \cup \mid U_{er}'$ ) $\rangle$ 
    by order
qed

moreover have  $\forall DE$ . Some  $C = \text{Some } DE \longrightarrow (\forall A \mid \in \mid$  trail-atms  $\Gamma'$ .
 $\exists L$ . ord-res.is-maximal-lit  $L\ DE \wedge A \preceq_t \text{atm-of } L)$ 
  using atms-in- $\Gamma'$ -lt-atm-K C-max-lit by fastforce

moreover have  $\forall Ln \in \text{set } \Gamma'$ . snd  $Ln = \text{None} \longleftrightarrow \text{is-neg } (\text{fst } Ln)$ 
  using  $\Gamma$ -deci-iff-neg  $\langle$ suffix  $\Gamma'$   $\Gamma$  $\rangle$ 
  by (metis (no-types, opaque-lifting) in-set-conv-decomp suffixE suffix-appendD)

```

**moreover have** *trail-true-cls*  $\Gamma' C$   
**if**  $L_n \in \text{set } \Gamma'$  **and**  $\text{snd } L_n = \text{None}$  **and**  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $C \prec_c \{\#fst L_n\}$   
**for**  $L_n C$   
**proof** –  
**have**  $L_n \in \text{set } \Gamma$   
**using**  $\langle L_n \in \text{set } \Gamma' \rangle \langle \text{suffix } \Gamma' \Gamma \rangle$  *set-mono-suffix* **by** *blast*  
  
**have**  $C = \text{eres } D E \vee C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\langle C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \rangle$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \rangle$   
**by** *simp*  
  
**thus** *trail-true-cls*  $\Gamma' C$   
**proof** (*elim disjE*)  
**assume**  $C = \text{eres } D E$   
  
**hence**  $K \prec_t fst L_n$   
**using**  $\langle C \prec_c \{\#fst L_n\} \rangle$  *eres-max-lit*  
**by** (*simp add: linorder-lit.is-maximal-in-mset-iff*)  
  
**hence** *atm-of*  $K \preceq_t \text{atm-of } (fst L_n)$   
**by** (*cases K; cases fst L\_n*) *simp-all*  
  
**moreover have** *atm-of*  $(fst L_n) \in | \text{trail-atms } \Gamma'$   
**using**  $\langle L_n \in \text{set } \Gamma' \rangle$  **by** (*simp add: fset-trail-atms*)  
  
**moreover have** *atm-of*  $K \in | \text{atms-of-cls } (N \mid \cup \mid U_{er})$   
**by** (*metis*  $\langle \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \rangle$   
*atm-of-in-atms-of-clsI eres-max-lit finsert-iff linorder-lit.is-maximal-in-mset-iff*)  
  
**ultimately have** *atm-of*  $K \in | \text{trail-atms } \Gamma'$   
**using**  $\Gamma'$ -*lower*  
**unfolding** *linorder-trm.is-lower-set-iff*  
**by** *fastforce*  
  
**moreover have** *atm-of*  $K \notin | \text{trail-atms } \Gamma'$   
**using** *atms-in- $\Gamma'$ -lt-atm-K* **by** *blast*  
  
**ultimately show** *trail-true-cls*  $\Gamma' C$   
**by** *contradiction*  
**next**  
**assume**  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
  
**hence** *trail-true-cls*  $\Gamma C$   
**using**  $\Gamma$ -*deci-ball-lt-true*  $\langle L_n \in \text{set } \Gamma \rangle \langle \text{snd } L_n = \text{None} \rangle \langle C \prec_c \{\#fst L_n\} \rangle$

by *metis*

then obtain  $L_C$  where  $L_C \in \# C$  and *trail-true-lit*  $\Gamma L_C$   
unfolding *trail-true-cls-def* by *auto*

hence  $\forall x \in \# C. x \prec_l \text{fst } Ln$   
using  $\langle C \prec_c \{\# \text{fst } Ln\} \rangle$   
unfolding *multp-singleton-right*[*OF linorder-lit.transp-on-less*]  
by *simp*

hence  $L_C \prec_l \text{fst } Ln$   
using  $\langle L_C \in \# C \rangle$  by *metis*

hence *atm-of*  $L_C \preceq_t \text{atm-of } (\text{fst } Ln)$   
by (*cases*  $L_C$ ; *cases*  $\text{fst } Ln$ ) *simp-all*

moreover have *atm-of*  $(\text{fst } Ln) \prec_t \text{atm-of } K$   
using *atms-in- $\Gamma'$ -lt-atm- $K$*   
by (*simp add: fset-trail-atms that*(1))

ultimately have *atm-of*  $L_C \prec_t \text{atm-of } K$   
by *order*

have  $L_C \in \text{fst } \text{'set } \Gamma$   
using  $\langle \text{trail-true-lit } \Gamma L_C \rangle$   
unfolding *trail-true-lit-def* .

hence  $L_C \in \text{fst } \text{'set } \Gamma'$   
using *mem-set- $\Gamma'$ -iff*  
using  $\langle \text{atm-of } L_C \prec_t \text{atm-of } K \rangle$  *linorder-trm.not-le* by *auto*

hence *trail-true-lit*  $\Gamma' L_C$   
unfolding *trail-true-lit-def* .

thus *trail-true-cls*  $\Gamma' C$   
using  $\langle L_C \in \# C \rangle$   
unfolding *trail-true-cls-def* by *auto*

qed  
qed

moreover have  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \cup U_{er})$   
using  $\Gamma\text{-prop-in } \langle \text{suffix } \Gamma' \Gamma \rangle \text{set-mono-suffix } \langle U_{er}' = \text{finsert } (\text{eres } D E) U_{er} \rangle$   
by *blast*

moreover have  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C (\text{fst } Ln)$   
using  $\Gamma\text{-prop-greatest } \langle \text{suffix } \Gamma' \Gamma \rangle \text{set-mono-suffix}$  by *blast*



**moreover have**  $\forall \Gamma_1 L C \Gamma_0. \Gamma' = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls}$   
 $\Gamma_0 \{ \#K \in \# C. K \neq L \# \}$   
**using**  $\Gamma\text{-prop-almost-false} \langle \text{suffix } \Gamma' \Gamma \rangle$   
**by**  $(\text{metis (no-types, lifting) append.assoc suffix-def})$

**moreover have**  $\forall \Gamma_1 L D \Gamma_0. \Gamma' = \Gamma_1 @ (L, \text{Some } D) \# \Gamma_0 \longrightarrow$   
 $(\forall C |\in| \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \rangle. C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma_0 C)$   
**proof**  $(\text{intro allI impI ballI})$   
**fix**  
 $\Gamma_1 \Gamma_0 :: ('f \text{ gliteral} \times 'f \text{ gclause option}) \text{ list and}$   
 $L :: 'f \text{ gliteral and}$   
 $C_0 C_1 :: 'f \text{ gclause}$   
**assume**  
 $\Gamma'\text{-eq}: \Gamma' = \Gamma_1 @ (L, \text{Some } C_1) \# \Gamma_0 \text{ and}$   
 $C_0\text{-in}: C_0 |\in| \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \text{ and}$   
 $C_0 \prec_c C_1$

**have**  $C_0 = \text{eres } D E \vee C_0 |\in| \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er})$   
**using**  $C_0\text{-in}$   
**unfolding**  $\langle \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er})) \rangle$   
**by**  $\text{simp}$

**thus**  $\text{trail-true-cls } \Gamma_0 C_0$   
**proof**  $(\text{elim disjE})$   
**assume**  $C_0 = \text{eres } D E$

**have**  $\neg \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } (L, \text{Some } C_1)) \text{ and } (L, \text{Some } C_1) \in \text{set } \Gamma$   
**unfolding**  $\text{atomize-conj}$   
**using**  $\Gamma'\text{-eq} \langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle$   
**using**  $\text{mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone}[OF \Gamma\text{-sorted mono-atms-lt}]$   
**by**  $(\text{metis in-set-conv-decomp})$

**then have**  $\neg \text{atm-of } K \preceq_t \text{atm-of } L$   
**by**  $\text{simp}$

**hence**  $\text{atm-of } L \prec_t \text{atm-of } K$   
**by**  $\text{order}$

**moreover have**  $\text{linorder-lit.is-greatest-in-mset } C_1 L$   
**using**  $\Gamma\text{-prop-greatest} \langle (L, \text{Some } C_1) \in \text{set } \Gamma \rangle \text{ by fastforce}$

**ultimately have**  $\text{False}$   
**using**  $\langle C_0 \prec_c C_1 \rangle$   
**by**  $(\text{smt (verit)} \langle C_0 = \text{eres } D E \rangle \text{eres-max-lit linorder-cls.dual-order.asym}$   
 $\text{linorder-lit.dual-order.strict-trans}$   
 $\text{linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset}$   
 $\text{linorder-lit.mulp-if-maximal-less-than-maximal linorder-lit.not-less-iff-gr-or-eq}$

*literal.collapse(1) literal.collapse(2) ord-res.less-lit-simps(1)*  
*ord-res.less-lit-simps(4))*

**thus** *trail-true-cls*  $\Gamma_0$   $C_0$  ..  
**next**  
**assume**  $C_0 \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**thus** *trail-true-cls*  $\Gamma_0$   $C_0$   
**using**  $\Gamma$ -*prop-ball-lt-true*  $\Gamma'$ -*eq*  $\langle C_0 \prec_c C_1 \rangle$   
**by** (*metis (no-types, opaque-lifting)*)  $\langle \text{suffix } \Gamma' \Gamma \rangle$  *append-assoc suffixE*)  
**qed**  
**qed**

**ultimately show** *?thesis*  
**unfolding**  $\langle s' = (U_{er}', \mathcal{F}, \Gamma', \text{Some } C) \rangle$   
**proof** (*intro ord-res- $\gamma$ -invars.intros*)  
**have** *clause-true-in- $\Gamma'$ -if*: *trail-true-cls*  $\Gamma' x$  **and**  
 $\bigwedge K_x. \text{linorder-lit.is-maximal-in-mset } x K_x \implies$   
 $\neg (\exists A \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}'). A \prec_t \text{atm-of } K_x \wedge A \notin \mid \text{trail-atms } \Gamma')$   
**if**  $x$ -*lt*:  $\bigwedge DE. \text{Some } C = \text{Some } DE \longrightarrow x \prec_c DE$  **and**  $x$ -*in*:  $x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid$   
 $(N \mid \cup \mid U_{er}')$   
**for**  $x$   
**proof** –  
**have**  $x \prec_c C$   
**using**  $x$ -*lt* **by** *metis*

**hence**  $x \prec_c \text{eres } D E$   
**using**  $\langle C \prec_c \text{eres } D E \rangle$  **by** *order*

**hence**  $x \neq \text{eres } D E$   
**by** *order*

**have**  $x \prec_c E$   
**using**  $\langle x \prec_c \text{eres } D E \rangle \langle \text{eres } D E \prec_c E \rangle$  **by** *order*

**moreover have**  $x$ -*in'*:  $x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**using**  $x$ -*in*  $\langle x \neq \text{eres } D E \rangle$   
**unfolding**  $\langle \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})) \rangle$   
**by** *simp*

**ultimately have**  $x$ -*true*: *trail-true-cls*  $\Gamma x$   
**using** *clauses-lt-E-true* **by** *metis*

**then obtain**  $L_x$  **where**  $L_x \in \# x$  **and** *trail-true-lit*  $\Gamma L_x$   
**unfolding** *trail-true-cls-def* **by** *metis*

**have**  $L_x \preceq_l - K$   
**using**  $C$ -*max-lit*  $\langle x \prec_c C \rangle \langle L_x \in \# x \rangle$   
**by** (*smt (verit, ccfv-threshold) asympD empty-not-add-mset ord-res.transp-less-lit*)

*insert-DiffM*  
*linorder-lit.is-greatest-in-set-iff linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*  
*linorder-lit.is-maximal-in-mset-iff linorder-lit.is-maximal-in-set-eq-is-greatest-in-set*  
*linorder-lit.is-maximal-in-set-iff linorder-lit.leI ord-res.asymp-less-cls*  
*ord-res.mulp-if-all-left-smaller transpE)*

**have** *mono-atms-lt: monotone-on (set  $\Gamma$ ) ( $\lambda x y. \text{atm-of (fst } y) \prec_t \text{atm-of (fst } x)$ )* ( $\lambda x y. y \leq x$ )  
( $\lambda x. \text{atm-of } K \preceq_t \text{atm-of (fst } x)$ )  
**proof** (*intro monotone-onI, unfold le-bool-def, intro impI*)  
**fix**  $x y$   
**assume**  $\text{atm-of (fst } y) \prec_t \text{atm-of (fst } x)$  **and**  $\text{atm-of } K \preceq_t \text{atm-of (fst } y)$   
**thus**  $\text{atm-of } K \preceq_t \text{atm-of (fst } x)$   
**by** *order*  
**qed**

**obtain**  $\Gamma_1 \Gamma_0$  **where**  $\Gamma\text{-eq: } \Gamma = \Gamma_1 @ (- K, \text{Some } C) \# \Gamma_0$   
**using**  $\langle (- K, \text{Some } C) \in \text{set } \Gamma \rangle$  **by** (*metis split-list*)

**hence** *trail-true-cls*  $\Gamma_0 x$   
**using**  $\Gamma\text{-prop-ball-lt-true } x\text{-in' } \langle x \prec_c C \rangle$  **by** *metis*

**then obtain**  $L_x$  **where**  $L_x \in \# x$  **and**  $L_x\text{-true: trail-true-lit } \Gamma_0 L_x$   
**unfolding** *trail-true-cls-def* **by** *auto*

**moreover have**  $\Gamma' = \Gamma_0$   
**proof** –  
**have**  $\Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of (fst } Ln)) ((\Gamma_1 @ [(- K, \text{Some } C)]) @ \Gamma_0)$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of (fst } Ln)) \Gamma \rangle$   $\Gamma\text{-eq}$   
**by** *simp*

**also have**  $\dots = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of (fst } Ln)) \Gamma_0$   
**proof** (*rule dropWhile-append2*)  
**fix**  $Ln :: 'f \text{ gterm literal} \times 'f \text{ gclause option}$   
**assume**  $Ln \in \text{set } (\Gamma_1 @ [(- K, \text{Some } C)])$

**moreover have**  $\forall x \in \text{set } \Gamma_1. \text{atm-of } K \prec_t \text{atm-of (fst } x)$   
**using**  $\Gamma\text{-sorted}$  **by** (*simp add: \Gamma-eq sorted-wrt-append*)

**ultimately show**  $\text{atm-of } K \preceq_t \text{atm-of (fst } Ln)$   
**using**  $\langle \text{is-neg } K \rangle$  **by** *auto*  
**qed**

**also have**  $\dots = \Gamma_0$   
**proof** (*cases*  $\Gamma_0$ )  
**case** *Nil*  
**thus** *?thesis*  
**by** (*simp add: dropWhile-eq-self-iff*)

**next**  
**case**  $(\text{Cons } Ln \ \Gamma_0')$   
  
**hence**  $\text{atm-of } (fst \ Ln) \prec_t \text{atm-of } K$   
**using**  $\Gamma\text{-sorted}$  **by**  $(\text{simp add: } \Gamma\text{-eq sorted-wrt-append})$   
  
**hence**  $\neg \text{atm-of } K \preceq_t \text{atm-of } (fst \ Ln)$   
**by**  $\text{order}$   
  
**hence**  $\neg \text{atm-of } K \preceq_t \text{atm-of } (fst \ (hd \ \Gamma_0))$   
**by**  $(\text{simp add: } \langle \Gamma_0 = Ln \ \# \ \Gamma_0' \rangle)$   
  
**thus**  $?thesis$   
**by**  $(\text{simp add: } \text{dropWhile-eq-self-iff})$   
**qed**  
  
**finally show**  $?thesis$  .  
**qed**  
  
**ultimately have**  $\text{trail-true-lit } \Gamma' \ L_x$   
**by**  $\text{argo}$   
  
**thus**  $\text{trail-true-cls } \Gamma' \ x$   
**using**  $\langle L_x \in \# \ x \rangle$  **unfolding**  $\text{trail-true-cls-def}$  **by**  $\text{metis}$   
  
  
**fix**  $K_x$   
**assume**  $x\text{-max-lit: ord-res.is-maximal-lit } K_x \ x$   
  
**hence**  $K_x \preceq_l K$   
**using**  $\langle x \prec_c \text{eres } D \ E \rangle \text{eres-max-lit}$   
**using**  $\text{linorder-lit.multip-if-maximal-less-that-maximal}$  **by**  $\text{fastforce}$   
  
**hence**  $\text{atm-of } K_x \preceq_t \text{atm-of } K$   
**by**  $(\text{cases } K_x; \text{cases } K) \text{simp-all}$   
  
**have**  $A \ | \in \ | \ \text{trail-atms } \Gamma'$   
**if**  $A\text{-lt: } A \prec_t \text{atm-of } K_x$  **and**  $A\text{-in: } A \ | \in \ | \ \text{atms-of-cls } (N \ | \cup \ | \ U_{er})$  **for**  $A$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (fst \ Ln)) \ \Gamma \rangle$   
**proof**  $(\text{rule in-trail-atms-dropWhileI})$   
**show**  $\text{sorted-wrt } (\lambda x \ y. \text{atm-of } (fst \ y) \prec_t \text{atm-of } (fst \ x)) \ \Gamma$   
**using**  $\Gamma\text{-sorted}$  .  
**next**  
**show**  $\text{monotone-on } (\text{set } \Gamma) \ (\lambda x \ y. \text{atm-of } (fst \ y) \prec_t \text{atm-of } (fst \ x)) \ (\lambda x \ y. \ y$   
 $\leq x) \ (\lambda x. \text{atm-of } K \preceq_t \text{atm-of } (fst \ x))$   
**using**  $\text{mono-atms-lt}$  .  
**next**  
**show**  $\neg \text{atm-of } K \preceq_t A$   
**using**  $A\text{-lt } \langle \text{atm-of } K_x \preceq_t \text{atm-of } K \rangle$  **by**  $\text{order}$

**next**  
**have**  $\neg (\exists A | \in | \text{atms-of-clss } (N \cup | U_{er}). A \prec_t \text{atm-of } K_x \wedge A | \notin | \text{trail-atms } \Gamma)$   
**using**  $\langle \text{no-undef-atm-lt-max-lit-if-lt-E } \langle x | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}) \rangle \langle x \prec_c E \rangle \text{ x-max-lit} \rangle$   
**by** *metis*  
  
**thus**  $A | \in | \text{trail-atms } \Gamma$   
**using** *A-in A-lt by metis*  
**qed**  
  
**thus**  $\neg (\exists A | \in | \text{atms-of-clss } (N \cup | U_{er}'). A \prec_t \text{atm-of } K_x \wedge A | \notin | \text{trail-atms } \Gamma')$   
**using**  $\langle \text{atms-of-clss } (N \cup | U_{er}) = \text{atms-of-clss } (N \cup | U_{er}') \rangle$  **by** *metis*  
**qed**  
  
**thus**  
 $\forall x | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}'). (\forall DE. \text{Some } C = \text{Some } DE \longrightarrow x \prec_c DE)$   
 $\longrightarrow$   
 $\text{trail-true-cls } \Gamma' x$   
 $\forall x | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}'). (\forall DE. \text{Some } C = \text{Some } DE \longrightarrow x \prec_c DE)$   
 $\longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K x \longrightarrow$   
 $\neg (\exists A | \in | \text{atms-of-clss } (N \cup | U_{er}'). A \prec_t \text{atm-of } K \wedge A | \notin | \text{trail-atms } \Gamma'))$   
**unfolding** *atomize-conj by metis*  
  
**have**  $\text{trail-true-cls } \Gamma' \{ \#K \in \# B. K \neq L_B \# \} \wedge \text{is-pos } L_B$   
**if** *Some C = Some C' and*  
 $B\text{-in: } B | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}') \text{ and}$   
 $B\text{-lt: } B \prec_c C' \text{ and}$   
 $B\text{-max-lit: linorder-lit.is-maximal-in-mset } B L_B \text{ and}$   
 $L_B\text{-undef: } \neg \text{trail-defined-lit } \Gamma' L_B$   
**for**  $C' B L_B$   
**proof** –  
**have**  $C' = C$   
**using** *that by simp*  
**hence**  $B \prec_c C$   
**using** *B-lt by order*  
**hence**  $B \prec_c \text{eres } D E$   
**using**  $\langle C \prec_c \text{eres } D E \rangle$  **by** *order*  
**hence**  $B \neq \text{eres } D E$   
**by** *order*  
**hence**  $B | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er})$   
**using**  $B\text{-in } \langle \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}') = \text{finsert } (\text{eres } D E) (\text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er})) \rangle$   
**by** *simp*  
**moreover** **have**  $B \prec_c E$   
**using**  $\langle B \prec_c \text{eres } D E \rangle \langle \text{eres } D E \prec_c E \rangle$  **by** *order*

```

have  $L_B \preceq_l - K$ 
  using  $\langle B \prec_c C \rangle$  B-max-lit C-max-lit
    by (metis asympD linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset
linorder-lit.leI
      linorder-lit.mulp-if-maximal-less-that-maximal ord-res.asymp-less-cl)
hence  $L_B \prec_l - K \vee L_B = - K$ 
  by simp

thus ?thesis
proof (elim disjE)
  assume  $L_B \prec_l - K$ 
  hence atm-of  $L_B \prec_t$  atm-of  $K$ 
    using  $\langle is-neg K \rangle$ 
    by (cases  $L_B$ ; cases  $K$ ) simp-all

have trail-defined-lit  $\Gamma' L_B$ 
  unfolding  $\langle \Gamma' = dropWhile (\lambda Ln. atm-of K \preceq_t atm-of (fst Ln)) \Gamma \rangle$ 
proof (intro trail-defined-lit-dropWhileI ballI)
  show sorted-wrt  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$ 
    using  $\Gamma$ -sorted .
next
  show monotone-on (set  $\Gamma$ )  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) (\lambda x y. y \leq x)$ 
     $(\lambda x. atm-of K \preceq_t atm-of (fst x))$ 
    using mono-atms-lt .
next
  have  $\neg atm-of K \preceq_t atm-of L_B$ 
    using  $\langle atm-of L_B \prec_t atm-of K \rangle$  by order
  thus  $\neg atm-of K \preceq_t atm-of L_B \wedge \neg atm-of K \preceq_t atm-of (- L_B)$ 
    by simp
next
  have trail-defined-lit  $\Gamma K$ 
by (metis map-of-eq-None-iff option.discI step-hyps(12) trail-defined-lit-def)
  moreover have atm-of  $K \in |$  atms-of-clss  $(N \cup U_{er})$ 
    by (metis  $\langle atms-of-clss (N \cup U_{er}) = atms-of-clss (N \cup U_{er}') \rangle$ 
       $\langle iefac \mathcal{F} \mid \uparrow (N \cup U_{er}') = finsert (eres D E) (iefac \mathcal{F} \mid \uparrow (N \cup U_{er})) \rangle$ 
      atm-of-in-atms-of-clssI eres-max-lit finsertI1 linorder-lit.is-maximal-in-mset-iff)
  moreover have atm-of  $L_B \in |$  atms-of-clss  $(N \cup U_{er})$ 
    using  $\langle B \in |$  iefac  $\mathcal{F} \mid \uparrow (N \cup U_{er}) \rangle$  B-max-lit atm-of-in-atms-of-clssI
  unfolding linorder-lit.is-maximal-in-mset-iff
  by metis
  ultimately show trail-defined-lit  $\Gamma L_B$ 
    using  $\langle atm-of L_B \prec_t atm-of K \rangle$   $\Gamma$ -lower
  unfolding trail-defined-lit-iff-trail-defined-atm
  by (meson linorder-trm.is-lower-set-iff)
qed

hence False

```

**using**  $L_B$ -undef **by** contradiction

**thus** ?thesis ..

**next**

**have** trail-true-cls  $\Gamma' B$

**using** clause-true-in- $\Gamma'$ -if  $B$ -in  $\langle B \prec_c C \rangle$  **by** blast

**hence** trail-true-cls  $\Gamma' \{\#K \in\# B. K \neq L_B\# \}$

**using**  $L_B$ -undef

**by** (smt (verit, best) mem-Collect-eq set-mset-filter trail-defined-lit-iff-true-or-false trail-true-cls-def)

**moreover** assume  $L_B = - K$

**ultimately** show trail-true-cls  $\Gamma' \{\#K \in\# B. K \neq L_B\# \} \wedge$  is-pos  $L_B$

**using**  $\langle$ is-neg  $K \rangle$

**by** (cases  $K$ ) simp-all

**qed**

**qed**

**thus**  $\forall Da. \text{Some } C = \text{Some } Da \longrightarrow$

$(\forall x | \in | \text{iefac } \mathcal{F} | \uparrow | (N | \cup | U_{er} \wedge). x \prec_c Da \longrightarrow (\forall L_C. \text{ord-res.is-maximal-lit } L_C$

$x \longrightarrow$

$\neg \text{trail-defined-lit } \Gamma' L_C \longrightarrow \text{trail-true-cls } \Gamma' \{\#K \in\# x. K \neq L_C\# \} \wedge \text{is-pos}$

$L_C))$

**by** metis

**qed** simp-all

**qed**

**lemma** rtranclp-ord-res-7-preserves-ord-res-7-invars:

**assumes**

*step*: (ord-res-7  $N$ )\*\*  $s s'$  **and**

*invars*: ord-res-7-invars  $N s$

**shows** ord-res-7-invars  $N s'$

**using** step invars ord-res-7-preserves-invars

**by** (smt (verit, del-insts) rtranclp-induct)

**lemma** tranclp-ord-res-7-preserves-ord-res-7-invars:

**assumes**

*step*: (ord-res-7  $N$ )++  $s s'$  **and**

*invars*: ord-res-7-invars  $N s$

**shows** ord-res-7-invars  $N s'$

**using** step invars ord-res-7-preserves-invars

**by** (smt (verit, del-insts) tranclp-induct)

**lemma** propagating-clause-almost-false:

**assumes** invars: ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$  **and**  $(L, \text{Some } C) \in \text{set } \Gamma$

**shows** trail-false-cls  $\Gamma \{\#K \in\# C. K \neq L\# \}$

**proof** –

**have**  $\forall \Gamma_1 L C \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \longrightarrow \text{trail-false-cls } \Gamma_0 \{\#K \in\#$

$C. K \neq L\# \}$

**using** invars **unfolding** ord-res-7-invars-def **by** metis

**hence**  $\exists \Gamma_1 \Gamma_0. \Gamma = \Gamma_1 @ (L, \text{Some } C) \# \Gamma_0 \wedge \text{trail-false-cls } \Gamma_0 \{ \#K \in \# C. K \neq L \# \}$

**using**  $\langle (L, \text{Some } C) \in \text{set } \Gamma \rangle$  **unfolding** *in-set-conv-decomp* **by** *metis*

**thus** *trail-false-cls*  $\Gamma \{ \#K \in \# C. K \neq L \# \}$

**by** (*metis suffixI suffix-ConsD trail-false-cls-if-trail-false-suffix*)

**qed**

**lemma** *ex-ord-res-7-if-not-final*:

**assumes**

*not-final*:  $\neg \text{ord-res-7-final } (N, s)$  **and**

*invars*: *ord-res-7-invars*  $N s$

**shows**  $\exists s'. \text{ord-res-7 } N s s'$

**proof** –

**obtain**  $U_{er} \mathcal{F} \Gamma \mathcal{C}$  **where**  $s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$

**by** (*metis surj-pair*)

**note**  $\text{invars}' = \text{invars}[\text{unfolded } \text{ord-res-7-invars-def}, \text{rule-format}, OF \langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle]$

**have**  $\Gamma\text{-sorted}$ : *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$

**using**  $\text{invars}'$  **by** *argo*

**have**  $\Gamma\text{-consistent}$ : *trail-consistent*  $\Gamma$

**using**  $\Gamma\text{-sorted}$  *trail-consistent-if-sorted-wrt-atoms* **by** *metis*

**have** *distinct*  $(\text{map } fst \Gamma)$

**proof** (*rule distinct-if-sorted-wrt-asymp*)

**have** *sorted-wrt*  $(\lambda x y. fst y \prec_l fst x) \Gamma$

**proof** (*rule sorted-wrt-mono-rel*)

**show** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$

**using**  $\Gamma\text{-sorted}$  .

**next**

**fix**  $x y :: 'f \text{ gterm literal} \times 'f \text{ gterm literal multiset option}$

**assume**  $\text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$

**thus**  $fst y \prec_l fst x$

**by** (*cases*  $fst x$ ; *cases*  $fst y$ ) (*simp-all only: literal.sel ord-res.less-lit-simps*)

**qed**

**thus** *sorted-wrt*  $(\lambda x y. y \prec_l x) (\text{map } fst \Gamma)$

**unfolding** *sorted-wrt-map* .

**next**

**show** *asymp-on*  $(\text{set } (\text{map } fst \Gamma)) (\lambda x y. y \prec_l x)$

**using** *linorder-lit.asymp-on-greater* .

**qed**

**hence** *map-of- $\Gamma$ -eq-SomeI*:  $\bigwedge x y. (x, y) \in \text{set } \Gamma \implies \text{map-of } \Gamma x = \text{Some } y$

**by** (*metis map-of-is-SomeI*)



**have**  $\Gamma$ -lower: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-cls* ( $N \cup U_{er}$ ))  
**using** *invars'* **by** *argo*

**obtain**  $E$  **where**  $C = \text{Some } E$  **and**  $E \neq \{\#\}$   
**using** *not-final[unfolded*  $\langle s = (U_{er}, \mathcal{F}, \Gamma, C) \rangle$  *ord-res-7-final.simps, simplified]*  
**by** *metis*

**have**  $E$ -in:  $E \in \text{iefac } \mathcal{F} \mid \uparrow (N \cup U_{er})$   
**using** *invars'*  $\langle C = \text{Some } E \rangle$  **by** *metis*

**obtain**  $L$  **where**  $E$ -max-lit: *ord-res.is-maximal-lit*  $L E$   
**using**  $\langle E \neq \{\#\} \rangle$  *linorder-lit.ex-maximal-in-mset* **by** *metis*

**show** *?thesis*  
**proof** (*cases trail-false-cls*  $\Gamma E$ )  
**case**  $E$ -false: *True*

**hence**  $L$ -false: *trail-false-lit*  $\Gamma L$   
**using**  $E$ -max-lit **unfolding** *linorder-lit.is-maximal-in-mset-iff trail-false-cls-def*  
**by** *metis*

**hence**  $\exists \text{opt. } (- L, \text{opt}) \in \text{set } \Gamma$   
**by** (*auto simp: trail-false-lit-def*)

**have**  $\neg \text{is-pos } L$   
**proof** (*rule notI*)  
**assume**  $\text{is-pos } L$

**hence**  $\text{is-neg } (- L)$   
**by** (*metis*  $\langle \text{is-pos } L \rangle$  *is-pos-neg-not-is-pos*)

**hence**  $(- L, \text{None}) \in \text{set } \Gamma$   
**using** *invars'*  $\langle \exists \text{opt. } (- L, \text{opt}) \in \text{set } \Gamma \rangle$  **by** (*metis prod.sel*)

**moreover have**  $E \prec_c \{\# - L\}$   
**using**  $E$ -max-lit  $\langle \text{is-pos } L \rangle$   
**by** (*smt (verit, best) Neg-atm-of-iff add-mset-remove-trivial empty-iff*  
*ord-res.less-lit-simps(4) linorder-cls.less-irrefl linorder-lit.is-greatest-in-mset-iff*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset linorder-lit.less-linear*  
*linorder-lit.mulp-if-maximal-less-than-maximal literal.collapse(1)*  
*ord-res.less-lit-simps(1) set-mset-empty uminus-literal-def union-single-eq-member*)

**ultimately have** *trail-true-cls*  $\Gamma E$   
**using** *invars'*  $E$ -in **by** *force*

**hence**  $\neg \text{trail-false-cls } \Gamma E$   
**by** (*metis*  $\Gamma$ -consistent *not-trail-true-cls-and-trail-false-cls*)

**thus** *False*  
**using** *E-false by contradiction*  
**qed**

**hence** *L-neg: is-neg L*  
**by** *argo*

**then obtain** *D* **where**  $(- L, \text{Some } D) \in \text{set } \Gamma$   
**using** *invars' <is-neg L> <\exists opt. (- L, opt) \in set \Gamma>*  
**by** *(metis prod.sel is-pos-neg-not-is-pos not-Some-eq)*

**hence** *map-of \Gamma (- L) = Some (Some D)*  
**using** *map-of-\Gamma-eq-SomeI by metis*

**have**  $\exists \Gamma_1 \Gamma_0. \Gamma = \Gamma_1 @ (- L, \text{Some } D) \# \Gamma_0 \wedge \text{trail-false-cls } \Gamma_0 \{ \#K \in \# D. K \neq - L \# \}$   
**using** *invars' <(- L, Some D) \in set \Gamma> propagating-clause-almost-false*  
**unfolding** *in-set-conv-decomp by metis*

**have** *D-almost-false: trail-false-cls \Gamma \{ \#K \in \# D. K \neq - L \# \}*  
**using** *invars <s = (U\_{er}, \mathcal{F}, \Gamma, \mathcal{C})> <(- L, Some D) \in set \Gamma> propagating-clause-almost-false*  
**by** *metis*

**show** *?thesis*  
**proof** *(cases eres D E = { \# })*  
**case** *True*  
**thus** *?thesis*  
**unfolding** *<s = (U\_{er}, \mathcal{F}, \Gamma, \mathcal{C})> <\mathcal{C} = Some E>*  
**using** *E-false E-max-lit L-neg <map-of \Gamma (- L) = Some (Some D)>*  
**using** *ord-res-7.resolution-bot by metis*

**next**  
**case** *False*

**then obtain** *K* **where** *eres-max-lit: ord-res.is-maximal-lit K (eres D E)*  
**using** *linorder-lit.ex-maximal-in-mset by metis*

**hence**  $K \in \# \text{eres } D \ E$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff by argo*

**hence**  $K \in \# D \wedge K \neq - L \vee K \in \# E$   
**using** *strong-lit-in-one-of-resolvents-if-in-eres[OF E-max-lit] by metis*

**hence** *K-false: trail-false-lit \Gamma K*  
**using** *D-almost-false E-false unfolding trail-false-cls-def by auto*

**hence**  $\exists \text{opt. } (- K, \text{opt}) \in \text{set } \Gamma$   
**by** *(auto simp: trail-false-lit-def)*

```

show ?thesis
proof (cases K)
  case (Pos AK)

  hence is-pos K
  by simp

  thus ?thesis
  unfolding ⟨s = (Uer, F, Γ, C)⟩ ⟨C = Some E⟩
  using E-false E-max-lit L-neg ⟨map-of Γ (- L) = Some (Some D)⟩ False
eres-max-lit
  using ord-res-7.resolution-pos by metis
next
  case (Neg AK)

  hence is-neg K
  by simp

  then obtain C where (- K, Some C) ∈ set Γ
  using invars' ⟨is-neg L⟩ ⟨∃ opt. (- K, opt) ∈ set Γ⟩
  by (metis prod.sel is-pos-neg-not-is-pos not-Some-eq)

  hence map-of Γ (- K) = Some (Some C)
  using map-of-Γ-eq-SomeI by metis

  thus ?thesis
  unfolding ⟨s = (Uer, F, Γ, C)⟩ ⟨C = Some E⟩
  using E-false E-max-lit L-neg ⟨map-of Γ (- L) = Some (Some D)⟩ False
eres-max-lit
  ⟨is-neg K⟩
  using ord-res-7.resolution-neg by metis
  qed
qed
next
  case E-not-false: False
  show ?thesis
proof (cases ∃ A|∈|atms-of-clss (N |∪| Uer). A ≺t atm-of L ∧ A |∉| trail-atms
Γ)
  case True

  then obtain A where A-least: linorder-trm.is-least-in-fset
  { |A |∈| atms-of-clss (N |∪| Uer). A ≺t atm-of L ∧ A |∉| trail-atms Γ |} A
  by (metis (no-types, lifting) bot-fset.rep-eq empty-iff ffmembers-filter
linorder-trm.ex-least-in-fset)

  show ?thesis
  unfolding ⟨s = (Uer, F, Γ, C)⟩ ⟨C = Some E⟩
  using ord-res-7.decide-neg[OF E-not-false E-max-lit A-least refl, of F]
  by metis

```

```

next
case nex-undef-atm-lt-L: False
show ?thesis
proof (cases trail-defined-lit  $\Gamma$  L)
case True
thus ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle C = \text{Some } E \rangle$ 
  using ord-res-7.skip-defined[OF E-not-false E-max-lit nex-undef-atm-lt-L]
  by metis
next
case L-undef: False
show ?thesis
proof (cases L)
case L-eq: (Pos  $A_L$ )

hence L-pos: is-pos L
  by simp

show ?thesis
proof (cases trail-false-cls  $\Gamma$  {#K  $\in$  # E. K  $\neq$  L#})
case E-almost-false: True
show ?thesis
proof (cases ord-res.is-strictly-maximal-lit L E)
case True
thus ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle C = \text{Some } E \rangle$ 
  using ord-res-7.production[OF E-not-false E-max-lit nex-undef-atm-lt-L]
L-undef L-pos
  E-almost-false]
  by metis
next
case False
thus ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle C = \text{Some } E \rangle$ 
  using ord-res-7.factorizing[OF E-not-false E-max-lit nex-undef-atm-lt-L]
L-undef L-pos
  E-almost-false]
  by metis
qed
next
case E-not-almost-false: False
show ?thesis
proof (cases  $\exists D | \in |iefac \mathcal{F} | \uparrow (N | \cup | U_{er}). E \prec_c D$ )
case True

then obtain F where linorder-cls.is-least-in-fset
  (ffilter (( $\prec_c$ ) E) (iefac  $\mathcal{F} | \uparrow (N | \cup | U_{er}))$ ) F
by (metis bot-fset.rep-eq empty-iff ffilter linorder-cls.ex1-least-in-fset)

```

```

      thus ?thesis
        unfolding ⟨s = (Uer, F, Γ, C)⟩ ⟨C = Some E⟩
      using ord-res-7.skip-undefined-pos[OF E-not-false E-max-lit nex-undef-atm-lt-L
L-undef
        L-pos E-not-almost-false]
        by metis
      next
      case False
      thus ?thesis
        unfolding ⟨s = (Uer, F, Γ, C)⟩ ⟨C = Some E⟩
        using ord-res-7.skip-undefined-pos-ultimate[OF E-not-false E-max-lit
        nex-undef-atm-lt-L L-undef L-pos E-not-almost-false]
        by metis
      qed
    qed
  next
  case L-eq: (Neg AL)

  hence is-neg L
    by simp

  thus ?thesis
    unfolding ⟨s = (Uer, F, Γ, C)⟩ ⟨C = Some E⟩
  using ord-res-7.skip-undefined-neg[OF E-not-false E-max-lit nex-undef-atm-lt-L
L-undef]
    by metis
  qed
  qed
  qed
  qed
  qed
end

lemma ord-res-7.safe-state-if-invars:
  fixes N :: 'f gclause fset and s
  assumes invars: ord-res-7-invars N s
  shows safe-state (constant-context ord-res-7) ord-res-7-final (N, s)
  using safe-state-constant-context-if-invars[where
    R = ord-res-7 and F = ord-res-7-final and I = ord-res-7-invars]
  using ord-res-7-preserves-invars ex-ord-res-7-if-not-final invars by metis

end

end

theory Clause-Could-Propagate
  imports
    Background
    Implicit-Exhaustive-Factorization
begin

```

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**definition** *clause-could-propagate* **where**

*clause-could-propagate*  $\Gamma C L \longleftrightarrow \neg \text{trail-defined-lit } \Gamma L \wedge$   
*linorder-lit.is-maximal-in-mset*  $C L \wedge \text{trail-false-cls } \Gamma \{\#K \in \# C. K \neq L\# \}$

**lemma** *trail-false-if-could-have-propagated*:

*clause-could-propagate*  $\Gamma C L \implies \text{trail-false-cls } ((- L, n) \# \Gamma) C$

**unfolding** *clause-could-propagate-def trail-false-cls-def trail-false-lit-def* **by** *auto*

**lemma** *atoms-of-trail-lit-atom-of-propagatable-literal*:

**assumes**

$\Gamma$ -*lower*: *linorder-trm.is-lower-set* (*fset* (*trail-atms*  $\Gamma$ ))  $\mathcal{A}$  **and**

*C-prop*: *clause-could-propagate*  $\Gamma C L$  **and**

*atm-of*  $L \in \mathcal{A}$

**shows**  $\forall A \in \text{trail-atms } \Gamma. A \prec_t \text{atm-of } L$

**proof** –

**have** *atm-of*  $L \notin \text{trail-atms } \Gamma$

**using** *C-prop*

**unfolding** *clause-could-propagate-def trail-defined-lit-iff-trail-defined-atm*

**by** *argo*

**then show** *?thesis*

**using**  $\Gamma$ -*lower*  $\langle \text{atm-of } L \in \mathcal{A} \rangle$

**by** (*metis linorder-trm.is-lower-set-iff linorder-trm.neqE*)

**qed**

**lemma** *trail-false-cls-filter-mset-iff*:

*trail-false-cls*  $\Gamma \{\#Ka \in \# C. Ka \neq K\# \} \longleftrightarrow (\forall L \in \# C. L \neq K \longrightarrow \text{trail-false-lit } \Gamma L)$

**unfolding** *trail-false-cls-def* **by** *auto*

**lemma** *clause-could-propagate-iff*: *clause-could-propagate*  $\Gamma C K \longleftrightarrow$

$\neg \text{trail-defined-lit } \Gamma K \wedge \text{ord-res.is-maximal-lit } K C \wedge (\forall L \in \# C. L \neq K \longrightarrow \text{trail-false-lit } \Gamma L)$

**unfolding** *clause-could-propagate-def trail-false-cls-filter-mset-iff* ..

**lemma** *clause-could-propagate-efac*: *clause-could-propagate*  $\Gamma (\text{efac } C) = \text{clause-could-propagate } \Gamma C$

**proof** (*rule ext*)

**fix**  $L$

**show** *clause-could-propagate*  $\Gamma (\text{efac } C) L \longleftrightarrow \text{clause-could-propagate } \Gamma C L$

**unfolding** *clause-could-propagate-def*

**by** (*metis (mono-tags, lifting) ex1-efac-eq-factoring-chain mem-Collect-eq*

*ord-res.ground-factorings-preserves-maximal-literal set-mset-efac set-mset-filter trail-false-cls-def*)

**qed**

**lemma** *bex-clause-could-propagate-simp*:

```

fixes  $\mathcal{F} N \Gamma L$ 
shows  $fBex (iefac \mathcal{F} \mid \uparrow N) (\lambda C. clause\text{-}could\text{-}propagate \Gamma C L) \longleftrightarrow$ 
 $fBex N (\lambda C. clause\text{-}could\text{-}propagate \Gamma C L)$ 
sketch (rule iffI; elim bexE)
proof (rule iffI; elim bexE)
  fix  $C :: 'f gclause$ 
  assume  $C \mid \in \mid iefac \mathcal{F} \mid \uparrow N$  and  $clause\text{-}could\text{-}propagate \Gamma C L$ 
  thus  $\exists C \mid \in \mid N. clause\text{-}could\text{-}propagate \Gamma C L$ 
    by (metis clause\text{-}could\text{-}propagate\text{-}efac fimageE iefac\text{-}def)
next
  fix  $C :: 'f gclause$ 
  assume  $C \mid \in \mid N$  and  $clause\text{-}could\text{-}propagate \Gamma C L$ 
  thus  $\exists C \mid \in \mid iefac \mathcal{F} \mid \uparrow N. clause\text{-}could\text{-}propagate \Gamma C L$ 
    by (metis clause\text{-}could\text{-}propagate\text{-}efac fimageI iefac\text{-}def)
qed

end

end

theory ORD-RES-8
  imports
    Background
    Implicit-Exhaustive-Factorization
    Exhaustive-Resolution
    Clause-Could-Propagate
begin

```

## 23 ORD-RES-8 (atom-guided literal trail construction)

```

type-synonym  $'f ord\text{-}res\text{-}8\text{-}state =$ 
 $'f gclause fset \times 'f gclause fset \times 'f gclause fset \times ('f gliteral \times 'f gclause option)$ 
 $list$ 

```

```

context simulation-SCLFOL-ground-ordered-resolution begin

```

```

inductive ord-res-8 where

```

```

  decide-neg:

```

```

   $\neg (\exists C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). trail\text{-}false\text{-}cls \Gamma C) \implies$ 
 $linorder\text{-}trm.is\text{-}least\text{-}in\text{-}fset \{ \mid A_2 \mid \in \mid atms\text{-}of\text{-}clss (N \mid \cup \mid U_{er}).$ 
 $\forall A_1 \mid \in \mid trail\text{-}atms \Gamma. A_1 \prec_t A_2 \} A \implies$ 
 $\neg (\exists C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). clause\text{-}could\text{-}propagate \Gamma C (Pos A)) \implies$ 
 $\Gamma' = (Neg A, None) \# \Gamma \implies$ 
 $ord\text{-}res\text{-}8 N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}, \Gamma') \mid$ 

```

```

  propagate:

```

```

   $\neg (\exists C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). trail\text{-}false\text{-}cls \Gamma C) \implies$ 
 $linorder\text{-}trm.is\text{-}least\text{-}in\text{-}fset \{ \mid A_2 \mid \in \mid atms\text{-}of\text{-}clss (N \mid \cup \mid U_{er}).$ 

```

$\forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2 \mid \} A \implies$   
 $\text{linorder-cls.is-least-in-fset } \{ \mid C \mid \in \mid \text{iefac } \mathcal{F} \mid \} (N \mid \cup \mid U_{er}).$   
 $\text{clause-could-propagate } \Gamma C (Pos A) \mid \} C \implies$   
 $\text{linorder-lit.is-greatest-in-mset } C (Pos A) \implies$   
 $\Gamma' = (Pos A, Some C) \# \Gamma \implies$   
 $\text{ord-res-8 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}, \Gamma') \mid$

*factorize:*

$\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \} (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma C) \implies$   
 $\text{linorder-trm.is-least-in-fset } \{ \mid A_2 \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}).$   
 $\forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2 \mid \} A \implies$   
 $\text{linorder-cls.is-least-in-fset } \{ \mid C \mid \in \mid \text{iefac } \mathcal{F} \mid \} (N \mid \cup \mid U_{er}).$   
 $\text{clause-could-propagate } \Gamma C (Pos A) \mid \} C \implies$   
 $\neg \text{linorder-lit.is-greatest-in-mset } C (Pos A) \implies$   
 $\mathcal{F}' = \text{finsert } C \mathcal{F} \implies$   
 $\text{ord-res-8 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}', \Gamma) \mid$

*resolution:*

$\text{linorder-cls.is-least-in-fset } \{ \mid D \mid \in \mid \text{iefac } \mathcal{F} \mid \} (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma D \mid \}$   
 $D \implies$   
 $\text{linorder-lit.is-maximal-in-mset } D (Neg A) \implies$   
 $\text{map-of } \Gamma (Pos A) = Some (Some C) \implies$   
 $U_{er}' = \text{finsert } (eres C D) U_{er} \implies$   
 $\Gamma' = \text{dropWhile } (\lambda Ln. \forall K.$   
 $\text{linorder-lit.is-maximal-in-mset } (eres C D) K \longrightarrow \text{atm-of } K \preceq_t \text{atm-of } (fst$   
 $Ln)) \Gamma \implies$   
 $\text{ord-res-8 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}', \mathcal{F}, \Gamma')$

**lemma** *right-unique-ord-res-8:*

**fixes**  $N :: 'f \text{ gclause fset}$

**shows** *right-unique* ( $\text{ord-res-8 } N$ )

**proof** (*rule right-uniqueI*)

**fix**  $x y z$

**assume** *step1:*  $\text{ord-res-8 } N x y$  **and** *step2:*  $\text{ord-res-8 } N x z$

**show**  $y = z$

**using** *step1*

**proof** (*cases*  $N x y$  *rule:*  $\text{ord-res-8.cases}$ )

**case** *step1-hyps:* ( $\text{decide-neg } \mathcal{F} U_{er} \Gamma A \Gamma'$ )

**show** *?thesis*

**using** *step2 unfolding*  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**proof** (*cases*  $N (U_{er}, \mathcal{F}, \Gamma) z$  *rule:*  $\text{ord-res-8.cases}$ )

**case** ( $\text{decide-neg } A \Gamma'$ )

**with** *step1-hyps show* *?thesis*

**by** (*metis* (*no-types, lifting*)  $\text{linorder-trm.dual-order.asym linorder-trm.is-least-in-fset-iff}$ )

**next**

**case** (*propagate*  $A2 C2 \Gamma2'$ )

**with** *step1-hyps have* *False*

**by** (*metis* (*no-types, lifting*)  $\text{linorder-cls.is-least-in-fset-ffilterD}(1)$

$\text{linorder-cls.is-least-in-fset-ffilterD}(2)$   $\text{linorder-trm.dual-order.asym}$ )



```

      linorder-trm.is-least-in-fset-iff)
thus ?thesis ..
next
  case (factorize A2 C2 F2')
  with step1-hyps have False
    by (metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD(1)
      linorder-cls.is-least-in-fset-ffilterD(2) linorder-trm.dual-order.asym
      linorder-trm.is-least-in-fset-iff)
    thus ?thesis ..
next
  case (resolution D2 A2 C2 Uer2' Γ2')
  with step1-hyps have False
    by (metis linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
qed
next
  case step1-hyps: (propagate F Uer Γ A C Γ')
  show ?thesis
    using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
  proof (cases N (Uer, F, Γ) z rule: ord-res-8.cases)
    case (decide-neg A2 Γ2')
    with step1-hyps have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD(1)
        linorder-cls.is-least-in-fset-ffilterD(2) linorder-trm.dual-order.asym
        linorder-trm.is-least-in-fset-iff)
      thus ?thesis ..
    next
      case (propagate A2 C2 Γ2')
      with step1-hyps show ?thesis
        by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
          linorder-trm.dual-order.asym linorder-trm.is-least-in-fset-iff The-optional-eq-SomeI)
    next
      case (factorize A2 C2 F2')
      with step1-hyps have False
        by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
          linorder-trm.Uniq-is-least-in-fset the1-equality')
        thus ?thesis ..
    next
      case (resolution D2 A2 C2 Uer2' Γ2')
      with step1-hyps have False
        by (metis linorder-cls.is-least-in-ffilter-iff)
        thus ?thesis ..
    qed
  next
    case step1-hyps: (factorize F Uer Γ A C F')
    show ?thesis
      using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
    proof (cases N (Uer, F, Γ) z rule: ord-res-8.cases)
      case (decide-neg A2 Γ2')

```

```

with step1-hyps have False
  by (metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD(1)
        linorder-cls.is-least-in-fset-ffilterD(2) linorder-trm.dual-order.asym
        linorder-trm.is-least-in-fset-iff)
thus ?thesis ..
next
case (propagate A2 C2  $\Gamma 2'$ )
with step1-hyps have False
  by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
        linorder-trm.Uniq-is-least-in-fset the1-equality')
thus ?thesis ..
next
case (factorize A2 C2  $\mathcal{F} 2'$ )
with step1-hyps show ?thesis
  by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
        linorder-trm.Uniq-is-least-in-fset the1-equality')
next
case (resolution D2 A2 C2  $U_{er} 2' \Gamma 2'$ )
with step1-hyps have False
  by (metis linorder-cls.is-least-in-ffilter-iff)
thus ?thesis ..
qed
next
case step1-hyps: (resolution  $\Gamma \mathcal{F} U_{er} D A C U_{er}' \Gamma'$ )
show ?thesis
  using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
proof (cases  $N (U_{er}, \mathcal{F}, \Gamma) z$  rule: ord-res-8.cases)
  case (decide-neg A  $\Gamma'$ )
  with step1-hyps have False
  by (metis (no-types, opaque-lifting) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
next
case (propagate A C  $\Gamma'$ )
with step1-hyps have False
  by (metis (no-types, opaque-lifting) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
next
case (factorize A C  $\mathcal{F}'$ )
with step1-hyps have False
  by (metis (no-types, opaque-lifting) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
next
case step2-hyps: (resolution D2 A2 C2  $U_{er} 2' \Gamma 2'$ )
have  $D2 = D$ 
  using  $\langle \text{linorder-cls.is-least-in-fset (ffilter (trail-false-cls } \Gamma) (\text{iefac } \mathcal{F} \mid \langle (N \cup U_{er}) \rangle) D) \rangle$ 
  using  $\langle \text{linorder-cls.is-least-in-fset (ffilter (trail-false-cls } \Gamma) (\text{iefac } \mathcal{F} \mid \langle (N \cup U_{er}) \rangle) D2) \rangle$ 
  by (metis linorder-cls.Uniq-is-least-in-fset Uniq-D)

```

```

have A2 = A
  using ⟨linorder-lit.is-maximal-in-mset D (Neg A)⟩
  using ⟨linorder-lit.is-maximal-in-mset D2 (Neg A2)⟩
  unfolding ⟨D2 = D⟩
  by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset literal.sel(2))

have C2 = C
  using step1-hyps(5) step2-hyps(4)
  unfolding ⟨A2 = A⟩
  by simp

show ?thesis
  unfolding ⟨y = (Uer',  $\mathcal{F}$ ,  $\Gamma'$ )⟩ ⟨z = (Uer2',  $\mathcal{F}$ ,  $\Gamma$ 2')⟩ prod.inject
  proof (intro conjI)
    show Uer' = Uer2'
    unfolding ⟨Uer' = finsert (eres C D) Uer⟩ ⟨Uer2' = finsert (eres C2 D2)
Uer⟩
      unfolding ⟨D2 = D⟩ ⟨C2 = C⟩ ..
    next
      show  $\mathcal{F}$  =  $\mathcal{F}$  ..
    next
      show  $\Gamma'$  =  $\Gamma$ 2'
      using step1-hyps(7) step2-hyps(6)
      unfolding ⟨D2 = D⟩ ⟨C2 = C⟩
      by argo
    qed
  qed
qed
qed
qed

inductive ord-res-8-final :: 'f ord-res-8-state  $\Rightarrow$  bool where
  model-found:
     $\neg (\exists A \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}). A \mid \notin \mid \text{trail-atms } \Gamma) \Longrightarrow$ 
     $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma C) \Longrightarrow$ 
    ord-res-8-final (N, Uer,  $\mathcal{F}$ ,  $\Gamma$ ) |

  contradiction-found:
    {#}  $\mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}) \Longrightarrow$ 
    ord-res-8-final (N, Uer,  $\mathcal{F}$ ,  $\Gamma$ )

sublocale ord-res-8-semantics: semantics where
  step = constant-context ord-res-8 and
  final = ord-res-8-final
proof unfold-locales
  fix S :: 'f ord-res-8-state

obtain
  N Uer  $\mathcal{F}$  :: 'f gterm clause fset and

```

$\Gamma :: ('f\ gterm\ literal \times 'f\ gterm\ literal\ multiset\ option)\ list$  **where**  
*S-def*:  $S = (N, U_{er}, \mathcal{F}, \Gamma)$   
**by** (*metis prod.exhaust*)

**assume** *ord-res-8-final S*

**hence**  $\nexists x. ord-res-8\ N\ (U_{er}, \mathcal{F}, \Gamma)\ x$

**unfolding** *S-def*

**proof** (*cases (N, U<sub>er</sub>, F, Γ) rule: ord-res-8-final.cases*)

**case** *model-found*

**have**  $\nexists A. linorder-trm.is-least-in-fset\ \{|A_2| \in|\ atm\text{-of-cls}\ (N \cup| U_{er}).$

$\forall A_1 | \in|\ trail-atms\ \Gamma. A_1 \prec_t A_2\}| A$

**using**  $\langle \neg (\exists A | \in|\ atm\text{-of-cls}\ (N \cup| U_{er}). A | \notin|\ trail-atms\ \Gamma) \rangle$

**using** *linorder-trm.is-least-in-ffilter-iff* **by** *fastforce*

**moreover have**  $\nexists C. linorder-cls.is-least-in-fset$

$(ffilter\ (trail-false-cls\ \Gamma)\ (iefac\ \mathcal{F}\ | \uparrow\ (N \cup| U_{er})))\ C$

**using**  $\langle \neg fBex\ (iefac\ \mathcal{F}\ | \uparrow\ (N \cup| U_{er}))\ (trail-false-cls\ \Gamma) \rangle$

**by** (*metis linorder-cls.is-least-in-ffilter-iff*)

**ultimately show** *?thesis*

**unfolding** *ord-res-8.simps* **by** *blast*

**next**

**case** *contradiction-found*

**hence**  $\exists C | \in|\ iefac\ \mathcal{F}\ | \uparrow\ (N \cup| U_{er}).\ trail-false-cls\ \Gamma\ C$

**using** *trail-false-cls-empty* **by** *metis*

**moreover have**  $\nexists D A. linorder-cls.is-least-in-fset\ (ffilter\ (trail-false-cls\ \Gamma)$

$(iefac\ \mathcal{F}\ | \uparrow\ (N \cup| U_{er})))\ D \wedge ord-res.is-maximal-lit\ (Neg\ A)\ D$

**by** (*metis empty-iff linorder-cls.is-least-in-ffilter-iff linorder-cls.leD*

*linorder-lit.is-maximal-in-mset-iff local.contradiction-found(1) mempty-lesseq-cls set-mset-empty trail-false-cls-empty*)

**ultimately show** *?thesis*

**unfolding** *ord-res-8.simps* **by** *blast*

**qed**

**thus finished** (*constant-context ord-res-8*) *S*

**by** (*simp add: S-def finished-def constant-context.simps*)

**qed**

**definition** *trail-is-sorted* **where**

*trail-is-sorted N s*  $\longleftrightarrow$

$(\forall U_{er}\ \mathcal{F}\ \Gamma. s = (U_{er}, \mathcal{F}, \Gamma) \longrightarrow$

*sorted-wrt*  $(\lambda x\ y. atm\text{-of}\ (fst\ y) \prec_t atm\text{-of}\ (fst\ x))\ \Gamma)$ )

**lemma** *ord-res-8-preserves-trail-is-sorted*:

**assumes**  
*step*: *ord-res-8*  $N$   $s$   $s'$  **and**  
*invar*: *trail-is-sorted*  $N$   $s$   
**shows** *trail-is-sorted*  $N$   $s'$   
**using** *step*  
**proof** (*cases*  $N$   $s$   $s'$  *rule*: *ord-res-8.cases*)  
**case** *step-hyps*: (*decide-neg*  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $\Gamma'$ )

**have**  $\forall x \mid \in \mid$  *trail-atms*  $\Gamma$ .  $x \prec_t A$   
**using** *step-hyps* **unfolding** *linorder-trm.is-least-in-filter-iff* **by** *simp*

**hence**  $\forall x \in \text{set } \Gamma$ . *atm-of* (*fst*  $x$ )  $\prec_t A$   
**by** (*simp add*: *fset-trail-atms*)

**moreover have** *sorted-wrt* ( $\lambda x y$ . *atm-of* (*fst*  $y$ )  $\prec_t$  *atm-of* (*fst*  $x$ ))  $\Gamma$   
**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *trail-is-sorted-def*)

**ultimately have** *sorted-wrt* ( $\lambda x y$ . *atm-of* (*fst*  $y$ )  $\prec_t$  *atm-of* (*fst*  $x$ ))  $\Gamma'$   
**by** (*simp add*:  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$ )

**thus** *?thesis*  
**by** (*simp add*:  $\langle s' = (U_{er}, \mathcal{F}, \Gamma') \rangle$  *trail-is-sorted-def*)

**next**  
**case** *step-hyps*: (*propagate*  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $C$   $\Gamma'$ )

**have**  $\forall x \mid \in \mid$  *trail-atms*  $\Gamma$ .  $x \prec_t A$   
**using** *step-hyps* **unfolding** *linorder-trm.is-least-in-filter-iff* **by** *simp*

**hence**  $\forall x \in \text{set } \Gamma$ . *atm-of* (*fst*  $x$ )  $\prec_t A$   
**by** (*simp add*: *fset-trail-atms*)

**moreover have** *sorted-wrt* ( $\lambda x y$ . *atm-of* (*fst*  $y$ )  $\prec_t$  *atm-of* (*fst*  $x$ ))  $\Gamma$   
**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *trail-is-sorted-def*)

**ultimately have** *sorted-wrt* ( $\lambda x y$ . *atm-of* (*fst*  $y$ )  $\prec_t$  *atm-of* (*fst*  $x$ ))  $\Gamma'$   
**by** (*simp add*:  $\langle \Gamma' = (\text{Pos } A, \text{Some } C) \# \Gamma \rangle$ )

**thus** *?thesis*  
**by** (*simp add*:  $\langle s' = (U_{er}, \mathcal{F}, \Gamma') \rangle$  *trail-is-sorted-def*)

**next**  
**case** (*factorize*  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $C$   $\mathcal{F}'$ )

**have** *sorted-wrt* ( $\lambda x y$ . *atm-of* (*fst*  $y$ )  $\prec_t$  *atm-of* (*fst*  $x$ ))  $\Gamma$   
**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *trail-is-sorted-def*)

**thus** *?thesis*  
**by** (*simp add*:  $\langle s' = (U_{er}, \mathcal{F}', \Gamma) \rangle$  *trail-is-sorted-def*)

**next**  
**case** *step-hyps*: (*resolution*  $\Gamma$   $\mathcal{F}$   $U_{er}$   $D$   $A$   $C$   $U_{er}'$   $\Gamma'$ )

**have** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma$   
**using** *invar* **by**  $(\text{simp add: } \langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle \text{trail-is-sorted-def})$

**hence** *sorted-wrt*  $(\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) \Gamma'$   
**unfolding** *step-hyps*(7) **by**  $(\text{rule sorted-wrt-dropWhile})$

**thus** *?thesis*  
**by**  $(\text{simp add: } \langle s' = (U_{er}', \mathcal{F}, \Gamma') \rangle \text{trail-is-sorted-def})$   
**qed**

**inductive** *trail-annotations-invars*  
**for**  $N :: 'f \text{ gterm literal multiset fset}$   
**where**

*Nil:*  
 $\text{trail-annotations-invars } N (U_{er}, \mathcal{F}, []) \mid$   
*Cons-None:*  
 $\text{trail-annotations-invars } N (U_{er}, \mathcal{F}, (L, \text{None}) \# \Gamma)$   
**if**  $\text{trail-annotations-invars } N (U_{er}, \mathcal{F}, \Gamma) \mid$   
*Cons-Some:*  
 $\text{trail-annotations-invars } N (U_{er}, \mathcal{F}, (L, \text{Some } D) \# \Gamma)$   
**if** *linorder-lit.is-greatest-in-mset*  $D L$  **and**  
 $D \in \text{iefac } \mathcal{F} \mid (N \mid \cup \mid U_{er})$  **and**  
 $\text{trail-false-cls } \Gamma \{ \#K \in \# D. K \neq L \# \}$  **and**  
*linorder-cls.is-least-in-fset*  
 $\{ \mid D \in \text{iefac } \mathcal{F} \mid (N \mid \cup \mid U_{er}). \text{clause-could-propagate } \Gamma D L \}$   $D$  **and**  
 $\text{trail-annotations-invars } N (U_{er}, \mathcal{F}, \Gamma)$

**lemma**  
**assumes**

$\text{linorder-lit.is-greatest-in-mset } C L$  **and**  
 $\text{trail-false-cls } \Gamma \{ \#K \in \# C. K \neq L \# \}$  **and**  
 $\neg \text{trail-defined-cls } \Gamma C$

**shows**  $\text{clause-could-propagate } \Gamma C L$   
**unfolding** *clause-could-propagate-iff*  
**proof**  $(\text{intro conjI})$   
**show**  $\text{linorder-lit.is-maximal-in-mset } C L$   
**using**  $\langle \text{linorder-lit.is-greatest-in-mset } C L \rangle$   
**by**  $(\text{metis linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset})$

**hence**  $L \in \# C$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff* ..

**show**  $\forall K \in \# C. K \neq L \longrightarrow \text{trail-false-lit } \Gamma K$   
**using**  $\langle \text{trail-false-cls } \Gamma \{ \#K \in \# C. K \neq L \# \} \rangle$   
**unfolding** *trail-false-cls-def*  
**by** *simp*

**hence**  $\forall K \in \# C. K \neq L \longrightarrow \text{trail-defined-lit } \Gamma K$

```

    using trail-defined-lit-iff-true-or-false by metis

  thus  $\neg$  trail-defined-lit  $\Gamma$   $L$ 
    using  $\langle \neg$  trail-defined-cls  $\Gamma$   $C \rangle$   $\langle L \in \# C \rangle$ 
    unfolding trail-defined-cls-def
    by metis
qed

lemma propagating-clause-in-clauses:
  assumes trail-annotations-invars  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ) and map-of  $\Gamma$   $L = \text{Some} (\text{Some } C)$ 
  shows  $C \in | \text{iefac } \mathcal{F} |^1 (N \cup U_{er})$ 
  using assms
proof (induction ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ) arbitrary:  $\Gamma$  rule: trail-annotations-invars.induct)
  case Nil
  hence False
  by simp
  thus ?case ..
next
  case (Cons-None  $\Gamma$   $K$ )
  thus ?case
  by (metis map-of-Cons-code(2) option.discI option.inject)
next
  case (Cons-Some  $D$   $K$   $\Gamma$ )
  thus ?case
  by (metis map-of-Cons-code(2) option.inject)
qed

lemma trail-annotations-invars-mono-wrt-trail-suffix:
  assumes suffix  $\Gamma' \Gamma$  trail-annotations-invars  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ )
  shows trail-annotations-invars  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma'$ )
  using assms(2,1)
proof (induction ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ) arbitrary:  $\Gamma \Gamma'$  rule: trail-annotations-invars.induct)
  case Nil
  thus ?case
  by (simp add: trail-annotations-invars.Nil)
next
  case (Cons-None  $\Gamma$   $L$ )
  have  $\Gamma' = (L, \text{None}) \# \Gamma \vee \text{suffix } \Gamma' \Gamma$ 
  using Cons-None.premis unfolding suffix-Cons .
  thus ?case
  using Cons-None.hyps
  by (metis trail-annotations-invars.Cons-None)
next
  case (Cons-Some  $C$   $L$   $\Gamma$ )
  have  $\Gamma' = (L, \text{Some } C) \# \Gamma \vee \text{suffix } \Gamma' \Gamma$ 
  using Cons-Some.premis unfolding suffix-Cons .
  then show ?case
  using Cons-Some.hyps

```

by (*metis trail-annotations-invars.Cons-Some*)  
 qed

**lemma** *ord-res-8-preserves-trail-annotations-invars*:  
 assumes  
   *step: ord-res-8 N s s' and*  
   *invars:*  
     *trail-annotations-invars N s*  
     *trail-is-sorted N s*  
 shows *trail-annotations-invars N s'*  
 using *step*  
**proof** (*cases N s s' rule: ord-res-8.cases*)  
 case *step-hyps: (decide-neg F U<sub>er</sub> Γ A Γ')*  
 show ?*thesis*  
   **unfolding**  $\langle s' = (U_{er}, \mathcal{F}, \Gamma) \rangle \langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$   
**proof** (*rule trail-annotations-invars.Cons-None*)  
   **show** *trail-annotations-invars N (U<sub>er</sub>, F, Γ)*  
     **using** *invars(1) by (simp add: ⟨s = (U<sub>er</sub>, F, Γ)⟩)*  
 qed  
**next**  
 case *step-hyps: (propagate F U<sub>er</sub> Γ A C Γ')*  
 show ?*thesis*  
   **unfolding**  $\langle s' = (U_{er}, \mathcal{F}, \Gamma) \rangle \langle \Gamma' = (\text{Pos } A, \text{Some } C) \# \Gamma \rangle$   
**proof** (*rule trail-annotations-invars.Cons-Some*)  
   **show** *ord-res.is-strictly-maximal-lit (Pos A) C*  
     **using** *step-hyps by argo*  
**next**  
   **show**  $C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
     **using** *step-hyps(5)*  
     **by** (*simp add: linorder-cls.is-least-in-fset-iff*)  
**next**  
   **show** *trail-false-cl Γ {#K ∈# C. K ≠ Pos A#}*  
     **using** *step-hyps(5)*  
     **by** (*simp add: linorder-cls.is-least-in-fset-iff clause-could-propagate-def*)  
**next**  
   **show** *linorder-cls.is-least-in-fset*  
      $\{ \mid C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ clause-could-propagate } \Gamma \ C \ (Pos \ A) \mid \}$   
     **using** *step-hyps by argo*  
**next**  
   **show** *trail-annotations-invars N (U<sub>er</sub>, F, Γ)*  
     **using** *invars(1) by (simp add: ⟨s = (U<sub>er</sub>, F, Γ)⟩)*  
 qed  
**next**  
 case *step-hyps: (factorize F U<sub>er</sub> Γ A E F')*  
  
**hence** *efac E ≠ E*  
   **by** (*metis (no-types, lifting) ex1-efac-eq-factoring-chain ex-ground-factoringI*  
     *linorder-cls.is-least-in-filter-iff clause-could-propagate-iff*)



**moreover have** *clause-could-propagate*  $\Gamma E$  (*Pos A*)  
**using** *step-hyps unfolding linorder-cls.is-least-in-filter-iff* **by** *metis*

**ultimately have** *ord-res.is-strictly-maximal-lit* (*Pos A*) (*efac E*)  
**unfolding** *clause-could-propagate-def*  
**using** *ex1-efac-eq-factoring-chain ex-ground-factoringI*  
*ord-res.ground-factorings-preserves-maximal-literal* **by** *blast*

**have**  $\mathcal{F} \mid\subseteq\mid \mathcal{F}'$   
**unfolding**  $\langle \mathcal{F}' = \text{finsert } E \ \mathcal{F} \rangle$  **by** *auto*

**have** *trail-annotations-invars*  $N$  ( $U_{er}, \mathcal{F}, \Gamma$ )  
**using** *invars(1)* **by** (*simp add:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$* )

**moreover have**  $\forall A_1 \mid\in\mid \text{trail-atms } \Gamma. A_1 \prec_t A$   
**using** *step-hyps unfolding linorder-trm.is-least-in-filter-iff* **by** *blast*

**ultimately show** *?thesis*  
**unfolding**  $\langle s' = (U_{er}, \mathcal{F}', \Gamma) \rangle$   
**proof** (*induction* ( $U_{er}, \mathcal{F}, \Gamma$ ) *arbitrary:  $\Gamma$  rule: trail-annotations-invars.induct*)  
**case** *Nil*  
**show** *?case*  
**by** (*simp add: trail-annotations-invars.Nil*)

**next**  
**case** (*Cons-None*  $\Gamma L$ )  
**show** *?case*  
**proof** (*rule trail-annotations-invars.Cons-None*)  
**have**  $\forall A_1 \mid\in\mid \text{trail-atms } \Gamma. A_1 \prec_t A$   
**using** *Cons-None.prem*s **by** (*simp add: fset-trail-atms*)

**thus** *trail-annotations-invars*  $N$  ( $U_{er}, \mathcal{F}', \Gamma$ )  
**using** *Cons-None.hyps* **by** *argo*

**qed**

**next**  
**case** (*Cons-Some*  $C L \Gamma$ )

**have**  
*clause-could-propagate*  $\Gamma C L$  **and**  
*C-least:  $\forall y \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er}). y \neq C \longrightarrow \text{clause-could-propagate } \Gamma y$*   
 $L \longrightarrow C \prec_c y$   
**using** *Cons-Some.hyps(4)* **unfolding** *linorder-cls.is-least-in-filter-iff* **by** *metis+*

**hence** *ord-res.is-maximal-lit*  $L C$   
**unfolding** *clause-could-propagate-def* **by** *argo*

**show** *?case*  
**proof** (*rule trail-annotations-invars.Cons-Some*)  
**show** *ord-res.is-strictly-maximal-lit*  $L C$

```

using ⟨ord-res.is-strictly-maximal-lit L C⟩ .

have efac C = C
using Cons-Some
by (metis (no-types, opaque-lifting) efac-spec is-pos-def linorder-lit. Uniq-is-maximal-in-mset
linorder-lit.is-greatest-in-mset-iff-is-maximal-and-count-eq-one
nex-strictly-maximal-pos-lit-if-neq-efac the1-equality')

hence C |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )
using ⟨C |∈| iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )⟩ ⟨ $\mathcal{F}$  | $\subseteq$ |  $\mathcal{F}'$ ⟩
by (smt (verit, best) assms fimage-iff fsubsetD iefac-def)

show C |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )
using ⟨C |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )⟩ .

show trail-false-clc  $\Gamma$  {#x ∈# C. x ≠ L#}
using ⟨trail-false-clc  $\Gamma$  {#x ∈# C. x ≠ L#}⟩ .

show linorder-clc.is-least-in-fset
{C |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ). clause-could-propagate  $\Gamma$  C L} C
unfolding linorder-clc.is-least-in-filter-iff
proof (intro conjI ballI impI)
show C |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )
using ⟨C |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )⟩ .
next
show clause-could-propagate  $\Gamma$  C L
using ⟨clause-could-propagate  $\Gamma$  C L⟩ .
next
fix D :: 'f gterm literal multiset
assume D |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ ) and D ≠ C and clause-could-propagate
 $\Gamma$  D L

have atm-of L <t A
using Cons-Some.prem by (simp add: fset-trail-atms)

hence L <l Pos A
by (cases L) simp-all

moreover have ord-res.is-maximal-lit L D
using ⟨clause-could-propagate  $\Gamma$  D L⟩ unfolding clause-could-propagate-iff
by metis

ultimately have D <c efac E
using ⟨ord-res.is-strictly-maximal-lit (Pos A) (efac E)⟩
by (metis linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset
linorder-lit.mulp-if-maximal-less-than-maximal)

hence D |∈| iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )
using ⟨D |∈| iefac  $\mathcal{F}'$  | $\uparrow$  (N | $\cup$ |  $U_{er}$ )⟩

```

```

unfolding ⟨ $\mathcal{F}' = \text{finsert } E \ \mathcal{F}$ ⟩
using iefac-def by force

thus  $C \prec_c D$ 
using C-least ⟨ $D \neq C$ ⟩ clause-could-propagate  $\Gamma \ D \ L$  by metis
qed

have  $\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t A$ 
using Cons-Some.premis by (simp add: fset-trail-atms)

thus trail-annotations-invars  $N \ (U_{er}, \mathcal{F}', \Gamma)$ 
using Cons-Some.hyps by argo
qed
qed
next
case step-hyps: (resolution  $\Gamma \ \mathcal{F} \ U_{er} \ E \ A \ D \ U_{er}' \ \Gamma'$ )

show ?thesis
proof (cases eres  $D \ E = \{\#\}$ )
case True
hence  $\nexists K. \text{ord-res.is-maximal-lit } K \ (\text{eres } D \ E)$ 
by (simp add: linorder-lit.is-maximal-in-mset-iff)
hence  $\Gamma' = []$ 
unfolding step-hyps by simp
thus ?thesis
unfolding ⟨ $s' = (U_{er}', \mathcal{F}, \Gamma')$ ⟩
using trail-annotations-invars.Nil by metis
next
case False

then obtain  $K$  where eres-max-lit: linorder-lit.is-maximal-in-mset (eres  $D \ E$ )
using linorder-lit.ex-maximal-in-mset by metis

have  $\Gamma'.\text{eq}: \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \ \Gamma$ 
unfolding step-hyps(7)
proof (rule dropWhile-cong)
show  $\Gamma = \Gamma ..$ 
next
fix  $x :: 'f \text{ gterm literal} \times 'f \text{ gterm literal multiset option}$ 
assume  $x \in \text{set } \Gamma$ 
show ( $\forall K. \text{ord-res.is-maximal-lit } K \ (\text{eres } D \ E) \longrightarrow \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } x)$ ) =
  (atm-of  $K \preceq_t \text{atm-of } (\text{fst } x)$ )
unfolding case-prod-beta
using eres-max-lit
by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset)
qed

```

```

have trail-annotations-invars  $N (U_{er}, \mathcal{F}, \Gamma)$ 
  using invars(1) by (simp add:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$ )

moreover have suffix  $\Gamma' \Gamma$ 
  unfolding step-hyps
  using suffix-dropWhile by metis

moreover have  $\forall Ln \in \text{set } \Gamma'. \neg (\text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln))$ 
  unfolding  $\Gamma'\text{-eq}$ 
proof (rule ball-set-dropWhile-if-sorted-wrt-and-monotone-on)
  have sorted-wrt  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$ 
    using invars(2)
    by (simp add:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  trail-is-sorted-def sorted-wrt-map)

  thus sorted-wrt  $(\lambda x y. \text{fst } y \prec_l \text{fst } x) \Gamma$ 
proof (rule sorted-wrt-mono-rel[rotated])
  show  $\bigwedge x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x) \implies \text{fst } y \prec_l \text{fst } x$ 
  by (metis (no-types, lifting) linorder-lit.antisym-conv3 linorder-trm.dual-order.asym
    literal.exhaust-sel ord-res.less-lit-simps(1) ord-res.less-lit-simps(3)
    ord-res.less-lit-simps(4))

  qed
next
  show monotone-on  $(\text{set } \Gamma) (\lambda x y. \text{fst } y \prec_l \text{fst } x) (\lambda Ln y. y \leq Ln)$ 
   $(\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln))$ 
  apply (simp add: monotone-on-def)
  by (smt (verit, best) Neg-atm-of-iff Pos-atm-of-iff linorder-lit.order.asym
    linorder-trm.less-linear linorder-trm.order.strict-trans ord-res.less-lit-simps(1)
    ord-res.less-lit-simps(3) ord-res.less-lit-simps(4))

  qed

ultimately show ?thesis
  unfolding  $\langle s' = (U_{er}', \mathcal{F}, \Gamma') \rangle$ 
proof (induction  $(U_{er}, \mathcal{F}, \Gamma)$  arbitrary:  $\Gamma \Gamma'$  rule: trail-annotations-invars.induct)
  case Nil
  thus ?case
  by (simp add: trail-annotations-invars.Nil)
next
  case  $(\text{Cons-None } \Gamma L)$ 
  thus ?case
  by (metis insert-iff list.simps(15) suffix-Cons suffix-order.dual-order.refl
    trail-annotations-invars.Cons-None)
next
  case  $(\text{Cons-Some } C L \Gamma)$ 

  have
    clause-could-propagate  $\Gamma C L$  and
    C-least:  $\forall y \in |\text{iefac } \mathcal{F} |^{\uparrow} (N \cup U_{er}). y \neq C \implies \text{clause-could-propagate } \Gamma$ 
     $y L \implies C \prec_c y$ 
    using Cons-Some.hyps(4) unfolding linorder-cls.is-least-in-filter-iff by

```

*metis+*

```

hence C-max-lit: ord-res.is-maximal-lit L C
unfolding clause-could-propagate-def by argo

obtain  $\Gamma''$  where  $(L, \text{Some } C) \# \Gamma = \Gamma'' @ \Gamma'$ 
using Cons-Some.premis by  $(\text{auto elim: suffixE})$ 

show ?case
proof  $(\text{cases } \Gamma')$ 
case Nil
hence  $\Gamma' = (L, \text{Some } C) \# \Gamma$ 
using  $\langle (L, \text{Some } C) \# \Gamma = \Gamma'' @ \Gamma' \rangle$  by simp

show ?thesis
unfolding  $\langle \Gamma' = (L, \text{Some } C) \# \Gamma \rangle$ 
proof  $(\text{rule trail-annotations-invars.Cons-Some})$ 
show ord-res.is-strictly-maximal-lit L C
using  $\langle \text{ord-res.is-strictly-maximal-lit L C} \rangle$  .

show  $C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er'})$ 
using  $\langle C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}) \rangle \langle U_{er'} = \text{finsert } (\text{eres } D E) U_{er} \rangle$ 
by simp

show trail-false-cls  $\Gamma \{ \#K \in \# C. K \neq L \# \}$ 
using  $\langle \text{trail-false-cls } \Gamma \{ \#K \in \# C. K \neq L \# \} \rangle$  .

show linorder-cls.is-least-in-fset  $\{ | C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er'}) .$ 
clause-could-propagate  $\Gamma C L \}$  C
unfolding linorder-cls.is-least-in-ffilter-iff
proof  $(\text{intro conjI ballI impI})$ 
show  $C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er'})$ 
using  $\langle C \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er'}) \rangle$  .
next
show clause-could-propagate  $\Gamma C L$ 
using Cons-Some.hyps(4) unfolding linorder-cls.is-least-in-ffilter-iff
by metis
next
fix  $x :: 'f \text{ gterm literal multiset}$ 
assume  $x \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er'})$ 
and  $x \neq C$ 
and clause-could-propagate  $\Gamma x L$ 

have linorder-lit.is-maximal-in-mset  $x L$ 
using  $\langle \text{clause-could-propagate } \Gamma x L \rangle$  unfolding clause-could-propagate-def
by argo

moreover have  $L \prec_t K$ 
using  $\langle \forall Ln \in \text{set } \Gamma'. \neg (\text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \rangle$ 

```

**unfolding**  $\langle \Gamma' = (L, \text{Some } C) \# \Gamma \rangle$   
**apply** *simp*  
**by** (*metis* *Neg-atm-of-iff* *linorder-lit.antisym-conv3* *linorder-trm.less-linear*  
*literal.collapse(1)* *ord-res.less-lit-simps(1)* *ord-res.less-lit-simps(3)*  
*ord-res.less-lit-simps(4)*)

**ultimately have** *set-mset*  $x \neq \text{set-mset } (\text{eres } D \ E)$   
**using** *eres-max-lit*  
**by** (*metis* *linorder-lit.is-maximal-in-mset-iff* *linorder-lit.neq-iff*)

**hence**  $x \neq \text{iefac } \mathcal{F} \ (\text{eres } D \ E)$   
**using** *iefac-def* **by** *auto*

**hence**  $x \in | \text{iefac } \mathcal{F} \ |^{\dagger} (N \ |\cup| \ U_{er})$   
**using**  $\langle x \in | \text{iefac } \mathcal{F} \ |^{\dagger} (N \ |\cup| \ U_{er}) \rangle$   
**unfolding**  $\langle U_{er}' = \text{finsert } (\text{eres } D \ E) \ U_{er} \rangle$   
**by** *simp*

**then show**  $C \prec_c x$   
**using** *C-least*  $\langle x \neq C \rangle$  *clause-could-propagate*  $\Gamma \ x \ L$  **by** *metis*  
**qed**

**show** *trail-annotations-invars*  $N \ (U_{er}', \mathcal{F}, \Gamma)$   
**using** *Cons-Some*  
**by** (*simp* *add*:  $\langle \Gamma' = (L, \text{Some } C) \# \Gamma \rangle$ )  
**qed**

**next**  
**case** (*Cons a list*)  
**then show** *?thesis*  
**by** (*metis* *Cons-Some.hyps(6)* *Cons-Some.prem*s  $\langle (L, \text{Some } C) \# \Gamma = \Gamma''$   
 $\text{@ } \Gamma' \rangle$  *empty-iff*  
*list.set(1)* *list.set-intros(1)* *self-append-conv2* *suffix-Cons*)  
**qed**

**qed**  
**qed**  
**qed**  
**qed**

**definition** *trail-is-lower-set* **where**  
*trail-is-lower-set*  $N \ s \longleftrightarrow$   
 $(\forall U_{er} \ \mathcal{F} \ \Gamma. \ s = (U_{er}, \mathcal{F}, \Gamma) \longrightarrow$   
*linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-clss*  $(N \ |\cup| \ U_{er}))$ )

**lemma** *atoms-not-in-clause-set-undefined-if-trail-is-sorted-lower-set*:  
**assumes** *invar*: *trail-is-lower-set*  $N \ (U_{er}, \mathcal{F}, \Gamma)$   
**shows**  $\forall A. \ A \notin | \text{atms-of-clss } (N \ |\cup| \ U_{er}) \longrightarrow A \notin | \text{trail-atms } \Gamma$   
**using** *invar*[*unfolded* *trail-is-lower-set-def*, *simplified*]  
**by** (*metis* *Un-iff* *linorder-trm.is-lower-set-iff* *sup.absorb2*)

**lemma** *ord-res-8-preserves-atoms-in-trail-lower-set*:

**assumes**  
*step*: *ord-res-8*  $N$   $s$   $s'$  **and**  
*invars*:  
*trail-is-lower-set*  $N$   $s$   
*trail-annotations-invars*  $N$   $s$   
*trail-is-sorted*  $N$   $s$   
**shows** *trail-is-lower-set*  $N$   $s'$   
**using** *step*  
**proof** (*cases*  $N$   $s$   $s'$  *rule*: *ord-res-8.cases*)  
**case** *step-hyps*: (*decide-neg*  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $\Gamma'$ )

**have**  
*A-in*:  $A \in |atms-of-clss (N \cup U_{er})|$  **and**  
*A-gt*:  $\forall x \in |trail-atms \Gamma|. x \prec_t A$  **and**  
*A-least*:  $\forall x \in |atms-of-clss (N \cup U_{er})|. (\forall w \in |trail-atms \Gamma|. w \prec_t x) \longrightarrow x \neq A \longrightarrow A \prec_t x$   
**using** *step-hyps* **unfolding** *linorder-trm.is-least-in-filter-iff* **by** *simp-all*

**have** *trail-atms*  $\Gamma' = \text{finsert } A \text{ (trail-atms } \Gamma)$   
**using** *step-hyps* **by** *simp*

**have**  
*inv1*: *sorted-wrt*  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$  **and**  
*inv2*: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-clss*  $(N \cup U_{er})$ )  
**using** *invars*(1,3)  
**by** (*simp-all* *only*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *trail-is-lower-set-def* *trail-is-sorted-def*)

**have** *sorted-wrt*  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma'$   
**unfolding**  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  *list.map sorted-wrt.simps*  
**proof** (*intro conjI ballI*)  
**fix**  $x$   
**assume**  $x \in set \Gamma$   
**hence** *atm-of*  $(fst x) \in |trail-atms \Gamma|$   
**by** (*auto simp: fset-trail-atms*)  
**hence** *atm-of*  $(fst x) \prec_t atm-of (Neg A)$   
**using** *A-gt* **by** *simp*  
**thus** *atm-of*  $(fst x) \prec_t atm-of (fst (Neg A, None))$   
**unfolding** *comp-def prod.sel* .

**next**  
**show** *sorted-wrt*  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$   
**using** *inv1* .

**qed**

**moreover** **have** *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma'$ ) (*atms-of-clss*  $(N \cup U_{er})$ )  
**unfolding**  $\langle trail-atms \Gamma' = \text{finsert } A \text{ (trail-atms } \Gamma) \rangle$  *finsert.rep-eq*  
**proof** (*rule linorder-trm.is-lower-set-insertI*)  
**show**  $A \in |atms-of-clss (N \cup U_{er})|$   
**using** *A-in* .

```

next
  show  $\forall w \in | \text{atms-of-clss } (N \cup U_{er}). w \prec_t A \longrightarrow w \in | \text{trail-atms } \Gamma$ 
    using A-least inv2
    by (metis linorder-trm.is-lower-set-iff linorder-trm.not-less-iff-gr-or-eq)
next
  show linorder-trm.is-lower-fset (trail-atms  $\Gamma$ )
    (atms-of-clss ( $N \cup U_{er}$ ))
    using inv2 .
qed

ultimately show ?thesis
  by (simp add:  $\langle s' = (U_{er}, \mathcal{F}, \Gamma') \rangle$  trail-is-lower-set-def)
next
  case step-hyps: (propagate  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $C$   $\Gamma'$ )

  have
    A-in: A  $\in | \text{atms-of-clss } (N \cup U_{er})$  and
    A-gt:  $\forall x \in | \text{trail-atms } \Gamma. x \prec_t A$  and
    A-least:  $\forall x \in | \text{atms-of-clss } (N \cup U_{er}). (\forall w \in | \text{trail-atms } \Gamma. w \prec_t x) \longrightarrow x$ 
 $\neq A \longrightarrow A \prec_t x$ 
    using step-hyps unfolding linorder-trm.is-least-in-filter-iff by simp-all

  have trail-atms  $\Gamma' = \text{finsert } A$  (trail-atms  $\Gamma$ )
    using step-hyps by simp

  have
    inv1: sorted-wrt ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma$  and
    inv2: linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-clss ( $N \cup U_{er}$ ))
    using invars(1,3)
    by (simp-all only:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  trail-is-lower-set-def trail-is-sorted-def)

  have sorted-wrt ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma'$ 
    unfolding  $\langle \Gamma' = (Pos A, Some C) \# \Gamma \rangle$  list.map sorted-wrt.simps
  proof (intro conjI ballI)
    fix  $x$ 
    assume  $x \in \text{set } \Gamma$ 
    hence atm-of ( $fst x$ )  $\in | \text{trail-atms } \Gamma$ 
      by (auto simp: fset-trail-atms)
    hence atm-of ( $fst x$ )  $\prec_t \text{atm-of } (Pos A)$ 
      using A-gt by simp
    thus atm-of ( $fst x$ )  $\prec_t \text{atm-of } (fst (Pos A, Some C))$ 
      unfolding comp-def prod.sel .
  next
    show sorted-wrt ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma$ 
      using inv1 .
  qed

moreover have linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-clss ( $N \cup U_{er}$ ))

```



**unfolding**  $\langle \text{trail-atms } \Gamma' = \text{finsert } A (\text{trail-atms } \Gamma) \rangle \text{finsert.rep-eq}$   
**proof** (rule *linorder-trm.is-lower-set-insertI*)  
**show**  $A \in | \text{atms-of-clss } (N \cup U_{er})$   
**using** *A-in* .  
**next**  
**show**  $\forall w \in | \text{atms-of-clss } (N \cup U_{er}). w \prec_t A \longrightarrow w \in | \text{trail-atms } \Gamma$   
**using** *A-least inv2*  
**by** (metis *linorder-trm.is-lower-set-iff linorder-trm.not-less-iff-gr-or-eq*)  
**next**  
**show** *linorder-trm.is-lower-fset* (trail-atms  $\Gamma$ ) (atms-of-clss  $(N \cup U_{er})$ )  
**using** *inv2* .  
**qed**

**ultimately show** *?thesis*  
**by** (simp add:  $\langle s' = (U_{er}, \mathcal{F}, \Gamma) \rangle \text{trail-is-lower-set-def}$ )  
**next**  
**case** (factorize  $\mathcal{F} U_{er} \Gamma A C \mathcal{F}'$ )  
**thus** *?thesis*  
**using** *invars(1)* **by** (simp add: *trail-is-lower-set-def fset-trail-atms*)  
**next**  
**case** *step-hyps*: (resolution  $\Gamma \mathcal{F} U_{er} D A C U_{er}' \Gamma'$ )

**have** *suffix*  $\Gamma' \Gamma$   
**using** *step-hyps suffix-dropWhile* **by** *blast*

**then obtain**  $\Gamma''$  **where**  $\Gamma = \Gamma'' @ \Gamma'$   
**unfolding** *suffix-def* **by** *metis*

**have** *atms-of-clss*  $(N \cup U_{er}') = \text{atms-of-clss } (\text{finsert } (\text{eres } C D) (N \cup U_{er}))$   
**unfolding**  $\langle U_{er}' = \text{finsert } (\text{eres } C D) U_{er} \rangle$  **by** *simp*  
**also have**  $\dots = \text{atms-of-clss } (\text{eres } C D) \cup | \text{atms-of-clss } (N \cup U_{er})$   
**unfolding** *atms-of-clss-finsert ..*  
**also have**  $\dots = \text{atms-of-clss } (N \cup U_{er})$   
**proof** –  
**have**  $\forall A \in | \text{atms-of-clss } (\text{eres } C D). A \in | \text{atms-of-clss } C \vee A \in | \text{atms-of-clss } D$   
**using** *lit-in-one-of-resolvents-if-in-eres*  
**by** (smt (verit, best) *atms-of-clss-def fimage-iff fset-fset-mset*)

**moreover have**  $C \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er})$   
**using** *invars(2)[unfolded  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$ ]* *step-hyps(5)*  
**by** (metis *propagating-clause-in-clauses*)

**moreover have**  $D \in | \text{iefac } \mathcal{F} \uparrow (N \cup U_{er})$   
**using** *linorder-clss.is-least-in-filter-iff* *step-hyps(3)* **by** *blast*

**ultimately have**  $\forall A \in | \text{atms-of-clss } (\text{eres } C D). A \in | \text{atms-of-clss } (\text{iefac } \mathcal{F} \uparrow (N \cup U_{er}))$   
**by** (metis *atms-of-clss-finsert funion-iff mk-disjoint-finsert*)

**hence**  $\forall A \mid \in \mid \text{atms-of-cls } (\text{eres } C D). A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**unfolding**  $\text{atms-of-clss-fimage-iefac}$  .

**thus** *?thesis*  
**by** *blast*

**qed**

**finally have**  $\text{atms-of-clss } (N \mid \cup \mid U_{er}') = \text{atms-of-clss } (N \mid \cup \mid U_{er})$  .

**have**

*inv1*:  $\text{sorted-wrt } (\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$  **and**  
*inv2*:  $\text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er}))$   
**using**  $\text{invars}(1,3)$   
**by**  $(\text{simp-all only: } \langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle \text{ trail-is-lower-set-def trail-is-sorted-def})$

**have**  $\text{sorted-wrt } (\lambda x y. y \prec_t x) (\text{map } (\text{atm-of } \circ \text{fst}) \Gamma)$   
**using** *inv1* **by**  $(\text{simp add: sorted-wrt-map})$

**hence**  $\text{sorted-wrt } (\lambda x y. y \prec_t x) (\text{map } (\text{atm-of } \circ \text{fst}) \Gamma'' @ \text{map } (\text{atm-of } \circ \text{fst}) \Gamma')$   
**by**  $(\text{simp add: } \langle \Gamma = \Gamma'' @ \Gamma' \rangle)$

**moreover have**  $\text{linorder-trm.is-lower-set } (\text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \Gamma'' @ \text{map } (\text{atm-of } \circ \text{fst}) \Gamma'))$   
 $(\text{fset } (\text{atms-of-clss } (N \mid \cup \mid U_{er})))$   
**using** *inv2*  $\langle \Gamma = \Gamma'' @ \Gamma' \rangle$   
**by**  $(\text{metis image-comp list.set-map map-append fset-trail-atms})$

**ultimately have**  $\text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma') (\text{atms-of-clss } (N \mid \cup \mid U_{er}'))$   
**using**  $\text{linorder-trm.sorted-and-lower-set-appendD-right}$   
**using**  $\langle \text{atms-of-clss } (N \mid \cup \mid U_{er}') = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$   
**by**  $(\text{metis (no-types, lifting) image-comp list.set-map fset-trail-atms})$

**thus** *?thesis*

**by**  $(\text{simp add: } \langle s' = (U_{er}', \mathcal{F}, \Gamma') \rangle \text{ trail-is-lower-set-def})$

**qed**

**definition**  $\text{false-cls-is-empty-or-has-neg-max-lit}$  **where**

$\text{false-cls-is-empty-or-has-neg-max-lit } N s \longleftrightarrow$   
 $(\forall U_{er} \mathcal{F} \Gamma. s = (U_{er}, \mathcal{F}, \Gamma) \longrightarrow (\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}).$   
 $\text{trail-false-cls } \Gamma C \longrightarrow C = \{\#\} \vee (\exists A. \text{linorder-lit.is-maximal-in-mset } C$   
 $(\text{Neg } A))))$

**lemma**  $\text{ord-res-8-preserves-false-cls-is-empty-or-has-neg-max-lit}$ :

**assumes**

*step*:  $\text{ord-res-8 } N s s'$  **and**

*invars*:

$\text{false-cls-is-empty-or-has-neg-max-lit } N s$

$\text{trail-is-lower-set } N s$

*trail-is-sorted*  $N\ s$   
**shows** *false-cls-is-mempty-or-has-neg-max-lit*  $N\ s'$   
**using** *step*  
**proof** (*cases*  $N\ s\ s'$  *rule: ord-res-8.cases*)  
**case** *step-hyps*: (*decide-neg*  $\mathcal{F}\ U_{er}\ \Gamma\ A\ \Gamma'$ )

**have**  $\Gamma$ -*lower*: *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-cls* ( $N\ |\cup|\ U_{er}$ ))  
**using**  $\langle$ *trail-is-lower-set*  $N\ s$  $\rangle$ [*unfolded trail-is-lower-set-def*,  
*rule-format*, *OF*  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$ ].

**have**  $\Gamma$ -*sorted*: *sorted-wrt* ( $\lambda x\ y.\ \text{atm-of}\ (fst\ y)\ \prec_t\ \text{atm-of}\ (fst\ x)$ )  $\Gamma$   
**using**  $\langle$ *trail-is-sorted*  $N\ s$  $\rangle$ [*unfolded trail-is-sorted-def*,  
*rule-format*, *OF*  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$ ].

**have** *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms*[*OF*  $\Gamma$ -*sorted*].

**hence** *trail-consistent*  $\Gamma'$   
**unfolding**  $\langle \Gamma' = (Neg\ A,\ None)\ \# \ \Gamma \rangle$   
**proof** (*rule trail-consistent.Cons* [*rotated*])  
**show**  $\neg$  *trail-defined-lit*  $\Gamma\ (Neg\ A)$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm*  
**using** *linorder-trm.is-least-in-fset-ffilterD*(2) *linorder-trm.less-irrefl* *step-hyps*(4)  
*trail-defined-lit-iff-trail-defined-atm* **by** *force*

**qed**

**have** *atm-defined-iff-lt-A*:  $x\ |\in|\ \text{trail-atms}\ \Gamma\ \longleftrightarrow\ x\ \prec_t\ A$   
**if**  $x$ -*in*:  $x\ |\in|\ \text{atms-of-cls}\ (N\ |\cup|\ U_{er})$  **for**  $x$   
**proof** (*rule iffI*)  
**assume**  $x\ |\in|\ \text{trail-atms}\ \Gamma$   
**thus**  $x\ \prec_t\ A$   
**using** *step-hyps*(4)  
**unfolding** *linorder-trm.is-least-in-ffilter-iff*  
**by** *blast*

**next**  
**assume**  $x\ \prec_t\ A$   
**thus**  $x\ |\in|\ \text{trail-atms}\ \Gamma$   
**using** *step-hyps*(4)[*unfolded linorder-trm.is-least-in-ffilter-iff*]  
**using**  $\Gamma$ -*lower*[*unfolded linorder-trm.is-lower-set-iff*]  
**by** (*metis linorder-trm.dual-order.asym linorder-trm.neq-iff x-in*)

**qed**

**have**  $\neg$  *trail-false-cls*  $\Gamma'\ C$  **if**  $C$ -*in*:  $C\ |\in|\ \text{iefac}\ \mathcal{F}\ |\cdot|\ (N\ |\cup|\ U_{er})$  **for**  $C$   
**proof** –  
**have**  $\neg$  *trail-false-cls*  $\Gamma\ C$   
**using**  $C$ -*in* *step-hyps*(3) **by** *metis*  
**hence** *trail-true-cls*  $\Gamma\ C\ \vee\ \neg$  *trail-defined-cls*  $\Gamma\ C$   
**using** *trail-true-or-false-cls-if-defined* **by** *metis*  
**thus**  $\neg$  *trail-false-cls*  $\Gamma'\ C$

```

proof (elim disjE)
  assume trail-true-cls  $\Gamma$   $C$ 
  hence trail-true-cls  $\Gamma'$   $C$ 
    unfolding  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$  trail-true-cls-def
    by (metis image-insert insert-iff list.set(2) trail-true-lit-def)
  thus  $\neg$  trail-false-cls  $\Gamma'$   $C$ 
    using  $\langle$ trail-consistent  $\Gamma'$  $\rangle$ 
    by (metis not-trail-true-cls-and-trail-false-cls)
next
  assume  $\neg$  trail-defined-cls  $\Gamma$   $C$ 
  then obtain  $L$  where  $L$ -max: ord-res.is-maximal-lit  $L$   $C$ 
  by (metis  $\langle$  $\neg$  trail-false-cls  $\Gamma$   $C$  $\rangle$  linorder-lit.ex-maximal-in-mset trail-false-cls-empty)

  have  $L \in \#$   $C$ 
    using  $L$ -max linorder-lit.is-maximal-in-mset-iff by metis

  have atm-of  $L \mid \in \mid$  atms-of-cls ( $N \mid \cup \mid$   $U_{er}$ )
    using  $C$ -in  $\langle L \in \#$   $C \rangle$  by (metis atm-of-in-atms-of-clsI)

  show  $\neg$  trail-false-cls  $\Gamma'$   $C$ 
  proof (cases atm-of  $L = A$ )
    case True

      have  $\neg$  trail-defined-lit  $\Gamma$  ( $\text{Pos } A$ )
        using step-hyps(4)
      unfolding trail-defined-lit-iff-trail-defined-atm linorder-trm.is-least-in-ffilter-iff
        by auto

      hence ( $\exists K \in \#$   $C$ .  $K \neq \text{Pos } A \wedge \neg$  trail-false-lit  $\Gamma$   $K$ )  $\vee$ 
         $\neg$  ord-res.is-maximal-lit ( $\text{Pos } A$ )  $C$ 
        using step-hyps(5)  $C$ -in
        unfolding clause-could-propagate-iff
        unfolding bex-simps de-Morgan-conj not-not by blast

    thus ?thesis
  proof (elim disjE bexE conjE)
    fix  $K :: 'f$  gterm literal
    assume  $K \in \#$   $C$  and  $K \neq \text{Pos } A$  and  $\neg$  trail-false-lit  $\Gamma$   $K$ 
    thus  $\neg$  trail-false-cls  $\Gamma'$   $C$ 
      by (smt (verit) fst-conv image-iff insertE list.simps(15) step-hyps(6)
        trail-false-cls-def trail-false-lit-def uminus-Pos uminus-lit-swap)
  next
  assume  $\neg$  ord-res.is-maximal-lit ( $\text{Pos } A$ )  $C$ 

  hence  $L = \text{Neg } A$ 
    using  $\langle$ atm-of  $L = A \rangle$   $L$ -max by (metis literal.exhaust-sel)

  thus  $\neg$  trail-false-cls  $\Gamma'$   $C$ 
    unfolding  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$ 

```

**unfolding** *trail-false-cls-def trail-false-lit-def*  
**using**  $\langle L \in \# C \rangle [\text{unfolded } \langle L = \text{Neg } A \rangle]$   
**by** (*metis*  $\langle \neg \text{trail-defined-cls } \Gamma C \rangle$  *fst-conv image-insert insertE*  
*list.simps(15)*  
*trail-defined-cls-def trail-defined-lit-def uminus-lit-swap uminus-not-id'*)  
**qed**  
**next**  
**case** *False*

**moreover have**  $\neg \text{atm-of } L \prec_t A$   
**proof** (*rule notI*)  
**assume** *atm-of*  $L \prec_t A$   
**moreover have**  $\forall K \in \# C. \text{atm-of } K \preceq_t \text{atm-of } L$   
**using** *L-max linorder-lit.is-maximal-in-mset-iff*  
**by** (*metis Neg-atm-of-iff linorder-trm.le-less-linear linorder-trm.linear*  
*literal.collapse(1) ord-res.less-lit-simps(1) ord-res.less-lit-simps(2)*  
*ord-res.less-lit-simps(3) ord-res.less-lit-simps(4)*)  
**ultimately have**  $\forall K \in \# C. \text{atm-of } K \prec_t A$   
**by** (*metis linorder-trm.antisym-conv1 linorder-trm.dual-order.strict-trans*)  
**moreover have**  $\forall K \in \# C. \text{atm-of } K \in | \text{atms-of-cls } (N \cup U_{er})$   
**using** *C-in* **by** (*metis atm-of-in-atms-of-clsI*)  
**ultimately have**  $\forall K \in \# C. \text{atm-of } K \in | \text{trail-atms } \Gamma$   
**using** *atm-defined-iff-lt-A* **by** *metis*  
**hence**  $\forall K \in \# C. \text{trail-defined-lit } \Gamma K$   
**using** *trail-defined-lit-iff-trail-defined-atm* **by** *metis*  
**hence** *trail-defined-cls*  $\Gamma C$   
**unfolding** *trail-defined-cls-def* **by** *argo*  
**thus** *False*  
**using**  $\langle \neg \text{trail-defined-cls } \Gamma C \rangle$  **by** *contradiction*  
**qed**

**ultimately have**  $A \prec_t \text{atm-of } L$   
**by** *order*

**hence** *atm-of*  $L \notin | \text{trail-atms } \Gamma'$   
**unfolding**  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$   
**unfolding** *trail-atms.simps prod.sel literal.sel*  
**using** *atm-defined-iff-lt-A[OF*  $\langle \text{atm-of } L \in | \text{atms-of-cls } (N \cup U_{er}) \rangle]$   
**using**  $\langle \neg \text{atm-of } L \prec_t A \rangle$  **by** *simp*

**hence**  $\neg \text{trail-defined-lit } \Gamma' L$   
**using** *trail-defined-lit-iff-trail-defined-atm* **by** *blast*

**hence**  $\neg \text{trail-false-lit } \Gamma' L$   
**using** *trail-defined-lit-iff-true-or-false* **by** *blast*

**thus**  $\neg \text{trail-false-cls } \Gamma' C$   
**unfolding** *trail-false-cls-def*  
**using**  $\langle L \in \# C \rangle$  **by** *metis*

qed  
 qed  
 qed

hence  $\forall C \in |\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})|. \text{ trail-false-cls } \Gamma' C \longrightarrow$   
 $C = \{\#\} \vee (\exists A. \text{ ord-res.is-maximal-lit } (\text{Neg } A) C)$   
 by *metis*

thus *?thesis*

unfolding *false-cls-is-mempty-or-has-neg-max-lit-def*  $\langle s' = (U_{er}, \mathcal{F}, \Gamma') \rangle$  by  
*simp*  
 next

case *step-hyps*: (*propagate*  $\mathcal{F} U_{er} \Gamma A C \Gamma'$ )

have  $E = \{\#\} \vee (\exists A. \text{ ord-res.is-maximal-lit } (\text{Neg } A) E)$

if *E-in*:  $E \in |\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})|$  and *E-false*: *trail-false-cls*  $\Gamma' E$  for *E*

proof (*cases*  $A \in \text{ atm-of } \langle \text{ set-mset } E \rangle$ )

case *True*

have  $\neg \text{ trail-false-cls } \Gamma E$

using  $\langle \neg \text{ fBex } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) (\text{trail-false-cls } \Gamma) \rangle$  *E-in* by *metis*

hence  $\text{Neg } A \in \# E$

using *E-false*[*unfolded*  $\langle \Gamma' = (\text{Pos } A, \text{Some } C) \# \Gamma \rangle$ ]

by (*metis subtrail-falseI uminus-Pos*)

have *atm-of*  $L \in |\text{ trail-atms } \Gamma'|$  if  $L \in \# E$  for *L*

using *E-false*  $\langle L \in \# E \rangle$

by (*simp add: atm-of-in-atm-of-set-iff-in-set-or-uminus-in-set fset-trail-atms*  
*trail-false-cls-def trail-false-lit-def*)

moreover have  $x \prec_t A$  if  $x \in |\text{ trail-atms } \Gamma|$  for *x*

using *step-hyps*(4) that

by (*simp add: linorder-trm.is-least-in-ffilter-iff*)

ultimately have *atm-of*  $L \preceq_t A$  if  $L \in \# E$  for *L*

unfolding  $\langle \Gamma' = (\text{Pos } A, \text{Some } C) \# \Gamma \rangle$  *trail-atms.simps prod.sel literal.sel*

using  $\langle L \in \# E \rangle$  by *blast*

hence  $L \preceq_l \text{Neg } A$  if  $L \in \# E$  for *L*

using  $\langle L \in \# E \rangle$

by (*metis linorder-lit.leI linorder-trm.leD literal.collapse(1) literal.collapse(2)*  
*ord-res.less-lit-simps(3) ord-res.less-lit-simps(4)*)

hence  $\exists A. \text{ ord-res.is-maximal-lit } (\text{Neg } A) E$

using  $\langle \text{Neg } A \in \# E \rangle$

by (*metis*  $\langle \neg \text{ trail-false-cls } \Gamma E \rangle$  *linorder-lit.ex-maximal-in-mset*  
*linorder-lit.is-maximal-in-mset-iff reflclp-iff trail-false-cls-mempty*)

thus *?thesis* ..

**next**  
**case** *False*  
**hence** *trail-false-cls*  $\Gamma$  *E*  
**using** *E-false*[*unfolded*  $\langle \Gamma' = (Pos\ A, Some\ C) \# \Gamma \rangle$ ]  
**by** (*metis atm-of-in-atm-of-set-iff-in-set-or-uminus-in-set literal.sel(1) sub-trail-falseI*)

**moreover have**  $\neg$  *trail-false-cls*  $\Gamma$  *E*  
**using**  $\langle \neg$  *fBex* (*iefac*  $\mathcal{F}$   $| \uparrow$  ( $N \mid \cup \mid U_{er}$ )) (*trail-false-cls*  $\Gamma$ )  $\rangle$  *E-in* **by** *metis*

**ultimately have** *False*  
**by** *contradiction*

**thus** *?thesis ..*  
**qed**

**thus** *?thesis*  
**unfolding** *false-cls-is-empty-or-has-neg-max-lit-def*  $\langle s' = (U_{er}, \mathcal{F}, \Gamma) \rangle$  **by**  
*simp*  
**next**  
**case** *step-hyps*: (*factorize*  $\mathcal{F}$   $U_{er}$   $\Gamma$  *A* *C*  $\mathcal{F}'$ )

**have** *trail-false-cls*  $\Gamma$  (*iefac*  $\mathcal{F}$  *C*)  $\longleftrightarrow$  *trail-false-cls*  $\Gamma$  *C* **if**  $C \in \mid N \mid \cup \mid U_{er}$  **for**  
 $\mathcal{F}$  *C*  
**using** *that* **by** (*simp add: iefac-def trail-false-cls-def*)

**hence**  $\forall C \in \mid iefac\ \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$ .  
*trail-false-cls*  $\Gamma$  *C*  $\longrightarrow$   $C = \{\#\} \vee (\exists A. ord-res.is-maximal-lit (Neg\ A)\ C)$   
**using** *step-hyps(3)* **by** *force*

**thus** *?thesis*  
**by** (*simp add:*  $\langle s' = (U_{er}, \mathcal{F}', \Gamma) \rangle$  *false-cls-is-empty-or-has-neg-max-lit-def*)  
**next**  
**case** *step-hyps*: (*resolution*  $\Gamma$   $\mathcal{F}$   $U_{er}$  *D* *A* *C*  $U_{er}'$   $\Gamma'$ )

**have** *false-wrt- $\Gamma$ -if-false-wrt- $\Gamma'$* : *trail-false-cls*  $\Gamma$  *E* **if** *trail-false-cls*  $\Gamma'$  *E* **for** *E*  
**using** *that*  
**unfolding** *step-hyps(7)* *trail-false-cls-def* *trail-false-lit-def*  
**using** *image-iff set-dropWhileD* **by** *fastforce*

**have** *iefac*  $\mathcal{F}$  *E*  $= \{\#\} \vee (\exists A. ord-res.is-maximal-lit (Neg\ A) (iefac\ \mathcal{F}\ E))$   
**if**  $E \in \mid N \mid \cup \mid U_{er}'$  *trail-false-cls*  $\Gamma'$  (*iefac*  $\mathcal{F}$  *E*) **for** *E*  
**proof** (*cases*  $E = eres\ C\ D$ )  
**case** *True*

**show** *?thesis*  
**proof** (*cases*  $eres\ C\ D = \{\#\}$ )  
**case** *True*  
**thus** *?thesis*

```

    by (simp add: ⟨E = eres C D⟩)
next
case False
then obtain K where K-max: ord-res.is-maximal-lit K (eres C D)
  using linorder-lit.ex-maximal-in-mset by metis

have Γ' = dropWhile (λx. atm-of K ≼t atm-of (fst x)) Γ
  unfolding step-hyps(7)
proof (rule dropWhile-cong)
  show Γ = Γ ..
next
fix Ln :: 'f gterm literal × 'f gterm literal multiset option
obtain L annot where Ln = (L, annot)
  by force
have (∀ K. ord-res.is-maximal-lit K (eres C D) ⟶ atm-of K ≼t atm-of L)
⟷
  (atm-of K ≼t atm-of L)
  using K-max by (metis linorder-lit.Uniq-is-maximal-in-mset the1-equality')
thus (∀ K. ord-res.is-maximal-lit K (eres C D) ⟶ atm-of K ≼t atm-of (fst
Ln)) ⟷
  (atm-of K ≼t atm-of (fst Ln))
  unfolding ⟨Ln = (L, annot)⟩ prod.case prod.sel .
qed

have K ∈# eres C D
  using K-max linorder-lit.is-maximal-in-mset-iff by metis

moreover have ¬ trail-defined-lit Γ' K
proof -
  have sorted-wrt (λx y. atm-of (fst y) <t atm-of (fst x)) Γ
    using invars[unfolded ⟨s = (Uer, F, Γ)⟩ trail-is-sorted-def]
    by (simp add: sorted-wrt-map)

  have ∀ Ln ∈ set Γ'. ¬ (atm-of K ≼t atm-of (fst Ln))
    unfolding ⟨Γ' = dropWhile (λx. atm-of K ≼t atm-of (fst x)) Γ⟩
  proof (rule ball-set-dropWhile-if-sorted-wrt-and-monotone-on)
    show sorted-wrt (λx y. atm-of (fst y) <t atm-of (fst x)) Γ
      using invars[unfolded ⟨s = (Uer, F, Γ)⟩ trail-is-sorted-def]
      by (simp add: sorted-wrt-map)
  next
    show monotone-on (set Γ) (λx y. atm-of (fst y) <t atm-of (fst x)) (≥)
      (λx. atm-of K ≼t atm-of (fst x))
      by (rule monotone-onI) auto
  qed

  hence ∀ Ln ∈ set Γ'. atm-of (fst Ln) <t atm-of K
    by auto

  hence atm-of K ∉ trail-atms Γ'

```



by (*smt (verit, best) fset-trail-atms image-iff linorder-trm.dual-order.asym*)

**thus** *?thesis*  
 using *trail-defined-lit-iff-trail-defined-atm* **by** *metis*  
**qed**

**ultimately have**  $\neg$  *trail-false-cls*  $\Gamma'$  (*eres C D*)  
 using *trail-defined-lit-iff-true-or-false trail-false-cls-def* **by** *metis*

**hence**  $\neg$  *trail-false-cls*  $\Gamma'$   $E$   
 unfolding  $\langle E = \text{eres } C D \rangle$  .

**hence**  $\neg$  *trail-false-cls*  $\Gamma'$  (*iefac*  $\mathcal{F}$   $E$ )  
 unfolding *trail-false-cls-def* **by** (*metis iefac-def set-mset-efac*)

**thus** *?thesis*  
 using  $\langle \text{trail-false-cls } \Gamma' (\text{iefac } \mathcal{F} E) \rangle$   
**by** *contradiction*  
**qed**

**next**  
**case** *False*  
**hence**  $E \mid \in \mid N \mid \cup \mid U_{er}$   
 using *step-hyps(6) that(1)* **by** *force*

**moreover hence** *iefac*  $\mathcal{F}$   $E \neq \{\#\}$   
 using *step-hyps(3-)*  
**by** (*metis (no-types, opaque-lifting) empty-iff linorder-cls.is-least-in-filter-iff linorder-cls.not-less linorder-lit.is-maximal-in-mset-iff mempty-lesseq-cls rev-fimage-eqI set-mset-empty trail-false-cls-mempty*)

**moreover have** *trail-false-cls*  $\Gamma$  (*iefac*  $\mathcal{F}$   $E$ )  
 using  $\langle \text{trail-false-cls } \Gamma' (\text{iefac } \mathcal{F} E) \rangle$  *false-wrt- $\Gamma$ -if-false-wrt- $\Gamma'$*  **by** *metis*

**ultimately have**  $\exists A.$  *ord-res.is-maximal-lit (Neg A) (iefac*  $\mathcal{F}$   $E$ )  
 using *invars(1)[unfolded  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  false-cls-is-mempty-or-has-neg-max-lit-def]*  
**by** *auto*

**thus** *?thesis ..*  
**qed**

**thus** *?thesis*  
**by** (*simp add:  $\langle s' = (U_{er}', \mathcal{F}, \Gamma') \rangle$  false-cls-is-mempty-or-has-neg-max-lit-def*)  
**qed**

**definition** *decided-literals-all-neg* **where**  
*decided-literals-all-neg*  $N$   $s \longleftrightarrow$   
 $(\forall U_{er} \mathcal{F} \Gamma. s = (U_{er}, \mathcal{F}, \Gamma) \longrightarrow$

$(\forall Ln \in set \Gamma. \forall L. Ln = (L, None) \longrightarrow is-neg L)$

**lemma** *ord-res-8-preserves-decided-literals-all-neg*:

**assumes**

*step*: *ord-res-8*  $N s s'$  **and**

*invar*: *decided-literals-all-neg*  $N s$

**shows** *decided-literals-all-neg*  $N s'$

**using** *step*

**proof** (*cases*  $N s s'$  *rule*: *ord-res-8.cases*)

**case** (*decide-neg*  $\mathcal{F} U_{er} \Gamma A \Gamma'$ )

**have**  $\forall Ln \in set \Gamma. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *decided-literals-all-neg-def*)

**hence**  $\forall Ln \in set \Gamma'. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**unfolding**  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  **by** *simp*

**thus** *?thesis*

**by** (*simp add*:  $\langle s' = (U_{er}, \mathcal{F}, \Gamma') \rangle$  *decided-literals-all-neg-def*)

**next**

**case** (*propagate*  $\mathcal{F} U_{er} \Gamma A C \Gamma'$ )

**have**  $\forall Ln \in set \Gamma. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *decided-literals-all-neg-def*)

**hence**  $\forall Ln \in set \Gamma'. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**unfolding**  $\langle \Gamma' = (Pos A, Some C) \# \Gamma \rangle$  **by** *simp*

**thus** *?thesis*

**by** (*simp add*:  $\langle s' = (U_{er}, \mathcal{F}, \Gamma') \rangle$  *decided-literals-all-neg-def*)

**next**

**case** (*factorize*  $\mathcal{F} U_{er} \Gamma A C \mathcal{F}'$ )

**have**  $\forall Ln \in set \Gamma. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *decided-literals-all-neg-def*)

**thus** *?thesis*

**by** (*simp add*:  $\langle s' = (U_{er}, \mathcal{F}', \Gamma) \rangle$  *decided-literals-all-neg-def*)

**next**

**case** *step-hyps*: (*resolution*  $\Gamma \mathcal{F} U_{er} D A C U_{er}' \Gamma'$ )

**have**  $\forall Ln \in set \Gamma. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**using** *invar* **by** (*simp add*:  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  *decided-literals-all-neg-def*)

**moreover have**  $set \Gamma' \subseteq set \Gamma$

**unfolding**  $\langle \Gamma' = dropWhile - \Gamma \rangle$

**by** (*meson set-mono-suffix suffix-dropWhile*)

**ultimately have**  $\forall Ln \in set \Gamma'. \forall L. Ln = (L, None) \longrightarrow is-neg L$

**by** *blast*

**thus** *?thesis*

**by** (*simp add:  $\langle s' = (U_{er'}, \mathcal{F}, \Gamma) \rangle$  decided-literals-all-neg-def*)

**qed**

**definition** *ord-res-8-invars* **where**

*ord-res-8-invars*  $N\ s \longleftrightarrow$   
*trail-is-sorted*  $N\ s \wedge$   
*trail-is-lower-set*  $N\ s \wedge$   
*false-cls-is-mempty-or-has-neg-max-lit*  $N\ s \wedge$   
*trail-annotations-invars*  $N\ s \wedge$   
*decided-literals-all-neg*  $N\ s$

**lemma** *ord-res-8-preserves-invars*:

**assumes**

*step: ord-res-8*  $N\ s\ s'$  **and**

*invars: ord-res-8-invars*  $N\ s$

**shows** *ord-res-8-invars*  $N\ s'$

**using** *invars*

**unfolding** *ord-res-8-invars-def*

**using**

*ord-res-8-preserves-trail-is-sorted*[*OF step*]

*ord-res-8-preserves-atoms-in-trail-lower-set*[*OF step*]

*ord-res-8-preserves-false-cls-is-mempty-or-has-neg-max-lit*[*OF step*]

*ord-res-8-preserves-trail-annotations-invars*[*OF step*]

*ord-res-8-preserves-decided-literals-all-neg*[*OF step*]

**by** *metis*

**lemma** *rtranclp-ord-res-8-preserves-invars*:

**assumes**

*step: (ord-res-8)*<sup>\*\*</sup>  $s\ s'$  **and**

*invars: ord-res-8-invars*  $N\ s$

**shows** *ord-res-8-invars*  $N\ s'$

**using** *step invars ord-res-8-preserves-invars*

**by** (*smt (verit, del-insts) rtranclp-induct*)

**lemma** *tranclp-ord-res-8-preserves-invars*:

**assumes**

*step: (ord-res-8)*<sup>++</sup>  $s\ s'$  **and**

*invars: ord-res-8-invars*  $N\ s$

**shows** *ord-res-8-invars*  $N\ s'$

**using** *step invars ord-res-8-preserves-invars*

**by** (*smt (verit, del-insts) tranclp-induct*)

**lemma** *ex-ord-res-8-if-not-final*:

**assumes**

*not-final:  $\neg$  ord-res-8-final* ( $N, s$ ) **and**

*invars*: *ord-res-8-invars*  $N$   $s$   
**shows**  $\exists s'. \text{ord-res-8 } N \ s \ s'$   
**proof** –  
**obtain**  $U_{er} \ \mathcal{F} \ \Gamma$  **where**  $s = (U_{er}, \mathcal{F}, \Gamma)$   
**by** (*metis surj-pair*)

**note**  $\text{invars}' = \text{invars}[\text{unfolded } \langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle \text{ord-res-8-invars-def}]$

**have**  
*undef-atm-or-false-cls*:  
 $(\exists x \ | \in \ | \text{atms-of-clss } (N \ | \cup \ | U_{er}). \ x \ | \notin \ | \text{trail-atms } \Gamma) \wedge$   
 $\neg (\exists x \ | \in \ | \text{iefac } \mathcal{F} \ | \uparrow \ (N \ | \cup \ | U_{er}). \ \text{trail-false-cls } \Gamma \ x) \vee$   
 $(\exists x \ | \in \ | \text{iefac } \mathcal{F} \ | \uparrow \ (N \ | \cup \ | U_{er}). \ \text{trail-false-cls } \Gamma \ x)$  **and**  
 $\{\#\} \ | \notin \ | \text{iefac } \mathcal{F} \ | \uparrow \ (N \ | \cup \ | U_{er})$   
**unfolding** *atomize-conj*  
**using** *not-final[unfolded ord-res-8-final.simps]*  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  **by** *metis*

**show** *?thesis*  
**using** *undef-atm-or-false-cls*  
**proof** (*elim disjE conjE*)  
**assume**  
*ex-undef-atm*:  $\exists x \ | \in \ | \text{atms-of-clss } (N \ | \cup \ | U_{er}). \ x \ | \notin \ | \text{trail-atms } \Gamma$  **and**  
*no-conflict*:  $\neg (\exists x \ | \in \ | \text{iefac } \mathcal{F} \ | \uparrow \ (N \ | \cup \ | U_{er}). \ \text{trail-false-cls } \Gamma \ x)$

**moreover have**  $\{|A_2 \ | \in \ | \text{atms-of-clss } (N \ | \cup \ | U_{er}). \ \forall A_1 \ | \in \ | \text{trail-atms } \Gamma. \ A_1$   
 $\prec_t A_2\} \neq \{\|\}$   
**proof** –  
**obtain**  $A_2 :: 'f \text{ gterm}$  **where**  
 $A_2\text{-in}: A_2 \ | \in \ | \text{atms-of-clss } (N \ | \cup \ | U_{er})$  **and**  
 $A_2\text{-undef}: A_2 \ | \notin \ | \text{trail-atms } \Gamma$   
**using** *ex-undef-atm* **by** *metis*

**have**  $A_1 \prec_t A_2$  **if**  $A_1\text{-in}: A_1 \ | \in \ | \text{trail-atms } \Gamma$  **for**  $A_1 :: 'f \text{ gterm}$   
**proof** –  
**have**  $A_1 \neq A_2$   
**using**  $A_1\text{-in}$   $A_2\text{-undef}$  **by** *metis*

**moreover have** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma)$   $(\text{atms-of-clss } (N$   
 $\ | \cup \ | U_{er}))$   
**using** *invars'[unfolded trail-is-lower-set-def, simplified]* **by** *argo*

**ultimately show** *?thesis*  
**by** (*meson*  $A_2\text{-in}$   $A_2\text{-undef}$  *linorder-trm.is-lower-set-iff linorder-trm.linorder-cases*  
*that*)  
**qed**

**thus** *?thesis*  
**using**  $A_2\text{-in}$   
**by** (*smt* (*verit*, *ccfv-threshold*) *femptyE fmember-filter*)

qed

**ultimately obtain**  $A :: 'f\ gterm$  **where**

$A$ -least:  $linorder-trm.is-least-in-fset \{|A_2 | \in | atms-of-clss (N \cup | U_{er}).$

$\forall A_1 | \in | trail-atms \Gamma. A_1 \prec_t A_2 | \}$   $A$

**using**  $ex-undef-atm\ linorder-trm.ex-least-in-fset$  **by**  $presburger$

**show**  $\exists s'. ord-res-8\ N\ s\ s'$

**proof** ( $cases\ \exists C | \in | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er}).\ clause-could-propagate\ \Gamma\ C\ (Pos\ A)$ )

**case**  $True$

**hence**  $\{|C | \in | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er}).\ clause-could-propagate\ \Gamma\ C\ (Pos\ A)|\} \neq \{\}\}$

**by**  $force$

**then obtain**  $C$  **where**

$C$ -least:  $linorder-cls.is-least-in-fset$

$\{|C | \in | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er}).\ clause-could-propagate\ \Gamma\ C\ (Pos\ A)|\}$   $C$

**using**  $linorder-cls.ex1-least-in-fset$  **by**  $meson$

**show**  $?thesis$

**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**using**  $ord-res-8.propagate[OF\ no-conflict\ A-least\ C-least]$

**using**  $ord-res-8.factorize[OF\ no-conflict\ A-least\ C-least]$

**by**  $metis$

**next**

**case**  $False$

**thus**  $?thesis$

**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**using**  $ord-res-8.decide-neg[OF\ no-conflict\ A-least]$  **by**  $metis$

qed

**next**

**assume**  $\exists x | \in | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er}).\ trail-false-clcs\ \Gamma\ x$

**then obtain**  $D :: 'f\ gclause$  **where**

$D$ -least:  $linorder-clcs.is-least-in-fset$

$(ffilter\ (trail-false-clcs\ \Gamma)\ (iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er})))\ D$

**by**  $(metis\ femptyE\ ffilter-filter\ linorder-clcs.ex-least-in-fset)$

**hence**  $D | \in | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er})$  **and**  $trail-false-clcs\ \Gamma\ D$

**unfolding**  $atomize-conj\ linorder-clcs.is-least-in-filter-iff$  **by**  $argo$

**moreover have**  $D \neq \{\#\}$

**using**  $\langle D | \in | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er}) \rangle \langle \{\#\} | \notin | iefac\ \mathcal{F}\ | \uparrow (N \cup | U_{er}) \rangle$  **by**  $metis$

**ultimately obtain**  $A$  **where**  $Neg-A-max$ :  $linorder-lit.is-maximal-in-mset\ D$   $(Neg\ A)$

**using**  $invars'\ false-clcs-is-memty-or-has-neg-max-lit-def$  **by**  $metis$

```

hence trail-false-lit  $\Gamma$  (Neg A)
  using  $\langle \text{trail-false-cls } \Gamma \ D \rangle$ 
  by (metis linorder-lit.is-maximal-in-mset-iff trail-false-cls-def)

hence  $\text{Pos } A \in \text{fst } \text{' set } \Gamma$ 
  unfolding trail-false-lit-def by simp

hence  $\exists C. (\text{Pos } A, \text{Some } C) \in \text{set } \Gamma$ 
  using invars'[unfolded decided-literals-all-neg-def, simplified]
  by fastforce

then obtain  $C :: \text{'gclause where}$ 
   $\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C)$ 
by (metis invars' is-pos-def map-of-SomeD not-Some-eq decided-literals-all-neg-def
  weak-map-of-SomeI)

thus  $\exists s'. \text{ord-res-8 } N \ s \ s'$ 
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
  using ord-res-8.resolution D-least Neg-A-max by blast
qed
qed

lemma ord-res-8-safe-state-if-invars:
  fixes  $N \ s$ 
  assumes invars: ord-res-8-invars N s
  shows safe-state (constant-context ord-res-8) ord-res-8-final (N, s)
  using safe-state-constant-context-if-invars where
     $\mathcal{R} = \text{ord-res-8}$  and  $\mathcal{F} = \text{ord-res-8-final}$  and  $\mathcal{I} = \text{ord-res-8-invars}$ 
  using ord-res-8-preserves-invars ex-ord-res-8-if-not-final invars by metis

end

end
theory ORD-RES-9
  imports
    ORD-RES-8
begin

```

## 24 ORD-RES-9 (factorize when propagating)

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-9* **where**

*decide-neg*:

$$\neg (\exists C \mid \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \ C) \implies$$

$$\text{linorder-trm.is-least-in-fset } \{ \mid A_2 \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}).$$

$$\forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2 \mid \} A \implies$$

$$\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{clause-could-propagate } \Gamma \ C (\text{Pos } A)) \implies$$

$$\Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \implies$$

*ord-res-9*  $N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}, \Gamma') \mid$

*propagate*:

$\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma C) \implies$   
 $\text{linorder-trm.is-least-in-fset } \{|A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}).$   
 $\forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A \implies$   
 $\text{linorder-cls.is-least-in-fset } \{|C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$   
 $\text{clause-could-propagate } \Gamma C (\text{Pos } A)\} C \implies$   
 $\Gamma' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma \implies$   
 $\mathcal{F}' = (\text{if linorder-lit.is-greatest-in-mset } C (\text{Pos } A) \text{ then } \mathcal{F} \text{ else finsert } C \mathcal{F}) \implies$   
 $\text{ord-res-9 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}', \Gamma') \mid$

*resolution*:

$\text{linorder-cls.is-least-in-fset } \{|D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma D\}$   
 $D \implies$   
 $\text{linorder-lit.is-maximal-in-mset } D (\text{Neg } A) \implies$   
 $\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C) \implies$   
 $U_{er}' = \text{finsert } (\text{eres } C D) U_{er} \implies$   
 $\Gamma' = \text{dropWhile } (\lambda Ln. \forall K.$   
 $\text{linorder-lit.is-maximal-in-mset } (\text{eres } C D) K \longrightarrow \text{atm-of } K \preceq_t \text{atm-of } (\text{fst}$   
 $\text{Ln})) \Gamma \implies$   
 $\text{ord-res-9 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}', \mathcal{F}, \Gamma')$

**lemma** *right-unique-ord-res-9*:

**fixes**  $N :: 'f \text{ gclause fset}$

**shows** *right-unique* (*ord-res-9*  $N$ )

**proof** (*rule right-uniqueI*)

**fix**  $x y z$

**assume** *step1*: *ord-res-9*  $N x y$  **and** *step2*: *ord-res-9*  $N x z$

**show**  $y = z$

**using** *step1*

**proof** (*cases*  $N x y$  *rule*: *ord-res-9.cases*)

**case** *hyps1*: (*decide-neg*  $\mathcal{F} U_{er} \Gamma A_1 \Gamma_1'$ )

**show** *?thesis*

**using** *step2* **unfolding**  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**proof** (*cases*  $N (U_{er}, \mathcal{F}, \Gamma) z$  *rule*: *ord-res-9.cases*)

**case** (*decide-neg*  $A \Gamma'$ )

**with** *hyps1* **show** *?thesis*

**by** (*metis* (*no-types, lifting*) *linorder-trm.dual-order.asym*  
*linorder-trm.is-least-in-fset-iff*)

**next**

**case** (*propagate*  $A C \Gamma' \mathcal{F}'$ )

**with** *hyps1* **have** *False*

**by** (*metis* (*no-types, lifting*) *linorder-cls.is-least-in-filter-iff*  
*linorder-trm.dual-order.asym linorder-trm.is-least-in-filter-iff*)

**thus** *?thesis ..*

**next**

**case** (*resolution*  $D A C U_{er}' \Gamma'$ )

**with** *hyps1* **have** *False*

```

    by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
qed
next
case hyps1: (propagate  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A_1$   $C_1$   $\Gamma_1'$   $\mathcal{F}_1'$ )
show ?thesis
  using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
proof (cases  $N (U_{er}, \mathcal{F}, \Gamma)$  z rule: ord-res-9.cases)
  case (decide-neg  $A \Gamma'$ )
  with hyps1 have False
  by (metis (no-types, lifting) Uniq-D linorder-cls.is-least-in-ffilter-iff
      linorder-trm.Uniq-is-least-in-fset)
  thus ?thesis ..
next
case (propagate  $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )
with hyps1 show ?thesis
  by (metis (no-types, lifting) linorder-cls.dual-order.asym
      linorder-cls.is-least-in-ffilter-iff linorder-trm.dual-order.asym
      linorder-trm.is-least-in-fset-iff)
next
case (resolution  $D$   $A$   $C$   $U_{er}' \Gamma'$ )
with hyps1 have False
  by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
qed
next
case hyps1: (resolution  $\Gamma$   $\mathcal{F}$   $U_{er}$   $D_1$   $A_1$   $C_1$   $U_{er1}' \Gamma_1'$ )
show ?thesis
  using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
proof (cases  $N (U_{er}, \mathcal{F}, \Gamma)$  z rule: ord-res-9.cases)
  case (decide-neg  $A \Gamma'$ )
  with hyps1 have False
  by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
next
case (propagate  $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )
with hyps1 have False
  by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
next
case hyps2: (resolution  $D$   $A$   $C$   $U_{er}' \Gamma'$ )
have  $D_1 = D$ 
  using hyps1 hyps2
  by (metis (no-types) Uniq-D linorder-cls.Uniq-is-least-in-fset)
have  $C_1 = C$ 
  using hyps1 hyps2
  unfolding  $\langle D_1 = D \rangle$ 
  by (metis (no-types) Uniq-D linorder-lit.Uniq-is-maximal-in-mset option.inject
      uminus-Neg)

```



```

    show ?thesis
      using hyps1 hyps2
      unfolding ⟨D1 = D⟩ ⟨C1 = C⟩
      by argo
  qed
qed
qed

lemma ord-res-9-is-one-or-two-ord-res-9-steps:
  fixes N s s'
  assumes step: ord-res-9 N s s'
  shows ord-res-8 N s s' ∨ (ord-res-8 N OO ord-res-8 N) s s'
  using step
proof (cases N s s' rule: ord-res-9.cases)
  case step-hyps: (decide-neg  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $\Gamma'$ )
  show ?thesis
  proof (rule disjI1)
    show ord-res-8 N s s'
      using step-hyps ord-res-8.decide-neg by (simp only:)
  qed
next
  case step-hyps: (propagate  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )

  have
    C-in:  $C \in | \in | \text{iefac } \mathcal{F} \ | \uparrow \ (N \ | \cup \ | U_{er})$  and
    C-could-prop: clause-could-propagate  $\Gamma$   $C$  (Pos  $A$ ) and
    C-lt:  $\forall D \ | \in | \text{iefac } \mathcal{F} \ | \uparrow \ (N \ | \cup \ | U_{er}).$ 
       $D \neq C \longrightarrow \text{clause-could-propagate } \Gamma \ D \ (\text{Pos } A) \longrightarrow C \prec_c D$ 
    using step-hyps unfolding atomize-conj linorder-cls.is-least-in-filter-iff by
  argo

  hence C-max-lit: ord-res.is-maximal-lit (Pos  $A$ )  $C$ 
    unfolding clause-could-propagate-def by argo

  show ?thesis
  proof (cases ord-res.is-strictly-maximal-lit (Pos  $A$ )  $C$ )
    case True
      hence efac  $C = C$ 
        using nex-strictly-maximal-pos-lit-if-neq-efac by force
      thus ?thesis
        using True step-hyps ord-res-8.propagate by simp
    next
      case False
        have  $\mathcal{F}' = \text{finsert } C \ \mathcal{F}$ 
          using False step-hyps by simp
  qed

```

```

have efac  $C \neq C$ 
  using False C-max-lit by (metis greatest-literal-in-efacI literal.disc(1))

hence C-in':  $C \in N \cup U_{er}$ 
  using C-in
  by (smt (verit, ccfv-threshold) fimage-iff iefac-def ex1-efac-eq-factoring-chain
    factorizable-if-neq-efac)

have  $C \notin \mathcal{F}$ 
  by (smt (verit, ccfv-threshold) C-in <efac C ≠ C> factorizable-if-neq-efac
    fimage-iff ex1-efac-eq-factoring-chain iefac-def)

have fimage-iefac- $\mathcal{F}'$ -eq:
  iefac  $\mathcal{F}' \mid \uparrow (N \cup U_{er}) = finsert (efac C) ((iefac \mathcal{F} \mid \uparrow (N \cup U_{er})) - \{C\})$ 
proof (intro fsubset-antisym fsubsetI)
  fix  $x :: 'f$  gclause
  assume  $x \in iefac \mathcal{F}' \mid \uparrow (N \cup U_{er})$ 
  then obtain  $x'$  where  $x' \in N \cup U_{er}$  and  $x = iefac \mathcal{F}' x'$ 
    by blast
  thus  $x \in finsert (efac C) ((iefac \mathcal{F} \mid \uparrow (N \cup U_{er})) \mid - \{C\})$ 
  proof (cases  $x' = C$ )
    case True
      hence  $x = efac C$ 
      using  $\langle x = iefac \mathcal{F}' x' \rangle$  by (simp add: < $\mathcal{F}' = finsert C \mathcal{F}$ > iefac-def)
      thus ?thesis
      using  $\langle efac C \neq C \rangle$  by simp
    next
      case False
        hence  $x = iefac \mathcal{F}' x'$ 
        using  $\langle x = iefac \mathcal{F}' x' \rangle$  by (auto simp add: < $\mathcal{F}' = finsert C \mathcal{F}$ > iefac-def)
        then show ?thesis
        by (metis (no-types, lifting) False < $x' \in N \cup U_{er}$ > ex1-efac-eq-factoring-chain
          factorizable-if-neq-efac fimage-eqI finsertCI fminus-iff fsingletonE
iefac-def)
        qed
    next
      fix  $x :: 'f$  gclause
      assume x-in:  $x \in finsert (efac C) (iefac \mathcal{F} \mid \uparrow (N \cup U_{er})) \mid - \{C\}$ 
      hence  $x = efac C \vee x \in (iefac \mathcal{F} \mid \uparrow (N \cup U_{er})) \mid - \{C\}$ 
      by blast
      thus  $x \in iefac \mathcal{F}' \mid \uparrow (N \cup U_{er})$ 
      proof (elim disjE)
        assume  $x = efac C$ 
        hence  $x = iefac \mathcal{F}' C$ 
        by (simp add: < $\mathcal{F}' = finsert C \mathcal{F}$ > iefac-def)
        thus  $x \in iefac \mathcal{F}' \mid \uparrow (N \cup U_{er})$ 
        using  $\langle C \in N \cup U_{er} \rangle$  by blast
      next

```

**assume**  $x \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \mid - \mid \{C\}$   
**hence**  $x \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  $x \neq C$   
**by** *simp-all*  
**then obtain**  $x'$  **where**  $x' \in N \mid \cup \mid U_{er}$  **and**  $x = \text{iefac } \mathcal{F} x'$   
**by** *auto*  
**moreover have**  $x' \neq C$   
**using**  $\langle x \neq C \rangle \langle x = \text{iefac } \mathcal{F} x' \rangle$   
**using**  $\langle C \notin \mathcal{F} \rangle$  *iefac-def* **by** *force*  
**ultimately show**  $x \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$   
**by** (*simp add:  $\langle \mathcal{F}' = \text{finsert } C \mathcal{F} \rangle$*  *iefac-def*)  
**qed**  
**qed**

**have** *first-step8: ord-res-8*  $N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}', \Gamma)$   
**using** *False step-hyps ord-res-8.factorize* **by** *simp*

**moreover have** *ord-res-8*  $N (U_{er}, \mathcal{F}', \Gamma) (U_{er}, \mathcal{F}', \Gamma')$   
**proof** (*rule ord-res-8.propagate*)  
**have**  $\neg \text{trail-false-cls } \Gamma C$   
**using** *step-hyps* **using** *C-in* **by** *metis*

**hence**  $\neg \text{trail-false-cls } \Gamma (\text{efac } C)$   
**by** (*simp add: trail-false-cls-def*)

**thus**  $\neg (\exists C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma C)$   
**using** *step-hyps* **unfolding** *fimage-iefac- $\mathcal{F}'$ -eq* **by** *blast*

**next**  
**show** *linorder-trm.is-least-in-fset*  $\{A_2 \in \text{atms-of-clss } (N \mid \cup \mid U_{er}). \forall A_1 \in \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A$   
**using** *step-hyps* **by** *argo*

**next**  
**show** *linorder-cls.is-least-in-fset*  $\{C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). \text{clause-could-propagate } \Gamma C (\text{Pos } A)\} (\text{efac } C)$   
**unfolding** *linorder-cls.is-least-in-ffilter-iff*  
**proof** (*intro conjI ballI impI*)  
**show**  $\text{efac } C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$   
**unfolding** *fimage-iefac- $\mathcal{F}'$ -eq* **by** *simp*

**next**  
**show** *clause-could-propagate*  $\Gamma (\text{efac } C) (\text{Pos } A)$   
**using** *C-could-prop*  
**unfolding** *clause-could-propagate-def*  
**by** (*simp add: trail-false-cls-def greatest-literal-in-efacI linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*)

**next**  
**fix**  $D :: \text{'f gterm literal multiset}$   
**assume**  
 $D \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $D \neq \text{efac } C$  **and**  
*clause-could-propagate*  $\Gamma D (\text{Pos } A)$

```

thus efac C  $\prec_c$  D
  using C-lt
  by (metis (no-types, opaque-lifting) C-in efac-properties-if-not-ident(1)
    fimage-iefac-F'-eq finsert-fminus finsert-iff linorder-cls.less-trans)
qed
next
show ord-res.is-strictly-maximal-lit (Pos A) (efac C)
  using step-hyps
  by (simp add: clause-could-propagate-def greatest-literal-in-efacI
    linorder-cls.is-least-in-filter-iff)
next
show  $\Gamma' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma$ 
  using step-hyps by argo
qed

ultimately have (ord-res-8 N OO ord-res-8 N) s s'
  using step-hyps by blast

thus ?thesis
  by argo
qed
next
case step-hyps: (resolution  $\Gamma \mathcal{F} U_{er} D A C U_{er}' \Gamma')$ 
show ?thesis
proof (rule disjI1)
  show ord-res-8 N s s'
  using step-hyps ord-res-8.resolution by (simp only:)
qed
qed

lemma ord-res-9-preserves-invars:
assumes
  step: ord-res-9 N s s' and
  invars: ord-res-8-invars N s
shows ord-res-8-invars N s'
using invars ord-res-9-is-one-or-two-ord-res-9-steps
by (metis local.step ord-res-8-preserves-invars relcomppE)

lemma rtranclp-ord-res-9-preserves-ord-res-8-invars:
assumes
  step: (ord-res-9 N)** s s' and
  invars: ord-res-8-invars N s
shows ord-res-8-invars N s'
using step invars ord-res-9-preserves-invars
by (smt (verit, del-insts) rtranclp-induct)

lemma ex-ord-res-9-if-not-final:
assumes
  not-final:  $\neg$  ord-res-8-final (N, s) and

```

*invars*: *ord-res-8-invars*  $N$   $s$   
**shows**  $\exists s'. \text{ord-res-9 } N \ s \ s'$   
**proof** –  
**obtain**  $U_{er} \ \mathcal{F} \ \Gamma$  **where**  $s = (U_{er}, \mathcal{F}, \Gamma)$   
**by** (*metis surj-pair*)

**note**  $\text{invars}' = \text{invars}[\text{unfolded } \langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle \text{ord-res-8-invars-def}]$

**have**  
*undef-atm-or-false-cls*:  
 $(\exists x \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). x \mid \notin \mid \text{trail-atms } \Gamma) \wedge$   
 $\neg (\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \ x) \vee$   
 $(\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \ x)$  **and**  
 $\{\#\} \mid \notin \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**unfolding** *atomize-conj*  
**using** *not-final[unfolded ord-res-8-final.simps]*  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  **by** *metis*

**show** *?thesis*  
**using** *undef-atm-or-false-cls*  
**proof** (*elim disjE conjE*)  
**assume**  
*ex-undef-atm*:  $\exists x \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). x \mid \notin \mid \text{trail-atms } \Gamma$  **and**  
*no-conflict*:  $\neg (\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \ x)$

**moreover have**  $\{ \mid A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1$   
 $\prec_t A_2 \mid \} \neq \{ \mid \}$   
**proof** –  
**obtain**  $A_2 :: 'f \text{ gterm}$  **where**  
 $A_2\text{-in}: A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$  **and**  
 $A_2\text{-undef}: A_2 \mid \notin \mid \text{trail-atms } \Gamma$   
**using** *ex-undef-atm* **by** *metis*

**have**  $A_1 \prec_t A_2$  **if**  $A_1\text{-in}: A_1 \mid \in \mid \text{trail-atms } \Gamma$  **for**  $A_1 :: 'f \text{ gterm}$   
**proof** –  
**have**  $A_1 \neq A_2$   
**using**  $A_1\text{-in } A_2\text{-undef}$  **by** *metis*

**moreover have** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma)$   $(\text{atms-of-clss } (N$   
 $\mid \cup \mid U_{er}))$   
**using** *invars'[unfolded trail-is-lower-set-def, simplified]* **by** *argo*

**ultimately show** *?thesis*  
**by** (*meson*  $A_2\text{-in } A_2\text{-undef } \text{linorder-trm.is-lower-set-iff } \text{linorder-trm.linorder-cases}$   
*that*)  
**qed**

**thus** *?thesis*  
**using**  $A_2\text{-in}$   
**by** (*smt* (*verit*, *ccfv-threshold*) *femptyE fmember-filter*)

qed

**ultimately obtain**  $A :: 'f\ gterm$  **where**

$A$ -least:  $linorder-trm.is-least-in-fset \{|A_2 | \in | atms-of-cls (N \cup | U_{er}).$

$\forall A_1 | \in | trail-atms \Gamma. A_1 \prec_t A_2 | \}$   $A$

**using**  $ex-undef-atm\ linorder-trm.ex-least-in-fset$  **by**  $presburger$

**show**  $\exists s'. ord-res-9\ N\ s\ s'$

**proof** ( $cases \exists C | \in | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er}). clause-could-propagate\ \Gamma\ C\ (Pos\ A)$ )

**case**  $True$

**hence**  $\{|C | \in | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er}). clause-could-propagate\ \Gamma\ C\ (Pos\ A)|\}$   
 $\neq \{| |\}$

**by**  $force$

**then obtain**  $C$  **where**

$C$ -least:  $linorder-cls.is-least-in-fset$

$\{|C | \in | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er}). clause-could-propagate\ \Gamma\ C\ (Pos\ A)|\}$   $C$

**using**  $linorder-cls.ex1-least-in-fset$  **by**  $meson$

**show**  $?thesis$

**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**using**  $ord-res-9.propagate[OF\ no-conflict\ A-least\ C-least]$

**by**  $metis$

**next**

**case**  $False$

**thus**  $?thesis$

**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**using**  $ord-res-9.decide-neg[OF\ no-conflict\ A-least]$  **by**  $metis$

qed

**next**

**assume**  $\exists x | \in | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er}). trail-false-cls\ \Gamma\ x$

**then obtain**  $D :: 'f\ gclause$  **where**

$D$ -least:  $linorder-cls.is-least-in-fset$

$(ffilter\ (trail-false-cls\ \Gamma)\ (iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er})))\ D$

**by**  $(metis\ femptyE\ ffilter-filter\ linorder-cls.ex-least-in-fset)$

**hence**  $D | \in | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er})$  **and**  $trail-false-cls\ \Gamma\ D$

**unfolding**  $atomize-conj\ linorder-cls.is-least-in-filter-iff$  **by**  $argo$

**moreover have**  $D \neq \{\#\}$

**using**  $\langle D | \in | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er}) \rangle \langle \{\#\} \notin | iefac\ \mathcal{F} \ | \uparrow (N \cup | U_{er}) \rangle$  **by**  
 $metis$

**ultimately obtain**  $A$  **where**  $Neg-A-max$ :  $linorder-lit.is-maximal-in-mset\ D$   
 $(Neg\ A)$

**using**  $invars'\ false-cls-is-mempty-or-has-neg-max-lit-def$  **by**  $metis$

**hence**  $trail-false-lit\ \Gamma\ (Neg\ A)$

```

using ⟨trail-false-cls  $\Gamma$   $D$ ⟩
by (metis linorder-lit.is-maximal-in-mset-iff trail-false-cls-def)

hence  $Pos\ A \in fst\ 'set\ \Gamma$ 
unfolding trail-false-lit-def by simp

hence  $\exists C. (Pos\ A, Some\ C) \in set\ \Gamma$ 
using invars'[unfolded decided-literals-all-neg-def, simplified]
by fastforce

then obtain  $C :: 'f\ gclause\ where$ 
   $map-of\ \Gamma\ (Pos\ A) = Some\ (Some\ C)$ 
by (metis invars' is-pos-def map-of-SomeD not-Some-eq decided-literals-all-neg-def
  weak-map-of-SomeI)

thus  $\exists s'. ord-res-9\ N\ s\ s'$ 
unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
using ord-res-9.resolution D-least Neg-A-max by blast
qed
qed

lemma ord-res-9-safe-state-if-invars:
fixes  $N\ s$ 
assumes invars: ord-res-8-invars  $N\ s$ 
shows safe-state (constant-context ord-res-9) ord-res-8-final  $(N, s)$ 
using safe-state-constant-context-if-invars [where
   $\mathcal{R} = ord-res-9$  and  $\mathcal{F} = ord-res-8-final$  and  $\mathcal{I} = ord-res-8-invars$ ]
using ord-res-9-preserves-invars ex-ord-res-9-if-not-final invars by metis

sublocale ord-res-9-semantic: semantics where
  step = constant-context ord-res-9 and
  final = ord-res-8-final
proof unfold-locales
fix  $S :: 'f\ gclause\ fset \times 'f\ gclause\ fset \times 'f\ gclause\ fset \times$ 
   $('f\ gliteral \times 'f\ gclause\ option)\ list$ 

obtain
   $N\ U_{er}\ \mathcal{F} :: 'f\ gterm\ clause\ fset$  and
   $\Gamma :: ('f\ gterm\ literal \times 'f\ gterm\ literal\ multiset\ option)\ list$  where
   $S-def: S = (N, U_{er}, \mathcal{F}, \Gamma)$ 
by (metis prod.exhaust)

assume ord-res-8-final  $S$ 

hence  $\nexists x. ord-res-9\ N\ (U_{er}, \mathcal{F}, \Gamma)\ x$ 
unfolding  $S-def$ 
proof (cases  $(N, U_{er}, \mathcal{F}, \Gamma)$  rule: ord-res-8-final.cases)
  case model-found

```

```

have  $\nexists A$ . linorder-trm.is-least-in-fset  $\{|A_2| \in| \textit{atms-of-clss} (N \cup| U_{er})$ .
 $\forall A_1 | \in| \textit{trail-atms} \Gamma. A_1 \prec_t A_2\}$   $A$ 
using  $\langle \neg (\exists A | \in| \textit{atms-of-clss} (N \cup| U_{er}). A | \notin| \textit{trail-atms} \Gamma) \rangle$ 
using linorder-trm.is-least-in-ffilter-iff by fastforce

moreover have  $\nexists C$ . linorder-cls.is-least-in-fset
(ffilter (trail-false-cls  $\Gamma$ ) (iefac  $\mathcal{F}$   $| \uparrow (N \cup| U_{er})$ ))  $C$ 
using  $\langle \neg \textit{fBex} (\textit{iefac} \mathcal{F} | \uparrow (N \cup| U_{er})) (\textit{trail-false-cls} \Gamma) \rangle$ 
by (metis linorder-cls.is-least-in-ffilter-iff)

ultimately show ?thesis
unfolding ord-res-9.simps by blast
next
case contradiction-found

hence  $\exists C | \in| \textit{iefac} \mathcal{F} | \uparrow (N \cup| U_{er}). \textit{trail-false-cls} \Gamma C$ 
using trail-false-cls-empty by metis

moreover have  $\nexists D$   $A$ . linorder-cls.is-least-in-fset (ffilter (trail-false-cls  $\Gamma$ )
(iefac  $\mathcal{F}$   $| \uparrow (N \cup| U_{er})$ ))  $D \wedge \textit{ord-res.is-maximal-lit} (\textit{Neg} A) D$ 
by (metis empty-iff linorder-cls.is-least-in-ffilter-iff linorder-cls.leD
linorder-lit.is-maximal-in-mset-iff local.contradiction-found(1) mempty-lesseq-cls
set-mset-empty trail-false-cls-empty)

ultimately show ?thesis
unfolding ord-res-9.simps by blast
qed

thus finished (constant-context ord-res-9)  $S$ 
by (simp add: S-def finished-def constant-context.simps)
qed

end

end
theory ORD-RES-10
imports ORD-RES-8
begin

```

## 25 ORD-RES-10 (propagate iff a conflict is produced)

```

type-synonym 'f ord-res-10-state =
'f gclause fset  $\times$  'f gclause fset  $\times$  'f gclause fset  $\times$  ('f gliteral  $\times$  'f gclause option)
list

```

```

context simulation-SCLFOL-ground-ordered-resolution begin

```



**inductive ord-res-10 where**

*decide-neg:*

$$\begin{aligned} & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma C) \implies \\ & \text{linorder-trm.is-least-in-fset } \{|A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \\ & \quad \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A \implies \\ & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ clause-could-propagate } \Gamma C (Pos A)) \implies \\ & \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \implies \\ & \text{ord-res-10 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}, \Gamma') \mid \end{aligned}$$

*decide-pos:*

$$\begin{aligned} & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma C) \implies \\ & \text{linorder-trm.is-least-in-fset } \{|A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \\ & \quad \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A \implies \\ & \text{linorder-cls.is-least-in-fset } \{|C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \\ & \quad \text{clause-could-propagate } \Gamma C (Pos A)\} C \implies \\ & \Gamma' = (\text{Pos } A, \text{None}) \# \Gamma \implies \\ & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma' C) \implies \\ & \mathcal{F}' = (\text{if linorder-lit.is-greatest-in-mset } C (Pos A) \text{ then } \mathcal{F} \text{ else finsert } C \mathcal{F}) \implies \\ & \text{ord-res-10 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}', \Gamma') \mid \end{aligned}$$

*propagate:*

$$\begin{aligned} & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma C) \implies \\ & \text{linorder-trm.is-least-in-fset } \{|A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \\ & \quad \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A \implies \\ & \text{linorder-cls.is-least-in-fset } \{|C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \\ & \quad \text{clause-could-propagate } \Gamma C (Pos A)\} C \implies \\ & \Gamma' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma \implies \\ & (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma' C) \implies \\ & \mathcal{F}' = (\text{if linorder-lit.is-greatest-in-mset } C (Pos A) \text{ then } \mathcal{F} \text{ else finsert } C \mathcal{F}) \implies \\ & \text{ord-res-10 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}, \mathcal{F}', \Gamma') \mid \end{aligned}$$

*resolution:*

$$\begin{aligned} & \text{linorder-cls.is-least-in-fset } \{|D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma D\} \\ & D \implies \\ & \text{linorder-lit.is-maximal-in-mset } D (\text{Neg } A) \implies \\ & \text{map-of } \Gamma (Pos A) = \text{Some } (\text{Some } C) \implies \\ & U_{er}' = \text{finsert } (\text{eres } C D) U_{er} \implies \\ & \Gamma' = \text{dropWhile } (\lambda Ln. \forall K. \\ & \quad \text{linorder-lit.is-maximal-in-mset } (\text{eres } C D) K \longrightarrow \text{atm-of } K \preceq_t \text{atm-of } (\text{fst} \\ & \quad Ln)) \Gamma \implies \\ & \text{ord-res-10 } N (U_{er}, \mathcal{F}, \Gamma) (U_{er}', \mathcal{F}, \Gamma') \end{aligned}$$

**lemma right-unique-ord-res-10:**

**fixes**  $N :: 'f \text{ gclause fset}$

**shows** *right-unique (ord-res-10 N)*

**proof** (*rule right-uniqueI*)

**fix**  $x \ y \ z$

**assume** *step1: ord-res-10 N x y and step2: ord-res-10 N x z*

**show**  $y = z$

```

using step1
proof (cases N x y rule: ord-res-10.cases)
  case hyps1: (decide-neg F Uer Γ A1 Γ1')
  show ?thesis
    using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
  proof (cases N (Uer, F, Γ) z rule: ord-res-10.cases)
    case (decide-neg A Γ')
    with hyps1 show ?thesis
      by (metis (no-types, lifting) linorder-trm.dual-order.asym
        linorder-trm.is-least-in-ffilter-iff)
  next
    case (decide-pos A C Γ' F')
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (propagate A C Γ' F')
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (resolution D A C Uer' Γ')
    with hyps1 have False
      by (metis (no-types, opaque-lifting) linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
  qed
next
  case hyps1: (decide-pos F Uer Γ A1 C1 Γ1' F1')
  show ?thesis
    using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
  proof (cases N (Uer, F, Γ) z rule: ord-res-10.cases)
    case (decide-neg A Γ')
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (decide-pos A C Γ' F')
    with hyps1 show ?thesis
      by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
        linorder-trm.Uniq-is-least-in-fset the1-equality')
  next
    case hyps2: (propagate A C Γ' F')
    have A1 = A
    using hyps1 hyps2
    by (metis (no-types, lifting) linorder-trm.dual-order.asym
      linorder-trm.is-least-in-ffilter-iff)

```

```

hence trail-false-cls  $\Gamma 1' = \text{trail-false-cls } \Gamma'$ 
  using hyps1 hyps2
  unfolding trail-false-cls-def trail-false-lit-def
  by simp
hence False
  using hyps1 hyps2 by argo
thus ?thesis ..
next
  case (resolution D A C Uer'  $\Gamma'$ )
  with hyps1 have False
    by (meson linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
qed
next
case hyps1: (propagate  $\mathcal{F}$  Uer  $\Gamma$  A1 C1  $\Gamma 1'$   $\mathcal{F} 1'$ )
show ?thesis
  using step2 unfolding  $\langle x = (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
proof (cases N (Uer,  $\mathcal{F}$ ,  $\Gamma$ ) z rule: ord-res-10.cases)
  case (decide-neg A  $\Gamma'$ )
  with hyps1 have False
    by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
      linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
  thus ?thesis ..
next
case hyps2: (decide-pos A C  $\Gamma'$   $\mathcal{F}'$ )
have A1 = A
  using hyps1 hyps2
  by (metis (no-types, lifting) linorder-trm.dual-order.asym
    linorder-trm.is-least-in-ffilter-iff)
hence trail-false-cls  $\Gamma 1' = \text{trail-false-cls } \Gamma'$ 
  using hyps1 hyps2
  unfolding trail-false-cls-def trail-false-lit-def
  by simp
hence False
  using hyps1 hyps2 by argo
thus ?thesis ..
next
case (propagate A C  $\Gamma'$   $\mathcal{F}'$ )
with hyps1 show ?thesis
  by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
    linorder-trm.Uniq-is-least-in-fset the1-equality)
next
case (resolution D A C Uer'  $\Gamma'$ )
with hyps1 have False
  by (meson linorder-cls.is-least-in-ffilter-iff)
thus ?thesis ..
qed
next
case hyps1: (resolution  $\Gamma$   $\mathcal{F}$  Uer D1 A1 C1 Uer1'  $\Gamma 1'$ )

```

```

show ?thesis
  using step2 unfolding ⟨x = (Uer, F, Γ)⟩
proof (cases N (Uer, F, Γ) z rule: ord-res-10.cases)
  case (decide-neg A Γ')
  with hyps1 have False
    by (meson linorder-cls.is-least-in-filter-iff)
  thus ?thesis ..
next
  case (decide-pos A C Γ' F')
  with hyps1 have False
    by (meson linorder-cls.is-least-in-filter-iff)
  thus ?thesis ..
next
  case (propagate A C Γ' F')
  with hyps1 have False
    by (meson linorder-cls.is-least-in-filter-iff)
  thus ?thesis ..
next
  case hyps2: (resolution D A C Uer' Γ')
  have D1 = D
    using hyps1 hyps2
    by (metis (no-types, opaque-lifting) linorder-cls.Uniq-is-least-in-fset Uniq-D)
  have A1 = A
    using hyps1 hyps2
    unfolding ⟨D1 = D⟩
  by (metis (mono-tags, opaque-lifting) Uniq-D linorder-lit.Uniq-is-maximal-in-mset
    literal.sel(2))
  have C1 = C
    using hyps1 hyps2
    unfolding ⟨A1 = A⟩
    by simp
  show ?thesis
    using hyps1 hyps2
    unfolding ⟨D1 = D⟩ ⟨A1 = A⟩ ⟨C1 = C⟩
    by argo
qed
qed
qed

sublocale ord-res-10-semantics: semantics where
  step = constant-context ord-res-10 and
  final = ord-res-8-final
proof unfold-locales
  fix S :: 'f ord-res-10-state

obtain
  N Uer F :: 'f gclause fset and
  Γ :: ('f gliteral × 'f gclause option) list where
  S-def: S = (N, Uer, F, Γ)

```

**by** (*metis prod.exhaust*)

**assume** *ord-res-8-final S*

**hence**  $\# x. \text{ord-res-10 } N (U_{er}, \mathcal{F}, \Gamma) x$   
**unfolding** *S-def*

**proof** (*cases (N, U<sub>er</sub>, F, Γ) rule: ord-res-8-final.cases*)  
**case** *model-found*

**have**  $\# A. \text{linorder-trm.is-least-in-fset } \{|A_2 | \in | \text{atms-of-clss } (N \cup U_{er}).$   
 $\forall A_1 | \in | \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A$   
**using**  $\langle \neg (\exists A | \in | \text{atms-of-clss } (N \cup U_{er}). A | \notin | \text{trail-atms } \Gamma) \rangle$   
**using** *linorder-trm.is-least-in-ffilter-iff* **by** *fastforce*

**moreover have**  $\# C. \text{linorder-cls.is-least-in-fset}$   
 $(\text{ffilter } (\text{trail-false-cls } \Gamma) (\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}))) C$   
**using**  $\langle \neg fBex (\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er})) (\text{trail-false-cls } \Gamma) \rangle$   
**by** (*metis linorder-cls.is-least-in-ffilter-iff*)

**ultimately show** *?thesis*  
**unfolding** *ord-res-10.simps* **by** *blast*

**next**  
**case** *contradiction-found*

**hence**  $\exists C | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}). \text{trail-false-cls } \Gamma C$   
**using** *trail-false-cls-empty* **by** *metis*

**moreover have**  $\# D A. \text{linorder-cls.is-least-in-fset } (\text{ffilter } (\text{trail-false-cls } \Gamma)$   
 $(\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}))) D \wedge \text{ord-res.is-maximal-lit } (\text{Neg } A) D$   
**by** (*metis empty-iff linorder-cls.is-least-in-ffilter-iff linorder-cls.leD*  
*linorder-lit.is-maximal-in-mset-iff local.contradiction-found(1) mempty-lesseq-clss*  
*set-mset-empty trail-false-clss-empty*)

**ultimately show** *?thesis*  
**unfolding** *ord-res-10.simps* **by** *blast*

**qed**

**thus finished** (*constant-context ord-res-10*) *S*  
**by** (*simp add: S-def finished-def constant-context.simps*)

**qed**

**inductive** *ord-res-10-invars* **for** *N* **where**  
*ord-res-10-invars N (U<sub>er</sub>, F, Γ) if*  
*sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$  **and**  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C (\text{fst } Ln)$  **and**  
*linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \cup U_{er}))$  **and**  
 $\forall Ln \Gamma'. \Gamma = Ln \# \Gamma' \longrightarrow$   
 $(\text{snd } Ln \neq \text{None} \longleftrightarrow (\exists C | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}). \text{trail-false-clss } \Gamma C))$

$\wedge$   
 $(\text{snd } Ln \neq \text{None} \longrightarrow \text{is-pos } (\text{fst } Ln)) \wedge$   
 $(\forall C. \text{snd } Ln = \text{Some } C \longrightarrow C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \wedge$   
 $(\forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{clause-could-propagate } \Gamma' C (\text{fst } Ln)) \wedge$   
 $(\forall x \in \text{set } \Gamma'. \text{snd } x = \text{None}) \text{ and}$   
 $\forall \Gamma_1 Ln \Gamma_0. \Gamma = \Gamma_1 @ Ln \# \Gamma_0 \longrightarrow$   
 $\text{snd } Ln = \text{None} \longrightarrow \neg(\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } (Ln \#$   
 $\Gamma_0) C)$

**lemma** *ord-res-10-preserves-invars*:

**assumes**

*step*: *ord-res-10*  $N s s'$  **and**

*invars*: *ord-res-10-invars*  $N s$

**shows** *ord-res-10-invars*  $N s'$

**using** *invars*

**proof** (*cases*  $N s$  *rule*: *ord-res-10-invars.cases*)

**case** *invars*:  $(1 \Gamma U_{er} \mathcal{F})$

**note**  $\Gamma\text{-sorted} = \langle \text{sorted-wrt } (\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma \rangle$

**note**  $\Gamma\text{-lower} = \langle \text{linorder-trm.is-lower-fset } (\text{trail-atms } \Gamma) (\text{atms-of-clss } (N \mid \cup \mid U_{er})) \rangle$

**have** *trail-consistent*  $\Gamma$

**using**  $\Gamma\text{-sorted}$  *trail-consistent-if-sorted-wrt-atoms* **by** *metis*

**show** *?thesis*

**using** *step unfolding*  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**proof** (*cases*  $N (U_{er}, \mathcal{F}, \Gamma) s'$  *rule*: *ord-res-10.cases*)

**case** *step-hyps*:  $(\text{decide-neg } A \Gamma')$

**have**

*A-in*:  $A \in | \text{atms-of-clss } (N \mid \cup \mid U_{er})$  **and**

*A-gt*:  $\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t A$  **and**

*A-lt*:  $\forall y \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}).$

$y \neq A \longrightarrow (\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$

**using**  $\langle \text{linorder-trm.is-least-in-fset} - A \rangle$

**unfolding** *atomize-conj linorder-trm.is-least-in-filter-iff*

**by** *argo*

**have** *trail-atms*  $\Gamma' = \text{finsert } A (\text{trail-atms } \Gamma)$

**unfolding**  $\langle \Gamma' = (\text{Neg } A, -) \# \Gamma \rangle$  **by** *simp*

**show** *?thesis*

**unfolding**  $\langle s' = (-, -, -) \rangle$

**proof** (*intro ord-res-10-invars.intros allI impI conjI*)

**show** *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma'$

**unfolding**  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$  *sorted-wrt.simps*

**proof** (*intro conjI ballI*)

**fix**  $Ln :: 'f \text{ gliteral} \times 'f \text{ gclause option}$

```

assume  $Ln \in set \Gamma$ 

hence  $atm-of (fst Ln) \in trail-atms \Gamma$ 
  by (simp add: fset-trail-atms)

thus  $atm-of (fst Ln) \prec_t atm-of (fst (Neg A, None))$ 
  unfolding prod.sel literal.sel
  using A-gt by metis
next
show sorted-wrt  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$ 
  using  $\Gamma$ -sorted .
qed

show  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow ord-res.is-strictly-maximal-lit$ 
(fst Ln) C
  unfolding  $\langle \Gamma' = (-, None) \# \Gamma \rangle$ 
  using invars by simp

show linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-clss ( $N \cup U_{er}$ ))
  unfolding  $\langle trail-atms \Gamma' = finsert A (trail-atms \Gamma) \rangle$  finsert.rep-eq
proof (intro linorder-trm.is-lower-set-insertI ballI impI)
  show  $A \in atms-of-clss (N \cup U_{er})$ 
  using A-in .
next
fix  $w :: 'f gterm$ 
assume  $w \in atms-of-clss (N \cup U_{er})$  and  $w \prec_t A$ 
thus  $w \in trail-atms \Gamma$ 
  by (metis A-lt  $\Gamma$ -lower linorder-trm.dual-order.asym linorder-trm.neq-iff
linorder-trm.not-in-lower-setI)
next
show linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-clss ( $N \cup U_{er}$ ))
  using  $\Gamma$ -lower .
qed

{
fix
   $Ln :: 'f gliteral \times 'f gclause option$  and
   $\Gamma'' :: ('f gliteral \times 'f gclause option) list$ 

assume  $\Gamma' = Ln \# \Gamma''$ 

have  $snd Ln = None$ 
  using  $\langle \Gamma' = Ln \# \Gamma'' \rangle$  step-hyps by auto

moreover have  $\neg (\exists C \in |iefac \mathcal{F} | \uparrow (N \cup U_{er}). trail-false-cls ((Neg A,$ 
None)  $\# \Gamma) C)$ 
proof (rule notI , elim bexE)
  fix  $C :: 'f gclause$ 
  assume

```

*C-in*:  $C \in | \text{iefac } \mathcal{F} \text{ } |^{\dagger} (N \cup U_{er})$  **and**  
*C-false*:  $\text{trail-false-cls } ((\text{Neg } A, \text{None}) \# \Gamma) C$

**have** *clause-could-propagate*  $\Gamma C (\text{Pos } A)$   
**unfolding** *clause-could-propagate-def*  
**proof** (*intro conjI*)  
**show**  $\neg \text{trail-defined-lit } \Gamma (\text{Pos } A)$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm literal.sel*  
**by** (*metis A-gt linorder-trm.less-irrefl*)

**next**  
**show** *ord-res.is-maximal-lit*  $(\text{Pos } A) C$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**proof** (*intro conjI ballI impI*)  
**have**  $\neg \text{trail-false-cls } \Gamma C$   
**using** *step-hyps C-in* **by** *metis*

**thus**  $\text{Pos } A \in \# C$   
**using** *C-false* **by** (*metis subtrail-falseI uminus-Neg*)

**next**

**fix**  $L :: 'f \text{ gliteral}$   
**assume** *L-in*:  $L \in \# C$  **and** *L-neq*:  $L \neq \text{Pos } A$

**have** *trail-false-lit*  $((\text{Neg } A, \text{None}) \# \Gamma) L$   
**using** *C-false L-in* **unfolding** *trail-false-cls-def* **by** *metis*

**hence**  $L \in \text{fst } \text{'set } \Gamma$   
**unfolding** *trail-false-lit-def*  
**using** *L-neq*  
**by** (*cases L*) *simp-all*

**hence** *trail-defined-lit*  $\Gamma L$   
**unfolding** *trail-defined-lit-def* **by** *argo*

**hence** *atm-of L*  $\in | \text{trail-atms } \Gamma$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm* .

**moreover** **have**  $\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t A$   
**using** *step-hyps* **unfolding** *linorder-trm.is-least-in-filter-iff* **by** *argo*

**ultimately** **have** *atm-of L*  $\prec_t A$   
**by** *metis*

**hence**  $L \preceq_l \text{Pos } A$   
**by** (*cases L*) *simp-all*

**thus**  $\neg \text{Pos } A \prec_l L$   
**by** *order*

**qed**



```

next
  show trail-false-cls  $\Gamma$   $\{\#K \in \# C. K \neq Pos A\}$ 
    using C-false
    unfolding trail-false-cls-def trail-false-lit-def
    by (smt (verit, ccfv-SIG) mem-Collect-eq set-mset-filter subtrail-falseI
      trail-false-cls-def trail-false-lit-def uminus-Neg)
qed

moreover have  $\neg$  clause-could-propagate  $\Gamma$  C (Pos A)
  using C-in step-hyps by metis

ultimately show False
  by contradiction
qed

ultimately show (snd Ln  $\neq$  None) = ( $\exists C \mid \in \mid$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup \mid$  Uer)).
trail-false-cls  $\Gamma'$  C)
  unfolding  $\langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  by argo

show snd Ln  $\neq$  None  $\implies$  is-pos (fst Ln)
  using  $\langle$  snd Ln = None  $\rangle$  by argo

have  $\neg$  fBex (iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup \mid$  Uer)) (trail-false-cls  $\Gamma$ )
  using step-hyps by argo

hence  $\forall x \in set \Gamma. snd x = None$ 
  using invars by (metis list.set-cases)

thus  $\forall x \in set \Gamma''. snd x = None$ 
  using  $\langle \Gamma' = Ln \# \Gamma'' \rangle \langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  by simp

show  $\bigwedge C. snd Ln = Some C \implies$  clause-could-propagate  $\Gamma''$  C (fst Ln)
  using  $\langle \Gamma' = Ln \# \Gamma'' \rangle \langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  by force

show  $\bigwedge D. snd Ln = Some D \implies D \mid \in \mid$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup \mid$  Uer)
  using  $\langle \Gamma' = Ln \# \Gamma'' \rangle \langle \Gamma' = (Neg A, None) \# \Gamma \rangle$  by force
}

have  $\neg$  ( $\exists C \mid \in \mid$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup \mid$  Uer)). trail-false-cls ( $(Neg A, None) \# \Gamma$ ) C)
proof (intro notI , elim bexE)
  fix C :: 'f g clause
  assume
    C-in:  $C \mid \in \mid$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup \mid$  Uer) and
    C-false: trail-false-cls ( $(Neg A, None) \# \Gamma$ ) C

  have clause-could-propagate  $\Gamma$  C (Pos A)
    unfolding clause-could-propagate-def
  proof (intro conjI)
    show  $\neg$  trail-defined-lit  $\Gamma$  (Pos A)

```

```

    unfolding trail-defined-lit-iff-trail-defined-atm
    by (metis A-gt linorder-trm.less-irrefl literal.sel(1))
next
  have  $\neg$  trail-false-cls  $\Gamma$  C
    using C-in step-hyps by metis

  thus trail-false-cls  $\Gamma$   $\{\#K \in\# C. K \neq \text{Pos } A\# \}$ 
    by (smt (verit) C-false mem-Collect-eq set-mset-filter subtrail-falseI
      trail-false-cls-def uminus-Neg)

  show ord-res.is-maximal-lit (Pos A) C
    unfolding linorder-lit.is-maximal-in-mset-iff
  proof (intro conjI ballI impI)
    show Pos A  $\in\# C$ 
      by (metis C-false  $\langle \neg$  trail-false-cls  $\Gamma$  C  $\rangle$  subtrail-falseI uminus-Neg)
  next
  fix L
  assume L  $\in\# C$  and L  $\neq \text{Pos } A$ 
  hence atm-of L  $|\in|$  trail-atms  $\Gamma$ 
    using  $\langle$  trail-false-cls  $\Gamma$   $\{\#K \in\# C. K \neq \text{Pos } A\# \}$   $\rangle$ 
  using trail-defined-lit-iff-trail-defined-atm trail-defined-lit-iff-true-or-false
    trail-false-cls-filter-mset-iff by blast
  hence atm-of L  $\prec_t A$ 
    using A-gt by metis
  hence L  $\prec_l \text{Pos } A$ 
    by (cases L) simp-all
  thus  $\neg$  Pos A  $\prec_l L$ 
    by order
  qed
qed

  moreover have  $\neg$  clause-could-propagate  $\Gamma$  C (Pos A)
    using C-in step-hyps by metis

  ultimately show False
    by contradiction
qed

  thus  $\bigwedge \Gamma_1 Ln \Gamma_0. \Gamma' = \Gamma_1 @ Ln \# \Gamma_0 \implies \text{snd } Ln = \text{None} \implies$ 
     $\neg (\exists C |\in| \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } (Ln \# \Gamma_0) C)$ 
  unfolding  $\langle \Gamma' = (-, \text{None}) \# \Gamma \rangle$ 
  by (metis suffixI trail-false-cls-if-trail-false-suffix)
qed
next
  case step-hyps: (decide-pos A C  $\Gamma'$   $\mathcal{F}'$ )

  have
  A-in: A  $|\in|$  atms-of-cls (N  $\mid \cup \mid U_{er}$ ) and
  A-gt:  $\forall A_1 |\in|$  trail-atms  $\Gamma. A_1 \prec_t A$  and

```

*A-lt*:  $\forall y \in |atms-of-clss (N \cup U_{er})$ .  
 $y \neq A \longrightarrow (\forall A_1 \in |trail-atms \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$   
**using**  $\langle linorder-trm.is-least-in-fset - A \rangle$   
**unfolding**  $\langle atomize-conj linorder-trm.is-least-in-filter-iff \rangle$   
**by** *argo*

**have**  $trail-atms \Gamma' = finsert A (trail-atms \Gamma)$   
**unfolding**  $\langle \Gamma' = (Pos A, -) \# \Gamma \rangle$  **by** *simp*

**show** *?thesis*  
**unfolding**  $\langle s' = (-, -, -) \rangle$   
**proof** (*intro ord-res-10-invars.intros allI impI conjI*)  
**show**  $sorted-wrt (\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma'$   
**unfolding**  $\langle \Gamma' = (Pos A, None) \# \Gamma \rangle$  *sorted-wrt.simps*  
**proof** (*intro conjI ballI*)  
**fix**  $Ln :: 'f gliteral \times 'f gclause option$   
**assume**  $Ln \in set \Gamma$   
  
**hence**  $atm-of (fst Ln) \in |trail-atms \Gamma$   
**by** (*simp add: fset-trail-atms*)  
  
**thus**  $atm-of (fst Ln) \prec_t atm-of (fst (Pos A, None))$   
**unfolding** *prod.sel literal.sel*  
**using** *A-gt* **by** *metis*  
**next**  
**show**  $sorted-wrt (\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$   
**using**  $\Gamma$ -*sorted* .  
**qed**

**show**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow ord-res.is-strictly-maximal-lit$   
 $(fst Ln) C$   
**unfolding**  $\langle \Gamma' = (-, None) \# \Gamma \rangle$   
**using** *invars* **by** *simp*

**show**  $linorder-trm.is-lower-fset (trail-atms \Gamma') (atms-of-clss (N \cup U_{er}))$   
**unfolding**  $\langle trail-atms \Gamma' = finsert A (trail-atms \Gamma) \rangle$  *finsert.rep-eq*  
**proof** (*intro linorder-trm.is-lower-set-insertI ballI impI*)  
**show**  $A \in |atms-of-clss (N \cup U_{er})$   
**using** *A-in* .  
**next**  
**fix**  $w :: 'f gterm$   
**assume**  $w \in |atms-of-clss (N \cup U_{er})$  **and**  $w \prec_t A$   
**thus**  $w \in |trail-atms \Gamma$   
**by** (*metis A-lt  $\Gamma$ -lower linorder-trm.dual-order.asym linorder-trm.neq-iff linorder-trm.not-in-lower-setI*)  
**next**  
**show**  $linorder-trm.is-lower-fset (trail-atms \Gamma) (atms-of-clss (N \cup U_{er}))$   
**using**  $\Gamma$ -*lower* .  
**qed**

```

{
  fix Ln and Γ''
  assume Γ' = Ln # Γ''

  have snd Ln = None
    using ⟨Γ' = Ln # Γ''⟩ step-hyps by auto

  moreover have ¬ (∃ C |∈| iefac F' |↑| (N |∪| Uer). trail-false-cls ((Pos A,
None) # Γ) C)
  proof (rule notI , elim bexE)
    fix D :: 'f gclause
    assume D-in: D |∈| iefac F' |↑| (N |∪| Uer)

    hence D = efac C ∨ D |∈| iefac F' |↑| (N |∪| Uer)
    unfolding ⟨F' = (if ord-res.is-strictly-maximal-lit (Pos A) C then F else
finsert C F)⟩
      by (smt (z3) fimage-iff finsert-iff iefac-def)

    hence ¬ trail-false-cls Γ' D
    proof (elim disjE)
      assume D = efac C

      hence trail-false-cls Γ' D ⟷ trail-false-cls Γ' C
      by (simp add: trail-false-cls-def)

      moreover have ¬ trail-false-cls Γ' C
      using step-hyps unfolding linorder-cls.is-least-in-filter-iff by metis

      ultimately show ¬ trail-false-cls Γ' D
      by argo
    next
      assume D |∈| iefac F' |↑| (N |∪| Uer)
      thus ¬ trail-false-cls Γ' D
      using step-hyps by metis
    qed

    moreover assume trail-false-cls ((Pos A, None) # Γ) D

    ultimately show False
    unfolding ⟨Γ' = (Pos A, None) # Γ⟩
      by contradiction
  qed

  ultimately show snd Ln ≠ None ⟷
  (∃ C |∈| iefac F' |↑| (N |∪| Uer). trail-false-cls Γ' C)
  unfolding ⟨Γ' = (Pos A, None) # Γ⟩ by argo

  show snd Ln ≠ None ⟹ is-pos (fst Ln)

```

**using**  $\langle \text{snd } Ln = \text{None} \rangle$  **by** *argo*

**have**  $\neg fBex (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) (trail\text{-}false\text{-}cls \Gamma)$   
**using** *step-hyps* **by** *argo*

**hence**  $\forall x \in set \Gamma. \text{snd } x = \text{None}$   
**using** *invars* **by** (*metis list.set-cases*)

**thus**  $\forall x \in set \Gamma''. \text{snd } x = \text{None}$   
**using**  $\langle \Gamma' = Ln \# \Gamma'' \rangle \langle \Gamma' = (Pos A, \text{None}) \# \Gamma \rangle$  **by** *simp*

**show**  $\bigwedge C. \text{snd } Ln = \text{Some } C \implies \text{clause-could-propagate } \Gamma'' C (fst Ln)$   
**using**  $\langle \Gamma' = Ln \# \Gamma'' \rangle \langle \Gamma' = (Pos A, \text{None}) \# \Gamma \rangle$  **by** *force*

**show**  $\bigwedge D. \text{snd } Ln = \text{Some } D \implies D \mid \in \mid iefac \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\langle \Gamma' = Ln \# \Gamma'' \rangle \langle \Gamma' = (Pos A, \text{None}) \# \Gamma \rangle$  **by** *force*

**}**

**show**  $\bigwedge \Gamma_1 Ln \Gamma_0. \Gamma' = \Gamma_1 @ Ln \# \Gamma_0 \implies \text{snd } Ln = \text{None} \implies$   
 $\neg (\exists C \mid \in \mid iefac \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). trail\text{-}false\text{-}cls (Ln \# \Gamma_0) C)$   
**using** *step-hyps*  
**unfolding** *bex-trail-false-cls-simp*  
**by** (*meson suffixI trail-false-cls-if-trail-false-suffix*)

**qed**

**next**

**case** *step-hyps*: (*propagate A C  $\Gamma'$   $\mathcal{F}'$* )

**have**

*A-in*:  $A \mid \in \mid \text{atms-of-cls} (N \mid \cup \mid U_{er})$  **and**  
*A-gt*:  $\forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A$  **and**  
*A-lt*:  $\forall y \mid \in \mid \text{atms-of-cls} (N \mid \cup \mid U_{er}).$   
 $y \neq A \longrightarrow (\forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$   
**using**  $\langle \text{linorder-trm.is-least-in-fset} - A \rangle$   
**unfolding** *atomize-conj linorder-trm.is-least-in-filter-iff*  
**by** *argo*

**have**

*C-in*:  $C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
*C-prop*: *clause-could-propagate*  $\Gamma C (Pos A)$  **and**  
*C-lt*:  $\forall D \mid \in \mid iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$   
 $D \neq C \longrightarrow \text{clause-could-propagate } \Gamma D (Pos A) \longrightarrow C \prec_c D$   
**using**  $\langle \text{linorder-cls.is-least-in-fset} - C \rangle$   
**unfolding** *atomize-conj linorder-cls.is-least-in-filter-iff* **by** *argo*

**have** *trail-atms*  $\Gamma' = \text{finsert } A (trail\text{-}atms \Gamma)$   
**unfolding**  $\langle \Gamma' = (Pos A, -) \# \Gamma \rangle$  **by** *simp*

**show** *?thesis*  
**unfolding**  $\langle s' = (-, -, -) \rangle$

```

proof (intro ord-res-10-invars.intros allI impI conjI)
  show sorted-wrt ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma'$ 
    unfolding  $\langle \Gamma' = (Pos A, -) \# \Gamma \rangle$  sorted-wrt.simps
proof (intro conjI ballI)
  fix Ln :: 'f gliteral  $\times$  'f gclause option
  assume Ln  $\in$  set  $\Gamma$ 

  hence atm-of (fst Ln)  $|\in|$  trail-atms  $\Gamma$ 
    by (simp add: fset-trail-atms)

  thus atm-of (fst Ln)  $\prec_t$  atm-of (fst (Pos A, Some (efac C)))
    unfolding prod.sel literal.sel
    using A-gt by metis
next
  show sorted-wrt ( $\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)$ )  $\Gamma$ 
    using  $\Gamma$ -sorted .
qed

  show  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{ord-res.is-strictly-maximal-lit}$ 
  (fst Ln) C
    unfolding  $\langle \Gamma' = (Pos A, \text{Some } (efac C)) \# \Gamma \rangle$ 
    using invars step-hyps
    by (simp add: clause-could-propagate-def greatest-literal-in-efacI
      linorder-cls.is-least-in-filter-iff)

  show linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-clss (N  $\cup$   $U_{er}$ ))
    unfolding  $\langle \text{trail-atms } \Gamma' = \text{finsert } A (\text{trail-atms } \Gamma) \rangle$  finsert.rep-eq
proof (intro linorder-trm.is-lower-set-insertI ballI impI)
  show A  $|\in|$  atms-of-clss (N  $\cup$   $U_{er}$ )
    using A-in .
next
  fix w :: 'f gterm
  assume w  $|\in|$  atms-of-clss (N  $\cup$   $U_{er}$ ) and w  $\prec_t$  A
  thus w  $|\in|$  trail-atms  $\Gamma$ 
    by (metis A-lt  $\Gamma$ -lower linorder-trm.dual-order.asym linorder-trm.neq-iff
      linorder-trm.not-in-lower-setI)
next
  show linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-clss (N  $\cup$   $U_{er}$ ))
    using  $\Gamma$ -lower .
qed

{
  fix Ln and  $\Gamma''$ 
  assume  $\Gamma' = Ln \# \Gamma''$ 
  hence Ln = (Pos A, Some (efac C)) and  $\Gamma'' = \Gamma$ 
  using  $\langle \Gamma' = (Pos A, \text{Some } (efac C)) \# \Gamma \rangle$  by simp-all

  obtain D where D-in: D  $|\in|$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\cup$   $U_{er}$ ) and D-false:
  trail-false-cls  $\Gamma' D$ 

```

```

using step-hyps by metis

have  $(\exists C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma' C)$ 
proof (rule bexI)
  show trail-false-cls  $\Gamma' D$ 
    using D-false .
next
  have  $\neg \text{trail-false-cls } \Gamma' C$ 
    by (metis clause-could-propagate-def linorder-cls.is-least-in-ffilter-iff
      linorder-lit.is-maximal-in-mset-iff ord-res.less-lit-simps(2) reflclp-iff
      step-hyps(2,4,5) subtrail-falseI uminus-Pos uminus-not-id')

hence  $D \neq C$ 
  using D-false by metis

thus  $D \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$ 
  unfolding  $\langle \mathcal{F}' = (\text{if } \text{ord-res.is-strictly-maximal-lit } (Pos A) C \text{ then } \mathcal{F} \text{ else } \text{finsert } C \mathcal{F}) \rangle$ 
  using D-in-iefac-def by auto
qed

moreover have snd  $Ln \neq None$ 
  using  $\langle \Gamma' = Ln \# \Gamma'' \rangle$  step-hyps by auto

ultimately show snd  $Ln \neq None \longleftrightarrow$ 
   $(\exists C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma' C)$ 
  by argo

show snd  $Ln \neq None \implies \text{is-pos } (fst Ln)$ 
  using  $\langle Ln = (Pos A, Some (efac C)) \rangle$  by auto

show  $\forall x \in \text{set } \Gamma''. \text{snd } x = None$ 
  unfolding  $\langle \Gamma'' = \Gamma \rangle$ 
  using invars by (meson list.set-cases step-hyps(2))

have clause-could-propagate  $\Gamma (efac C) (Pos A)$ 
  using C-prop clause-could-propagate-efac by metis

thus  $\bigwedge C. \text{snd } Ln = Some C \implies \text{clause-could-propagate } \Gamma'' C (fst Ln)$ 
  using  $\langle \Gamma' = Ln \# \Gamma'' \rangle$   $\langle \Gamma' = (Pos A, Some (efac C)) \# \Gamma \rangle$ 
  by force

have efac  $C \in \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})$ 
proof (cases ord-res.is-strictly-maximal-lit  $(Pos A) C$ )
  case True
  thus ?thesis
  unfolding  $\langle \mathcal{F}' = \rightarrow \rangle$ 
  using C-in
  by (metis (mono-tags, opaque-lifting) literal.discI(1))

```

```

      nex-strictly-maximal-pos-lit-if-neq-efac)
next
  case False
  then show ?thesis
    unfolding ⟨ $\mathcal{F}' = -$ ⟩
    using C-in
  by (smt (z3) fimage-iff finsert-iff iefac-def nex-strictly-maximal-pos-lit-if-neq-efac
      obtains-positive-greatest-lit-if-efac-not-ident)
qed

thus  $\bigwedge D. \text{snd } Ln = \text{Some } D \implies D \in | \text{iefac } \mathcal{F}' |^\uparrow (N \cup U_{er})$ 
  using ⟨ $\Gamma' = Ln \# \Gamma''$ ⟩ ⟨ $\Gamma' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma$ ⟩ by force
}

show  $\bigwedge \Gamma_1 Ln \Gamma_0. \Gamma' = \Gamma_1 @ Ln \# \Gamma_0 \implies \text{snd } Ln = \text{None} \implies$ 
 $\neg (\exists C \in | \text{iefac } \mathcal{F}' |^\uparrow (N \cup U_{er}). \text{trail-false-cls } (Ln \# \Gamma_0) C)$ 
  unfolding ⟨ $\Gamma' = (-, \text{Some } -) \# \Gamma$ ⟩
  using invars
  unfolding bex-trail-false-cls-simp
  by (metis list.inject not-None-eq split-pairs suffix-Cons suffix-def)
qed
next
case step-hyps: (resolution D A C Uer'  $\Gamma'$ )

note D-max-lit = ⟨ord-res.is-maximal-lit (Neg A) D⟩
have C-max-lit: ⟨ord-res.is-strictly-maximal-lit (Pos A) C⟩
  using invars by (metis map-of-SomeD split-pairs step-hyps(4))

have
  D-in:  $D \in | \text{iefac } \mathcal{F} |^\uparrow (N \cup U_{er})$  and
  D-false: trail-false-cls  $\Gamma$  D and
  D-lt:  $\forall E \in | \text{iefac } \mathcal{F} |^\uparrow (N \cup U_{er}). E \neq D \longrightarrow \text{trail-false-cls } \Gamma E \longrightarrow D \prec_c$ 
E
  using ⟨linorder-cls.is-least-in-fset - D⟩
  unfolding atomize-conj linorder-cls.is-least-in-filter-iff by argo

have A  $\in | \text{atms-of-clss } (N \cup U_{er})$ 
  using D-in D-max-lit
by (metis atm-of-in-atms-of-clssI linorder-lit.is-maximal-in-mset-iff literal.sel(2))

have ord-res.ground-resolution D C ((D - {#Neg A#}) + (C - {#Pos A#}))
proof (rule ord-res.ground-resolutionI)
  show D = add-mset (Neg A) (D - {#Neg A#})
    using D-max-lit unfolding linorder-lit.is-maximal-in-mset-iff by simp
next
  show C = add-mset (Pos A) (C - {#Pos A#})
    using C-max-lit unfolding linorder-lit.is-greatest-in-mset-iff by simp
next
show C  $\prec_c$  D

```



```

    using C-max-lit D-max-lit
    by (simp add: clause-lt-clause-if-max-lit-comp
        linorder-lit.is-greatest-in-mset-iff-is-maximal-and-count-eq-one)
next
show {#} = {#}  $\wedge$  ord-res.is-maximal-lit (Neg A) D  $\vee$  Neg A  $\in$  # {#}
    using D-max-lit by argo
next
show {#} = {#}
    by argo
next
show ord-res.is-strictly-maximal-lit (Pos A) C
    using C-max-lit .
next
show remove1-mset (Neg A) D + remove1-mset (Pos A) C = (D - {#Neg
A#}) + (C - {#Pos A#})
    ..
qed
hence eres C D  $\neq$  D
    unfolding eres-ident-iff not-not ground-resolution-def by metis

obtain  $\Gamma_b$  where  $\Gamma = (Pos A, Some C) \# \Gamma_b$ 
    using  $\langle$ map-of  $\Gamma$  (Pos A) = Some (Some C) $\rangle$  invars
    by (metis list.set-cases map-of-SomeD not-Some-eq snd-conv)

have A  $\in$  | trail-atms  $\Gamma$ 
    unfolding  $\langle$  $\Gamma = (Pos A, Some C) \# \Gamma_b$  $\rangle$  trail-atms-def by simp

moreover have A  $\notin$  | trail-atms  $\Gamma'$ 
proof (cases eres C D = {#})
case True

    hence  $\nexists$  K. ord-res.is-maximal-lit K (eres C D)
        unfolding linorder-lit.is-maximal-in-mset-iff by simp

    hence  $\Gamma' = dropWhile (\lambda Ln. True) \Gamma$ 
        using step-hyps(6) by simp

    also have ... = []
        by simp

    finally show ?thesis
        using  $\langle$  $\Gamma = (Pos A, Some C) \# \Gamma_b$  $\rangle$  by simp
next
case False

then obtain K where eres-max-lit: ord-res.is-maximal-lit K (eres C D)
    using linorder-lit.ex-maximal-in-mset by presburger

have atm-of K  $\prec_t$  atm-of (Neg A)

```

```

proof (rule lit-in-eres-lt-greatest-lit-in-greatest-resolvent [of C D])
  show eres C D  $\neq$  D
    using  $\langle$ eres C D  $\neq$  D $\rangle$  .
next
  show ord-res.is-maximal-lit (Neg A) D
    using D-max-lit .
next
  show  $-$  Neg A  $\notin\#$  D
    using D-false  $\langle$ trail-consistent  $\Gamma$  $\rangle$ 
    by (meson D-max-lit linorder-lit.is-maximal-in-mset-iff
      not-both-lit-and-comp-lit-in-false-clause-if-trail-consistent)
next
  show K  $\in\#$  eres C D
    using eres-max-lit unfolding linorder-lit.is-maximal-in-mset-iff by argo
qed

hence atm-of K  $\prec_t$  A
  unfolding literal.sel .

hence atm-of K  $\preceq_t$  A
  by order

have  $\Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma$ 
  using step-hyps(6) eres-max-lit
by (smt (verit, ccfv-threshold) Uniq-D dropWhile-cong linorder-lit.Uniq-is-maximal-in-mset)

also have  $\dots = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma_b$ 
  unfolding  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_b \rangle$ 
  unfolding dropWhile.simps prod.sel literal.sel
  using  $\langle$ atm-of K  $\preceq_t$  A $\rangle$  by simp

finally have  $\Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma_b$  .

hence trail-atms  $\Gamma' \mid\subseteq\mid$  trail-atms  $\Gamma_b$ 
  by (simp only: suffix-dropWhile trail-atms-subset-if-suffix)

moreover have A  $\notin\mid$  trail-atms  $\Gamma_b$ 
proof (rule notI)
  assume A  $\in\mid$  trail-atms  $\Gamma_b$ 
  then obtain Ln where Ln  $\in$  set  $\Gamma_b$  and atm-of (fst Ln) = A
    unfolding fset-trail-atms by blast
  moreover have  $\forall y \in \text{set } \Gamma_b. \text{atm-of } (\text{fst } y) \prec_t A$ 
    using  $\Gamma$ -sorted[unfolded  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_b \rangle$ ] by simp
  ultimately have A  $\prec_t$  A
    by metis
  thus False
    by order
qed

```

**moreover have**  $trail-atms \Gamma_b \sqsubseteq | trail-atms \Gamma$   
**proof** (rule *trail-atms-subset-if-suffix*)  
**show**  $suffix \Gamma_b \Gamma$   
**by** (*simp only*:  $\langle \Gamma = (Pos A, Some C) \# \Gamma_b \rangle suffix-ConsI$ )  
**qed**

**ultimately show** *?thesis*  
**by** *fast*  
**qed**

**ultimately have**  $\Gamma' \neq \Gamma$   
**by** *metis*

**have**  $C-in: C \in | iefac \mathcal{F} | \uparrow (N \cup | U_{er})$   
**using** *invars*  
**by** (*meson*  $\langle \Gamma = (Pos A, Some C) \# \Gamma_b \rangle snd-conv$ )

**define**  $P :: 'f gliteral \times 'f gclause option \Rightarrow bool$  **where**  
 $P \equiv \lambda Ln. \forall K. ord-res.is-maximal-lit K (eres C D) \longrightarrow atm-of K \preceq_t atm-of$   
 $(fst Ln)$

**have**  $\Gamma = takeWhile P \Gamma @ \Gamma'$   
**unfolding** *takeWhile-dropWhile-id*  
**unfolding**  $P-def \langle \Gamma' = dropWhile - \Gamma \rangle$  **by** *simp*

**hence**  $suffix \Gamma' \Gamma$   
**unfolding** *suffix-def* **by** *metis*

**show** *?thesis*  
**unfolding**  $\langle s' = (-, -, -) \rangle$   
**proof** (*intro ord-res-10-invars.intros allI impI conjI*)  
**show**  $\Gamma'-sorted: sorted-wrt (\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma'$   
**by** (*metis (no-types, lifting) \Gamma'-sorted \langle suffix \Gamma' \Gamma \rangle sorted-wrt-append suffix-def*)

**show**  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow ord-res.is-strictly-maximal-lit$   
 $(fst Ln) C$   
**using**  $\langle suffix \Gamma' \Gamma \rangle invars(3) set-mono-suffix$  **by** *blast*

**have**  $\bigwedge xs. fset (trail-atms xs) = atm-of 'fst ' set xs$   
**unfolding** *fset-trail-atms ..*  
**also have**  $\bigwedge xs. atm-of 'fst ' set xs = set (map (atm-of o fst) xs)$   
**by** (*simp add: image-comp*)  
**finally have**  $\bigwedge xs. fset (trail-atms xs) = set (map (atm-of o fst) xs) .$

**have**  $atms-of-clss (N \cup | U_{er}') = atms-of-cl (eres C D) \cup | atms-of-clss (N$   
 $\cup | U_{er})$   
**unfolding**  $\langle U_{er}' = finsert (eres C D) U_{er} \rangle$  **by** *simp*

**also have**  $\dots = \text{atms-of-clss } (N \mid\cup\mid U_{er})$   
**proof** –  
**have**  $\text{atms-of-cl } (\text{eres } C \ D) \mid\subseteq\mid \text{atms-of-cl } C \mid\cup\mid \text{atms-of-cl } D$   
**by** (*smt (verit, best) atms-of-cl-def fimage-iff fset-fset-mset fsubsetI*  
*unionCI*  
*lit-in-one-of-resolvents-if-in-eres*)

**moreover have**  $\text{atms-of-cl } C \mid\subseteq\mid \text{atms-of-clss } (N \mid\cup\mid U_{er})$   
**using** *C-in*  
**by** (*metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb fsubset-funion-eq*)

**moreover have**  $\text{atms-of-cl } D \mid\subseteq\mid \text{atms-of-clss } (N \mid\cup\mid U_{er})$   
**using** *D-in*  
**by** (*metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb fsubset-funion-eq*)

**ultimately show** *?thesis*  
**by** *blast*  
**qed**

**finally have**  $\text{atms-of-clss } (N \mid\cup\mid U_{er}') = \text{atms-of-clss } (N \mid\cup\mid U_{er}) .$

**show**  $\Gamma'$ -lower: *linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-clss  $(N \mid\cup\mid U_{er}')$ )*  
**unfolding**  $\langle \text{fset } (\text{trail-atms } \Gamma') = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \Gamma') \rangle$   
**proof** (*rule linorder-trm.sorted-and-lower-set-appendD-right(2)*)  
**have** *sorted-wrt*  $(\lambda x y. y \prec_t x)$   $(\text{map } (\text{atm-of } \circ \text{fst}) \Gamma)$   
**using**  $\Gamma$ -sorted **by** (*simp add: sorted-wrt-map*)

**thus** *sorted-wrt*  $(\lambda x y. y \prec_t x)$   $(\text{map } (\text{atm-of } \circ \text{fst}) (\text{takeWhile } P \ \Gamma) @ \text{map } (\text{atm-of } \circ \text{fst}) \Gamma')$   
**using**  $\langle \Gamma = \text{takeWhile } P \ \Gamma @ \Gamma' \rangle$   
**by** (*metis map-append*)

**next**  
**show** *linorder-trm.is-lower-set*  
 $(\text{set } (\text{map } (\text{atm-of } \circ \text{fst}) (\text{takeWhile } P \ \Gamma) @ \text{map } (\text{atm-of } \circ \text{fst}) \Gamma'))$   
 $(\text{fset } (\text{atms-of-clss } (N \mid\cup\mid U_{er}')))$   
**unfolding** *map-append[symmetric]*  
**unfolding**  $\langle \Gamma = \text{takeWhile } P \ \Gamma @ \Gamma' \rangle$ [*symmetric*]  
**using**  $\Gamma$ -lower  
**unfolding**  $\langle \text{fset } (\text{trail-atms } \Gamma) = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \Gamma) \rangle$   
**unfolding**  $\langle \text{atms-of-clss } (N \mid\cup\mid U_{er}') = \text{atms-of-clss } (N \mid\cup\mid U_{er}) \rangle$   
**qed**

**have** *no-false-cl-in- $\Gamma'$* :  $\neg (\exists C \mid\in\mid \text{iefac } \mathcal{F} \mid\uparrow\mid (N \mid\cup\mid U_{er}'). \text{trail-false-cl } \Gamma' \ C)$   
**if** *eres-max-lit: ord-res.is-maximal-lit*  $K$  (*eres*  $C \ D$ )  
**for**  $K$

**proof** –  
**have**  $FOO: \bigwedge Ln.$   
 $(\forall K. \text{ord-res.is-maximal-lit } K \text{ (eres } C D) \longrightarrow \text{atm-of } K \preceq_t \text{atm-of (fst } Ln)) \longleftrightarrow$   
 $\text{atm-of } K \preceq_t \text{atm-of (fst } Ln)$   
**using**  $\text{eres-max-lit}$   
**by**  $(\text{metis linorder-lit.Uniq-is-maximal-in-mset the1-equality})$   
  
**have**  $\forall x \in \text{set } \Gamma'. \text{atm-of (fst } x) \prec_t \text{atm-of } K$   
**using**  $\langle \Gamma' = \text{dropWhile - } \Gamma \rangle [\text{unfolded } FOO] \Gamma\text{-sorted}$   
**by**  $(\text{metis linorder-trm.not-le mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone mono-atms-lt})$   
  
**hence**  $\forall x \in \text{set } \Gamma'. \text{atm-of (fst } x) \neq \text{atm-of } K$   
**by**  $\text{fastforce}$   
  
**hence**  $\text{atm-of } K \notin \text{trail-atms } \Gamma'$   
**unfolding**  $\text{fset-trail-atms}$  **by**  $\text{auto}$   
  
**hence**  $\neg \text{trail-defined-lit } \Gamma' K$   
**unfolding**  $\text{trail-defined-lit-iff-trail-defined-atm .}$   
  
**hence**  $\neg \text{trail-false-lit } \Gamma' K$   
**by**  $(\text{metis trail-defined-lit-iff-true-or-false})$   
  
**hence**  $\neg \text{trail-false-cls } \Gamma' \text{ (eres } C D)$   
**using**  $\text{eres-max-lit}$   
**unfolding**  $\text{linorder-lit.is-maximal-in-mset-iff trail-false-cls-def}$   
**by**  $\text{metis}$   
  
**show**  $\neg (\exists C \in |\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er}'). \text{trail-false-cls } \Gamma' C)$   
**unfolding**  $\langle U_{er}' = \text{finsert (eres } C D) U_{er} \rangle$   
**proof**  $(\text{rule notI , elim bexE})$   
**fix**  $E :: 'f \text{ gclause}$   
**assume**  
 $E\text{-in: } E \in |\text{iefac } \mathcal{F} | \uparrow (N \cup \text{finsert (eres } C D) U_{er})$  **and**  
 $E\text{-false: } \text{trail-false-cls } \Gamma' E$   
  
**have**  $E = \text{iefac } \mathcal{F} \text{ (eres } C D) \vee E \in |\text{iefac } \mathcal{F} | \uparrow (N \cup U_{er})$   
**using**  $E\text{-in}$  **by**  $\text{simp}$   
  
**thus**  $\text{False}$   
**proof**  $(\text{elim disjE})$   
**assume**  $E = \text{iefac } \mathcal{F} \text{ (eres } C D)$   
  
**hence**  $\text{trail-false-cls } \Gamma' \text{ (eres } C D)$   
**using**  $E\text{-false}$   
**by**  $(\text{simp add: iefac-def trail-false-cls-def})$

**thus** *False*  
**using**  $\langle \neg \text{trail-false-cls } \Gamma' \text{ (eres } C D) \rangle$  **by** *contradiction*  
**next**  
**assume**  $E \in | \text{iefac } \mathcal{F} \text{ |}^q \text{ (} N \text{ |} \cup \text{ | } U_{er} \text{)}$   
  
**moreover have** *trail-false-cls*  $\Gamma' E$   
**using**  $E\text{-false } \langle \text{suffix } \Gamma' \Gamma \rangle$  **by** (*metis trail-false-cls-if-trail-false-suffix*)  
  
**ultimately have**  $D \preceq_c E$   
**using**  $D\text{-lt } E\text{-false}$  **by** *fast*  
  
**then obtain**  $L$  **where**  $L \in \# E$  **and**  $\text{Neg } A \preceq_l L$   
**using**  $D\text{-max-lit}$   
**by** (*metis empty-iff linorder-cls.leD linorder-lit.is-maximal-in-mset-iff linorder-lit.leI ord-res.mulp-if-all-left-smaller set-mset-empty*)  
  
**hence**  $A \preceq_t \text{atm-of } L$   
**by** (*cases L simp-all*)  
  
**moreover have**  $A \in | \text{atms-of-cls } (N \text{ |} \cup \text{ | } U_{er} \text{)}$   
**using**  $\langle A \in | \text{atms-of-cls } (N \text{ |} \cup \text{ | } U_{er} \text{}) \rangle$   
**using**  $\langle \text{atms-of-cls } (N \text{ |} \cup \text{ | } U_{er} \text{ }) = \text{atms-of-cls } (N \text{ |} \cup \text{ | } U_{er} \text{ }) \rangle$   
**by** (*simp only:*)  
  
**ultimately have**  $\text{atm-of } L \notin | \text{trail-atms } \Gamma'$   
**using**  $\text{linorder-trm.not-in-lower-setI}[OF \Gamma'\text{-lower}] \langle A \notin | \text{trail-atms } \Gamma' \rangle$   
**by** *blast*  
  
**hence**  $\neg \text{trail-defined-lit } \Gamma' L$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm* .  
  
**hence**  $\neg \text{trail-false-lit } \Gamma' L$   
**by** (*metis trail-defined-lit-iff-true-or-false*)  
  
**moreover have** *trail-false-lit*  $\Gamma' L$   
**using**  $E\text{-false } \langle L \in \# E \rangle$  **unfolding** *trail-false-cls-def* **by** *metis*  
  
**ultimately show** *False*  
**by** *contradiction*  
**qed**  
**qed**  
**qed**  
  
**have** *ex-max-lit-in-eres-if*:  $\exists K. \text{ord-res.is-maximal-lit } K \text{ (eres } C D)$   
**if**  $\Gamma' = \Gamma_1 @ Ln \# \Gamma_0$  **for**  $Ln$  **and**  $\Gamma_1 \Gamma_0$   
**proof** –  
**have**  $\exists x xs. \Gamma' = x \# xs$   
**using** *that neq-Nil-conv* **by** *blast*

**hence**  $\exists x. \neg (\forall K. \text{ord-res.is-maximal-lit } K \text{ (eres } C D) \longrightarrow \text{atm-of } K \preceq_t \text{atm-of (fst } x))$   
**unfolding** *step-hyps(6) dropWhile-eq-Cons-conv by auto*

**thus** *?thesis*  
**by** *metis*  
**qed**

{  
**fix** *Ln and  $\Gamma''$*   
**assume**  $\Gamma' = Ln \# \Gamma''$

**then obtain** *K where*  
*eres-max-lit: ord-res.is-maximal-lit K (eres C D)*  
**using** *ex-max-lit-in-eres-if[of [], simplified] by metis*

**have**  $\neg (\exists C | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} \wedge). \text{trail-false-cls } \Gamma' C)$   
**using** *no-false-cls-in- $\Gamma'$  eres-max-lit by metis*

**moreover have** *snd Ln = None*  
**using** *invars  $\langle \Gamma' \neq \Gamma \rangle \langle \text{suffix } \Gamma' \Gamma \rangle$*   
**by** (*metis  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_b \rangle \langle \Gamma' = Ln \# \Gamma'' \rangle \text{in-set-conv-decomp}$*   
*suffix-Cons*  
*suffix-def*)

**ultimately show**  $\text{snd } Ln \neq \text{None} \longleftrightarrow (\exists C | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} \wedge). \text{trail-false-cls } \Gamma' C)$   
**by** *argo*

**show**  $\text{snd } Ln \neq \text{None} \implies \text{is-pos (fst } Ln)$   
**using**  $\langle \text{snd } Ln = \text{None} \rangle$  **by** *argo*

**show**  $\forall x \in \text{set } \Gamma''. \text{snd } x = \text{None}$   
**using** *invars  $\langle \Gamma' \neq \Gamma \rangle \langle \text{suffix } \Gamma' \Gamma \rangle$*   
**by** (*metis  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_b \rangle \langle \Gamma' = Ln \# \Gamma'' \rangle \text{set-mono-suffix}$*   
*subsetD suffix-ConsD2*)

**show**  $\bigwedge C. \text{snd } Ln = \text{Some } C \implies \text{clause-could-propagate } \Gamma'' C \text{ (fst } Ln)$   
**by** (*simp add:  $\langle \text{snd } Ln = \text{None} \rangle$* )

**show**  $\bigwedge E. \text{snd } Ln = \text{Some } E \implies E | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} \wedge)$   
**by** (*simp add:  $\langle \text{snd } Ln = \text{None} \rangle$* )

}

**show**  $\bigwedge \Gamma_1 Ln \Gamma_0. \Gamma' = \Gamma_1 @ Ln \# \Gamma_0 \implies \text{snd } Ln = \text{None} \implies$   
 $\neg (\exists C | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} \wedge). \text{trail-false-cls } (Ln \# \Gamma_0) C)$   
**using** *ex-max-lit-in-eres-if no-false-cls-in- $\Gamma'$*

by (*metis suffixI trail-false-cls-if-trail-false-suffix*)  
 qed  
 qed  
 qed

**lemma** *rtranclp-ord-res-10-preserves-invars*:

**assumes**  
*step*: (*ord-res-10 N*)<sup>\*\*</sup> *s s'* **and**  
*invars*: *ord-res-10-invars N s*  
**shows** *ord-res-10-invars N s'*  
**using** *step invars ord-res-10-preserves-invars*  
**by** (*smt (verit, del-insts) rtranclp-induct*)

**lemma** *ex-ord-res-10-if-not-final*:

**assumes**  
*not-final*:  $\neg$  *ord-res-8-final (N, s)* **and**  
*invars*: *ord-res-10-invars N s*  
**shows**  $\exists s'. \text{ord-res-10 } N \text{ } s \text{ } s'$   
**using** *invars*  
**proof** (*cases N s rule: ord-res-10-invars.cases*)  
**case** *invars'*: ( $1 \Gamma U_{er} \mathcal{F}$ )

**have**

*undef-atm-or-false-cls*:  
 $(\exists x \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). x \mid \notin \mid \text{trail-atms } \Gamma) \wedge$   
 $\neg (\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \text{ } x) \vee$   
 $(\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \text{ } x)$  **and**  
 $\{\#\} \mid \notin \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
**unfolding** *atomize-conj*  
**using** *not-final[unfolded ord-res-8-final.simps]*  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$  **by** *metis*

**show** *?thesis*

**using** *undef-atm-or-false-cls*  
**proof** (*elim disjE conjE*)

**assume**

*ex-undef-atm*:  $\exists x \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). x \mid \notin \mid \text{trail-atms } \Gamma$  **and**  
*no-conflict*:  $\neg (\exists x \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \text{ } x)$

**moreover have**  $\{\mid A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1$   
 $\prec_t A_2 \mid \} \neq \{\mid \}$

**proof** –

**obtain**  $A_2 :: 'f \text{ gterm}$  **where**

$A_2\text{-in}$ :  $A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$  **and**

$A_2\text{-undef}$ :  $A_2 \mid \notin \mid \text{trail-atms } \Gamma$

**using** *ex-undef-atm* **by** *metis*

**have**  $A_1 \prec_t A_2$  **if**  $A_1\text{-in}$ :  $A_1 \mid \in \mid \text{trail-atms } \Gamma$  **for**  $A_1 :: 'f \text{ gterm}$

**proof** –

**have**  $A_1 \neq A_2$



**using**  $A_1$ -in  $A_2$ -undef **by** *metis*

**moreover have** *linorder-trm.is-lower-fset* (*trail-atms*  $\Gamma$ ) (*atms-of-cls* ( $N$   $| \cup |$   $U_{er}$ ))

**using** *invars'*[*unfolded trail-is-lower-set-def, simplified*] **by** *argo*

**ultimately show** *?thesis*

**by** (*meson*  $A_2$ -in  $A_2$ -undef *linorder-trm.is-lower-set-iff* *linorder-trm.linorder-cases* *that*)

**qed**

**thus** *?thesis*

**using**  $A_2$ -in

**by** (*smt* (*verit, ccfv-threshold*) *femptyE fmember-filter*)

**qed**

**ultimately obtain**  $A :: 'f$  *gterm* **where**

*A-least: linorder-trm.is-least-in-fset*  $\{|A_2| \in | \text{atms-of-cls } (N | \cup | U_{er}).$

$\forall A_1 | \in | \text{trail-atms } \Gamma. A_1 \prec_t A_2 | \}$   $A$

**using** *ex-undef-atm linorder-trm.ex-least-in-fset* **by** *presburger*

**show**  $\exists s'. \text{ord-res-10 } N \ s \ s'$

**proof** (*cases*  $\exists C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N | \cup | U_{er}). \text{ clause-could-propagate } \Gamma \ C \ (Pos \ A)$ )

**case** *True*

**hence**  $\{|C| \in | \text{iefac } \mathcal{F} \ | \uparrow (N | \cup | U_{er}). \text{ clause-could-propagate } \Gamma \ C \ (Pos \ A) | \}$

$\neq \{ | \}$

**by** *force*

**then obtain**  $C$  **where**

*C-least: linorder-cls.is-least-in-fset*

$\{|C| \in | \text{iefac } \mathcal{F} \ | \uparrow (N | \cup | U_{er}). \text{ clause-could-propagate } \Gamma \ C \ (Pos \ A) | \}$   $C$

**using** *linorder-cls.ex1-least-in-fset* **by** *meson*

**show** *?thesis*

**proof** (*cases*  $\exists D | \in | \text{iefac } \mathcal{F} \ | \uparrow (N | \cup | U_{er}). \text{ trail-false-cls } ((Pos \ A, \text{Some } (efac \ C)) \# \Gamma) \ D)$ )

**case** *True*

**then show** *?thesis*

**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

**using** *ord-res-10.propagate[OF no-conflict A-least C-least]*

**by** *metis*

**next**

**case** *False*

**hence**  $\neg (\exists C | \in | \text{iefac } \mathcal{F} \ | \uparrow (N | \cup | U_{er}). \text{ trail-false-cls } ((Pos \ A, \text{None}) \# \Gamma)$

$C)$

**by** (*simp add: trail-false-cls-def trail-false-lit-def*)

**then show** *?thesis*

**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$

```

    using ord-res-10.decide-pos[OF no-conflict A-least C-least]
    by metis
qed
next
case False
thus ?thesis
  unfolding ‹s = (Uer, F, Γ)›
  using ord-res-10.decide-neg[OF no-conflict A-least]
  by metis
qed
next
assume ∃ x |∈| iefac F |↑| (N |∪| Uer). trail-false-cls Γ x

then obtain D :: 'f gclause where
  D-least: linorder-cls.is-least-in-fset
    (ffilter (trail-false-cls Γ) (iefac F |↑| (N |∪| Uer))) D
  by (metis femptyE fmember-filter linorder-cls.ex-least-in-fset)

hence D-in: D |∈| iefac F |↑| (N |∪| Uer) and D-false: trail-false-cls Γ D
  unfolding atomize-conj linorder-cls.is-least-in-ffilter-iff by argo

have D ≠ {#}
  using D-in ‹{#} |∉| iefac F |↑| (N |∪| Uer)› by metis

hence Γ ≠ []
  using D-false
  by (auto simp add: trail-false-cls-def trail-false-lit-def)

then obtain Ln Γ' where Γ = Ln # Γ'
  using neq-Nil-conv by metis

hence snd Ln ≠ None
  using invars' ‹∃ x |∈| iefac F |↑| (N |∪| Uer). trail-false-cls Γ x› by presburger

have ∀ x ∈ set Γ'. snd x = None
  using invars' ‹Γ = Ln # Γ'› by metis

have ¬(∃ x |∈| iefac F |↑| (N |∪| Uer). trail-false-cls Γ' x)
proof (cases Γ')
case Nil
thus ?thesis
  using ‹{#} |∉| iefac F |↑| (N |∪| Uer)›
  by (simp add: trail-false-cls-def trail-false-lit-def)
next
case (Cons x xs)
hence snd x = None
  using ‹∀ x ∈ set Γ'. snd x = None› by simp
then show ?thesis
  using ‹∀ Γ1 Ln Γ0. Γ = Γ1 @ Ln # Γ0 ⟶ snd Ln = None ⟶
```

$\neg (\exists C \mid \in \cdot \text{trail-false-cls } (Ln \# \Gamma_0) C)$  [rule-format, of [Ln] x xs,  
*simplified*]  
**using** *Cons*  $\langle \Gamma = Ln \# \Gamma' \rangle$  **by** *blast*  
**qed**

**hence**  $\neg \text{trail-false-cls } \Gamma' D$   
**using** *D-in* **by** *metis*

**hence**  $\neg \text{fst } Ln \in \# D$   
**using** *D-false*  $\langle \Gamma = Ln \# \Gamma' \rangle$  **by** (*metis eq-fst-iff subtrail-falseI*)

**moreover have** *is-pos* (*fst Ln*)  
**using** *invars'*  $\langle \Gamma = Ln \# \Gamma' \rangle$   $\langle \text{snd } Ln \neq \text{None} \rangle$  **by** *metis*

**moreover have**  $\forall L \in \# D. \text{atm-of } L \mid \in \mid \text{trail-atms } \Gamma$   
**using** *D-false*  
**unfolding** *trail-false-cls-def*  
**using** *trail-defined-lit-iff-true-or-false* [of  $\Gamma$ ]  
**using** *trail-defined-lit-iff-trail-defined-atm* [of  $\Gamma$ ]  
**by** *metis*

**moreover have**  $\forall x \mid \in \mid \text{trail-atms } \Gamma'. x \prec_t \text{atm-of } (\text{fst } Ln)$   
**using**  $\langle \text{sorted-wrt } (\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma \rangle$  [  
*unfolded*  $\langle \Gamma = Ln \# \Gamma' \rangle$  *sorted-wrt.simps*]  
**by** (*simp add: fset-trail-atms*)

**ultimately have** *linorder-lit.is-maximal-in-mset*  $D$  ( $\neg \text{fst } Ln$ )  
**unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**by** (*smt* (*verit*, *best*)  $\langle \Gamma = Ln \# \Gamma' \rangle$  *insertE ord-res.less-lit-simps*(4)  
*linorder-lit.not-less-iff-gr-or-eq literal.exhaust-sel ord-res.less-lit-simps*(3)  
*trail-atms.simps*(2) *uminus-literal-def*)

**moreover obtain**  $A$  **where**  $\text{fst } Ln = \text{Pos } A$   
**using**  $\langle \text{is-pos } (\text{fst } Ln) \rangle$  **by** (*cases fst Ln*) *simp-all*

**ultimately have** *eres-max-lit: ord-res.is-maximal-lit* ( $\text{Neg } A$ )  $D$   
**by** *simp*

**obtain**  $C :: 'f \text{ gclause where}$   
 $\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C)$   
**unfolding**  $\langle \Gamma = Ln \# \Gamma' \rangle$   
**using**  $\langle \text{fst } Ln = \text{Pos } A \rangle$   $\langle \text{snd } Ln \neq \text{None} \rangle$  **by** *force*

**thus**  $\exists s'. \text{ord-res-10 } N s s'$   
**unfolding**  $\langle s = (U_{er}, \mathcal{F}, \Gamma) \rangle$   
**using** *ord-res-10.resolution* [*OF D-least eres-max-lit*] **by** *blast*  
**qed**  
**qed**

**lemma** *ord-res-10-safe-state-if-invars*:  
**fixes**  $N s$   
**assumes** *invars*: *ord-res-10-invars*  $N s$   
**shows** *safe-state* (*constant-context ord-res-10*) *ord-res-8-final* ( $N, s$ )  
**using** *safe-state-constant-context-if-invars*[**where**  
 $\mathcal{R} = \text{ord-res-10}$  **and**  $\mathcal{F} = \text{ord-res-8-final}$  **and**  $\mathcal{I} = \text{ord-res-10-invars}$ ]  
**using** *ord-res-10-preserves-invars* *ex-ord-res-10-if-not-final invars* **by** *metis*

**end**

**end**

**theory** *ORD-RES-11*  
**imports** *ORD-RES-10*  
**begin**

## 26 ORD-RES-11 (SCL strategy)

**type-synonym** *'f ord-res-11-state* =  
*'f gclause fset*  $\times$  *'f gclause fset*  $\times$  *'f gclause fset*  $\times$  (*'f gliteral*  $\times$  *'f gclause option*)  
*list*  $\times$   
*'f gclause option*

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**lemma**

**fixes**  $N U_{er} \mathcal{F} \Gamma A$

**assumes**

*no-false-cls*:  $\neg (\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma C)$  **and**

*A-least*: *linorder-trm.is-least-in-fset*  $\{|A_2 \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}).$

$\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t A_2\}$   $A$  **and**

*C-least*: *linorder-cls.is-least-in-fset*  $\{|C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}).$

*clause-could-propagate*  $\Gamma C (Pos A)\}$   $C$

**defines**

$\Gamma' \equiv (Pos A, None) \# \Gamma$  **and**

$\mathcal{F}' \equiv (\text{if } \text{linorder-lit.is-greatest-in-mset } C (Pos A) \text{ then } \mathcal{F} \text{ else } \text{finsert } C \mathcal{F})$

**shows**

$(\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma' C) \longleftrightarrow$

$(\exists C \in | \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma' C)$

**by** (*meson bex-trail-false-cls-simp*)

**inductive** *ord-res-11* **where**

*decide-neg*:

$\neg (\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma C) \implies$

*linorder-trm.is-least-in-fset*  $\{|A_2 \in | \text{atms-of-clss } (N \mid \cup \mid U_{er}).$

$\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t A_2\}$   $A \implies$

$\neg (\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{clause-could-propagate } \Gamma C (Pos A)) \implies$

$\Gamma' = (Neg A, None) \# \Gamma \implies$

*ord-res-11*  $N (U_{er}, \mathcal{F}, \Gamma, None) (U_{er}, \mathcal{F}, \Gamma', None) \mid$

*decide-pos:*

$$\begin{aligned} & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma C) \implies \\ & \text{linorder-trm.is-least-in-fset } \{|A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \\ & \quad \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A \implies \\ & \text{linorder-cls.is-least-in-fset } \{|C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \\ & \quad \text{clause-could-propagate } \Gamma C (Pos A)\} C \implies \\ & \Gamma' = (Pos A, None) \# \Gamma \implies \\ & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma' C) \implies \\ & \mathcal{F}' = (\text{if linorder-lit.is-greatest-in-mset } C (Pos A) \text{ then } \mathcal{F} \text{ else finsert } C \mathcal{F}) \implies \\ & \text{ord-res-11 } N (U_{er}, \mathcal{F}, \Gamma, None) (U_{er}, \mathcal{F}', \Gamma', None) \mid \end{aligned}$$

*propagate:*

$$\begin{aligned} & \neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma C) \implies \\ & \text{linorder-trm.is-least-in-fset } \{|A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \\ & \quad \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A \implies \\ & \text{linorder-cls.is-least-in-fset } \{|C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \\ & \quad \text{clause-could-propagate } \Gamma C (Pos A)\} C \implies \\ & \Gamma' = (Pos A, Some (efac C)) \# \Gamma \implies \\ & (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma' C) \implies \\ & \mathcal{F}' = (\text{if linorder-lit.is-greatest-in-mset } C (Pos A) \text{ then } \mathcal{F} \text{ else finsert } C \mathcal{F}) \implies \\ & \text{ord-res-11 } N (U_{er}, \mathcal{F}, \Gamma, None) (U_{er}, \mathcal{F}', \Gamma', None) \mid \end{aligned}$$

*conflict:*

$$\begin{aligned} & \text{linorder-cls.is-least-in-fset } \{|D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma D\} \\ & D \implies \\ & \text{ord-res-11 } N (U_{er}, \mathcal{F}, \Gamma, None) (U_{er}, \mathcal{F}, \Gamma, Some D) \mid \end{aligned}$$

*skip:*  $- L \notin \# C \implies$

$$\text{ord-res-11 } N (U_{er}, \mathcal{F}, (L, n) \# \Gamma, Some C) (U_{er}, \mathcal{F}, \Gamma, Some C) \mid$$

*resolution:*

$$\begin{aligned} & \Gamma = (L, Some D) \# \Gamma' \implies - L \in \# C \implies \\ & \text{ord-res-11 } N (U_{er}, \mathcal{F}, \Gamma, Some C) (U_{er}, \mathcal{F}, \Gamma, Some ((C - \{\#- L\# \}) + (D \\ & - \{\#L\# \}))) \mid \end{aligned}$$

*backtrack:*

$$\begin{aligned} & \Gamma = (L, None) \# \Gamma' \implies - L \in \# C \implies \\ & \text{ord-res-11 } N (U_{er}, \mathcal{F}, \Gamma, Some C) (\text{finsert } C U_{er}, \mathcal{F}, \Gamma', None) \end{aligned}$$

**lemma** *right-unique-ord-res-11:*

**fixes**  $N :: 'f \text{ gclause fset}$

**shows** *right-unique (ord-res-11 N)*

**proof** (*rule right-uniqueI*)

**fix**  $x y z$

**assume** *step1: ord-res-11 N x y and step2: ord-res-11 N x z*

**show**  $y = z$

**using** *step1*

```

proof (cases N x y rule: ord-res-11.cases)
  case hyps1: (decide-neg  $\mathcal{F}$   $U_{er}$   $\Gamma$  A1  $\Gamma 1'$ )
  show ?thesis
    using step2 unfolding  $\langle x = - \rangle$ 
  proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , None :: 'f gclause option) z rule: ord-res-11.cases)
    case (decide-neg A  $\Gamma'$ )
    with hyps1 show ?thesis
      by (metis (no-types, lifting) linorder-trm.dual-order.asym
        linorder-trm.is-least-in-fset-iff)
  next
    case (decide-pos A C  $\Gamma'$   $\mathcal{F}'$ )
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (propagate A C  $\Gamma'$   $\mathcal{F}'$ )
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (conflict D)
    with hyps1 have False
      by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
  qed
next
  case hyps1: (decide-pos  $\mathcal{F}$   $U_{er}$   $\Gamma$  A1 C1  $\Gamma 1'$   $\mathcal{F} 1'$ )
  show ?thesis
    using step2 unfolding  $\langle x = - \rangle$ 
  proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , None :: 'f gclause option) z rule: ord-res-11.cases)
    case (decide-neg A  $\Gamma'$ )
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (decide-pos A C  $\Gamma'$   $\mathcal{F}'$ )
    with hyps1 show ?thesis
      by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
        linorder-trm.Uniq-is-least-in-fset the1-equality)
  next
    case hyps2: (propagate A C  $\Gamma'$   $\mathcal{F}'$ )
    have A1 = A
      using  $\langle$ linorder-trm.is-least-in-fset - A1 $\rangle$   $\langle$ linorder-trm.is-least-in-fset - A $\rangle$ 
      by (metis (no-types) linorder-trm.Uniq-is-least-in-fset Uniq-D)
    hence trail-false-cls  $\Gamma 1' =$  trail-false-cls  $\Gamma'$ 
    unfolding  $\langle \Gamma 1' = - \# \Gamma \rangle$   $\langle \Gamma' = - \# \Gamma \rangle$ 

```

```

    unfolding trail-false-cls-def trail-false-lit-def
    by simp
  hence False
    using hyps1 hyps2 by argo
  thus ?thesis ..
next
  case (conflict D)
  with hyps1 have False
    by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
  thus ?thesis ..
qed
next
  case hyps1: (propagate  $\mathcal{F}$   $U_{er}$   $\Gamma$   $A1$   $C1$   $\Gamma1'$   $\mathcal{F}1'$ )
  show ?thesis
    using step2 unfolding  $\langle x = - \rangle$ 
  proof (cases  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ,  $None$  :: 'f gclause option) z rule: ord-res-11.cases)
    case (decide-neg  $A$   $\Gamma'$ )
    with hyps1 have False
      by (metis (no-types, lifting) linorder-cls.is-least-in-ffilter-iff
        linorder-trm.dual-order.asym linorder-trm.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case hyps2: (decide-pos  $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )
    have  $A1 = A$ 
      using  $\langle \text{linorder-trm.is-least-in-fset} - A1 \rangle \langle \text{linorder-trm.is-least-in-fset} - A \rangle$ 
      by (metis (no-types) linorder-trm.Uniq-is-least-in-fset Uniq-D)
    hence trail-false-cls  $\Gamma1' = \text{trail-false-cls } \Gamma'$ 
      unfolding  $\langle \Gamma1' = - \# \Gamma \rangle \langle \Gamma' = - \# \Gamma \rangle$ 
      unfolding trail-false-cls-def trail-false-lit-def
      by simp
    hence False
      using hyps1 hyps2 by argo
    thus ?thesis ..
  next
    case (propagate  $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )
    with hyps1 show ?thesis
      by (metis (no-types, lifting) linorder-cls.Uniq-is-least-in-fset
        linorder-trm.Uniq-is-least-in-fset the1-equality)
  next
    case (conflict  $D$ )
    with hyps1 have False
      by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
  qed
next
  case hyps1: (conflict  $\Gamma$   $\mathcal{F}$   $U_{er}$   $D1$ )
  show ?thesis
    using step2 unfolding  $\langle x = - \rangle$ 
  proof (cases  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ,  $None$  :: 'f gclause option) z rule: ord-res-11.cases)

```

```

    case (decide-neg A  $\Gamma'$ )
    with hyps1 have False
      by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (decide-pos A C  $\Gamma'$   $\mathcal{F}'$ )
    with hyps1 have False
      by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (propagate A C  $\Gamma'$   $\mathcal{F}'$ )
    with hyps1 have False
      by (metis (no-types) linorder-cls.is-least-in-ffilter-iff)
    thus ?thesis ..
  next
    case (conflict D)
    with hyps1 show ?thesis
      by (metis (no-types) linorder-cls.Uniq-is-least-in-fset Uniq-D)
  qed
next
  case hyps1: (skip L C  $U_{er}$   $\mathcal{F}$  n  $\Gamma$ )
  show ?thesis
    using step2 unfolding  $\langle x = \rightarrow$ 
  proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ , (L, n)  $\#$   $\Gamma$ , Some C) z rule: ord-res-11.cases)
    case skip
    with hyps1 show ?thesis
      by argo
  next
    case (resolution L' D'  $\Gamma'$ )
    with hyps1 have False
      by simp
    thus ?thesis ..
  next
    case (backtrack K  $\Gamma'$ )
    with hyps1 have False
      by simp
    thus ?thesis ..
  qed
next
  case hyps1: (resolution  $\Gamma$  L1 D1  $\Gamma1'$  C  $U_{er}$   $\mathcal{F}$ )
  show ?thesis
    using step2 unfolding  $\langle x = \rightarrow$ 
  proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , Some C) z rule: ord-res-11.cases)
    case (skip L n  $\Gamma$ )
    with hyps1 have False
      by simp
    thus ?thesis ..
  next
    case (resolution L D  $\Gamma'$ )

```



```

    with hyps1 show ?thesis
      by simp
  next
    case (backtrack K  $\Gamma'$ )
    with hyps1 have False
      by simp
    thus ?thesis ..
  qed
next
case hyps1: (backtrack  $\Gamma$  L  $\Gamma'$  C  $U_{er}$   $\mathcal{F}$ )
show ?thesis
  using step2 unfolding  $\langle x = \rightarrow \rangle$ 
proof (cases N ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , Some C) z rule: ord-res-11.cases)
  case (skip L n  $\Gamma$ )
  with hyps1 have False
    by simp
  thus ?thesis ..
next
case (resolution L D  $\Gamma'$ )
with hyps1 have False
  by simp
thus ?thesis ..
next
case (backtrack K  $\Gamma'$ )
with hyps1 show ?thesis
  by simp
qed
qed
qed

```

**inductive** *ord-res-11-final* :: '*f* *ord-res-11-state*  $\Rightarrow$  *bool* **where**  
*model-found*:  
 $\neg (\exists A \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er}). A \mid \notin \mid \text{trail-atms } \Gamma) \Longrightarrow$   
 $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \mid C) \Longrightarrow$   
*ord-res-11-final* (*N*,  $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ , *None*) |

*contradiction-found*:  
*ord-res-11-final* (*N*,  $U_{er}$ ,  $\mathcal{F}$ , [], *Some* {#})

**sublocale** *ord-res-11-antics: semantics* **where**  
*step* = *constant-context ord-res-11* **and**  
*final* = *ord-res-11-final*  
**proof** *unfold-locales*  
**fix** *S* :: '*f* *ord-res-11-state*

**assume** *ord-res-11-final* *S*  
**thus** *finished* (*constant-context ord-res-11*) *S*  
**proof** (*cases* *S* rule: *ord-res-11-final.cases*)

**case** (*model-found*  $N U_{er} \Gamma \mathcal{F}$ )

**have**  $\nexists A$ . *linorder-trm.is-least-in-fset*  $\{|A_2| \in | \text{atms-of-clss} (N \cup U_{er})$ .  
 $\forall A_1 | \in | \text{trail-atms} \Gamma. A_1 \prec_t A_2\}$   $A$   
**using**  $\langle \neg (\exists A | \in | \text{atms-of-clss} (N \cup U_{er}). A | \notin | \text{trail-atms} \Gamma) \rangle$   
**unfolding** *linorder-trm.is-least-in-filter-iff*  
**by** *blast*

**moreover have**  $\nexists D$ . *linorder-cls.is-least-in-fset*  
*(ffilter (trail-false-cls  $\Gamma$ ) (iefac  $\mathcal{F} \mid \uparrow (N \cup U_{er}))$ )  $D$*   
**using**  $\langle \neg (\exists C | \in | \text{iefac} \mathcal{F} \mid \uparrow (N \cup U_{er}). \text{trail-false-cls} \Gamma C) \rangle$   
**unfolding** *linorder-cls.is-least-in-filter-iff*  
**by** *metis*

**ultimately have**  $\nexists x$ . *ord-res-11*  $N (U_{er}, \mathcal{F}, \Gamma, \text{None}) x$   
**by** (*auto elim: ord-res-11.cases*)

**thus** *?thesis*  
**by** (*simp add:  $\langle S = \rightarrow \text{finished-def constant-context.simps} \rangle$* )

**next**

**case** (*contradiction-found*  $N U_{er} \mathcal{F}$ )

**hence**  $\nexists x$ . *ord-res-11*  $N (U_{er}, \mathcal{F}, [], \text{Some } \{\#\}) x$   
**by** (*auto elim: ord-res-11.cases*)

**thus** *?thesis*  
**by** (*simp add:  $\langle S = \rightarrow \text{finished-def constant-context.simps} \rangle$* )

**qed**  
**qed**

**inductive** *ord-res-11-invars* **where**  
*ord-res-11-invars*  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$  **if**  
*ord-res-10-invars*  $N (U_{er}, \mathcal{F}, \Gamma)$  **and**  
 $\{\#\} | \in | N \cup U_{er} \longrightarrow \Gamma = []$  **and**  
 $\forall C. \mathcal{C} = \text{Some } C \longrightarrow \text{atms-of-cls } C | \subseteq | \text{atms-of-clss} (N \cup U_{er})$  **and**  
 $\forall C. \mathcal{C} = \text{Some } C \longrightarrow \text{trail-false-cls} \Gamma C$  **and**  
 $\text{atms-of-clss } U_{er} | \subseteq | \text{atms-of-clss } N$  **and**  
 $\forall C | \in | \mathcal{F}. \exists L. \text{is-pos } L \wedge \text{linorder-lit.is-maximal-in-mset } C L$

**lemma** *ord-res-11-invars-initial-state*: *ord-res-11-invars*  $N (\{\|\}, \{\|\}, [], \text{None})$   
**by** (*intro ord-res-11-invars.intros ord-res-10-invars.intros simp-all*)

**lemma** *mempty-in-fimage-iefac[simp]*:  $\{\#\} | \in | \text{iefac} \mathcal{F} \mid \uparrow N \longleftrightarrow \{\#\} | \in | N$   
**using** *iefac-def by auto*

**lemma** *ord-res-11-preserves-invars*:  
**assumes**  
*step: ord-res-11*  $N s s'$  **and**  
*invars: ord-res-11-invars*  $N s$   
**shows** *ord-res-11-invars*  $N s'$   
**using** *invars*

**proof** (cases  $N$  s rule: ord-res-11-invars.cases)  
**case** more-invars: (1  $U_{er}$   $\mathcal{F}$   $\Gamma$   $\mathcal{C}$ )  
**show** ?thesis  
**using**  $\langle$ ord-res-10-invars  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ) $\rangle$   
**proof** (cases  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ) rule: ord-res-10-invars.cases)  
**case** invars: 1  
  
**note**  $\Gamma$ -sorted =  $\langle$ sorted-wrt ( $\lambda x y.$  atm-of (fst  $y$ )  $\prec_t$  atm-of (fst  $x$ ))  $\Gamma$  $\rangle$   
**note**  $\Gamma$ -lower =  $\langle$ linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-clss ( $N \cup U_{er}$ )) $\rangle$   
  
**have** trail-consistent  $\Gamma$   
**using**  $\Gamma$ -sorted trail-consistent-if-sorted-wrt-atoms **by** metis  
  
**show** ?thesis  
**using** step unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle$   
**proof** (cases  $N$  ( $U_{er}$ ,  $\mathcal{F}$ ,  $\Gamma$ ,  $\mathcal{C}$ )  $s'$  rule: ord-res-11.cases)  
**case** step-hyps: (decide-neg  $A$   $\Gamma'$ )  
  
**have**  
 $A$ -in:  $A \in |$  atms-of-clss ( $N \cup U_{er}$ ) **and**  
 $A$ -gt:  $\forall A_1 \in |$  trail-atms  $\Gamma.$   $A_1 \prec_t A$  **and**  
 $A$ -lt:  $\forall y \in |$  atms-of-clss ( $N \cup U_{er}$ ).  
 $y \neq A \longrightarrow (\forall A_1 \in |$  trail-atms  $\Gamma.$   $A_1 \prec_t y) \longrightarrow A \prec_t y$   
**using**  $\langle$ linorder-trm.is-least-in-fset -  $A$  $\rangle$   
**unfolding** atomize-conj linorder-trm.is-least-in-filter-iff  
**by** argo  
  
**have** trail-atms  $\Gamma' =$  fininsert  $A$  (trail-atms  $\Gamma$ )  
**unfolding**  $\langle \Gamma' = (\text{Neg } A, -) \# \Gamma \rangle$  **by** simp  
  
**show** ?thesis  
**unfolding**  $\langle s' = (-, -, -) \rangle$   
**proof** (intro ord-res-11-invars.intros ord-res-10-invars.intros allI impI conjI)  
**show** sorted-wrt ( $\lambda x y.$  atm-of (fst  $y$ )  $\prec_t$  atm-of (fst  $x$ ))  $\Gamma'$   
**unfolding**  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$  sorted-wrt.simps  
**proof** (intro conjI ballI)  
**fix**  $Ln :: 'f$  gliteral  $\times 'f$  gclause option  
**assume**  $Ln \in$  set  $\Gamma$   
  
**hence** atm-of (fst  $Ln$ )  $\in |$  trail-atms  $\Gamma$   
**by** (simp add: fset-trail-atms)  
  
**thus** atm-of (fst  $Ln$ )  $\prec_t$  atm-of (fst ( $\text{Neg } A, \text{None}$ ))  
**unfolding** prod.sel literal.sel  
**using**  $A$ -gt **by** metis  
**next**  
**show** sorted-wrt ( $\lambda x y.$  atm-of (fst  $y$ )  $\prec_t$  atm-of (fst  $x$ ))  $\Gamma$   
**using**  $\Gamma$ -sorted .

```

qed

show  $\forall Ln \in \text{set } \Gamma'. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{ord-res.is-strictly-maximal-lit}$ 
(fst Ln) C
  unfolding  $\langle \Gamma' = (-, \text{None}) \# \Gamma \rangle$ 
  using invars by simp

show linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-cls (N | $\cup$ |  $U_{er}$ ))
  unfolding  $\langle \text{trail-atms } \Gamma' = \text{finsert } A (\text{trail-atms } \Gamma) \rangle \text{finsert.rep-eq}$ 
proof (intro linorder-trm.is-lower-set-insertI ballI impI)
  show  $A \in | \text{atms-of-cls } (N | \cup | U_{er})$ 
    using A-in .
next
  fix  $w :: 'f \text{ gterm}$ 
  assume  $w \in | \text{atms-of-cls } (N | \cup | U_{er})$  and  $w \prec_t A$ 
  thus  $w \in | \text{trail-atms } \Gamma$ 
    by (metis A-lt  $\Gamma$ -lower linorder-trm.dual-order.asym linorder-trm.neq-iff
      linorder-trm.not-in-lower-setI)
next
  show linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-cls (N | $\cup$ |  $U_{er}$ ))
    using  $\Gamma$ -lower .
qed

{
  fix
     $Ln :: 'f \text{ gliteral} \times 'f \text{ gclause option}$  and
     $\Gamma'' :: ('f \text{ gliteral} \times 'f \text{ gclause option}) \text{ list}$ 

  assume  $\Gamma' = Ln \# \Gamma''$ 

  have  $\text{snd } Ln = \text{None}$ 
    using  $\langle \Gamma' = Ln \# \Gamma'' \rangle \text{step-hyps by auto}$ 

  moreover have  $\neg (\exists C \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}). \text{trail-false-cls } ((\text{Neg } A,$ 
None)  $\# \Gamma) C)$ 
  proof (rule notI , elim bexE)
    fix  $C :: 'f \text{ gclause}$ 
    assume
       $C\text{-in}: C \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er})$  and
       $C\text{-false}: \text{trail-false-cls } ((\text{Neg } A, \text{None}) \# \Gamma) C$ 

    have clause-could-propagate  $\Gamma C (\text{Pos } A)$ 
      unfolding clause-could-propagate-def
    proof (intro conjI)
      show  $\neg \text{trail-defined-lit } \Gamma (\text{Pos } A)$ 
        unfolding trail-defined-lit-iff-trail-defined-atm literal.sel
        by (metis A-gt linorder-trm.less-irrefl)
      next
      show ord-res.is-maximal-lit (Pos A) C

```

```

unfolding linorder-lit.is-maximal-in-mset-iff
proof (intro conjI ballI impI)
  have  $\neg$  trail-false-cls  $\Gamma$  C
    using step-hyps C-in by metis

  thus Pos A  $\in\#$  C
    using C-false by (metis subtrail-falseI uminus-Neg)
next

  fix L :: 'f gliteral
  assume L-in: L  $\in\#$  C and L-neq: L  $\neq$  Pos A

  have trail-false-lit ((Neg A, None)  $\#$   $\Gamma$ ) L
    using C-false L-in unfolding trail-false-cls-def by metis

  hence  $\neg$  L  $\in$  fst ' set  $\Gamma$ 
    unfolding trail-false-lit-def
    using L-neq
    by (cases L) simp-all

  hence trail-defined-lit  $\Gamma$  L
    unfolding trail-defined-lit-def by argo

  hence atm-of L  $\in$  trail-atms  $\Gamma$ 
    unfolding trail-defined-lit-iff-trail-defined-atm .

  moreover have  $\forall A_1 \in$  trail-atms  $\Gamma$ . A1  $\prec_t$  A
    using step-hyps unfolding linorder-trm.is-least-in-filter-iff by argo

  ultimately have atm-of L  $\prec_t$  A
    by metis

  hence L  $\preceq_i$  Pos A
    by (cases L) simp-all

  thus  $\neg$  Pos A  $\prec_i$  L
    by order
qed
next
  show trail-false-cls  $\Gamma$  { $\#K \in\#$  C. K  $\neq$  Pos A $\#$ }
    using C-false
    unfolding trail-false-cls-def trail-false-lit-def
    by (smt (verit, ccfv-SIG) mem-Collect-eq set-mset-filter subtrail-falseI
      trail-false-cls-def trail-false-lit-def uminus-Neg)
qed

  moreover have  $\neg$  clause-could-propagate  $\Gamma$  C (Pos A)
    using C-in step-hyps by metis

```

```

    ultimately show False
      by contradiction
  qed

  ultimately show (snd Ln ≠ None) = (∃ C |∈| iefac  $\mathcal{F}$  |!| (N |∪| Uer)).
trail-false-cls Γ' C)
    unfolding ⟨Γ' = (Neg A, None) # Γ⟩ by argo

  show snd Ln ≠ None ⇒ is-pos (fst Ln)
    using ⟨snd Ln = None⟩ by argo

  have ¬ fBex (iefac  $\mathcal{F}$  |!| (N |∪| Uer)) (trail-false-cls Γ)
    using step-hyps by argo

  hence ∀ x ∈ set Γ. snd x = None
    using invars by (metis list.set-cases)

  thus ∀ x ∈ set Γ''. snd x = None
    using ⟨Γ' = Ln # Γ''⟩ ⟨Γ' = (Neg A, None) # Γ⟩ by simp

  show ∧ C. snd Ln = Some C ⇒ clause-could-propagate Γ'' C (fst Ln)
    using ⟨Γ' = Ln # Γ''⟩ ⟨Γ' = (Neg A, None) # Γ⟩ by force

  show ∧ D. snd Ln = Some D ⇒ D |∈| iefac  $\mathcal{F}$  |!| (N |∪| Uer)
    using ⟨Γ' = Ln # Γ''⟩ ⟨Γ' = (Neg A, None) # Γ⟩ by force
}

C) have ¬ (∃ C |∈| iefac  $\mathcal{F}$  |!| (N |∪| Uer)). trail-false-cls ((Neg A, None) # Γ)
proof (intro notI , elim bexE)
  fix C :: 'f gclause
  assume
    C-in: C |∈| iefac  $\mathcal{F}$  |!| (N |∪| Uer) and
    C-false: trail-false-cls ((Neg A, None) # Γ) C

  have clause-could-propagate Γ C (Pos A)
    unfolding clause-could-propagate-def
  proof (intro conjI)
    show ¬ trail-defined-lit Γ (Pos A)
      unfolding trail-defined-lit-iff-trail-defined-atm
      by (metis A-gt linorder-trm.less-irrefl literal.sel(1))
    next
    have ¬ trail-false-cls Γ C
      using C-in step-hyps by metis

  thus trail-false-cls Γ {#K ∈# C. K ≠ Pos A#}
    by (smt (verit) C-false mem-Collect-eq set-mset-filter subtrail-falseI
      trail-false-cls-def uminus-Neg)

```

```

show ord-res.is-maximal-lit (Pos A) C
  unfolding linorder-lit.is-maximal-in-mset-iff
proof (intro conjI ballI impI)
  show Pos A ∈# C
    by (metis C-false ⟨¬ trail-false-cls Γ C⟩ subtrail-falseI uminus-Neg)
next
  fix L
  assume L ∈# C and L ≠ Pos A
  hence atm-of L |∈| trail-atms Γ
    using ⟨trail-false-cls Γ {#K ∈# C. K ≠ Pos A#}⟩
using trail-defined-lit-iff-trail-defined-atm trail-defined-lit-iff-true-or-false
  trail-false-cls-filter-mset-iff by blast
  hence atm-of L ≺t A
    using A-gt by metis
  hence L ≺l Pos A
    by (cases L) simp-all
  thus ¬ Pos A ≺l L
    by order
  qed
qed

moreover have ¬ clause-could-propagate Γ C (Pos A)
  using C-in-step-hyps by metis

ultimately show False
  by contradiction
qed

thus  $\bigwedge \Gamma_1 Ln \Gamma_0. \Gamma' = \Gamma_1 @ Ln \# \Gamma_0 \implies \text{snd } Ln = \text{None} \implies$ 
 $\neg (\exists C | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}). \text{trail-false-cls } (Ln \# \Gamma_0) C)$ 
  unfolding ⟨Γ' = (-, None) # Γ⟩
  by (metis suffixI trail-false-cls-if-trail-false-suffix)

show {#} |∈| N |∪| Uer  $\implies \Gamma' = []$ 
using bex-trail-false-cls-simp more-invars(3) step-hyps(3) trail-false-cls-mempty
by blast
next
show  $\bigwedge C. \text{None} = \text{Some } C \implies \text{atms-of-cls } C | \subseteq | \text{atms-of-cls } (N | \cup | U_{er})$ 
  by simp
next
show  $\bigwedge C. \text{None} = \text{Some } C \implies \text{trail-false-cls } \Gamma' C$ 
  by simp
next
show atms-of-cls Uer |⊆| atms-of-cls N
  using more-invars by argo
next
show  $\forall C | \in | \mathcal{F}. \exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L C$ 
  using more-invars by argo
qed

```

**next**

**case** *step-hyps*: (*decide-pos*  $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )

**have**

*A-in*:  $A \in | \text{atms-of-clss } (N \cup U_{er})$  **and**

*A-gt*:  $\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t A$  **and**

*A-lt*:  $\forall y \in | \text{atms-of-clss } (N \cup U_{er}).$

$y \neq A \longrightarrow (\forall A_1 \in | \text{trail-atms } \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$

**using**  $\langle \text{linorder-trm.is-least-in-fset} - A \rangle$

**unfolding** *atomize-conj linorder-trm.is-least-in-filter-iff*

**by** *argo*

**have** *trail-atms*  $\Gamma' = \text{finsert } A (\text{trail-atms } \Gamma)$

**unfolding**  $\langle \Gamma' = (\text{Pos } A, -) \# \Gamma \rangle$  **by** *simp*

**show** *?thesis*

**unfolding**  $\langle s' = (-, -, -) \rangle$

**proof** (*intro ord-res-11-invars.intros ord-res-10-invars.intros allI impI conjI*)

**show** *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma'$

**unfolding**  $\langle \Gamma' = (\text{Pos } A, \text{None}) \# \Gamma \rangle$  *sorted-wrt.simps*

**proof** (*intro conjI ballI*)

**fix**  $L_n :: 'f \text{ gliteral} \times 'f \text{ gclause option}$

**assume**  $L_n \in \text{set } \Gamma$

**hence** *atm-of*  $(\text{fst } L_n) \in | \text{trail-atms } \Gamma$

**by** (*simp add: fset-trail-atms*)

**thus** *atm-of*  $(\text{fst } L_n) \prec_t \text{atm-of } (\text{fst } (\text{Pos } A, \text{None}))$

**unfolding** *prod.sel literal.sel*

**using** *A-gt* **by** *metis*

**next**

**show** *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$

**using**  $\Gamma\text{-sorted}$  .

**qed**

**show**  $\forall L_n \in \text{set } \Gamma'. \forall C. \text{snd } L_n = \text{Some } C \longrightarrow \text{ord-res.is-strictly-maximal-lit}$   
 $(\text{fst } L_n) C$

**unfolding**  $\langle \Gamma' = (-, \text{None}) \# \Gamma \rangle$

**using** *invars* **by** *simp*

**show** *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma') (\text{atms-of-clss } (N \cup U_{er}))$

**unfolding**  $\langle \text{trail-atms } \Gamma' = \text{finsert } A (\text{trail-atms } \Gamma) \rangle$  *finsert.rep-eq*

**proof** (*intro linorder-trm.is-lower-set-insertI ballI impI*)

**show**  $A \in | \text{atms-of-clss } (N \cup U_{er})$

**using** *A-in* .

**next**

**fix**  $w :: 'f \text{ gterm}$

**assume**  $w \in | \text{atms-of-clss } (N \cup U_{er})$  **and**  $w \prec_t A$

**thus**  $w \in | \text{trail-atms } \Gamma$



```

    by (metis A-lt  $\Gamma$ -lower linorder-trm.dual-order.asym linorder-trm.neq-iff
        linorder-trm.not-in-lower-setI)
next
  show linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-cls (N  $\cup$   $U_{er}$ ))
    using  $\Gamma$ -lower .
qed

{
  fix Ln and  $\Gamma''$ 
  assume  $\Gamma' = Ln \# \Gamma''$ 

  have snd Ln = None
    using  $\langle \Gamma' = Ln \# \Gamma'' \rangle$  step-hyps by auto

  moreover have  $\neg (\exists C | \in | \text{iefac } \mathcal{F}' | \uparrow | (N \cup U_{er}). \text{trail-false-cls } ((\text{Pos } A, \text{None}) \# \Gamma) C)$ 
    proof (rule notI , elim bexE)
      fix D :: 'f gclause
      assume D-in: D  $| \in | \text{iefac } \mathcal{F}' | \uparrow | (N \cup U_{er})$ 

      hence D = efac C  $\vee$  D  $| \in | \text{iefac } \mathcal{F} | \uparrow | (N \cup U_{er})$ 
        unfolding  $\langle \mathcal{F}' = (\text{if ord-res.is-strictly-maximal-lit } (\text{Pos } A) C \text{ then } \mathcal{F} \text{ else } \text{finsert } C \mathcal{F}) \rangle$ 
        by (smt (z3) fimage-iff finsert-iff iefac-def)

      hence  $\neg \text{trail-false-cls } \Gamma' D$ 
        proof (elim disjE)
          assume D = efac C

          hence trail-false-cls  $\Gamma' D \longleftrightarrow \text{trail-false-cls } \Gamma' C$ 
            by (simp add: trail-false-cls-def)

          moreover have  $\neg \text{trail-false-cls } \Gamma' C$ 
            using step-hyps unfolding linorder-cls.is-least-in-ffilter-iff by metis

          ultimately show  $\neg \text{trail-false-cls } \Gamma' D$ 
            by argo
        next
          assume D  $| \in | \text{iefac } \mathcal{F} | \uparrow | (N \cup U_{er})$ 
          thus  $\neg \text{trail-false-cls } \Gamma' D$ 
            using step-hyps by metis
        qed

      moreover assume trail-false-cls ((Pos A, None)  $\#$   $\Gamma$ ) D

      ultimately show False
        unfolding  $\langle \Gamma' = (\text{Pos } A, \text{None}) \# \Gamma \rangle$ 
        by contradiction
    qed

```

```

ultimately show snd Ln ≠ None ↔
(∃ C|∈|iefac F' |↑ (N |∪| Uer). trail-false-cls Γ' C)
  unfolding ⟨Γ' = (Pos A, None) # Γ⟩ by argo

show snd Ln ≠ None ⇒ is-pos (fst Ln)
  using ⟨snd Ln = None⟩ by argo

have ¬ fBex (iefac F |↑ (N |∪| Uer)) (trail-false-cls Γ)
  using step-hyps by argo

hence ∀ x∈set Γ. snd x = None
  using invars by (metis list.set-cases)

thus ∀ x∈set Γ''. snd x = None
  using ⟨Γ' = Ln # Γ''⟩ ⟨Γ' = (Pos A, None) # Γ⟩ by simp

show ∧ C. snd Ln = Some C ⇒ clause-could-propagate Γ'' C (fst Ln)
  using ⟨Γ' = Ln # Γ''⟩ ⟨Γ' = (Pos A, None) # Γ⟩ by force

show ∧ D. snd Ln = Some D ⇒ D |∈| iefac F' |↑ (N |∪| Uer)
  using ⟨Γ' = Ln # Γ''⟩ ⟨Γ' = (Pos A, None) # Γ⟩ by force
}

show ∧ Γ1 Ln Γ0. Γ' = Γ1 @ Ln # Γ0 ⇒ snd Ln = None ⇒
¬ (∃ C|∈|iefac F' |↑ (N |∪| Uer). trail-false-cls (Ln # Γ0) C)
  using step-hyps
  unfolding bex-trail-false-cls-simp
  by (meson suffixI trail-false-cls-if-trail-false-suffix)

show {#} |∈| N |∪| Uer ⇒ Γ' = []
  by (meson bex-trail-false-cls-simp step-hyps(3) trail-false-cls-empty)
next
show ∧ C. None = Some C ⇒ atms-of-cls C |⊆| atms-of-cls (N |∪| Uer)
  by simp
next
show ∧ C. None = Some C ⇒ trail-false-cls Γ' C
  by simp
next
show atms-of-cls Uer |⊆| atms-of-cls N
  using more-invars by argo
next
have ∃ L. is-pos L ∧ ord-res.is-maximal-lit L C
  using step-hyps
  by (metis (no-types, lifting) clause-could-propagate-def
    linorder-cls.is-least-in-filter-iff literal.discI(1))

thus ∀ C|∈|F'. ∃ L. is-pos L ∧ ord-res.is-maximal-lit L C
  using more-invars ⟨F' = (if - then F else finset C F)⟩ by simp

```

```

qed
next
case step-hyps: (propagate  $A$   $C$   $\Gamma'$   $\mathcal{F}'$ )

have
  A-in:  $A \in |atms-of-clss (N \cup U_{er})$  and
  A-gt:  $\forall A_1 \in |trail-atms \Gamma. A_1 \prec_t A$  and
  A-lt:  $\forall y \in |atms-of-clss (N \cup U_{er}).$ 
   $y \neq A \longrightarrow (\forall A_1 \in |trail-atms \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$ 
using  $\langle linorder-trm.is-least-in-fset - A \rangle$ 
unfolding atomize-conj linorder-trm.is-least-in-ffilter-iff
by argo

have
  C-in:  $C \in |iefac \mathcal{F} |' (N \cup U_{er})$  and
  C-prop: clause-could-propagate  $\Gamma$   $C$  (Pos  $A$ ) and
  C-lt:  $\forall D \in |iefac \mathcal{F} |' (N \cup U_{er}).$ 
   $D \neq C \longrightarrow clause-could-propagate \Gamma D (Pos A) \longrightarrow C \prec_c D$ 
using  $\langle linorder-cls.is-least-in-fset - C \rangle$ 
unfolding atomize-conj linorder-cls.is-least-in-ffilter-iff by argo

have trail-atms  $\Gamma' = finsert A (trail-atms \Gamma)$ 
unfolding  $\langle \Gamma' = (Pos A, -) \# \Gamma \rangle$  by simp

show ?thesis
unfolding  $\langle s' = (-, -, -) \rangle$ 
proof (intro ord-res-11-invars.intros ord-res-10-invars.intros allI impI conjI)
show sorted-wrt  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma'$ 
unfolding  $\langle \Gamma' = (Pos A, -) \# \Gamma \rangle$  sorted-wrt.simps
proof (intro conjI ballI)
  fix  $Ln :: 'f gliteral \times 'f gclause option$ 
  assume  $Ln \in set \Gamma$ 

  hence atm-of  $(fst Ln) \in |trail-atms \Gamma$ 
  by (simp add: fset-trail-atms)

  thus atm-of  $(fst Ln) \prec_t atm-of (fst (Pos A, Some (efac C)))$ 
  unfolding prod.sel literal.sel
  using A-gt by metis
next
  show sorted-wrt  $(\lambda x y. atm-of (fst y) \prec_t atm-of (fst x)) \Gamma$ 
  using  $\Gamma$ -sorted .
qed

show  $\forall Ln \in set \Gamma'. \forall C. snd Ln = Some C \longrightarrow ord-res.is-strictly-maximal-lit$ 
(fst Ln)  $C$ 
unfolding  $\langle \Gamma' = (Pos A, Some (efac C)) \# \Gamma \rangle$ 
using invars step-hyps
by (simp add: clause-could-propagate-def greatest-literal-in-efacI)

```

```

linorder-cls.is-least-in-filter-iff)

show linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-cls (N | $\cup$ |  $U_{er}$ ))
  unfolding  $\langle$ trail-atms  $\Gamma' = \text{finsert } A \text{ (trail-atms } \Gamma)\rangle \text{finsert.rep-eq}$ 
proof (intro linorder-trm.is-lower-set-insertI ballI impI)
  show  $A \in | \text{atms-of-cls } (N | \cup | U_{er})$ 
    using  $A\text{-in}$  .
next
fix  $w :: 'f \text{ gterm}$ 
assume  $w \in | \text{atms-of-cls } (N | \cup | U_{er})$  and  $w \prec_t A$ 
thus  $w \in | \text{trail-atms } \Gamma$ 
  by (metis  $A\text{-lt } \Gamma\text{-lower linorder-trm.dual-order.asym linorder-trm.neq-iff}$ 
linorder-trm.not-in-lower-setI)
next
show linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-cls (N | $\cup$ |  $U_{er}$ ))
  using  $\Gamma\text{-lower}$  .
qed

{
fix  $L_n$  and  $\Gamma''$ 
assume  $\Gamma' = L_n \# \Gamma''$ 
hence  $L_n = (\text{Pos } A, \text{Some } (\text{efac } C))$  and  $\Gamma'' = \Gamma$ 
  using  $\langle \Gamma' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma \rangle$  by simp-all

  obtain  $D$  where  $D\text{-in}: D \in | \text{iefac } \mathcal{F} \uparrow | (N | \cup | U_{er})$  and  $D\text{-false}: \text{trail-false-cls } \Gamma' D$ 
  using step-hyps by metis

have  $(\exists C \in | \text{iefac } \mathcal{F}' \uparrow | (N | \cup | U_{er}). \text{trail-false-cls } \Gamma' C)$ 
proof (rule beXI)
  show  $\text{trail-false-cls } \Gamma' D$ 
    using  $D\text{-false}$  .
next
have  $\neg \text{trail-false-lit } \Gamma' (\text{Pos } A)$ 
by (metis  $C\text{-prop map-of-Cons-code}(2) \text{map-of-eq-None-iff clause-could-propagate-def}$ 
step-hyps}(6) \text{trail-defined-lit-def trail-false-lit-def uminus-not-id})

moreover have  $\text{Pos } A \in \# C$ 
  using  $C\text{-prop}$ 
  unfolding clause-could-propagate-def linorder-lit.is-maximal-in-mset-iff
  by argo

ultimately have  $\neg \text{trail-false-cls } \Gamma' C$ 
  unfolding trail-false-cls-def by metis

hence  $D \neq C$ 
  using  $D\text{-false}$  by metis

thus  $D \in | \text{iefac } \mathcal{F}' \uparrow | (N | \cup | U_{er})$ 

```

```

      unfolding ⟨ $\mathcal{F}' = (\text{if ord-res.is-strictly-maximal-lit } (Pos A) C \text{ then } \mathcal{F}$ 
else  $\text{finsert } C \mathcal{F})$ ⟩
      using  $D\text{-in iefac-def by auto}$ 
    qed

  moreover have  $\text{snd } Ln \neq None$ 
    using ⟨ $\Gamma' = Ln \# \Gamma''$ ⟩  $\text{step-hyps by auto}$ 

  ultimately show  $\text{snd } Ln \neq None \longleftrightarrow$ 
    ( $\exists C | \in | \text{iefac } \mathcal{F}' | \uparrow (N \cup U_{er}). \text{trail-false-cls } \Gamma' C$ )
    by  $\text{argo}$ 

  show  $\text{snd } Ln \neq None \implies \text{is-pos } (\text{fst } Ln)$ 
    using ⟨ $Ln = (Pos A, \text{Some } (efac C))$ ⟩  $\text{by auto}$ 

  show  $\forall x \in \text{set } \Gamma''. \text{snd } x = None$ 
    unfolding ⟨ $\Gamma'' = \Gamma$ ⟩
    using  $\text{invars by (meson list.set-cases step-hyps)}$ 

  have  $\text{clause-could-propagate } \Gamma (efac C) (Pos A)$ 
    using  $C\text{-prop clause-could-propagate-efac by metis}$ 

  thus  $\bigwedge C. \text{snd } Ln = \text{Some } C \implies \text{clause-could-propagate } \Gamma'' C (\text{fst } Ln)$ 
    using ⟨ $\Gamma' = Ln \# \Gamma''$ ⟩ ⟨ $\Gamma' = (Pos A, \text{Some } (efac C)) \# \Gamma$ ⟩
    by  $\text{force}$ 

  have  $efac C | \in | \text{iefac } \mathcal{F}' | \uparrow (N \cup U_{er})$ 
  proof (cases  $\text{ord-res.is-strictly-maximal-lit } (Pos A) C$ )
    case  $True$ 
      thus  $?thesis$ 
        unfolding ⟨ $\mathcal{F}' = -$ ⟩
        using  $C\text{-in}$ 
        by (metis (mono-tags, opaque-lifting)  $\text{literal.discI}(1)$ 
 $\text{nex-strictly-maximal-pos-lit-if-neq-efac}$ )
    next
      case  $False$ 
        then show  $?thesis$ 
          unfolding ⟨ $\mathcal{F}' = -$ ⟩
          using  $C\text{-in}$ 
        by (smt (z3)  $\text{fimage-iff finsert-iff iefac-def nex-strictly-maximal-pos-lit-if-neq-efac}$ 
 $\text{obtains-positive-greatest-lit-if-efac-not-ident}$ )
  qed

  thus  $\bigwedge D. \text{snd } Ln = \text{Some } D \implies D | \in | \text{iefac } \mathcal{F}' | \uparrow (N \cup U_{er})$ 
    using ⟨ $\Gamma' = Ln \# \Gamma''$ ⟩ ⟨ $\Gamma' = (Pos A, \text{Some } (efac C)) \# \Gamma$ ⟩  $\text{by force}$ 
}

show  $\bigwedge \Gamma_1 Ln \Gamma_0. \Gamma' = \Gamma_1 @ Ln \# \Gamma_0 \implies \text{snd } Ln = None \implies$ 
 $\neg (\exists C | \in | \text{iefac } \mathcal{F}' | \uparrow (N \cup U_{er}). \text{trail-false-cls } (Ln \# \Gamma_0) C)$ 

```

```

unfolding ⟨ $\Gamma' = (-, \text{Some } -) \# \Gamma$ ⟩
using invars
unfolding be-trail-false-cls-simp
by (metis list.inject not-None-eq split-pairs suffix-Cons suffix-def)

show {#} |∈|  $N \mid \cup \mid U_{er} \implies \Gamma' = []$ 
by (meson be-trail-false-cls-simp step-hyps(3) trail-false-cls-mempty)
next
show  $\bigwedge C. \text{None} = \text{Some } C \implies \text{atms-of-cls } C \mid \subseteq \mid \text{atms-of-cls } (N \mid \cup \mid U_{er})$ 
by simp
next
show  $\bigwedge C. \text{None} = \text{Some } C \implies \text{trail-false-cls } \Gamma' C$ 
by simp
next
show atms-of-cls  $U_{er} \mid \subseteq \mid \text{atms-of-cls } N$ 
using more-invars by argo
next
have  $\exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L C$ 
using step-hyps
by (metis (no-types, lifting) clause-could-propagate-def
linorder-cls.is-least-in-ffilter-iff literal.discI(1))

thus  $\forall C \mid \in \mid \mathcal{F}'. \exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L C$ 
using more-invars ⟨if - then  $\mathcal{F}$  else  $\text{finsert } C \mathcal{F}$ ⟩ by simp
qed
next
case step-hyps: (conflict D)
show ?thesis
unfolding ⟨s' = -⟩
proof (intro ord-res-11-invars.intros allI impI)
show ord-res-10-invars  $N (U_{er}, \mathcal{F}, \Gamma)$ 
using more-invars by argo
next
show {#} |∈|  $N \mid \cup \mid U_{er} \implies \Gamma = []$ 
using more-invars by argo
next
show  $\bigwedge C. \text{Some } D = \text{Some } C \implies \text{atms-of-cls } C \mid \subseteq \mid \text{atms-of-cls } (N \mid \cup \mid U_{er})$ 
by (metis atm-of-in-atms-of-clsI atm-of-cls-def fimage-fsubsetI fset-fset-mset
linorder-cls.is-least-in-ffilter-iff option.inject step-hyps(3))
next
show  $\bigwedge C. \text{Some } D = \text{Some } C \implies \text{trail-false-cls } \Gamma C$ 
using linorder-cls.is-least-in-fset - D
unfolding linorder-cls.is-least-in-ffilter-iff
by simp
next
show atms-of-cls  $U_{er} \mid \subseteq \mid \text{atms-of-cls } N$ 
using more-invars by argo
next

```

```

    show  $\forall C \in |\mathcal{F}. \exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L \ C$ 
      using more-invars by argo
  qed
next
case step-hyps: (skip  $L \ C \ n \ \Gamma'$ )

  have  $\bigwedge xs. \text{fset } (\text{trail-atms } xs) = \text{atm-of } \langle \text{fst } \langle \text{set } xs$ 
    unfolding fset-trail-atms ..
  also have  $\bigwedge xs. \text{atm-of } \langle \text{fst } \langle \text{set } xs = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ xs)$ 
    by (simp add: image-comp)
  finally have  $\bigwedge xs. \text{fset } (\text{trail-atms } xs) = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ xs) .$ 

  show ?thesis
    unfolding  $\langle s' = (U_{er}, \mathcal{F}, \Gamma', \text{Some } C) \rangle$ 
  proof (intro ord-res-11-invars.intros ord-res-10-invars.intros ballI allI impI
conjI)
    show sorted-wrt  $(\lambda x \ y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \ \Gamma'$ 
      using invars unfolding  $\langle \Gamma = (L, n) \# \Gamma' \rangle$  by simp
  next
  fix  $Ln :: 'f \text{ gliteral} \times 'f \text{ gclause option}$  and  $C :: 'f \text{ gclause}$ 
  assume  $Ln \in \text{set } \Gamma'$  and  $\text{snd } Ln = \text{Some } C$ 
  then show ord-res.is-strictly-maximal-lit  $(\text{fst } Ln) \ C$ 
    using invars unfolding  $\langle \Gamma = (L, n) \# \Gamma' \rangle$  by simp
  next
  show linorder-trm.is-lower-fset  $(\text{trail-atms } \Gamma') \ (\text{atms-of-cls } (N \cup U_{er}))$ 
    unfolding  $\langle \text{fset } (\text{trail-atms } \Gamma') = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma') \rangle$ 
  proof (rule linorder-trm.sorted-and-lower-set-appendD-right(2))
    have sorted-wrt  $(\lambda x \ y. y \prec_t x) \ (\text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma)$ 
      using  $\Gamma$ -sorted by (simp add: sorted-wrt-map)

    thus sorted-wrt  $(\lambda x \ y. y \prec_t x) \ ([\text{atm-of } L] \ @ \ \text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma')$ 
      unfolding  $\langle \Gamma = - \# \Gamma' \rangle$  by simp
  next
  have  $\text{set } ([\text{atm-of } L] \ @ \ \text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma') = \text{fset } (\text{trail-atms } \Gamma)$ 
    unfolding append-Cons append-Nil list.set
  unfolding  $\langle \text{fset } (\text{trail-atms } \Gamma') = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma') \rangle$  [symmetric]
  unfolding  $\langle \Gamma = - \# \Gamma' \rangle$  trail-atms.simps prod.sel
  by simp

  thus linorder-trm.is-lower-set  $(\text{set } ([\text{atm-of } L] \ @ \ \text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma'))$ 
     $(\text{fset } (\text{atms-of-cls } (N \cup U_{er})))$ 
    using  $\Gamma$ -lower
    by (simp only:)
  qed
next
fix
   $Ln :: 'f \text{ gliteral} \times 'f \text{ gclause option}$  and
   $\Gamma'' :: ('f \text{ gliteral} \times 'f \text{ gclause option}) \text{ list}$ 
  assume  $\Gamma' = Ln \# \Gamma''$ 

```

```

have snd Ln = None
  by (metis  $\langle \Gamma' = Ln \# \Gamma'' \rangle$  in-set-conv-decomp invars(4) step-hyps(1)
suffixE suffix-Cons)

show (snd Ln  $\neq$  None) =  $(\exists C | \in | \text{iefac } \mathcal{F} \ | \uparrow | (N \ | \cup | \ U_{er}). \text{trail-false-cls } \Gamma'$ 
C)
  using  $\langle \text{snd } Ln = None \rangle$ 
  by (metis  $\langle \Gamma' = Ln \# \Gamma'' \rangle$  invars(5) step-hyps(1) suffixE suffix-Cons)

show snd Ln  $\neq$  None  $\implies$  is-pos (fst Ln)
  using  $\langle \text{snd } Ln = None \rangle$  by simp

show  $\bigwedge C. \text{snd } Ln = \text{Some } C \implies \text{clause-could-propagate } \Gamma'' \ C \ (\text{fst } Ln)$ 
  using  $\langle \text{snd } Ln = None \rangle$  by simp

show  $\bigwedge C. \text{snd } Ln = \text{Some } C \implies C \ | \in | \text{iefac } \mathcal{F} \ | \uparrow | (N \ | \cup | \ U_{er})$ 
  using  $\langle \text{snd } Ln = None \rangle$  by simp

show  $\bigwedge x. x \in \text{set } \Gamma'' \implies \text{snd } x = None$ 
  by (simp add:  $\langle \Gamma' = Ln \# \Gamma'' \rangle$  invars(4) step-hyps(1))
next
fix
  Ln :: 'f gliteral  $\times$  'f gclause option and
   $\Gamma_1 \ \Gamma_0$  :: ('f gliteral  $\times$  'f gclause option) list
  assume  $\Gamma' = \Gamma_1 \ @ \ Ln \ \# \ \Gamma_0$  and snd Ln = None
  thus  $\neg (\exists C | \in | \text{iefac } \mathcal{F} \ | \uparrow | (N \ | \cup | \ U_{er}). \text{trail-false-cls } (Ln \ \# \ \Gamma_0) \ C)$ 
  by (metis append-Cons invars(5) step-hyps(1))
next
  show  $\{ \# \} \ | \in | \ N \ | \cup | \ U_{er} \implies \Gamma' = []$ 
  using step-hyps(1) more-invars(3) by fastforce
next
  show  $\bigwedge D. \text{Some } C = \text{Some } D \implies \text{atms-of-cls } D \ | \subseteq | \ \text{atms-of-cls } (N \ | \cup | \ U_{er})$ 
  using more-invars(4) step-hyps(2) by presburger
next
  show  $\bigwedge D. \text{Some } C = \text{Some } D \implies \text{trail-false-cls } \Gamma' \ D$ 
  using more-invars  $\langle C = \text{Some } C \rangle \langle \Gamma = (L, n) \ \# \ \Gamma' \rangle \langle \leftarrow L \notin \# \ C \rangle$ 
  using subtrail-falseI by auto
next
  show atms-of-cls  $U_{er} \ | \subseteq | \ \text{atms-of-cls } N$ 
  using more-invars by argo
next
  show  $\bigwedge C. C \ | \in | \ \mathcal{F} \implies \exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L \ C$ 
  using more-invars by metis
qed
next
case step-hyps: (resolution L D  $\Gamma'$  C)

```



```

have D-max-lit: ord-res.is-strictly-maximal-lit L D
  using invars ⟨Γ = (L, Some D) # Γ'⟩ by simp

show ?thesis
  unfolding ⟨s' = -⟩
proof (intro ord-res-11-invars.intros allI impI)
  show ord-res-10-invars N (Uer,  $\mathcal{F}$ ,  $\Gamma$ )
    using more-invars by argo
next
  show {#} |∈| N |∪| Uer ⇒ Γ = []
    using more-invars by argo
next
  have D |∈| iefac  $\mathcal{F}$  |∧| (N |∪| Uer)
    using invars ⟨Γ = (L, Some D) # Γ'⟩
    by (metis snd-conv)

  hence atms-of-cls D |⊆| atms-of-clss (N |∪| Uer)
    by (metis atms-of-clss-fimage-iefac atms-of-clss-finsert finsert-absorb funion-upper1)

  moreover have atms-of-cls C |⊆| atms-of-clss (N |∪| Uer)
    by (smt (verit) add-mset-add-single atms-of-cls-def dual-order.trans fimage-fsubsetI
      fimage-iff fset-fset-mset more-invars(4) step-hyps(1) union-iff)

  ultimately show ∧E. Some (remove1-mset (- L) C + remove1-mset L D) = Some E ⇒
    atms-of-cls E |⊆| atms-of-clss (N |∪| Uer)
    by (smt (verit, ccfv-threshold) add-mset-add-single atms-of-cls-def diff-single-trivial
      fimage-iff fset-fset-mset fsubsetI fsubset-funion-eq funionI1 insert-DiffM
      option.inject union-iff)
next
  fix E :: 'f gclause
  assume Some (remove1-mset (- L) C + remove1-mset L D) = Some E

  hence E = remove1-mset (- L) C + remove1-mset L D
    by simp

  show trail-false-cls  $\Gamma$  E
    unfolding trail-false-cls-def
proof (intro ballI)
  fix K :: 'f gliteral
  assume K ∈# E

  hence K ∈# remove1-mset (- L) C ∨ K ∈# remove1-mset L D
    unfolding ⟨E = -⟩ by simp

  thus trail-false-lit  $\Gamma$  K
proof (elim disjE)

```

```

assume  $K \in\# \text{remove1-mset } (- L) C$ 

moreover have  $\text{trail-false-cls } \Gamma C$ 
  using  $\text{more-invars } \langle C = \text{Some } C \rangle$  by  $\text{metis}$ 

ultimately show  $\text{trail-false-lit } \Gamma K$ 
  unfolding  $\text{trail-false-cls-def}$ 
  by  $(\text{metis in-diffD})$ 
next
assume  $K\text{-in}: K \in\# \text{remove1-mset } L D$ 

have  $\text{clause-could-propagate } \Gamma' D L$ 
  using  $\text{invars } \langle \Gamma = (L, \text{Some } D) \# \Gamma' \rangle$  by  $\text{simp}$ 

hence  $\text{trail-false-cls } \Gamma' \{ \#K \in\# D. K \neq L\# \}$ 
  by  $(\text{simp only: clause-could-propagate-def})$ 

hence  $\text{trail-false-cls } \Gamma \{ \#K \in\# D. K \neq L\# \}$ 
  unfolding  $\langle \Gamma = (L, \text{Some } D) \# \Gamma' \rangle$ 
  by  $(\text{simp add: trail-false-cls-def trail-false-lit-def})$ 

moreover have  $K \in\# \{ \#K \in\# D. K \neq L\# \}$ 
  using  $D\text{-max-lit } K\text{-in in-diffD linorder-lit.is-greatest-in-mset-iff}$  by
fastforce

ultimately show  $\text{trail-false-lit } \Gamma K$ 
  unfolding  $\text{trail-false-cls-def}$  by  $\text{metis}$ 
qed
qed
next
show  $\text{atms-of-cls } U_{er} \mid\subseteq \mid \text{atms-of-cls } N$ 
  using  $\text{more-invars}$  by  $\text{argo}$ 
next
show  $\forall C \mid\in \mid \mathcal{F}. \exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L C$ 
  using  $\text{more-invars}$  by  $\text{metis}$ 
qed
next
case  $\text{step-hyps}: (\text{backtrack } L \Gamma' C)$ 

have  $\{ \# \} \mid\neq \mid N \mid\cup \mid U_{er}$ 
  using  $\text{more-invars step-hyps(3)}$  by  $\text{fastforce}$ 

hence  $\{ \# \} \mid\neq \mid \text{iefac } \mathcal{F} \mid\uparrow \mid (N \mid\cup \mid U_{er})$ 
  unfolding  $\text{mempty-in-fimage-iefac}$  .

hence  $\neg(\exists C \mid\in \mid \text{iefac } \mathcal{F} \mid\uparrow \mid (N \mid\cup \mid U_{er}). \text{trail-false-cls } \Gamma C)$ 
  using  $\text{invars } \langle \Gamma = - \# \Gamma' \rangle$ 
  by  $(\text{metis (no-types, opaque-lifting) split-pairs})$ 

```

**hence**  $\neg(\exists C | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ trail-false-cls } \Gamma' C)$   
**by** (*metis append.simps(1) step-hyps(3) suffix-ConsD suffix-def trail-false-cls-if-trail-false-suffix*)

**moreover have**  $\neg \text{ trail-false-cls } \Gamma' C$   
**by** (*metis <trail-consistent  $\Gamma$ > fst-conv list.distinct(1) list.inject step-hyps(3,4) trail-consistent.simps trail-defined-lit-def trail-false-cls-def trail-false-lit-def uminus-lit-swap*)

**ultimately have**  $\neg(\exists C | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid \text{finsert } C \ U_{er}). \text{ trail-false-cls } \Gamma'$   
*C)* **by** (*simp add: iefac-def trail-false-cls-def*)

**have**  $\bigwedge xs. \text{fset } (\text{trail-atms } xs) = \text{atm-of } \text{'fst' } \text{set } xs$   
**unfolding** *fset-trail-atms ..*  
**also have**  $\bigwedge xs. \text{atm-of } \text{'fst' } \text{set } xs = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ xs)$   
**by** (*simp add: image-comp*)  
**finally have**  $\bigwedge xs. \text{fset } (\text{trail-atms } xs) = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ xs) .$

**have**  $\text{atms-of-cls } C \mid \subseteq \mid \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using** *more-invars <C = Some C> by metis*

**hence**  $\text{atms-of-clss } (N \mid \cup \mid \text{finsert } C \ U_{er}) = \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**by** *auto*

**show** *?thesis*  
**unfolding**  $\langle s' = (\text{finsert } C \ U_{er}, \mathcal{F}, \Gamma', \text{None}) \rangle$   
**proof** (*intro ord-res-11-invars.intros ord-res-10-invars.intros ballI allI impI conjI*)

**show** *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \ \Gamma'$   
**using** *invars unfolding < $\Gamma = - \# \Gamma'$ > by simp*

**next**  
**fix** *Ln :: 'f gliteral  $\times$  'f gclause option and C :: 'f gclause*  
**assume** *Ln  $\in$  set  $\Gamma'$  and snd Ln = Some C*  
**then show** *ord-res.is-strictly-maximal-lit (fst Ln) C*  
**using** *invars unfolding < $\Gamma = - \# \Gamma'$ > by simp*

**next**  
**show** *linorder-trm.is-lower-fset (trail-atms  $\Gamma'$ ) (atms-of-clss (N  $\mid \cup \mid$  finsert C  $U_{er}$ ))*  
**unfolding**  $\langle \text{fset } (\text{trail-atms } \Gamma') = \text{set } (\text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma') \rangle$   
**unfolding**  $\langle \text{atms-of-clss } (N \mid \cup \mid \text{finsert } C \ U_{er}) = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$   
**proof** (*rule linorder-trm.sorted-and-lower-set-appendD-right(2)*)  
**have** *sorted-wrt*  $(\lambda x y. y \prec_t x) \ (\text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma)$   
**using**  $\Gamma$ -*sorted by (simp add: sorted-wrt-map)*

**thus** *sorted-wrt*  $(\lambda x y. y \prec_t x) \ ([\text{atm-of } L] \ @ \ \text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma)$   
**unfolding**  $\langle \Gamma = - \# \Gamma' \rangle$  **by** *simp*

**next**  
**have**  $\text{set } ([\text{atm-of } L] \ @ \ \text{map } (\text{atm-of } \circ \text{fst}) \ \Gamma) = \text{fset } (\text{trail-atms } \Gamma)$

```

unfolding append-Cons append-Nil list.set
unfolding ⟨fset (trail-atms Γ') = set (map (atm-of o fst) Γ')⟩[symmetric]
unfolding ⟨Γ = - # Γ'⟩ trail-atms.simps prod.sel
by simp

thus linorder-trm.is-lower-set (set ([atm-of L] @ map (atm-of o fst) Γ'))
  (fset (atms-of-clss (N |∪| Uer)))
using Γ-lower
by (simp only:)
qed
next
fix
  Ln :: 'f gliteral × 'f gclause option and
  Γ'' :: ('f gliteral × 'f gclause option) list
assume Γ' = Ln # Γ''

have snd Ln = None
by (simp add: ⟨Γ' = Ln # Γ'⟩ invars(4) step-hyps(3))

thus (snd Ln ≠ None) ↔ (∃ C|∈|iefac F |' (N |∪| finsert C Uer).
trail-false-cls Γ' C)
using ⟨¬(∃ C|∈|iefac F |' (N |∪| finsert C Uer). trail-false-cls Γ' C)⟩
by argo

show snd Ln ≠ None ⇒ is-pos (fst Ln)
using ⟨snd Ln = None⟩ by simp

show ∧C. snd Ln = Some C ⇒ clause-could-propagate Γ'' C (fst Ln)
using ⟨snd Ln = None⟩ by simp

show ∧D. snd Ln = Some D ⇒ D |∈| iefac F |' (N |∪| finsert C Uer)
using ⟨snd Ln = None⟩ by simp

show ∧x. x ∈ set Γ'' ⇒ snd x = None
by (simp add: ⟨Γ' = Ln # Γ'⟩ invars(4) step-hyps(3))
next
fix
  Ln :: 'f gliteral × 'f gclause option and
  Γ1 Γ0 :: ('f gliteral × 'f gclause option) list
assume Γ' = Γ1 @ Ln # Γ0 and snd Ln = None
thus ¬ (∃ C|∈|iefac F |' (N |∪| finsert C Uer). trail-false-cls (Ln # Γ0) C)
using ⟨¬(∃ C|∈|iefac F |' (N |∪| finsert C Uer). trail-false-cls Γ' C)⟩
by (metis suffixI trail-false-cls-if-trail-false-suffix)
next
have C ≠ {#}
using ⟨¬ L ∈ # C⟩ by force

hence {#} |∉| N |∪| finsert C Uer
using ⟨{#} |∉| N |∪| Uer⟩ by simp

```

```

thus {#} |∈| N |∪| finsert C Uer ⇒ Γ' = []
by contradiction
next
show ∧D. None = Some D ⇒ atms-of-cls D |⊆| atms-of-clss (N |∪| finsert
C Uer)
by simp
next
show ∧C. None = Some C ⇒ trail-false-cls Γ' C
by simp
next
have atms-of-cls C |⊆| atms-of-clss N
by (smt (verit, ccfv-threshold) ⟨atms-of-cls C |⊆| atms-of-clss (N |∪| Uer)⟩
atms-of-clss-def fin-mono fmember-ffUnion-iff fsubsetI funion-iff
more-invars(6))

thus atms-of-clss (finsert C Uer) |⊆| atms-of-clss N
using ⟨atms-of-clss Uer |⊆| atms-of-clss N⟩ by simp
next
show ∧C. C |∈| ℱ ⇒ ∃L. is-pos L ∧ ord-res.is-maximal-lit L C
using more-invars by metis
qed
qed
qed
qed

```

**lemma** *rtranclp-ord-res-11-preserves-invars*:

```

assumes
  step: (ord-res-11 N)** s s' and
  invars: ord-res-11-invars N s
shows ord-res-11-invars N s'
using step invars ord-res-11-preserves-invars
by (smt (verit, del-insts) rtranclp-induct)

```

**lemma** *tranclp-ord-res-11-preserves-invars*:

```

assumes
  step: (ord-res-11 N)++ s s' and
  invars: ord-res-11-invars N s
shows ord-res-11-invars N s'
using step invars ord-res-11-preserves-invars
by (smt (verit, del-insts) tranclp-induct)

```

**lemma** *ex-ord-res-11-if-not-final*:

```

assumes
  not-final: ¬ ord-res-11-final (N, s) and
  invars: ord-res-11-invars N s
shows ∃s'. ord-res-11 N s s'
using invars
proof (cases N s rule: ord-res-11-invars.cases)

```

```

case more-invars: (1  $U_{er}$   $\mathcal{F}$   $\Gamma$   $\mathcal{C}$ )
show ?thesis
  using  $\langle \text{ord-res-10-invars } N (U_{er}, \mathcal{F}, \Gamma) \rangle$ 
proof (cases  $N (U_{er}, \mathcal{F}, \Gamma)$  rule: ord-res-10-invars.cases)
  case invars: 1

  show ?thesis
  proof (cases  $\mathcal{C}$ )
    case None
    show ?thesis
    proof (cases  $\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } \Gamma \ C$ )
      case True

      then obtain  $C$  where
         $\text{linorder-cls.is-least-in-fset } (\text{ffilter } (\text{trail-false-cls } \Gamma) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) \ C$ 
        using linorder-cls.ex-is-least-in-ffilter-iff by metis

      thus ?thesis
      unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, C) \rangle \langle C = \text{None} \rangle$ 
      using ord-res-11.conflict by metis
    next
    case no-false-cls: False

    hence  $\exists A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2$ 
    using not-final
    unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, C) \rangle \langle C = \text{None} \rangle$ 
    by (smt (verit) invars( $\beta$ ) linorder-trm.antisym-conv3 linorder-trm.not-in-lower-setI
      ord-res-11-final.model-found)

    then obtain  $A$  where
       $A\text{-least: linorder-trm.is-least-in-fset } \{ \mid A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2 \mid \}$ 
      using linorder-trm.ex-is-least-in-ffilter-iff by presburger

    show ?thesis
    proof (cases  $\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{clause-could-propagate } \Gamma \ C \ (\text{Pos } A)$ )
      case True

      then obtain  $C$  where
         $C\text{-least: linorder-cls.is-least-in-fset } \{ \mid C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{clause-could-propagate } \Gamma \ C \ (\text{Pos } A) \mid \}$ 
        using linorder-cls.ex-is-least-in-ffilter-iff by presburger

      show ?thesis
      proof (cases  $\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-cls } ((\text{Pos } A, \text{None}) \# \Gamma) \ C$ )

```

```

case True

moreover have trail-false-cls ((Pos A, None) #  $\Gamma$ ) =
  trail-false-cls ((Pos A, Some (efac C)) #  $\Gamma$ )
  unfolding trail-false-cls-def trail-false-lit-def by simp

ultimately show ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \mathcal{C} = None \rangle$ 
  using ord-res-11.propagate[OF no-false-cls A-least C-least]
  by metis
next
case False
thus ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \mathcal{C} = None \rangle$ 
  using ord-res-11.decide-pos[OF no-false-cls A-least C-least]
  by metis
qed
next
case False
thus ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \mathcal{C} = None \rangle$ 
  using ord-res-11.decide-neg[OF no-false-cls A-least] by metis
qed
qed
next
case (Some D)

hence D-false: trail-false-cls  $\Gamma$  D
  using more-invars by metis

show ?thesis
proof (cases D = {#})
  case True

  hence  $\Gamma \neq []$ 
  using not-final
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \mathcal{C} = Some D \rangle$ 
  unfolding ord-res-11-final.simps by metis

then obtain L n  $\Gamma'$  where  $\Gamma = (L, n) \# \Gamma'$ 
  by (metis list.exhaust prod.exhaust)

moreover have  $- L \notin \# D$ 
  unfolding  $\langle D = \{ \# \} \rangle$  by simp

ultimately show ?thesis
  unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \mathcal{C} = Some D \rangle$ 
  using ord-res-11.skip by metis
next

```

```

case False

then obtain  $L\ n\ \Gamma'$  where  $\Gamma = (L, n) \# \Gamma'$ 
  using D-false[unfolded trail-false-cls-def trail-false-lit-def]
  by (metis eq-fst-iff image-iff list.set-cases multiset-nonemptyE)

show ?thesis
proof (cases - L ∈# D)
  case True
  show ?thesis
  proof (cases n)
    case None
    show ?thesis
    proof (cases - L ∈# D)
      case True
      thus ?thesis
      unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \Gamma = (L, n) \# \Gamma' \rangle \langle n = None \rangle \langle \mathcal{C} =$ 
Some D  $\rangle$ 
        using ord-res-11.backtrack by metis
      next
      case False
      thus ?thesis
      using ord-res-11.skip
      unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \Gamma = (L, n) \# \Gamma' \rangle \langle n = None \rangle \langle \mathcal{C} =$ 
Some D  $\rangle$  by metis
      qed
    next
    case (Some C)
    thus ?thesis
    unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \Gamma = (L, n) \# \Gamma' \rangle \langle n = Some\ C \rangle \langle \mathcal{C} =$ 
Some D  $\rangle$ 
      using ord-res-11.resolution[OF - <- L ∈# D] by metis
      qed
    next
    case False
    thus ?thesis
    using ord-res-11.skip
    unfolding  $\langle s = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle \langle \Gamma = (L, n) \# \Gamma' \rangle \langle \mathcal{C} = Some\ D \rangle$  by
metis
    qed
  qed
  qed
  qed
  qed
qed

lemma ord-res-11-safe-state-if-invars:
  fixes  $N\ s$ 
  assumes invars: ord-res-11-invars N s
  shows safe-state (constant-context ord-res-11) ord-res-11-final (N, s)

```



```

using safe-state-constant-context-if-invars[where
   $\mathcal{R} = \text{ord-res-11}$  and  $\mathcal{F} = \text{ord-res-11-final}$  and  $\mathcal{I} = \text{ord-res-11-invars}$ ]
using ord-res-11-preserves-invars ex-ord-res-11-if-not-final invars by metis

lemma rtrancl-ord-res-11-all-resolution-steps:
  assumes C-max-lit: ord-res.is-strictly-maximal-lit K C
  shows (ord-res-11 N)** (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some D) (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some (eres C D))
  proof –
    obtain CD where eres C D = CD and run: full-run (ground-resolution C) D CD
    using ex1-eres-eq-full-run-ground-resolution by metis

    have (ord-res-11 N)** (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some D) (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some CD)
    proof (rule full-run-preserves-invariant[OF run])
      show (ord-res-11 N)** (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some D) (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some D)
      by simp
    next
      fix x y
      assume ground-resolution C x y

      hence ord-res-11 N (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some x) (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some y)
      unfolding ground-resolution-def
      proof (cases x C y rule: ord-res.ground-resolution.cases)
        case res-hyps: (ground-resolutionI A P1' P2')

        have K = – Neg A
          using C-max-lit
        by (metis ord-res.Uniq-strictly-maximal-lit-in-ground-cls res-hyps(6) the1-equality' uminus-Neg)

        hence – K ∈# x
          by (simp add: ⟨x = add-mset (Neg A) P1'⟩)

        moreover have y = remove1-mset (– K) x + remove1-mset K C
          using res-hyps ⟨K = – Neg A⟩ by force

        ultimately show ?thesis
          using ord-res-11.resolution[OF refl, of K x N U  $\mathcal{F}$  C  $\Gamma$ ]
          by metis
      qed

    moreover assume (ord-res-11 N)**
      (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some D)
      (U,  $\mathcal{F}$ , (K, Some C) #  $\Gamma$ , Some x)

```

```

ultimately show (ord-res-11 N)**
  (U, F, (K, Some C) # Γ, Some D)
  (U, F, (K, Some C) # Γ, Some y)
  by simp
qed

then show ?thesis
  unfolding ⟨eres C D = CD⟩
  by argo
qed

lemma rtrancl-ord-res-11-all-skip-steps:
  (ord-res-11 N)** (U, F, Γ, Some C) (U, F, dropWhile (λLn. - fst Ln ∉# C)
  Γ, Some C)
proof (induction Γ)
  case Nil
  show ?case by simp
next
  case (Cons Ln Γ)
  show ?case
  proof (cases - fst Ln ∈# C)
    case True
    hence dropWhile (λLn. - fst Ln ∉# C) (Ln # Γ) = Ln # Γ
    by simp
    thus ?thesis
    by simp
  next
    case False
    hence *: dropWhile (λLn. - fst Ln ∉# C) (Ln # Γ) = dropWhile (λLn. - fst
  Ln ∉# C) Γ
    by simp
  obtain L C where Ln = (L, C)
  by (metis prod.exhaust)

  have ord-res-11 N (U, F, Ln # Γ, Some C) (U, F, Γ, Some C)
  unfolding ⟨Ln = (L, C)⟩
  proof (rule ord-res-11.skip)
  show - L ∉# C
  using False unfolding ⟨Ln = (L, C)⟩ by simp
qed

thus ?thesis
  unfolding *
  using Cons.IH
  by simp
qed
qed

```

```

end

end
theory Simulation-SCLFOL-ORDRES
  imports
    Background
    ORD-RES
    ORD-RES-1
    ORD-RES-2
    ORD-RES-3
    ORD-RES-4
    ORD-RES-5
    ORD-RES-6
    ORD-RES-7
    ORD-RES-8
    ORD-RES-9
    ORD-RES-10
    ORD-RES-11
    Clause-Could-Propagate
  begin

```

## 27 ORD-RES-1 (deterministic)

```

type-synonym 'f ord-res-1-state = 'f gclause fset

context simulation-SCLFOL-ground-ordered-resolution begin

sublocale backward-simulation-with-measuring-function where
  step1 = ord-res and
  step2 = ord-res-1 and
  final1 = ord-res-final and
  final2 = ord-res-1-final and
  order =  $\lambda$ -. False and
  match = (=) and
  measure =  $\lambda$ -. ()
proof unfold-locales
  show wfP ( $\lambda$ -. False)
    by simp
next
  show  $\bigwedge N1\ N2. N1 = N2 \implies \text{ord-res-1-final } N2 \implies \text{ord-res-final } N1$ 
    unfolding ord-res-1-final-def by metis
next
  fix N1 N2 N2' :: 'f ord-res-1-state
  assume match: N1 = N2 and step2: ord-res-1 N2 N2'
  show  $(\exists N1'. \text{ord-res}^{++} N1\ N1' \wedge N1' = N2') \vee N1 = N2' \wedge \text{False}$ 
  proof (intro disjI1 exI conjI)

  have empty-no-in:  $\{\#\} \notin N2$ 
    if C-least: linorder-cls.is-least-in-fset  $\{C \mid C \in N2\}$ .

```

```

    ¬ ord-res.interp (fset N2) C ∪ ord-res.production (fset N2) C ⊨ C} C and
    L-max: linorder-lit.is-maximal-in-mset C L
  for C L
  proof (rule notI)
    assume {#} |∈| N2
    moreover have ¬ ord-res.interp (fset N2) {#} ∪ ord-res.production (fset
N2) {#} ⊨ {#}
      by simp
    moreover have ∧ C. {#} ≼c C
      using mempty-lesseq-cls by metis
    ultimately have C = {#}
      using C-least
      by (metis (no-types, lifting) ffilter-member-filter linorder-cls.is-least-in-fset-iff
linorder-cls.less-le-not-le)
    moreover have L ∈# C
      using L-max by (simp add: linorder-lit.is-maximal-in-mset-iff)
    ultimately show False
      by simp
  qed

  have ord-res N2 N2'
    using step2
  proof (cases N2 N2' rule: ord-res-1.cases)
    case hyps: (factoring C L C')
    show ?thesis
    proof (rule ord-res.factoring)
      show {#} |∉| N2
        using hyps mempty-no-in is-least-false-clause-def by simp
    next
      show ex-false-clause (fset N2)
        unfolding ex-false-clause-def
        using hyps is-least-false-clause-def
        by (metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD)
    next
      show C |∈| N2
        using hyps is-least-false-clause-def linorder-cls.is-least-in-fset-ffilterD(1)
  by blast
  next
    show ord-res.ground-factoring C C'
      using hyps by argo
  next
    show N2' = finsert C' N2
      using hyps by argo
  qed
  next
  case hyps: (resolution C L D CD)
  show ?thesis
  proof (rule ord-res.resolution)
    show {#} |∉| N2

```

```

    using hyps mempty-no-in is-least-false-clause-def by simp
  next
  show ex-false-clause (fset N2)
  unfolding ex-false-clause-def
  using hyps is-least-false-clause-def
  by (metis (no-types, lifting) linorder-cls.is-least-in-fset-ffilterD)
  next
  show C |∈| N2
  using hyps is-least-false-clause-def linorder-cls.is-least-in-fset-ffilterD(1)
by blast
  next
  show D |∈| N2
  using hyps by argo
  next
  show ord-res.ground-resolution C D CD
  using hyps by argo
  next
  show N2' = finsert CD N2
  using hyps by argo
  qed
  qed
  thus ord-res++ N1 N2'
  unfolding match by simp
  next
  show N2' = N2' ..
  qed
  qed
end

```

## 28 ORD-RES-2 (full factorization)

**type-synonym** 'f ord-res-2-state = 'f gclause fset × 'f gclause fset × 'f gclause fset

**context** simulation-SCLFOL-ground-ordered-resolution **begin**

**fun** ord-res-1-matches-ord-res-2

:: 'f ord-res-1-state ⇒ - ⇒ bool **where**

ord-res-1-matches-ord-res-2 S1 (N, (U<sub>r</sub>, U<sub>ef</sub>)) ⟷ (∃ U<sub>f</sub>.

S1 = N |∪| U<sub>r</sub> |∪| U<sub>ef</sub> |∪| U<sub>f</sub> ∧

(∀ C<sub>f</sub> |∈| U<sub>f</sub>. ∃ C |∈| N |∪| U<sub>r</sub> |∪| U<sub>ef</sub>. ord-res.ground-factoring<sup>++</sup> C C<sub>f</sub> ∧

C<sub>f</sub> ≠ efac C<sub>f</sub> ∧

(efac C<sub>f</sub> |∈| U<sub>ef</sub> ∨ is-least-false-clause (N |∪| U<sub>r</sub> |∪| U<sub>ef</sub>) C)))

**lemma** ord-res-1-matches-ord-res-2-simps':

ord-res-1-matches-ord-res-2 S1 (N, (U<sub>r</sub>, U<sub>ef</sub>)) ⟷

(∃ U<sub>f</sub>. S1 = N |∪| U<sub>r</sub> |∪| U<sub>ef</sub> |∪| U<sub>f</sub> ∧

(∀ C<sub>f</sub> |∈| U<sub>f</sub>. C<sub>f</sub> ≠ efac C<sub>f</sub> ∧ (∃ C |∈| N |∪| U<sub>r</sub> |∪| U<sub>ef</sub>. ord-res.ground-factoring<sup>++</sup>

$C \ C_f \wedge$   
 $(\text{efac } C_f \mid \in \mid U_{ef} \vee \text{is-least-false-clause } (N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ C))$   
**unfolding** *ord-res-1-matches-ord-res-2.simps by metis*

**lemma** *ord-res-1-matches-ord-res-2.simps''*:  
 $\text{ord-res-1-matches-ord-res-2 } S1 \ (N, (U_r, U_{ef})) \longleftrightarrow$   
 $(\exists U_f. S1 = N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f \wedge$   
 $(\forall C_f \mid \in \mid U_f. C_f \neq \text{efac } C_f \wedge (\exists C \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}. \text{ord-res.ground-factoring}^{++}$   
 $C \ C_f \wedge$   
 $(\text{efac } C \mid \in \mid U_{ef} \vee \text{is-least-false-clause } (N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ C))))$   
**unfolding** *ord-res-1-matches-ord-res-2.simps'*  
**by** (*metis ground-factorings-preserves-efac tranclp-into-rtranclp*)

**lemma** *ord-res-1-final-iff-ord-res-2-final*:  
**assumes** *match: ord-res-1-matches-ord-res-2*  $S_1 \ S_2$   
**shows** *ord-res-1-final*  $S_1 \longleftrightarrow$  *ord-res-2-final*  $S_2$

**proof** –

**obtain**  $N \ U_r \ U_{ef}$  **where**  $S_2 = (N, (U_r, U_{ef}))$   
**by** (*metis prod.exhaust*)  
**with** *match* **obtain**  $U_f$  **where**  
 $S_1\text{-def: } S_1 = N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f$  **and**  
 $U_f\text{-spec: } \forall C_f \mid \in \mid U_f. \exists C \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}. \text{ord-res.ground-factoring}^{++}$   
 $C \ C_f \wedge C_f \neq \text{efac } C_f \wedge$   
 $(\text{efac } C_f \mid \in \mid U_{ef} \vee \text{is-least-false-clause } (N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ C)$   
**by** *auto*

**have**  $U_f\text{-unproductive: } \forall C_f \mid \in \mid U_f. \text{ord-res.production } (fset \ (N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f)) \ C_f = \{\}$

**proof** (*intro ballI*)

**fix**  $C_f$   
**assume**  $C_f \mid \in \mid U_f$   
**hence**  $C_f \neq \text{efac } C_f$   
**using**  $U_f\text{-spec}$  **by** *metis*  
**hence**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C_f$   
**using** *nex-strictly-maximal-pos-lit-if-neq-efac* **by** *metis*  
**thus**  $\text{ord-res.production } (fset \ (N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f)) \ C_f = \{\}$   
**using** *unproductive-if-nex-strictly-maximal-pos-lit* **by** *metis*

**qed**

**have**  $\text{Interp-eq: } \bigwedge C. \text{ord-res-Interp } (fset \ (N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f)) \ C =$   
 $\text{ord-res-Interp } (fset \ (N \mid \cup \mid U_r \mid \cup \mid U_{ef})) \ C$   
**using** *Interp-union-unproductive*[*of fset (N |cup| U\_r |cup| U\_{ef}) fset U\_f, folded*  
*union-fset,*  
*OF finite-fset finite-fset U\_f-unproductive*] .

**have**  $\{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f \longleftrightarrow \{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}$

**proof** (*rule iffI*)

**assume**  $\{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \mid \cup \mid U_f$   
**hence**  $\{\#\} \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \vee \{\#\} \mid \in \mid U_f$

```

    by simp
  thus {#} |∈| N |∪| Ur |∪| Uef
  proof (elim disjE)
    assume {#} |∈| N |∪| Ur |∪| Uef
    thus {#} |∈| N |∪| Ur |∪| Uef
      by assumption
  next
    assume {#} |∈| Uf
    hence {#} ≠ efac {#}
      using Uf-spec[rule-format, of {#}] by metis
    hence False
      by simp
    thus {#} |∈| N |∪| Ur |∪| Uef ..
  qed
next
  assume {#} |∈| N |∪| Ur |∪| Uef
  thus {#} |∈| N |∪| Ur |∪| Uef |∪| Uf
    by simp
  qed

moreover have ¬ ex-false-clause (fset (N |∪| Ur |∪| Uef |∪| Uf)) ↔
  ¬ ex-false-clause (fset (N |∪| Ur |∪| Uef))
  proof (rule iffI; erule contrapos-nn)
    assume ex-false-clause (fset (N |∪| Ur |∪| Uef))
    thus ex-false-clause (fset (N |∪| Ur |∪| Uef |∪| Uf))
      unfolding ex-false-clause-def Interp-eq by auto
  next
    assume ex-false-clause (fset (N |∪| Ur |∪| Uef |∪| Uf))
    then obtain C where
      C |∈| N |∪| Ur |∪| Uef |∪| Uf and
      C-false: ¬ ord-res-Interp (fset (N |∪| Ur |∪| Uef)) C ⊨ C
      unfolding ex-false-clause-def Interp-eq by metis
    hence C |∈| N |∪| Ur |∪| Uef ∨ C |∈| Uf
      by simp
    thus ex-false-clause (fset (N |∪| Ur |∪| Uef))
      proof (elim disjE)
        assume C |∈| N |∪| Ur |∪| Uef
        thus ex-false-clause (fset (N |∪| Ur |∪| Uef))
          unfolding ex-false-clause-def using C-false by metis
      next
        assume C |∈| Uf
        then obtain C' where C' |∈| N |∪| Ur |∪| Uef and
          ord-res.ground-factoring++ C' C and
          C ≠ efac C and
          efac C |∈| Uef ∨ is-least-false-clause (N |∪| Ur |∪| Uef) C'
          using Uf-spec[rule-format, of C] by metis
        thus ex-false-clause (fset (N |∪| Ur |∪| Uef))
          proof (elim disjE exE conjE)
            assume efac C |∈| Uef

```

```

show ex-false-clause (fset (N | $\cup$ | Ur | $\cup$ | Uef))
proof (cases ord-res-Interp (fset (N | $\cup$ | Ur | $\cup$ | Uef)) (efac C)  $\models$  efac C)
  case efac-C-true: True
  have efac C  $\subseteq_{\#}$  C
    using efac-subset[of C] .
  hence efac C  $\preceq_c$  C
    using subset-implies-reflclp-multip by metis
  hence ord-res-Interp (fset (N | $\cup$ | Ur | $\cup$ | Uef)) C  $\models$  efac C
    using efac-C-true ord-res.entailed-clause-stays-entailed by fastforce
  hence ord-res-Interp (fset (N | $\cup$ | Ur | $\cup$ | Uef)) C  $\models$  C
    using efac-C-true by (simp add: true-cls-def)
  with C-false have False
    by contradiction
  thus ?thesis ..
next
  case False

  moreover have efac C  $\in$  |N | $\cup$ | Ur | $\cup$ | Uef
    using  $\langle$ efac C  $\in$  |Uef $\rangle$  by simp

  ultimately show ex-false-clause (fset (N | $\cup$ | Ur | $\cup$ | Uef))
    unfolding ex-false-clause-def by metis
  qed
next
  assume is-least-false-clause (N | $\cup$ | Ur | $\cup$ | Uef) C'
  hence C'  $\in$  |N | $\cup$ | Ur | $\cup$ | Uef and  $\neg$  ord-res-Interp (fset (N | $\cup$ | Ur | $\cup$ |
Uef)) C'  $\models$  C'
    using linorder-cls.is-least-in-ffilter-iff is-least-false-clause-def by simp-all
  thus ex-false-clause (fset (N | $\cup$ | Ur | $\cup$ | Uef))
    unfolding ex-false-clause-def by metis
  qed
  qed
  qed

  ultimately show ?thesis
  by (simp add: S1-def  $\langle$ S2 = (N, Ur, Uef) $\rangle$  ord-res-1-final-def ord-res-2-final.simps
ord-res-final-def)
qed

lemma safe-states-if-ord-res-1-matches-ord-res-2:
  assumes match: ord-res-1-matches-ord-res-2 S1 S2
  shows safe-state ord-res-1 ord-res-1-final S1  $\wedge$  safe-state ord-res-2-step ord-res-2-final
S2
proof –
  have safe-state ord-res-1 ord-res-1-final S1
    using safe-state-if-all-states-safe ord-res-1-safe by metis

  moreover have safe-state ord-res-2-step ord-res-2-final S2

```



using *safe-state-if-all-states-safe ord-res-2-step-safe* by *metis*

ultimately show *?thesis*

by *argo*

qed

**definition** *ord-res-1-measure* where

*ord-res-1-measure* *s1* =

(if  $\exists C. \text{is-least-false-clause } s1 \ C$  then

The (*is-least-false-clause* *s1*)

else

{#})

**lemma** *forward-simulation*:

assumes *match*: *ord-res-1-matches-ord-res-2* *s1* *s2* and

*step1*: *ord-res-1* *s1* *s1'*

shows ( $\exists s2'. \text{ord-res-2-step}^{++} \ s2 \ s2' \wedge \text{ord-res-1-matches-ord-res-2} \ s1' \ s2'$ )  $\vee$

*ord-res-1-matches-ord-res-2* *s1'* *s2*  $\wedge \text{ord-res-1-measure} \ s1' \ \subset\# \ \text{ord-res-1-measure}$

*s1*

**proof** –

let

*?match* = *ord-res-1-matches-ord-res-2* and

*?measure* = *ord-res-1-measure* and

*?order* = ( $\subset\#$ )

**obtain** *N* *U<sub>r</sub>* *U<sub>ef</sub>* :: '*f* *gterm* *clause* *fset* where

*s2-def*: *s2* = (*N*, (*U<sub>r</sub>*, *U<sub>ef</sub>*))

by (*metis prod.exhaust*)

**from** *match* **obtain** *U<sub>f</sub>* where

*s1-def*: *s1* = *N*  $\cup$  *U<sub>r</sub>*  $\cup$  *U<sub>ef</sub>*  $\cup$  *U<sub>f</sub>* and

*U<sub>f</sub>-spec*:  $\forall C_f \mid \in \ U_f. \exists C \mid \in \ N \cup U_r \cup U_{ef}. \text{ord-res.ground-factoring}^{++}$   
*C* *C<sub>f</sub>*  $\wedge C_f \neq \text{efac } C_f \wedge$

(*efac* *C<sub>f</sub>*  $\mid \in \ U_{ef} \vee \text{is-least-false-clause} \ (N \cup U_r \cup U_{ef}) \ C$ )

**unfolding** *s2-def* *ord-res-1-matches-ord-res-2.simps* by *metis*

**have** *U<sub>f</sub>-unproductive*:  $\forall C_f \mid \in \ U_f. \text{ord-res.production} \ (\text{fset} \ (N \cup U_r \cup U_{ef} \cup U_f)) \ C_f = \{\}$

**proof** (*intro ballI*)

fix *C<sub>f</sub>*

assume *C<sub>f</sub>*  $\mid \in \ U_f$

hence *C<sub>f</sub>*  $\neq \text{efac } C_f$

using *U<sub>f</sub>-spec* by *metis*

hence  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C_f$

using *nex-strictly-maximal-pos-lit-if-neq-efac* by *metis*

**thus** *ord-res.production* (*fset* (*N*  $\cup$  *U<sub>r</sub>*  $\cup$  *U<sub>ef</sub>*  $\cup$  *U<sub>f</sub>*)) *C<sub>f</sub>* =  $\{\}$

using *unproductive-if-nex-strictly-maximal-pos-lit* by *metis*

qed

```

have Interp-eq:  $\bigwedge C. \text{ord-res-Interp} (\text{fset} (N \mid \cup U_r \mid \cup U_{ef} \mid \cup U_f)) C =$ 
   $\text{ord-res-Interp} (\text{fset} (N \mid \cup U_r \mid \cup U_{ef})) C$ 
  using Interp-union-unproductive[of fset (N |cup U_r |cup U_{ef}) fset U_f, folded
union-fset,
  OF finite-fset finite-fset U_f-unproductive] .

show  $(\exists s2'. \text{ord-res-2-step}^{++} s2 s2' \wedge ?\text{match } s1' s2') \vee$ 
   $?\text{match } s1' s2 \wedge ?\text{order} (?\text{measure } s1') (?\text{measure } s1)$ 
  using step1
proof (cases s1 s1' rule: ord-res-1.cases)
  case (factoring C L C')

  have C-least-false: is-least-false-clause  $(N \mid \cup U_r \mid \cup U_{ef} \mid \cup U_f) C$ 
  using factoring
  unfolding is-least-false-clause-def s1-def by argo

  hence C-in:  $C \mid \in N \mid \cup U_r \mid \cup U_{ef} \mid \cup U_f$ 
  unfolding is-least-false-clause-def linorder.cls.is-least-in-filter-iff s1-def by
argo
  hence C-in-disj:  $C \mid \in N \mid \cup U_r \mid \cup U_{ef} \vee C \mid \in U_f$ 
  by simp

  show ?thesis
  proof (cases C' = efac C')
  case True
  let  $?s2' = (N, (U_r, \text{finsert } C' U_{ef}))$ 

  have ord-res-2-step++  $s2 ?s2'$ 
  proof (rule tranclp.r-into-trancl)
  show ord-res-2-step  $s2 (N, U_r, \text{finsert } C' U_{ef})$ 
  using C-in-disj
  proof (elim disjE)
  assume  $C \mid \in N \mid \cup U_r \mid \cup U_{ef}$ 
  show ?thesis
  unfolding s2-def
  proof (intro ord-res-2-step.intros ord-res-2.factoring)
  show is-least-false-clause  $(N \mid \cup U_r \mid \cup U_{ef}) C$ 
  using is-least-false-clause-if-is-least-false-clause-in-union-unproductive[
  OF U_f-unproductive <C |in N |cup U_r |cup U_{ef}> C-least-false]
  unfolding is-least-false-clause-def .
  next
  show ord-res.is-maximal-lit L C
  using  $\langle \text{ord-res.is-maximal-lit } L C \rangle$  .
  next
  show is-pos L
  using  $\langle \text{is-pos } L \rangle$  .
  next
  show  $\text{finsert } C' U_{ef} = \text{finsert} (\text{efac } C) U_{ef}$ 
  using True factoring ground-factoring-preserves-efac by metis

```

```

qed
next
assume  $C \in U_f$ 
then obtain  $x$  where
   $x \in N \cup U_r \cup U_{ef}$  and
   $\text{ord-res.ground-factoring}^{++} x C$  and
   $C \neq \text{efac } C$  and
   $\text{efac } C \in U_{ef} \vee \text{is-least-false-clause } (N \cup U_r \cup U_{ef}) x$ 
  using  $U_f\text{-spec}$  by metis

show ?thesis
  unfolding  $s2\text{-def}$ 
proof (intro  $\text{ord-res-2-step.intros}$   $\text{ord-res-2.factoring}$ )
  have  $\langle \text{efac } C \notin U_{ef} \rangle$ 
  proof (rule notI)
    have  $\text{efac } C \preceq_c C$ 
    using  $\text{efac-subset}[of C]$   $\text{subset-implies-reflclp-multp}$  by metis
    hence  $\text{efac } C \prec_c C$ 
    using  $\langle C \neq \text{efac } C \rangle$  by order

    moreover assume  $\text{efac } C \in U_{ef}$ 

  ultimately show False
    using  $C\text{-least-false}[unfolding \text{is-least-false-clause-def}$ 
       $\text{linorder-cls.is-least-in-filter-iff}]$ 
    by (metis  $\langle C \neq \text{efac } C \rangle$   $\text{funionCI}$   $\text{linorder-cls.not-less-iff-gr-or-eq}$ 
       $\text{ord-res.entailed-clause-stays-entailed}$   $\text{set-mset-efac}$   $\text{true-cls-def}$ )
  qed
  thus  $\text{is-least-false-clause } (N \cup U_r \cup U_{ef}) x$ 
    using  $\langle \text{efac } C \in U_{ef} \vee \text{is-least-false-clause } (N \cup U_r \cup U_{ef}) x \rangle$ 
by argo
next
show  $\text{ord-res.is-maximal-lit } L x$ 
  using  $\langle \text{ord-res.ground-factoring}^{++} x C \rangle$   $\langle \text{ord-res.is-maximal-lit } L C \rangle$ 
  using  $\text{ord-res.ground-factorings-preserves-maximal-literal}$ 
  by (metis  $\text{tranclp-into-rtranclp}$ )
next
show  $\text{is-pos } L$ 
  using  $\langle \text{is-pos } L \rangle$  .
next
show  $\text{finsert } C' U_{ef} = \text{finsert } (\text{efac } x) U_{ef}$ 
  using  $\langle \text{ord-res.ground-factoring}^{++} x C \rangle$   $\langle \text{ord-res.ground-factoring } C C' \rangle$ 
  using  $\text{True}$   $\text{ground-factorings-preserves-efac}$   $\text{ground-factoring-preserves-efac}$ 
  by (metis  $\text{tranclp-into-rtranclp}$ )
qed
qed
qed

moreover have ?match  $s1' ?s2'$ 

```

**proof** –  
**have**  $s1' = N \mid\cup\mid U_r \mid\cup\mid \text{finsert } C' U_{ef} \mid\cup\mid U_f$   
**unfolding**  $\langle s1' = \text{finsert } C' s1 \rangle$   $s1\text{-def}$  **by** *simp*

**moreover have**  $\exists C \mid\in\mid N \mid\cup\mid U_r \mid\cup\mid \text{finsert } C' U_{ef}$ .  
 $\text{ord-res.ground-factorizing}^{++} C C_f \wedge C_f \neq \text{efac } C_f \wedge$   
 $(\text{efac } C_f \mid\in\mid \text{finsert } C' U_{ef} \vee \text{is-least-false-clause } (N \mid\cup\mid U_r \mid\cup\mid \text{finsert } C'$   
 $U_{ef}) C)$   
**if**  $C_f \mid\in\mid U_f$  **for**  $C_f$   
**proof** –  
**obtain**  $x$  **where**  
 $x \mid\in\mid N \mid\cup\mid U_r \mid\cup\mid U_{ef}$  **and**  
 $\text{ord-res.ground-factorizing}^{++} x C_f$  **and**  
 $C_f \neq \text{efac } C_f$  **and**  
 $\text{efac } C_f \mid\in\mid U_{ef} \vee \text{is-least-false-clause } (N \mid\cup\mid U_r \mid\cup\mid U_{ef}) x$   
**using**  $\langle C_f \mid\in\mid U_f \rangle$   $U_f\text{-spec}$  **by** *metis*

**show** *?thesis*  
**proof** (*intro* *bexI* *conjI*)  
**show**  $x \mid\in\mid N \mid\cup\mid U_r \mid\cup\mid \text{finsert } C' U_{ef}$   
**using**  $\langle x \mid\in\mid N \mid\cup\mid U_r \mid\cup\mid U_{ef} \rangle$  **by** *simp*  
**next**  
**show**  $\text{ord-res.ground-factorizing}^{++} x C_f$   
**using**  $\langle \text{ord-res.ground-factorizing}^{++} x C_f \rangle$  .  
**next**  
**show**  $C_f \neq \text{efac } C_f$   
**using**  $\langle C_f \neq \text{efac } C_f \rangle$  .  
**next**  
**show**  $\text{efac } C_f \mid\in\mid \text{finsert } C' U_{ef} \vee \text{is-least-false-clause } (N \mid\cup\mid U_r \mid\cup\mid$   
 $\text{finsert } C' U_{ef}) x$   
**using**  $\langle \text{efac } C_f \mid\in\mid U_{ef} \vee \text{is-least-false-clause } (N \mid\cup\mid U_r \mid\cup\mid U_{ef}) x \rangle$   
**proof** (*elim* *disjE*)  
**assume**  $\text{efac } C_f \mid\in\mid U_{ef}$   
**thus** *?thesis*  
**by** *simp*  
**next**  
**assume**  $\text{is-least-false-clause } (N \mid\cup\mid U_r \mid\cup\mid U_{ef}) x$   
**show** *?thesis*  
**proof** (*cases*  $C' = \text{efac } x$ )  
**case** *True*  
**moreover have**  $\text{efac } x = \text{efac } C_f$   
**using**  $\langle \text{ord-res.ground-factorizing}^{++} x C_f \rangle$  *ground-factorings-preserves-efac*  
**by** (*metis* *tranclp-into-rtranclp*)  
**ultimately show** *?thesis*  
**by** *simp*  
**next**  
**case** *False*  
**show** *?thesis*  
**using** *C-in-disj*

```

proof (elim disjE)
  assume  $C \in N \cup U_r \cup U_{ef}$ 
  then show ?thesis
    by (smt (verit) C-least-false True  $U_f$ -unproductive
   $x C_f$   $\langle$ is-least-false-clause  $(N \cup U_r \cup U_{ef}) x \rangle$   $\langle$ ord-res.ground-factoring++

  finsert-iff ground-factoring-preserves-efac ground-factorings-preserves-efac
  linorder-cls.Uniq-is-least-in-fset local.factoring(4)
  is-least-false-clause-def
  is-least-false-clause-if-is-least-false-clause-in-union-unproductive
  the1-equality' tranclp-into-rtranclp)
next
  assume  $C \in U_f$ 
  then show ?thesis
    using C-least-false
  using is-least-false-clause-if-is-least-false-clause-in-union-unproductive[
   $OF U_f$ -unproductive]
  by (smt (z3) True  $U_f$ -spec  $\langle$ is-least-false-clause  $(N \cup U_r \cup$ 
   $U_{ef}) x \rangle$ 
   $\langle$ ord-res.ground-factoring++  $x C_f \rangle$  finsert-absorb finsert-iff
  ground-factoring-preserves-efac ground-factorings-preserves-efac
  linorder-cls.Uniq-is-least-in-fset local.factoring(4)
  is-least-false-clause-def the1-equality' tranclp-into-rtranclp)

  qed
  qed
  qed
  qed
  qed

  ultimately show ?thesis
    by auto
  qed

  ultimately show ?thesis
    by metis
next
  case False
  let  $?U_f' =$  finsert  $C' U_f$ 

  have ?match  $s1' s2$ 
proof –
  have finsert  $C' s1 = N \cup U_r \cup U_{ef} \cup ?U_f'$ 
    unfolding s1-def by simp

  moreover have  $\exists C \in N \cup U_r \cup U_{ef}$ .
    ord-res.ground-factoring++  $C C_f \wedge C_f \neq$  efac  $C_f \wedge$ 
    (efac  $C_f \in U_{ef} \vee$  is-least-false-clause  $(N \cup U_r \cup U_{ef}) C$ )
    if  $C_f \in ?U_f'$  for  $C_f$ 
proof –

```

```

from  $\langle C_f \mid \in \mid ?U_f \rangle$  have  $C_f = C' \vee C_f \mid \in \mid U_f$ 
  by simp
thus ?thesis
proof (elim disjE)
  assume  $C_f = C'$ 
  thus ?thesis
    using C-in-disj
  proof (elim disjE)
    assume  $C \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}$ 
    show ?thesis
      proof (intro beXI conjI)
        show  $C \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}$ 
          using  $\langle C \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \rangle$  .
        next
          show ord-res.ground-factorizing++ C Cf
            using  $\langle \text{ord-res.ground-factorizing } C \ C' \rangle \langle C_f = C' \rangle$  by simp
          next
            show  $C_f \neq \text{efac } C_f$ 
              using False  $\langle C_f = C' \rangle$  by argo
            next
              have is-least-false-clause  $(N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ C$ 
                using factoring
              using Interp-eq  $\langle C \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \rangle$  linorder-clis-least-in-filter-iff
                by (simp add: s1-def is-least-false-clause-def)
              thus efac  $C_f \mid \in \mid U_{ef} \vee \text{is-least-false-clause } (N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ C$  ..
              qed
            next
              assume  $C \mid \in \mid U_f$ 
              then obtain  $x$  where
                 $x \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}$  and
                ord-res.ground-factorizing++ x C and
                 $C \neq \text{efac } C$  and
                efac  $C \mid \in \mid U_{ef} \vee \text{is-least-false-clause } (N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ x$ 
                using Uf-spec by metis

              show ?thesis
              proof (intro beXI conjI)
                show  $x \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef}$ 
                  using  $\langle x \mid \in \mid N \mid \cup \mid U_r \mid \cup \mid U_{ef} \rangle$  .
                next
                  show ord-res.ground-factorizing++ x Cf
                    using  $\langle \text{ord-res.ground-factorizing++ } x \ C \rangle \langle \text{ord-res.ground-factorizing } C$ 
                     $C' \rangle \langle C_f = C' \rangle$ 
                    by simp
                  next
                    show  $C_f \neq \text{efac } C_f$ 
                      using False  $\langle C_f = C' \rangle$  by argo
                    next
                      show efac  $C_f \mid \in \mid U_{ef} \vee \text{is-least-false-clause } (N \mid \cup \mid U_r \mid \cup \mid U_{ef}) \ x$ 

```

```

    using ⟨efac C |∈| Uef ∨ is-least-false-clause (N |∪| Ur |∪| Uef) x⟩
  proof (elim disjE)
    assume efac C |∈| Uef

    moreover have efac C = efac Cf
      unfolding ⟨Cf = C'⟩
    using ⟨ord-res.ground-factorizing C C'⟩ ground-factorizing-preserves-efac

  by metis

    ultimately show ?thesis
      by argo
    next
      assume is-least-false-clause (N |∪| Ur |∪| Uef) x
      thus ?thesis
        by argo
    qed
  qed
next
assume Cf |∈| Uf
thus ?thesis
  using Uf-spec by metis
qed
qed

ultimately have ord-res-1-matches-ord-res-2 (finsert C' s1) (N, (Ur, Uef))
  unfolding ord-res-1-matches-ord-res-2.simps by metis
thus ?thesis
  unfolding s2-def ⟨s1' = finsert C' s1⟩ by simp
qed

moreover have ?order (?measure s1') (?measure s1)
proof –
  have ?measure s1 = C
    unfolding ord-res-1-measure-def
    using C-least-false[folded s1-def]
    by (metis (mono-tags, lifting) linorder-cls.Uniq-is-least-in-fset
        is-least-false-clause-def the1-equality' the-equality)

  moreover have ?measure s1' = C'
proof –
  have C' <c C
    using factoring ord-res.ground-factorizing-smaller-conclusion by metis

  have unproductive: ∀ x ∈ {C'}. ord-res.production (fset s1 ∪ {C'}) x = {}
  using ⟨C' ≠ efac C'⟩
  by (simp add: nex-strictly-maximal-pos-lit-if-neq-efac
      unproductive-if-nex-strictly-maximal-pos-lit)

```

```

have Interp-eq:  $\wedge D. \text{ord-res-Interp } (\text{fset } s1) D = \text{ord-res-Interp } (\text{fset } (\text{finsert } C' s1)) D$ 
using Interp-union-unproductive[of fset s1 {C'}, folded union-fset, OF finite-fset - unproductive]
by simp

have is-least-false-clause (finsert C' s1) C'
unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
have  $\neg \text{ord-res-Interp } (\text{fset } s1) C \models C$ 
using C-least-false s1-def is-least-false-clause-def
linorder-cls.is-least-in-ffilter-iff by simp
thus  $\neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C' s1)) C' \models C'$ 
by (metis Interp-eq <C' <_c C> local.factorizing(4)
ord-res.entailed-clause-stays-entailed
ord-res.set-mset-eq-set-mset-if-ground-factoring subset-refl true-cls-mono)
next
fix y
assume y  $\in$  finsert C' s1 and y  $\neq$  C' and
y-false:  $\neg \text{ord-res-Interp } (\text{fset } (\text{finsert } C' s1)) y \models y$ 
hence y  $\in$  s1
by simp

moreover have  $\neg \text{ord-res-Interp } (\text{fset } s1) y \models y$ 
using y-false
unfolding Interp-eq .

ultimately have  $C \preceq_c y$ 
using C-least-false[folded s1-def, unfolded is-least-false-clause-def]
unfolding linorder-cls.is-least-in-ffilter-iff
by force
thus  $C' \prec_c y$ 
using  $\langle C' \prec_c C \rangle$  by order
qed simp
thus ?thesis
unfolding ord-res-1-measure-def <s1' = finsert C' s1>
by (metis (mono-tags, lifting) linorder-cls.Uniq-is-least-in-fset
is-least-false-clause-def the1-equality' the-equality)
qed

moreover have  $C' \subset\# C$ 
using factoring ord-res.strict-subset-mset-if-ground-factoring by metis

ultimately show ?thesis
unfolding s1-def by simp
qed

ultimately show ?thesis
by argo

```



**qed**  
**next**  
**case** (*resolution C L D CD*)  
  
**have** *is-least-false-clause s1 C*  
**using** *resolution unfolding is-least-false-clause-def* **by** *argo*  
**hence**  
 $C \in s1$  **and**  
 $\neg \text{ord-res-Interp} (fset\ s1)\ C \models C$  **and**  
 $\forall x \in s1. \neg \text{ord-res-Interp} (fset\ s1)\ x \models x \longrightarrow x \neq C \longrightarrow C \prec_c x$   
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-filter-iff* **by** *simp-all*  
  
**have**  $C \notin U_f$   
**proof** (*rule notI*)  
**assume**  $C \in U_f$   
**then show** *False*  
**by** (*metis U<sub>f</sub>-spec Uniq-D is-pos-def linorder-lit.Uniq-is-maximal-in-mset*  
*local.resolution(2)*  
*local.resolution(3) efac-spec*)  
**qed**  
**hence**  $C \in N \cup U_r \cup U_{ef}$   
**using**  $\langle C \in s1 \rangle$  **by** (*simp add: s1-def*)  
  
**have** *C-least-false: is-least-false-clause (N |U| U<sub>r</sub> |U| U<sub>ef</sub> |U| U<sub>f</sub>) C*  
**using** *resolution s1-def* **by** *metis*  
**hence** *C-least-false': is-least-false-clause (N |U| U<sub>r</sub> |U| U<sub>ef</sub>) C*  
**using** *is-least-false-clause-if-is-least-false-clause-in-union-unproductive[*  
*OF U<sub>f</sub>-unproductive ⟨C ∈ N |U| U<sub>r</sub> |U| U<sub>ef⟩</sub>* **by** *argo*  
  
**define** *s2'* **where**  
 $s2' = (N, (finsert\ CD\ U_r,\ U_{ef}))$   
  
**have** *ord-res-2-step<sup>++</sup> s2 s2'*  
**proof** –  
**have**  $D \notin U_f$   
**proof** (*rule notI*)  
**assume**  $D \in U_f$   
**thus** *False*  
**using**  $\langle \text{ord-res.production} (fset\ s1)\ D = \{atm-of\ L\} \rangle$   
**using** *U<sub>f</sub>-unproductive s1-def* **by** *simp*  
**qed**  
**hence** *D-in: D ∈ N |U| U<sub>r</sub> |U| U<sub>ef</sub>*  
**using**  $\langle D \in s1 \rangle$  [*unfolded s1-def*] **by** *simp*  
  
**have** *ord-res-2 N (U<sub>r</sub>, U<sub>ef</sub>) (finsert CD U<sub>r</sub>, U<sub>ef</sub>)*  
**proof** (*rule ord-res-2.resolution*)  
**show** *is-least-false-clause (N |U| U<sub>r</sub> |U| U<sub>ef</sub>) C*  
**using** *C-least-false'*.  
**next**

```

    show ord-res.is-maximal-lit L C
      using resolution by argo
  next
    show is-neg L
      using resolution by argo
  next
    show  $D \in N \cup U_r \cup U_{ef}$ 
      using D-in .
  next
    show  $D \prec_c C$ 
      using resolution by argo
  next
    show ord-res.production (fset (N |U| U_r |U| U_{ef})) D = {atm-of L}
      using resolution
      unfolding s1-def
      using production-union-unproductive[OF finite-fset finite-fset - D-in]
U_f-unproductive
      by (metis (no-types, lifting) union-fset)
  next
    show ord-res.ground-resolution C D CD
      using resolution by argo
qed simp-all
thus ?thesis
  by (auto simp: s2-def s2'-def ord-res-2-step.simps)
qed

moreover have ?match s1' s2'
proof -
  have finsert CD (N |U| U_r |U| U_{ef} |U| U_f) = N |U| finsert CD U_r |U| U_{ef}
|U| U_f
  by simp

moreover have  $\exists C \in N \cup U_r \cup U_{ef}$ .
ord-res.ground-factoring++ C C_f  $\wedge$  C_f  $\neq$  efac C_f  $\wedge$ 
(efac C_f  $\in U_{ef} \vee$  is-least-false-clause (N |U| finsert CD U_r |U| U_{ef}) C)
if C_f  $\in U_f$  for C_f
proof -
  obtain x where
    x  $\in N \cup U_r \cup U_{ef}$  and
    ord-res.ground-factoring++ x C_f and
    C_f  $\neq$  efac C_f and
    efac C_f  $\in U_{ef} \vee$  is-least-false-clause (N |U| U_r |U| U_{ef}) x
  using  $\langle C_f \in U_f \rangle$  U_f-spec by metis
  show ?thesis
proof (intro bexI conjI)
  show x  $\in N \cup U_r \cup U_{ef}$ 
    using  $\langle x \in N \cup U_r \cup U_{ef} \rangle$  by simp
  next
  show ord-res.ground-factoring++ x C_f

```

```

    using ⟨ord-res.ground-factorizing++ x Cf⟩ .
  next
  show Cf ≠ efac Cf
  using ⟨Cf ≠ efac Cf⟩ .
  next
  show ⟨efac Cf |∈| Uef ∨ is-least-false-clause (N |∪| finset CD Ur |∪|
Uef) x⟩
  using ⟨efac Cf |∈| Uef ∨ is-least-false-clause (N |∪| Ur |∪| Uef) x⟩ ⟨x
|∈| N |∪| Ur |∪| Uef⟩
  by (metis (no-types, lifting) C-least-false' Uniq-D ⟨ord-res.ground-factorizing++
x Cf⟩
      is-least-false-clause-def is-pos-def linorder-cls.Uniq-is-least-in-fset
      linorder-lit.Uniq-is-maximal-in-mset local.resolution(2) local.resolution(3)
      ord-res.ground-factorizing.cases tranclpD)
  qed
  qed

  ultimately show ?thesis
  unfolding s1-def resolution s2'-def by auto
  qed

  ultimately show ?thesis
  by metis
  qed
  qed

theorem bisimulation-ord-res-1-ord-res-2:
  defines match ≡ λi s1 s2. i = ord-res-1-measure s1 ∧ ord-res-1-matches-ord-res-2
s1 s2
  shows ∃ (MATCH :: nat × nat ⇒ 'f ord-res-1-state ⇒ 'f ord-res-2-state ⇒ bool)
 $\mathcal{R}_f \mathcal{R}_b$ .
  bisimulation ord-res-1 ord-res-1-final ord-res-2-step ord-res-2-final MATCH  $\mathcal{R}_f$ 
 $\mathcal{R}_b$ 
  proof (rule ex-bisimulation-from-forward-simulation)
  show right-unique ord-res-1
  using right-unique-ord-res-1 .
  next
  show right-unique ord-res-2-step
  using right-unique-ord-res-2-step .
  next
  show ∀ s1. ord-res-1-final s1 ⟶ (∃ s1'. ord-res-1 s1 s1')
  using ord-res-1-semantics.final-finished
  by (simp add: finished-def)
  next
  show ∀ s2. ord-res-2-final s2 ⟶ (∃ s2'. ord-res-2-step s2 s2')
  using ord-res-2-semantics.final-finished
  by (simp add: finished-def)
  next
  show ∀ i s1 s2. match i s1 s2 ⟶ ord-res-1-final s1 = ord-res-2-final s2

```

```

    using ord-res-1-final-iff-ord-res-2-final
    by (simp add: match-def)
next
show  $\forall i\ s1\ s2. \text{match } i\ s1\ s2 \longrightarrow$ 
    safe-state ord-res-1 ord-res-1-final s1  $\wedge$ 
    safe-state ord-res-2-step ord-res-2-final s2
proof (intro allI impI)
  fix i s1 S2
  assume match i s1 S2

  then obtain N s2 where
    S2-def:  $S2 = (N, s2)$  and
    i = ord-res-1-measure s1 and
    match: ord-res-1-matches-ord-res-2 s1 S2
  unfolding match-def
  by (metis prod.exhaust)

  show safe-state ord-res-1 ord-res-1-final s1  $\wedge$  safe-state ord-res-2-step ord-res-2-final
  S2
  using safe-states-if-ord-res-1-matches-ord-res-2[OF match] .
qed
next
show wfP ( $\subset\#$ )
  using wfp-subset-mset .
next
show  $\forall i\ s1\ s2\ s1'. \text{match } i\ s1\ s2 \longrightarrow \text{ord-res-1 } s1\ s1' \longrightarrow$ 
  ( $\exists i'\ s2'. \text{ord-res-2-step}^{++} s2\ s2' \wedge \text{match } i'\ s1'\ s2'$ )  $\vee$  ( $\exists i'. \text{match } i'\ s1'\ s2 \wedge$ 
   $i' \subset\# i$ )
proof (intro allI impI)
  fix i s1 S2 s1'
  assume match i s1 S2
  then obtain N s2 where
    S2-def:  $S2 = (N, s2)$  and i = ord-res-1-measure s1 and ord-res-1-matches-ord-res-2
  s1 S2
  unfolding match-def
  by (metis prod.exhaust)

  moreover assume ord-res-1 s1 s1'

  ultimately have ( $\exists S2'. \text{ord-res-2-step}^{++} S2\ S2' \wedge \text{ord-res-1-matches-ord-res-2}$ 
   $s1'\ S2'$ )  $\vee$ 
  ord-res-1-matches-ord-res-2 s1' S2  $\wedge$  ord-res-1-measure s1'  $\subset\#$  ord-res-1-measure
  s1
  using forward-simulation by metis

  thus ( $\exists i'\ S2'. \text{ord-res-2-step}^{++} S2\ S2' \wedge \text{match } i'\ s1'\ S2'$ )  $\vee$  ( $\exists i'. \text{match } i'$ 
   $s1'\ S2 \wedge i' \subset\# i$ )
  unfolding S2-def prod.case
  using lift-tranclp-to-pairs-with-constant-fst[of ord-res-2 N s2]

```

by (*metis (mono-tags, lifting) <i = ord-res-1-measure s1> match-def*)  
 qed  
 qed  
 end

## 29 ORD-RES-3 (full resolve)

**type-synonym** 'f ord-res-3-state = 'f gclause fset × 'f gclause fset × 'f gclause fset

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-2-matches-ord-res-3* :: - ⇒ 'f ord-res-3-state ⇒ bool **where**  
 (∀ C |∈| U<sub>pr</sub>. ∃ D1 |∈| N |∪| U<sub>er</sub> |∪| U<sub>ef</sub>. ∃ D2 |∈| N |∪| U<sub>er</sub> |∪| U<sub>ef</sub>.  
 (ground-resolution D1)<sup>++</sup> D2 C ∧ C ≠ eres D1 D2 ∧ eres D1 D2 |∈| U<sub>er</sub>)  
 ⇒  
*ord-res-2-matches-ord-res-3* (N, (U<sub>pr</sub> |∪| U<sub>er</sub>, U<sub>ef</sub>)) (N, (U<sub>er</sub>, U<sub>ef</sub>))

**lemma** *ord-res-2-final-iff-ord-res-3-final*:

**assumes** *match*: *ord-res-2-matches-ord-res-3* S<sub>2</sub> S<sub>3</sub>

**shows** *ord-res-2-final* S<sub>2</sub> ↔ *ord-res-3-final* S<sub>3</sub>

**using** *match*

**proof** (*cases* S<sub>2</sub> S<sub>3</sub> *rule*: *ord-res-2-matches-ord-res-3.cases*)

**case** *match-hyps*: (1 U<sub>pr</sub> N U<sub>er</sub> U<sub>ef</sub>)

**note** *invars* = *match-hyps*(3-)

**have** *U<sub>pr</sub>-spec*: ∀ C |∈| U<sub>pr</sub>. ∃ D1 |∈| N |∪| U<sub>er</sub> |∪| U<sub>ef</sub>. ∃ D2 |∈| N |∪| U<sub>er</sub> |∪| U<sub>ef</sub>.  
 (ground-resolution D1)<sup>++</sup> D2 C ∧ C ≠ eres D1 D2 ∧ eres D1 D2 |∈| U<sub>er</sub>  
**using** *invars* **by** *argo*

**have** *least-false-spec*: *is-least-false-clause* (N |∪| U<sub>pr</sub> |∪| U<sub>er</sub> |∪| U<sub>ef</sub>) =

*is-least-false-clause* (N |∪| U<sub>er</sub> |∪| U<sub>ef</sub>)

**using** *invars* *is-least-false-clause-conv-if-partial-resolution-invariant* **by** *metis*

**have** *U<sub>pr</sub>-unproductive*: ∀ C |∈| U<sub>pr</sub>. *ord-res.production* (fset (N |∪| U<sub>er</sub> |∪| U<sub>ef</sub> |∪| U<sub>pr</sub>)) C = {}

**proof** (*intro ballI*)

**fix** C

**assume** C |∈| U<sub>pr</sub>

**hence** ∄ L. *is-pos* L ∧ *ord-res.is-strictly-maximal-lit* L C

**using** *U<sub>pr</sub>-spec*

**by** (*metis eres-eq-after-tranclp-ground-resolution nex-strictly-maximal-pos-lit-if-neq-eres*)

**thus** *ord-res.production* (fset (N |∪| U<sub>er</sub> |∪| U<sub>ef</sub> |∪| U<sub>pr</sub>)) C = {}

**using** *unproductive-if-nex-strictly-maximal-pos-lit* **by** *metis*

qed

**hence**  $Interp-N-U_r-U_{ef}\text{-eq-}Interp-N-U_{er}-U_{ef}: \wedge C$ .  
 $ord\text{-res-}Interp (fset (N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef})) C =$   
 $ord\text{-res-}Interp (fset (N \mid \cup U_{er} \mid \cup U_{ef})) C$   
**using**  $Interp\text{-union-unproductive}[OF\ finite\text{-fset}\ finite\text{-fset},\ folded\ union\text{-fset},$   
 $of\ U_{pr}\ N \mid \cup U_{er} \mid \cup U_{ef}]$   
**by** ( $simp\ add: funion\text{-left-commute}\ sup\text{-commute}$ )

**have**  $ex\text{-false-clause} (fset (N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef})) \longleftrightarrow$   
 $ex\text{-false-clause} (fset (N \mid \cup U_{er} \mid \cup U_{ef}))$

**proof** ( $rule\ iffI$ )

**assume**  $ex\text{-false-clause} (fset (N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}))$   
**then obtain**  $C$  **where**  $is\text{-least-false-clause} (N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}) C$   
**using**  $obtains\text{-least-false-clause-if-ex-false-clause}$  **by**  $metis$   
**thus**  $ex\text{-false-clause} (fset (N \mid \cup U_{er} \mid \cup U_{ef}))$   
**using**  $least\text{-false-spec}\ ex\text{-false-clause-iff}$  **by**  $metis$

**next**

**assume**  $ex\text{-false-clause} (fset (N \mid \cup U_{er} \mid \cup U_{ef}))$   
**thus**  $ex\text{-false-clause} (fset (N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}))$   
**unfolding**  $ex\text{-false-clause-def}$   
**unfolding**  $Interp-N-U_r-U_{ef}\text{-eq-}Interp-N-U_{er}-U_{ef}$   
**by**  $auto$

**qed**

**moreover have**  $\{\#\} \mid \in N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef} \longleftrightarrow \{\#\} \mid \in N \mid \cup U_{er} \mid \cup U_{ef}$

**proof** ( $rule\ iffI$ )

**assume**  $\{\#\} \mid \in N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}$   
**hence**  $\{\#\} \mid \in N \mid \cup U_{er} \mid \cup U_{ef} \vee \{\#\} \mid \in U_{pr} \mid \cup U_{er}$   
**by**  $auto$

**thus**  $\{\#\} \mid \in N \mid \cup U_{er} \mid \cup U_{ef}$

**proof** ( $elim\ disjE$ )

**assume**  $\{\#\} \mid \in N \mid \cup U_{er}$   
**thus**  $?thesis$   
**by**  $auto$

**next**

**have**  $\{\#\} \mid \notin U_{pr}$   
**using**  $U_{pr}\text{-spec}[rule\text{-format},\ of\ \{\#\}]$   
**by** ( $metis\ eres\text{-eq-after-tranclp-ground-resolution}\ eres\text{-mempty-right}$ )  
**moreover assume**  $\{\#\} \mid \in U_{pr} \mid \cup U_{er}$   
**ultimately show**  $?thesis$   
**by**  $simp$

**qed**

**next**

**assume**  $\{\#\} \mid \in N \mid \cup U_{er} \mid \cup U_{ef}$   
**then show**  $\{\#\} \mid \in N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}$   
**by**  $auto$

**qed**

**ultimately have**  $ord\text{-res-final} (N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}) = ord\text{-res-final} (N$

$|\cup| U_{er} |\cup| U_{ef}$ )

**unfolding** *ord-res-final-def* **by** *argo*

**thus** *ord-res-2-final*  $S_2 \longleftrightarrow$  *ord-res-3-final*  $S_3$

**unfolding** *match-hyps(1,2)*

**by** (*simp add: ord-res-2-final.simps ord-res-3-final.simps sup-assoc*)

**qed**

**definition** *ord-res-2-measure* **where**

*ord-res-2-measure*  $S1 =$

(*let* ( $N, (U_r, U_{ef}) = S1$  *in*

(*if*  $\exists C.$  *is-least-false-clause* ( $N |\cup| U_r |\cup| U_{ef}$ )  $C$  *then*

*The* (*is-least-false-clause* ( $N |\cup| U_r |\cup| U_{ef}$ ))

*else*

{#}))

**definition** *resolvent-at* **where**

*resolvent-at*  $C D i = (THE CD. (ground-resolution C \rightsquigarrow i) D CD)$

**lemma** *resolvent-at-0[simp]*: *resolvent-at*  $C D 0 = D$

**by** (*simp add: resolvent-at-def*)

**lemma** *resolvent-at-less-cls-resolvent-at*:

**assumes** *reso-at*: (*ground-resolution*  $C \rightsquigarrow n$ )  $D CD$

**assumes**  $i < j$  **and**  $j \leq n$

**shows** *resolvent-at*  $C D j \prec_c$  *resolvent-at*  $C D i$

**proof** –

**obtain**  $j'$  **where**

$j = i + Suc j'$

**using**  $\langle i < j \rangle$  **by** (*metis less-iff-Suc-add nat-arith.suc1*)

**obtain**  $n'$  **where**

$n = j + n'$

**using**  $\langle j \leq n \rangle$  **by** (*metis le-add-diff-inverse*)

**obtain**  $CD_i CD_j CD_n$  **where**

(*ground-resolution*  $C \rightsquigarrow i$ )  $D CD_i$  **and**

(*ground-resolution*  $C \rightsquigarrow Suc j'$ )  $CD_i CD_j$

(*ground-resolution*  $C \rightsquigarrow n'$ )  $CD_j CD_n$

**using** *reso-at*  $\langle n = j + n' \rangle \langle j = i + Suc j' \rangle$  **by** (*metis relpowp-plusD*)

**have**  $*$ : *resolvent-at*  $C D i = CD_i$

**unfolding** *resolvent-at-def*

**using**  $\langle (ground-resolution C \rightsquigarrow i) D CD_i \rangle$

**by** (*simp add: Uniq-ground-resolution Uniq-relpowp the1-equality'*)

**have** (*ground-resolution*  $C \rightsquigarrow j$ )  $D CD_j$

**unfolding**  $\langle j = i + Suc j' \rangle$

**using**  $\langle (ground-resolution C \rightsquigarrow i) D CD_i \rangle \langle (ground-resolution C \rightsquigarrow Suc j') \rangle$

$CD_i CD_j$   
**by** (*metis relpowp-trans*)  
**hence** \*\*: *resolvent-at C D j = CD<sub>j</sub>*  
**unfolding** *resolvent-at-def*  
**by** (*simp add: Uniq-ground-resolution Uniq-relpowp the1-equality'*)  
  
**have** (*ground-resolution C*)<sup>++</sup>  $CD_i CD_j$   
**using**  $\langle (ground-resolution C \rightsquigarrow Suc j) CD_i CD_j \rangle$   
**by** (*metis Zero-not-Suc tranclp-if-relpowp*)  
**hence**  $CD_j \prec_c CD_i$   
**using** *resolvent-lt-right-premise-if-tranclp-ground-resolution* **by** *metis*  
**thus** *?thesis*  
**unfolding** \* \*\* .  
**qed**

**lemma**

**assumes** *reso-at: (ground-resolution C  $\rightsquigarrow$  n) D CD* **and**  $i < n$

**shows**

*left-premise-lt-resolvent-at: C  $\prec_c$  resolvent-at C D i* **and**

*max-lit-resolvent-at:*

*ord-res.is-maximal-lit L D  $\implies$  ord-res.is-maximal-lit L (resolvent-at C D i)*

**and**

*nex-pos-strictly-max-lit-in-resolvent-at:*

$\nexists L$ . *is-pos L  $\wedge$  ord-res.is-strictly-maximal-lit L (resolvent-at C D i)* **and**

*ground-resolution-resolvent-at-resolvent-at-Suc:*

*ground-resolution C (resolvent-at C D i) (resolvent-at C D (Suc i))* **and**

*relpowp-to-resolvent-at: (ground-resolution C  $\rightsquigarrow$  i) D (resolvent-at C D i)*

**proof** –

**obtain**  $j$  **where** *n-def: n = i + Suc j*

**using**  $\langle i < n \rangle$  *less-natE* **by** *auto*

**obtain**  $CD'$  **where** (*ground-resolution C  $\rightsquigarrow$  i*)  $D CD'$  **and** (*ground-resolution C  $\rightsquigarrow$  Suc j*)  $CD' CD$

**using** *reso-at n-def* **by** (*metis relpowp-plusD*)

**have** *resolvent-at C D i = CD'*

**unfolding** *resolvent-at-def*

**using**  $\langle (ground-resolution C \rightsquigarrow i) D CD' \rangle$

**by** (*simp add: Uniq-ground-resolution Uniq-relpowp the1-equality'*)

**have**  $C \prec_c CD'$

**proof** (*rule left-premise-lt-right-premise-if-tranclp-ground-resolution*)

**show** (*ground-resolution C*)<sup>++</sup>  $CD' CD$

**using**  $\langle (ground-resolution C \rightsquigarrow Suc j) CD' CD \rangle$

**by** (*metis Zero-not-Suc tranclp-if-relpowp*)

**qed**

**thus**  $C \prec_c$  *resolvent-at C D i*

**unfolding**  $\langle resolvent-at C D i = CD' \rangle$  **by** *argo*



**show** *ord-res.is-maximal-lit L (resolvent-at C D i) if ord-res.is-maximal-lit L D*  
**unfolding**  $\langle \text{resolvent-at } C \ D \ i = CD' \rangle$   
**using** *that*  
**using**  $\langle (\text{ground-resolution } C \ \sim i) \ D \ CD' \rangle$   
**by** (*smt (verit, ccfv-SIG) Uniq-ground-resolution Uniq-relpowp Zero-not-Suc*  
 $\langle \wedge \text{thesis. } (\wedge CD'. \llbracket (\text{ground-resolution } C \ \sim i) \ D \ CD'; (\text{ground-resolution } C$   
 $\sim \text{Suc } j) \ CD' \ CD \rrbracket \implies \text{thesis} \implies \text{thesis} \rangle$   
*linorder-lit.Uniq-is-greatest-in-mset linorder-lit.Uniq-is-maximal-in-mset lit-*  
*eral.sel(1)*  
*n-def relpowp-ground-resolutionD reso-at the1-equality' zero-eq-add-iff-both-eq-0)*

**show**  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ (\text{resolvent-at } C \ D \ i)$   
**unfolding**  $\langle \text{resolvent-at } C \ D \ i = CD' \rangle$   
**by** (*metis Zero-not-Suc*  $\langle (\text{ground-resolution } C \ \sim \text{Suc } j) \ CD' \ CD \rangle$   
*nex-strictly-maximal-pos-lit-if-resolvable tranclpD tranclp-if-relpowp)*

**show** *ground-resolution C (resolvent-at C D i) (resolvent-at C D (Suc i))*  
**proof** –  
**obtain**  $CD''$  **where** *ground-resolution C CD' CD'' and (ground-resolution C*  
 $\sim j) \ CD'' \ CD$   
**using**  $\langle (\text{ground-resolution } C \ \sim \text{Suc } j) \ CD' \ CD \rangle$  **by** (*metis relpowp-Suc-D2*)  
**hence**  $\langle (\text{ground-resolution } C \ \sim \text{Suc } i) \ D \ CD'' \rangle$   
**using**  $\langle (\text{ground-resolution } C \ \sim i) \ D \ CD' \rangle$  **by** *auto*  
**hence** *resolvent-at C D (Suc i) = CD''*  
**unfolding** *resolvent-at-def*  
**by** (*meson Uniq-ground-resolution Uniq-relpowp the1-equality'*)

**show** *?thesis*  
**unfolding**  $\langle \text{resolvent-at } C \ D \ i = CD' \rangle \ \langle \text{resolvent-at } C \ D \ (\text{Suc } i) = CD'' \rangle$   
**using**  $\langle \text{ground-resolution } C \ CD' \ CD'' \rangle$  .

**qed**

**show**  $\langle (\text{ground-resolution } C \ \sim i) \ D \ (\text{resolvent-at } C \ D \ i) \rangle$   
**using**  $\langle (\text{ground-resolution } C \ \sim i) \ D \ CD' \rangle \ \langle \text{resolvent-at } C \ D \ i = CD' \rangle$  **by** *argo*  
**qed**

**definition** *resolvents-upto where*

*resolvents-upto C D n = resolvent-at C D |' fset-upto (Suc 0) n*

**lemma** *resolvents-upto-0[simp]:*

*resolvents-upto C D 0 = {||}*

**by** (*simp add: resolvents-upto-def*)

**lemma** *resolvents-upto-Suc[simp]:*

*resolvents-upto C D (Suc n) = finsert (resolvent-at C D (Suc n)) (resolvents-upto C D n)*

**by** (*simp add: resolvents-upto-def*)

**lemma** *resolvent-at-fmember-resolvents-upto:*

**assumes**  $k \neq 0$   
**shows** *resolvent-at*  $C D k \mid \in \mid$  *resolvents-upto*  $C D k$   
**unfolding** *resolvents-upto-def*  
**proof** (*rule fimageI*)  
**show**  $k \mid \in \mid$  *fset-upto* (*Suc 0*)  $k$   
**using** *assms* **by** *simp*  
**qed**

**lemma** *backward-simulation-2-to-3*:  
**fixes** *match measure less*  
**defines** *match*  $\equiv$  *ord-res-2-matches-ord-res-3*  
**assumes**  
*match*: *match*  $S2 S3$  **and**  
*step2*: *ord-res-3-step*  $S3 S3'$   
**shows**  $(\exists S2'. \text{ord-res-2-step}^{++} S2 S2' \wedge \text{match } S2' S3')$   
**using** *match*[*unfolded match-def*]  
**proof** (*cases S2 S3 rule: ord-res-2-matches-ord-res-3.cases*)  
**case** *match-hyps*:  $(1 U_{pr} N U_{er} U_{ef})$   
**note** *invars* = *match-hyps*(3-)

**have**  $U_{pr}\text{-spec}: \forall C \mid \in \mid U_{pr}. \exists D1 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}. \exists D2 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}.$   
 $(\text{ground-resolution } D1)^{++} D2 C \wedge C \neq \text{eres } D1 D2 \wedge \text{eres } D1 D2 \mid \in \mid U_{er}$   
**using** *invars* **by** *argo*

**hence** *C-not-least-with-partial*:  $\neg$  *is-least-false-clause*  $(N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef}) C$   
**if** *C-in*:  $C \mid \in \mid U_{pr}$  **for**  $C$   
**proof** –  
**obtain**  $D1 D2$  **where**  
 $D1 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$  **and**  
 $D2 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$  **and**  
 $(\text{ground-resolution } D1)^{++} D2 C$  **and**  
 $C \neq \text{eres } D1 D2$  **and**  
 $\text{eres } D1 D2 \mid \in \mid U_{er}$   
**using**  $U_{pr}\text{-spec}$  *C-in* **by** *metis*

**have**  $\text{eres } D1 C = \text{eres } D1 D2$   
**using**  $\langle (\text{ground-resolution } D1)^{++} D2 C \rangle$  *eres-eq-after-tranclp-ground-resolution*  
**by** *metis*  
**hence**  $\text{eres } D1 C \prec_c C$   
**using**  $\text{eres-le}$ [*of D1 C*]  $\langle C \neq \text{eres } D1 D2 \rangle$  **by** *order*

**show** *?thesis*  
**proof** (*cases ord-res-Interp (fset (N |cup| U<sub>pr</sub> |cup| U<sub>er</sub> |cup| U<sub>ef</sub>)) (eres D1 D2)*)  
 $\models \text{eres } D1 D2$   
**case** *True*  
**then show** *?thesis*  
**by** (*metis (no-types, lifting)*)  $\langle (\text{ground-resolution } D1)^{++} D2 C \rangle \langle \text{eres } D1 C$

= *eres D1 D2*  
   *clause-true-if-eres-true is-least-false-clause-def*  
   *linorder-cls.is-least-in-fset-ffilterD(2)*  
 next  
   case *False*  
   then show *?thesis*  
     by (*metis (mono-tags, lifting) Un-iff*  $\langle \text{eres } D1 \ C = \text{eres } D1 \ D2 \rangle \langle \text{eres } D1 \ C \prec_c C \rangle$   
      $\langle \text{eres } D1 \ D2 \mid \in \mid U_{er} \rangle$  *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
     *linorder-cls.not-less-iff-gr-or-eq sup-fset.rep-eq*)  
 qed  
 qed  
  
 have *least-false-conv: is-least-false-clause* ( $N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ ) =  
   *is-least-false-clause* ( $N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ )  
   using *invars is-least-false-clause-conv-if-partial-resolution-invariant* by *metis*  
  
 have *U<sub>pr</sub>-unproductive:  $\bigwedge N. \forall C \mid \in \mid U_{pr}. \text{ord-res.production } N \ C = \{\}$*   
 proof (*intro ballI*)  
   fix *C*  
   assume  $C \mid \in \mid U_{pr}$   
   hence  $\exists D \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}. (\exists C'. \text{ground-resolution } D \ C \ C')$   
   using *U<sub>pr</sub>-spec* by (*metis eres-eq-after-tranclp-ground-resolution resolvable-if-neq-eres*)  
   hence  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \ C$   
   using *nex-strictly-maximal-pos-lit-if-resolvable* by *metis*  
   thus  $\bigwedge N. \text{ord-res.production } N \ C = \{\}$   
   using *unproductive-if-nex-strictly-maximal-pos-lit* by *metis*  
 qed  
  
 hence *Interp-N-U<sub>r</sub>-U<sub>ef</sub>-eq-Interp-N-U<sub>er</sub>-U<sub>ef</sub>:*  
    $\bigwedge C. \text{ord-res-Interp} (\text{fset} (N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef})) \ C = \text{ord-res-Interp} (\text{fset} (N \mid \cup \mid U_{er} \mid \cup \mid U_{ef})) \ C$   
   using *Interp-union-unproductive*[*OF finite-fset finite-fset, folded union-fset,*  
   *of U<sub>pr</sub> N  $\mid \cup \mid U_{er} \mid \cup \mid U_{ef}$* ]  
   by (*simp add: funion-left-commute sup-commute*)  
  
 have *U<sub>pr</sub>-have-generalization:  $\forall Ca \mid \in \mid U_{pr}. \exists D \mid \in \mid U_{er}. D \prec_c Ca \wedge \{D\} \models_e \{Ca\}$*   
 proof (*intro ballI*)  
   fix *Ca*  
   assume  $Ca \mid \in \mid U_{pr}$   
   then obtain *D1 D2* where  
      $D1 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$  and  
      $D2 \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$  and  
     (*ground-resolution D1*)<sup>++</sup> *D2 Ca* and  
      $Ca \neq \text{eres } D1 \ D2$  and  
      $\text{eres } D1 \ D2 \mid \in \mid U_{er}$   
   using *U<sub>pr</sub>-spec* by *metis*

**have**  $eres\ D1\ D2 = eres\ D1\ Ca$   
**using**  $\langle (ground-resolution\ D1)^{++}\ D2\ Ca \rangle\ eres-eq-after-tranclp-ground-resolution$   
**by** *metis*

**show**  $\exists D\ |\in|\ U_{er}.\ D\ \prec_c\ Ca \wedge \{D\}\ \models_e\ \{Ca\}$   
**proof** (*intro beXI conjI*)  
**have**  $eres\ D1\ Ca\ \preceq_c\ Ca$   
**using** *eres-le* .  
**thus**  $eres\ D1\ D2\ \prec_c\ Ca$   
**using**  $\langle Ca \neq\ eres\ D1\ D2 \rangle\ \langle eres\ D1\ D2 = eres\ D1\ Ca \rangle$  **by** *order*

**next**  
**show**  $\{eres\ D1\ D2\}\ \models_e\ \{Ca\}$   
**using**  $\langle (ground-resolution\ D1)^{++}\ D2\ Ca \rangle\ eres-entails-resolvent$  **by** *metis*

**next**  
**show**  $eres\ D1\ D2\ |\in|\ U_{er}$   
**using**  $\langle eres\ D1\ D2\ |\in|\ U_{er} \rangle$  **by** *simp*

**qed**  
**qed**

**from** *step2* **obtain**  $s3'$  **where**  $S3'$ -*def*:  $S3' = (N, s3')$  **and** *ord-res-3*  $N\ (U_{er}, U_{ef})\ s3'$   
**by** (*auto simp: match-hyps(1,2) elim: ord-res-3-step.cases*)

**show** *?thesis*  
**using**  $\langle ord-res-3\ N\ (U_{er}, U_{ef})\ s3' \rangle$   
**proof** (*cases N (U<sub>er</sub>, U<sub>ef</sub>) s3' rule: ord-res-3.cases*)  
**case** (*factoring C L U<sub>ef</sub>'*)

**define**  $S2'$  **where**  
 $S2' = (N, (U_{pr}\ |\cup|\ U_{er},\ finsert\ (efac\ C)\ U_{ef}))$

**have** *ord-res-2-step*<sup>++</sup>  $S2\ S2'$   
**unfolding** *match-hyps(1,2) S2'-def*  
**proof** (*intro tranclp.r-into-trancl ord-res-2-step.intros ord-res-2.factoring*)  
**show** *is-least-false-clause*  $(N\ |\cup|\ (U_{pr}\ |\cup|\ U_{er})\ |\cup|\ U_{ef})\ C$   
**using**  $\langle is-least-false-clause\ (N\ |\cup|\ U_{er}\ |\cup|\ U_{ef})\ C \rangle$   
**using** *least-false-conv*  
**by** (*metis sup-assoc*)

**next**  
**show** *ord-res.is-maximal-lit*  $L\ C$   
**using** *factoring* **by** *metis*

**next**  
**show** *is-pos*  $L$   
**using** *factoring* **by** *metis*

**next**  
**show**  $finsert\ (efac\ C)\ U_{ef} = finsert\ (efac\ C)\ U_{ef}$   
**by** *argo*

**qed**

**moreover have** *match S2' S3'*  
**unfolding** *S2'-def S3'-def*  
**unfolding** *factoring*  
**unfolding** *match-def*  
**proof** (*rule ord-res-2-matches-ord-res-3.intros*)  
**show**  $\forall Ca \in |U_{pr}.$   
 $\exists D1 \in |N \cup |U_{er} \cup |U_{ef}. \exists D2 \in |N \cup |U_{er} \cup |U_{ef}$   
 $(efac C) U_{ef}.$   
 $(ground-resolution D1)^{++} D2 Ca \wedge Ca \neq eres D1 D2 \wedge eres D1 D2 \in |U_{er}$   
**using** *U<sub>pr</sub>-spec* **by** *auto*  
**qed**

**ultimately show** *?thesis*  
**by** *metis*  
**next**  
**case** (*resolution C L D U<sub>rr</sub>'*)

**have** (*ground-resolution D*)<sup>\*\*</sup> *C (eres D C)  $\nexists x.$  ground-resolution D (eres D C) x*  
**unfolding** *atomize-conj*  
**by** (*metis ex1-eres-eq-full-run-ground-resolution full-run-def*)

**moreover have**  $\exists x.$  *ground-resolution D C x*  
**unfolding** *ground-resolution-def*  
**using** *resolution*  
**by** (*metis Neg-atm-of-iff ex-ground-resolutionI ord-res.mem-productionE singletonI*)

**ultimately have** (*ground-resolution D*)<sup>++</sup> *C (eres D C)*  
**by** (*metis rtranclpD*)

**then obtain** *n where (ground-resolution D  $\sim$  Suc n) C (eres D C)*  
**by** (*metis not0-implies-Suc not-gr-zero tranclp-power*)

**hence** *resolvent-at D C (Suc n) = eres D C*  
**by** (*metis Uniq-ground-resolution Uniq-relpowp resolvent-at-def the1-equality'*)

**have** *steps: k  $\leq$  Suc n  $\implies$  (ord-res-2-step  $\sim$  k)*  
 $(N, U_{pr} \cup |U_{er}, U_{ef}) (N, U_{pr} \cup |U_{er} \cup |U_{ef})$  *resolvents-upto D C k, U<sub>ef</sub>* **for**  
*k*

**proof** (*induction k*)  
**case** *0*  
**show** *?case*  
**by** *simp*  
**next**  
**case** (*Suc k*)  
**have** *k < Suc n*  
**using**  $\langle Suc k \leq Suc n \rangle$  **by** *presburger*  
**hence** *k  $\leq$  Suc n*

**by** *presburger*  
**hence** (*ord-res-2-step*  $\sim k$ ) ( $N, U_{pr} \mid \cup U_{er}, U_{ef}$ )  
 $(N, U_{pr} \mid \cup U_{er} \mid \cup \text{resolvents-upto } D C k, U_{ef})$   
**using** *Suc.IH* **by** *metis*

**moreover have** *ord-res-2-step*  
 $(N, U_{pr} \mid \cup U_{er} \mid \cup \text{resolvents-upto } D C k, U_{ef})$   
 $(N, U_{pr} \mid \cup U_{er} \mid \cup \text{resolvents-upto } D C (\text{Suc } k), U_{ef})$   
**unfolding** *resolvents-upto-Suc*  
**proof** (*intro ord-res-2-step.intros ord-res-2.resolution*)  
**show** *is-least-false-clause* ( $N \mid \cup (U_{pr} \mid \cup U_{er} \mid \cup \text{resolvents-upto } D C k)$   
 $\mid \cup U_{ef}$ )  
 $(\text{resolvent-at } D C k)$   
**using**  $\langle k < \text{Suc } n \rangle$   
**proof** (*induction k*)  
**case** 0  
**have** *is-least-false-clause* ( $N \mid \cup U_{pr} \mid \cup U_{er} \mid \cup U_{ef}$ )  $C$   
**using**  $\langle \text{is-least-false-clause } (N \mid \cup U_{er} \mid \cup U_{ef}) C \rangle$   
**unfolding** *least-false-conv* .  
**thus** *?case*  
**unfolding** *funion-fempty-right funion-assoc[symmetric]*  
**by** *simp*  
**next**  
**case** ( $\text{Suc } k'$ )

**have**  $\bigwedge x. \text{ord-res-Interp } (fset (N \mid \cup (U_{pr} \mid \cup U_{er} \mid \cup \text{resolvents-upto } D$   
 $C (\text{Suc } k')) \mid \cup U_{ef})) x =$   
 $\text{ord-res-Interp } (fset (N \mid \cup U_{er} \mid \cup U_{ef}) \cup fset (U_{pr} \mid \cup \text{resolvents-upto}$   
 $D C (\text{Suc } k')))$   $x$   
**by** (*simp add: funion-left-commute sup-assoc sup-commute*)  
**also have**  $\bigwedge x. \text{ord-res-Interp } (fset (N \mid \cup U_{er} \mid \cup U_{ef}) \cup fset (U_{pr} \mid \cup$   
 $\text{resolvents-upto } D C (\text{Suc } k')))$   $x =$   
 $\text{ord-res-Interp } (fset (N \mid \cup U_{er} \mid \cup U_{ef}))$   $x$   
**proof** (*intro Interp-union-unproductive ballI*)  
**fix**  $x y$  **assume**  $y \mid \in U_{pr} \mid \cup \text{resolvents-upto } D C (\text{Suc } k')$   
**hence**  $y \mid \in U_{pr} \vee y \mid \in \text{resolvents-upto } D C (\text{Suc } k')$   
**by** *blast*  
**thus** *ord-res.production* ( $fset (N \mid \cup U_{er} \mid \cup U_{ef}) \cup fset (U_{pr} \mid \cup$   
 $\text{resolvents-upto } D C (\text{Suc } k'))$ )  $y = \{\}$   
**proof** (*elim disjE*)  
**assume**  $y \mid \in U_{pr}$   
**thus** *?thesis*  
**using** *U<sub>pr</sub>-unproductive* **by** *metis*  
**next**  
**assume**  $y \mid \in \text{resolvents-upto } D C (\text{Suc } k')$   
**then obtain**  $i$  **where**  $i \mid \in fset\text{-upto } (\text{Suc } 0) (\text{Suc } k')$  **and**  $y =$   
 $\text{resolvent-at } D C i$   
**unfolding** *resolvents-upto-def* **by** *blast*

i)

```

have  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L \text{ (resolvent-at } D \ C$ 
proof (rule nex-pos-strictly-max-lit-in-resolvent-at)
  show (ground-resolution  $D \rightsquigarrow \text{Suc } n$ )  $C \text{ (eres } D \ C)$ 
    using  $\langle \text{ground-resolution } D \rightsquigarrow \text{Suc } n \rangle C \text{ (eres } D \ C) \rangle$  .
  next
    have  $i \leq \text{Suc } k'$ 
      using  $\langle i \in | \text{fset-upto } (\text{Suc } 0) (\text{Suc } k') \rangle$  by auto
      thus  $i < \text{Suc } n$ 
      using  $\langle \text{Suc } k' < \text{Suc } n \rangle$  by presburger
    qed

  then show ?thesis
using  $\langle y = \text{resolvent-at } D \ C \ i \rangle$  unproductive-if-nex-strictly-maximal-pos-lit
  by metis
  qed
qed simp-all
finally have Interp-simp:  $\bigwedge x.$ 
   $\text{ord-res-Interp (fset (} N \ | \cup \ (U_{pr} \ | \cup \ U_{er} \ | \cup \ \text{resolvents-upto } D \ C \ (\text{Suc}$ 
 $k')) \ | \cup \ U_{ef})) \ x =$ 
   $\text{ord-res-Interp (fset (} N \ | \cup \ U_{er} \ | \cup \ U_{ef})) \ x$  .

show ?case
  unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
  have  $\text{resolvent-at } D \ C \ (\text{Suc } k') \in | \text{resolvents-upto } D \ C \ (\text{Suc } k')$ 
    using resolvent-at-fmember-resolvents-upto by simp
  thus  $\text{resolvent-at } D \ C \ (\text{Suc } k') \in | N \ | \cup \ (U_{pr} \ | \cup \ U_{er} \ | \cup \ \text{resolvents-upto}$ 
 $D \ C \ (\text{Suc } k')) \ | \cup \ U_{ef}$ 
    by simp
  next

    show  $\neg \text{ord-res-Interp (fset (} N \ | \cup \ (U_{pr} \ | \cup \ U_{er} \ | \cup \ \text{resolvents-upto } D$ 
 $C \ (\text{Suc } k')) \ | \cup \ U_{ef}))$ 
       $(\text{resolvent-at } D \ C \ (\text{Suc } k')) \models \text{resolvent-at } D \ C \ (\text{Suc } k')$ 
    unfolding Interp-simp
    by (metis (no-types, lifting) Suc.prem0 Zero-not-Suc)
     $\langle \text{ground-resolution } D \rightsquigarrow \text{Suc } n \rangle C \text{ (eres } D \ C) \rangle$  clause-true-if-resolved-true
      insert-not-empty is-least-false-clause-def
      linorder-cls.is-least-in-fset-ffilterD(2) local.resolution(2) lo-
 $\text{cal.resolution}(7)$ 
      relpowp-to-resolvent-at tranclp-if-relpowp)
  next
    fix  $y$ 
    assume  $y \neq \text{resolvent-at } D \ C \ (\text{Suc } k')$ 
    assume  $\neg \text{ord-res-Interp (fset (} N \ | \cup \ (U_{pr} \ | \cup \ U_{er} \ | \cup \ \text{resolvents-upto}$ 
 $D \ C \ (\text{Suc } k')) \ | \cup \ U_{ef})) \ y \models y$ 
    hence  $\neg \text{ord-res-Interp (fset (} N \ | \cup \ U_{er} \ | \cup \ U_{ef})) \ y \models y$ 
    unfolding Interp-simp .

```

**hence**  $\neg \text{ord-res-Interp } (fset (N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef})) y \models y$   
**using** *Interp-N-U<sub>r</sub>-U<sub>ef</sub>-eq-Interp-N-U<sub>er</sub>-U<sub>ef</sub>* **by** *metis*

**assume**  $y \mid \in \mid N \mid \cup \mid (U_{pr} \mid \cup \mid U_{er} \mid \cup \mid \text{resolvents-upto } D \ C \ (Suc \ k')) \mid \cup \mid$   
 $U_{ef}$   
**hence**  $y \mid \in \mid N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef} \vee y \mid \in \mid \text{resolvents-upto } D \ C \ (Suc$   
 $k')$

**by** *auto*  
**thus** *resolvent-at D C (Suc k')  $\prec_c$  y*  
**proof** (*elim disjE*)  
**assume**  $y \mid \in \mid N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$   
**have**  $C \preceq_c y$   
**proof** (*cases y = C*)  
**case** *True*  
**thus** *?thesis*  
**by** *order*  
**next**  
**case** *False*  
**thus** *?thesis*  
**using**  $\langle y \mid \in \mid N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef} \rangle$   
**using**  $\langle \neg \text{ord-res-Interp } (fset (N \mid \cup \mid U_{pr} \mid \cup \mid U_{er} \mid \cup \mid U_{ef})) y \models y \rangle$   
**using**  $\langle \text{is-least-false-clause } (N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}) \ C \rangle$   
**unfolding** *least-false-conv[symmetric]*  
**unfolding** *is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*  
**by** *simp*  
**qed**

**moreover have** *resolvent-at D C (Suc k')  $\prec_c$  C*  
**by** (*metis Suc.prem*s  $\langle \text{ground-resolution } D \ \overset{\sim}{\sim} \text{Suc } n \rangle \ C \ (eres \ D \ C) \rangle$   
*less-or-eq-imp-le*  
*resolvent-at-less-cls-resolvent-at resolvent-at-0 zero-less-Suc*)

**ultimately show** *resolvent-at D C (Suc k')  $\prec_c$  y*  
**by** *order*  
**next**  
**assume**  $y \mid \in \mid \text{resolvents-upto } D \ C \ (Suc \ k')$   
**then obtain**  $i$  **where**  
*i-in: i  $\mid \in \mid fset-upto (Suc \ 0) (Suc \ k')$  and y-def: y = resolvent-at D C i*  
**unfolding** *resolvents-upto-def* **by** *blast*

**hence**  $i < Suc \ k'$   
**using**  $\langle y \neq \text{resolvent-at } D \ C \ (Suc \ k') \rangle$   
**by** *auto*

**show** *resolvent-at D C (Suc k')  $\prec_c$  y*  
**unfolding** *y-def*  
**proof** (*rule resolvent-at-less-cls-resolvent-at*)  
**show** (*ground-resolution D  $\overset{\sim}{\sim}$  Suc n*)  $C \ (eres \ D \ C)$   
**using**  $\langle \text{ground-resolution } D \ \overset{\sim}{\sim} \text{Suc } n \rangle \ C \ (eres \ D \ C) \rangle$  .



```

next
  show  $i < \text{Suc } k'$ 
    using  $\langle y \neq \text{resolvent-at } D \ C \ (\text{Suc } k') \rangle$  i-in y-def by auto
next
  show  $\text{Suc } k' \leq \text{Suc } n$ 
    using  $\langle \text{Suc } k' < \text{Suc } n \rangle$  by presburger
qed
qed
qed
next
show ord-res.is-maximal-lit L (resolvent-at D C k)
proof (rule max-lit-resolvent-at)
  show  $(\text{ground-resolution } D \ \overset{\sim}{\sim} \ \text{Suc } n) \ C \ (\text{eres } D \ C)$ 
    using  $\langle (\text{ground-resolution } D \ \overset{\sim}{\sim} \ \text{Suc } n) \ C \ (\text{eres } D \ C) \rangle$  .
next
  show  $k < \text{Suc } n$ 
    using  $\langle k < \text{Suc } n \rangle$  .
next
  show ord-res.is-maximal-lit L C
    using  $\langle \text{ord-res.is-maximal-lit } L \ C \rangle$  .
qed
next
show is-neg L
  using  $\langle \text{is-neg } L \rangle$  .
next
show  $D \ |\in| \ N \ |\cup| \ (U_{pr} \ |\cup| \ U_{er} \ |\cup| \ \text{resolvents-upto } D \ C \ k) \ |\cup| \ U_{ef}$ 
  using  $\langle D \ |\in| \ N \ |\cup| \ U_{er} \ |\cup| \ U_{ef} \rangle$  by auto
next
show  $D \ \prec_c \ \text{resolvent-at } D \ C \ k$ 
proof (rule left-premisse-lt-resolvent-at)
  show  $(\text{ground-resolution } D \ \overset{\sim}{\sim} \ \text{Suc } n) \ C \ (\text{eres } D \ C)$ 
    using  $\langle (\text{ground-resolution } D \ \overset{\sim}{\sim} \ \text{Suc } n) \ C \ (\text{eres } D \ C) \rangle$  .
next
  show  $k < \text{Suc } n$ 
    using  $\langle k < \text{Suc } n \rangle$  .
qed
next
have ord-res.production (fset (N |cup| (Upr |cup| Uer |cup| resolvents-upto D C k) |cup| Uef)) D =
  ord-res.production (fset (N |cup| Uer |cup| Uef) union fset (Upr |cup| resolvents-upto
  D C k)) D
  by (simp add: funion-left-commute sup-assoc sup-commute)
also have  $\dots = \text{ord-res.production (fset (N |cup| Uer |cup| Uef)) D$ 
proof (intro production-union-unproductive ballI)
  fix  $x$ 
  assume  $x \ |\in| \ U_{pr} \ |\cup| \ \text{resolvents-upto } D \ C \ k$ 
  hence  $\nexists L. \ \text{is-pos } L \ \wedge \ \text{ord-res.is-strictly-maximal-lit } L \ x$ 
    unfolding funion-iff

```

```

proof (elim disjE)
  assume  $x \in U_{pr}$ 
  thus ?thesis
    using  $U_{pr}\text{-spec}$ 
by (metis eres-eq-after-tranclp-ground-resolution nex-strictly-maximal-pos-lit-if-neq-eres)
next
  assume  $x \in \text{resolvents-upto } D \ C \ k$ 
then obtain  $i$  where  $i \in \text{fset-upto } (Suc \ 0) \ k$  and  $x\text{-def}: x = \text{resolvent-at}$ 
D C i
  unfolding resolvents-upto-def by auto

  have  $0 < i$  and  $i \leq k$ 
  using  $\langle i \in \text{fset-upto } (Suc \ 0) \ k \rangle$  by simp-all

  show ?thesis
    unfolding x-def
    proof (rule nex-pos-strictly-max-lit-in-resolvent-at)
      show (ground-resolution  $D \ \sim \ Suc \ n$ )  $C \ (eres \ D \ C)$ 
      using  $\langle \text{ground-resolution } D \ \sim \ Suc \ n \rangle C \ (eres \ D \ C)$  .
    next
      show  $i < Suc \ n$ 
      using  $\langle i \leq k \rangle \langle k < Suc \ n \rangle$  by presburger
    qed
  qed
thus ord-res.production (fset ( $N \ \cup \ U_{er} \ \cup \ U_{ef}$ )  $\cup$ 
  fset ( $U_{pr} \ \cup \ \text{resolvents-upto } D \ C \ k$ ))  $x = \{\}$ 
  using unproductive-if-nex-strictly-maximal-pos-lit by metis
next
  show  $D \in N \ \cup \ U_{er} \ \cup \ U_{ef}$ 
  using  $\langle D \in N \ \cup \ U_{er} \ \cup \ U_{ef} \rangle$  .
  qed simp-all
finally show ord-res.production (fset ( $N \ \cup \ (U_{pr} \ \cup \ U_{er} \ \cup \ \text{resolvents-upto}$ 
D C k) \ \cup \ U_{ef}))  $D = \{\text{atm-of } L\}$ 
  using  $\langle \text{ord-res.production } (fset \ (N \ \cup \ U_{er} \ \cup \ U_{ef})) \ D = \{\text{atm-of } L\} \rangle$  by
argo
next
  show ord-res.ground-resolution (resolvent-at  $D \ C \ k$ )  $D$  (resolvent-at  $D \ C$ 
(Suc k))
  unfolding ground-resolution-def[symmetric]
  proof (rule ground-resolution-resolvent-at-resolvent-at-Suc)
    show (ground-resolution  $D \ \sim \ Suc \ n$ )  $C \ (eres \ D \ C)$ 
    using  $\langle \text{ground-resolution } D \ \sim \ Suc \ n \rangle C \ (eres \ D \ C)$  .
  next
    show  $k < Suc \ n$ 
    using  $\langle k < Suc \ n \rangle$  .
  qed
next
  show  $U_{pr} \ \cup \ U_{er} \ \cup \ \text{finsert } (\text{resolvent-at } D \ C \ (Suc \ k)) \ (\text{resolvents-upto } D$ 

```

$C k) =$   
 $\text{finsert (resolvent-at } D C (Suc k)) (U_{pr} \mid \cup \mid U_{er} \mid \cup \mid \text{resolvents-upto } D C k)$   
 $\text{by simp}$   
**qed**

**ultimately show** *?case*  
 $\text{by (meson relpowp-Suc-I)}$   
**qed**

**hence**  $(ord\text{-res-2-step } \rightsquigarrow Suc n) S2 (N, U_{pr} \mid \cup \mid U_{er} \mid \cup \mid \text{resolvents-upto } D C$   
 $(Suc n), U_{ef})$   
**unfolding match-hyps(1,2) by blast**

**moreover have**  $\text{match } (N, U_{pr} \mid \cup \mid U_{er} \mid \cup \mid \text{resolvents-upto } D C (Suc n), U_{ef})$   
 $S3'$

**proof** –

**have 1:**  $S3' = (N, \text{finsert (eres } D C) U_{er}, U_{ef})$   
**unfolding**  $S3'\text{-def } \langle S3' = (U_{rr}', U_{ef}) \rangle \langle U_{rr}' = \text{finsert (eres } D C) U_{er} \rangle ..$

**have 2:**  $U_{pr} \mid \cup \mid U_{er} \mid \cup \mid \text{resolvents-upto } D C (Suc n) =$   
 $U_{pr} \mid \cup \mid \text{resolvents-upto } D C n \mid \cup \mid \text{finsert (eres } D C) U_{er}$   
**by**  $(\text{auto simp: } \langle \text{resolvent-at } D C (Suc n) = \text{eres } D C \rangle)$

**show** *?thesis*

**unfolding**  $\text{match-def } 1\ 2$

**proof**  $(\text{rule } ord\text{-res-2-matches-ord-res-3.intros})$

**show**  $\forall E \mid \in \mid U_{pr} \mid \cup \mid \text{resolvents-upto } D C n.$   
 $\exists D1 \mid \in \mid N \mid \cup \mid \text{finsert (eres } D C) U_{er} \mid \cup \mid U_{ef}. \exists D2 \mid \in \mid N \mid \cup \mid \text{finsert (eres}$   
 $D C) U_{er} \mid \cup \mid U_{ef}.$   
 $(\text{ground-resolution } D1)^{++} D2 E \wedge E \neq \text{eres } D1 D2 \wedge \text{eres } D1 D2 \mid \in \mid$   
 $\text{finsert (eres } D C) U_{er}$

**proof**  $(\text{intro ballI})$

**fix**  $Ca$

**assume**  $Ca \mid \in \mid U_{pr} \mid \cup \mid \text{resolvents-upto } D C n$

**hence**  $Ca \mid \in \mid U_{pr} \vee Ca \mid \in \mid \text{resolvents-upto } D C n$

**by**  $\text{simp}$

**thus**  $\exists D1 \mid \in \mid N \mid \cup \mid \text{finsert (eres } D C) U_{er} \mid \cup \mid U_{ef}. \exists D2 \mid \in \mid N \mid \cup \mid \text{finsert}$   
 $(\text{eres } D C) U_{er} \mid \cup \mid U_{ef}.$   
 $(\text{ground-resolution } D1)^{++} D2 Ca \wedge Ca \neq \text{eres } D1 D2 \wedge \text{eres } D1 D2 \mid \in \mid$   
 $\text{finsert (eres } D C) U_{er}$

**proof**  $(\text{elim disjE})$

**show**  $Ca \mid \in \mid U_{pr} \implies ?thesis$

**using**  $U_{pr}\text{-spec}$  **by**  $\text{auto}$

**next**

**assume**  $Ca \mid \in \mid \text{resolvents-upto } D C n$

**then obtain**  $i$  **where**  $i\text{-in: } i \mid \in \mid \text{fset-upto } (Suc 0) n$  **and**  $Ca\text{-def: } Ca =$   
 $\text{resolvent-at } D C i$

**unfolding**  $\text{resolvents-upto-def}$  **by**  $\text{auto}$

```

from i-in have  $0 < i \leq n$ 
  by simp-all

show ?thesis
proof (intro beXI conjI)
  have (ground-resolution  $D \rightsquigarrow i$ )  $C \text{ Ca}$ 
    unfolding  $\langle \text{Ca} = \text{resolvent-at } D \ C \ i \rangle$ 
  proof (rule relpowp-to-resolvent-at)
    show (ground-resolution  $D \rightsquigarrow \text{Suc } n$ )  $C \ (\text{eres } D \ C)$ 
      using  $\langle (\text{ground-resolution } D \rightsquigarrow \text{Suc } n) \ C \ (\text{eres } D \ C) \rangle$  .
  next
    show  $i < \text{Suc } n$ 
      using  $\langle i \leq n \rangle$  by presburger
  qed
  thus (ground-resolution  $D$ )++  $C \ \text{Ca}$ 
    using  $\langle 0 < i \rangle$  by (simp add: tranclp-if-relpowp)
next
  show  $\text{Ca} \neq \text{eres } D \ C$ 
    by (metis Ca-def  $\langle (\text{ground-resolution } D \rightsquigarrow \text{Suc } n) \ C \ (\text{eres } D \ C) \rangle$ 
       $\langle \nexists x. \text{ground-resolution } D \ (\text{eres } D \ C) \ x \ \langle i \leq n \rangle$ 
      ground-resolution-resolvent-at-resolvent-at-Suc less-Suc-eq-le)
next
  show  $\text{eres } D \ C \ |\in| \ \text{finsert} \ (\text{eres } D \ C) \ U_{er}$ 
    by simp
next
  show  $D \ |\in| \ N \ |\cup| \ \text{finsert} \ (\text{eres } D \ C) \ U_{er} \ |\cup| \ U_{ef}$ 
    using resolution by simp
next
  have  $C \ |\in| \ N \ |\cup| \ U_{er} \ |\cup| \ U_{ef}$ 
    using resolution
  by (simp add: is-least-false-clause-def linorder-cls.is-least-in-filter-iff)
  thus  $C \ |\in| \ N \ |\cup| \ \text{finsert} \ (\text{eres } D \ C) \ U_{er} \ |\cup| \ U_{ef}$ 
    by simp
  qed
qed
qed
qed
qed

ultimately have  $\exists S2'. (\text{ord-res-2-step} \rightsquigarrow \text{Suc } n) \ S2 \ S2' \wedge \text{match } S2' \ S3'$ 
  by metis

  thus  $\exists S2'. \text{ord-res-2-step}^{++} \ S2 \ S2' \wedge \text{match } S2' \ S3'$ 
    by (metis Zero-neq-Suc tranclp-if-relpowp)
qed
qed

lemma safe-states-if-ord-res-2-matches-ord-res-3:
  assumes match: ord-res-2-matches-ord-res-3  $S_2 \ S_3$ 

```

```

shows
  safe-state ord-res-2-step ord-res-2-final S2
  safe-state ord-res-3-step ord-res-3-final S3
proof –
  show safe-state ord-res-2-step ord-res-2-final S2
    using safe-state-if-all-states-safe ord-res-2-step-safe by metis

  show safe-state ord-res-3-step ord-res-3-final S3
    using safe-state-if-all-states-safe ord-res-3-step-safe by metis
qed

theorem bisimulation-ord-res-2-ord-res-3:
  defines match ≡ λ- S2 S3. ord-res-2-matches-ord-res-3 S2 S3
  shows ∃ (MATCH :: nat × nat ⇒ 'f ord-res-2-state ⇒ 'f ord-res-3-state ⇒ bool)
     $\mathcal{R}_f \mathcal{R}_b$ .
    bisimulation ord-res-2-step ord-res-2-final ord-res-3-step ord-res-3-final MATCH
     $\mathcal{R}_f \mathcal{R}_b$ 
proof (rule ex-bisimulation-from-backward-simulation)
  show right-unique ord-res-2-step
    using right-unique-ord-res-2-step .
next
  show right-unique ord-res-3-step
    using right-unique-ord-res-3-step .
next
  show  $\forall s1. \text{ord-res-2-final } s1 \longrightarrow (\nexists s1'. \text{ord-res-2-step } s1 \ s1')$ 
    by (metis finished-def ord-res-2-semantics.final-finished)
next
  show  $\forall s2. \text{ord-res-3-final } s2 \longrightarrow (\nexists s2'. \text{ord-res-3-step } s2 \ s2')$ 
    by (metis finished-def ord-res-3-semantics.final-finished)
next
  show  $\forall i \ s1 \ s2. \text{match } i \ s1 \ s2 \longrightarrow \text{ord-res-2-final } s1 = \text{ord-res-3-final } s2$ 
    unfolding match-def
    using ord-res-2-final-iff-ord-res-3-final by metis
next
  show  $\forall i \ s1 \ s2. \text{match } i \ s1 \ s2 \longrightarrow$ 
    safe-state ord-res-2-step ord-res-2-final s1  $\wedge$  safe-state ord-res-3-step ord-res-3-final
    s2
    unfolding match-def
    using safe-states-if-ord-res-2-matches-ord-res-3 by metis
next
  show wfP (λ- -. False)
    by simp
next
  show  $\forall i \ s1 \ s2 \ s2'.$ 
    match i s1 s2  $\longrightarrow$ 
    ord-res-3-step s2 s2'  $\longrightarrow$ 
     $(\exists i' \ s1'. \text{ord-res-2-step}^{++} \ s1 \ s1' \wedge \text{match } i' \ s1' \ s2') \vee (\exists i'. \text{match } i' \ s1 \ s2')$ 
 $\wedge$  False)
    unfolding match-def

```

using *backward-simulation-2-to-3* by *metis*  
qed

end

### 30 ORD-RES-4 (implicit factorization)

**type-synonym** 'f ord-res-4-state = 'f gclause fset × 'f gclause fset × 'f gclause fset

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-3-matches-ord-res-4* :: 'f ord-res-3-state ⇒ 'f ord-res-4-state ⇒ *bool* **where**

$\mathcal{F} \mid \subseteq \mid N \mid \cup \mid U_{er} \implies U_{ef} = \text{iefac } \mathcal{F} \mid \uparrow \{ \mid C \mid \in \mid N \mid \cup \mid U_{er}. \text{iefac } \mathcal{F} \ C \neq C \} \implies$   
*ord-res-3-matches-ord-res-4* (*N*, (*U<sub>er</sub>*, *U<sub>ef</sub>*)) (*N*, *U<sub>er</sub>*, *F*)

**lemma** *ord-res-3-final-iff-ord-res-4-final*:

**assumes** *match*: *ord-res-3-matches-ord-res-4* *S3* *S4*

**shows** *ord-res-3-final* *S3*  $\longleftrightarrow$  *ord-res-4-final* *S4*

**using** *match*

**proof** (*cases* *S3* *S4* *rule*: *ord-res-3-matches-ord-res-4.cases*)

**case** *match-hyps*: (*1* *F* *N* *U<sub>er</sub>* *U<sub>ef</sub>*)

**note** *invars* = *match-hyps*(3-)

**have**  $\{ \# \} \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef} \longleftrightarrow \{ \# \} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using** *invars* **by** (*auto simp*: *iefac-def*)

**moreover have** *ex-false-clause* (*fset* (*N*  $\mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ ))  $\longleftrightarrow$   
*ex-false-clause* (*fset* (*iefac* *F*  $\mid \uparrow (N \mid \cup \mid U_{er})$ ))

**unfolding** *ex-false-clause-iff*

**unfolding** *funion-funion-eq-funion-funion-fimage-iefac-if*[*OF invars*(2)]

**unfolding** *is-least-false-clause-with-iefac-conv* ..

**ultimately have** *ord-res-final* (*N*  $\mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ )  $\longleftrightarrow$  *ord-res-final* (*iefac* *F*  $\mid \uparrow (N \mid \cup \mid U_{er})$ )

**unfolding** *ord-res-final-def* **by** *argo*

**thus** *?thesis*

**unfolding** *match-hyps*(1,2)

**by** (*simp add*: *ord-res-3-final.simps* *ord-res-4-final.simps*)

qed

**lemma** *forward-simulation-between-3-and-4*:

**assumes**

*match*: *ord-res-3-matches-ord-res-4* *S3* *S4* **and**

*step*: *ord-res-3-step* *S3* *S3'*

**shows** ( $\exists S4'$ . *ord-res-4-step*<sup>++</sup> *S4* *S4'*  $\wedge$  *ord-res-3-matches-ord-res-4* *S3'* *S4'*)

**using** *match*

**proof** (*cases S3 S4 rule: ord-res-3-matches-ord-res-4.cases*)  
**case** *match-hyps*: (1  $\mathcal{F}$   $N$   $U_{er}$   $U_{ef}$ )  
**note** *match-invars* = *match-hyps*(3-)

**from** *step* **obtain**  $s3'$  **where** *step'*: *ord-res-3*  $N$  ( $U_{er}$ ,  $U_{ef}$ )  $s3'$  **and**  $S3' = (N, s3')$   
**unfolding** *match-hyps*(1,2)  
**by** (*auto elim: ord-res-3-step.cases*)

**from** *step'* **show** *?thesis*  
**proof** (*cases N (U<sub>er</sub>, U<sub>ef</sub>) s3' rule: ord-res-3.cases*)  
**case** (*factoring C L U<sub>ef</sub>'*)

**have**  $\neg$  *ord-res.is-strictly-maximal-lit L C*  
**using**  $\langle$ *is-least-false-clause* ( $N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ )  $C \rangle$   $\langle$ *ord-res.is-maximal-lit L C* $\rangle$   $\langle$ *is-pos L* $\rangle$   
**by** (*metis (no-types, lifting) is-least-false-clause-def is-pos-def pos-lit-not-greatest-if-maximal-in-least-false-clause*)

**have**  $C \mid \in \mid N \mid \cup \mid U_{er}$   
**proof** -  
**have**  $C \mid \in \mid N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$   
**using**  $\langle$ *is-least-false-clause* ( $N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ )  $C \rangle$   
**by** (*simp add: is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff*)  
**moreover** **have**  $C \not\mid \in \mid U_{ef}$   
**proof** (*rule notI*)  
**assume**  $C \mid \in \mid U_{ef}$   
**then** **obtain**  $C_0$  **where**  $C = \text{iefac } \mathcal{F} C_0$  **and**  $C_0 \mid \in \mid N \mid \cup \mid U_{er}$  **and** *iefac*  $\mathcal{F} C_0 \neq C_0$   
**using** *match-invars*(2) **by** *force*  
**then** **show** *False*  
**by** (*metis Uniq-D*  $\langle$  $\neg$  *ord-res.is-strictly-maximal-lit L C* $\rangle$  *iefac-def linorder-lit.Uniq-is-maximal-in-mset linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset local.factoring(3) obtains-positive-greatest-lit-if-efac-not-ident*)

**qed**  
**ultimately** **show** *?thesis*  
**by** *simp*  
**qed**

**show** *?thesis*  
**proof** (*intro exI conjI*)  
**show** *ord-res-4-step<sup>++</sup> S4* ( $N$ ,  $U_{er}$ , *fininsert C F*)  
**unfolding** *match-hyps*(1,2)  
**proof** (*intro tranclp.r-into-trancl ord-res-4-step.intros ord-res-4.factoring*)  
**have** *is-least-false-clause* ( $N \mid \cup \mid U_{er} \mid \cup \mid U_{ef}$ )  $C$   
**using** *factoring* **by** *argo*  
**hence** *is-least-false-clause* ( $N \mid \cup \mid U_{er} \mid \cup \mid \text{iefac } \mathcal{F} \mid \uparrow$  ( $N \mid \cup \mid U_{er}$ ))  $C$   
**unfolding** *funion-funion-eq-funion-funion-fimage-iefac-if*[*OF match-invars*(2)]

```

thus is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$  ( $N$  | $\cup$ |  $U_{er}$ ))  $C$ 
  unfolding is-least-false-clause-with-iefac-conv .
next
  show ord-res.is-maximal-lit  $L$   $C$ 
    using  $\langle$ ord-res.is-maximal-lit  $L$   $C$  $\rangle$  .
next
  show is-pos  $L$ 
    using  $\langle$ is-pos  $L$  $\rangle$  .
qed (rule refl) $+$ 
next
show ord-res-3-matches-ord-res-4  $S3'$  ( $N$ ,  $U_{er}$ , finsert  $C$   $\mathcal{F}$ )
  unfolding  $\langle$  $S3' = (N, s3')$  $\rangle$   $\langle$  $s3' = (U_{er}, U_{ef'})$  $\rangle$   $\langle$  $U_{ef'} = \text{finsert} (efac\ C)$ 
 $U_{ef}$  $\rangle$ 
proof (rule ord-res-3-matches-ord-res-4.intros)
  show finsert  $C$   $\mathcal{F}$  | $\subseteq$ |  $N$  | $\cup$ |  $U_{er}$ 
    using match-invars  $\langle$  $C$  | $\in$ |  $N$  | $\cup$ |  $U_{er}$  $\rangle$  by simp
next
  have  $\exists C'$ . ord-res.ground-factoring  $C$   $C'$ 
    using  $\langle$ ord-res.is-maximal-lit  $L$   $C$  $\rangle$   $\langle$ is-pos  $L$  $\rangle$ 
    by (metis  $\langle$  $\neg$  ord-res.is-strictly-maximal-lit  $L$   $C$  $\rangle$  ex-ground-factoringI
is-pos-def)
  hence efac  $C \neq C$ 
    by (metis ex1-efac-eq-factoring-chain)
  hence iefac (finsert  $C$   $\mathcal{F}$ )  $C \neq C$ 
    by (simp add: iefac-def)

have  $\{|Ca| \in |N \cup U_{er}. iefac (finsert\ C\ \mathcal{F})\ Ca \neq Ca|\} =$ 
  finsert  $C$   $\{|Ca| \in |N \cup U_{er}. iefac\ \mathcal{F}\ Ca \neq Ca|\}$ 
proof (intro fsubset-antisym fsubsetI)
  fix  $x$ 
  assume  $x \in \{|Ca| \in |N \cup U_{er}. iefac (finsert\ C\ \mathcal{F})\ Ca \neq Ca|\}$ 
  hence  $x \in |N \cup U_{er}$  and iefac (finsert  $C$   $\mathcal{F}$ )  $x \neq x$ 
    by simp-all
  then show  $x \in |finsert\ C\ \{|Ca| \in |N \cup U_{er}. iefac\ \mathcal{F}\ Ca \neq Ca|\}$ 
    by (smt (verit, best) ffmember-filter finsert-iff iefac-def)
next
  fix  $x$ 
  assume  $x \in |finsert\ C\ \{|Ca| \in |N \cup U_{er}. iefac\ \mathcal{F}\ Ca \neq Ca|\}$ 
  hence  $x = C \vee x \in |N \cup U_{er} \wedge iefac\ \mathcal{F}\ x \neq x$ 
    by auto
  thus  $x \in \{|Ca| \in |N \cup U_{er}. iefac (finsert\ C\ \mathcal{F})\ Ca \neq Ca|\}$ 
  proof (elim disjE conjE)
    assume  $x = C$ 
    thus ?thesis
    using  $\langle$  $C \in |N \cup U_{er}$  $\rangle$   $\langle$ iefac (finsert  $C$   $\mathcal{F}$ )  $C \neq C$  $\rangle$  by auto
next
  assume  $x \in |N \cup U_{er}$  and iefac  $\mathcal{F}$   $x \neq x$ 
  thus ?thesis

```



```

      by (smt (verit, best) fmember-filter finsertCI iefac-def)
    qed
  qed
  thus finsert (efac C) Uef =
    iefac (finsert C  $\mathcal{F}$ ) | $\uparrow$  {|Ca | $\in$ | N | $\cup$ | Uer. iefac (finsert C  $\mathcal{F}$ ) Ca  $\neq$  Ca|}
    using iefac-def match-invars(2) by auto
  qed
  qed
next
  case (resolution C L D Urr')

  have D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer)
  proof -
    have D | $\in$ | N | $\cup$ | Uer | $\cup$ | Uef
    using resolution by argo
    hence D | $\in$ | N | $\cup$ | Uer | $\cup$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer)
    unfolding union-union-eq-union-union-fimage-iefac-if[OF match-invars(2)]

    moreover have D | $\notin$ | N | $\cup$ | Uer - iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer)
    by (metis clauses-for-iefac-are-unproductive insert-not-empty local.resolution( $\gamma$ ))
    ultimately show ?thesis
    by blast
  qed

  show ?thesis
  proof (intro exI conjI)
    show ord-res-4-step++ S4 (N, finsert (eres D C) Uer,  $\mathcal{F}$ )
    unfolding match-hyps(1,2)
    proof (intro tranclp.r-into-trancl ord-res-4-step.intros ord-res-4.resolution)
      have is-least-false-clause (N | $\cup$ | Uer | $\cup$ | Uef) C
      using resolution by argo
      hence is-least-false-clause (N | $\cup$ | Uer | $\cup$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer)) C
      unfolding union-union-eq-union-union-fimage-iefac-if[OF match-invars(2)]

      thus is-least-false-clause (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer)) C
      unfolding is-least-false-clause-with-iefac-conv .
    next
      show ord-res.is-maximal-lit L C
      using resolution by argo
    next
      show is-neg L
      using resolution by argo
    next
      show D | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer)
      using  $\langle D | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \rangle$  .
    next
      show D  $\prec_c$  C
      using resolution by argo
    next
  end

```

```

have ord-res.production (fset (N |∪| Uer |∪| Uef)) D =
  ord-res.production (fset (N |∪| Uer |∪| iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer))) D
unfolding funion-funion-eq-funion-funion-fimage-iefac-if[OF match-invars(2)]
..
  also have ... = ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer)) ∪ fset
(N |∪| Uer)) D
    by (simp add: sup commute)
  also have ... = ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer))) D
  proof (rule production-union-unproductive-strong)
    show  $\forall x \in \text{fset } (N | \cup | U_{er}) - \text{fset } (\text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}))$ .
      ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer)) ∪ fset (N |∪| Uer)) x
= {}
    using clauses-for-iefac-are-unproductive[of N |∪| Uer  $\mathcal{F}$ ] by simp
  next
    show D |∈| iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer)
      using ⟨D |∈| iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer)⟩ .
    qed (rule finite-fset)+

    finally show ord-res.production (fset (iefac  $\mathcal{F}$  | $\uparrow$  (N |∪| Uer))) D =
{atm-of L}
      using resolution by argo
    qed (rule refl)+
  next
    show ord-res-3-matches-ord-res-4 S3' (N, finsert (eres D C) Uer,  $\mathcal{F}$ )
      unfolding ⟨S3' = (N, s3')⟩ ⟨s3' = (Urr', Uef)⟩ ⟨Urr' = finsert (eres D
C) Uer⟩
    proof (rule ord-res-3-matches-ord-res-4.intros)
      show  $\mathcal{F} | \subseteq | N | \cup | \text{finsert } (\text{eres } D \ C) \ U_{er}$ 
        using match-invars by auto
    next
      show Uef = iefac  $\mathcal{F}$  | $\uparrow$  {|C |∈| N |∪| finsert (eres D C) Uer. iefac  $\mathcal{F}$  C ≠
C|}
    proof (cases eres D C |∈|  $\mathcal{F}$ )
      case True
        then show ?thesis
          using ⟨ $\mathcal{F} | \subseteq | N | \cup | U_{er}$ ⟩
          using match-invars by force
        next
          case False
            hence iefac  $\mathcal{F}$  (eres D C) = eres D C
              by (simp add: iefac-def)
            hence {|C |∈| N |∪| finsert (eres D C) Uer. iefac  $\mathcal{F}$  C ≠ C|} = {|C |∈|
N |∪| Uer. iefac  $\mathcal{F}$  C ≠ C|}
              using ffilter-eq-ffilter-minus-singleton by auto
            thus ?thesis
              using match-invars by argo
          qed
        qed
      qed

```

```

qed
qed

theorem bisimulation-ord-res-3-ord-res-4:
  defines match  $\equiv \lambda S3 S4. \text{ord-res-3-matches-ord-res-4 } S3 S4$ 
  shows  $\exists (MATCH :: \text{nat} \times \text{nat} \Rightarrow 'f \text{ord-res-3-state} \Rightarrow 'f \text{ord-res-4-state} \Rightarrow \text{bool})$ 
   $\mathcal{R}_f \mathcal{R}_b.$ 
  bisimulation ord-res-3-step ord-res-3-final ord-res-4-step ord-res-4-final MATCH
   $\mathcal{R}_f \mathcal{R}_b$ 
proof (rule ex-bisimulation-from-forward-simulation)
  show right-unique ord-res-3-step
    using right-unique-ord-res-3-step .
next
  show right-unique ord-res-4-step
    using right-unique-ord-res-4-step .
next
  show  $\forall s1. \text{ord-res-3-final } s1 \longrightarrow (\nexists s1'. \text{ord-res-3-step } s1 s1')$ 
    by (metis finished-def ord-res-3-semantics.final-finished)
next
  show  $\forall s2. \text{ord-res-4-final } s2 \longrightarrow (\nexists s2'. \text{ord-res-4-step } s2 s2')$ 
    by (metis finished-def ord-res-4-semantics.final-finished)
next
  show  $\forall i s1 s2. \text{match } i s1 s2 \longrightarrow \text{ord-res-3-final } s1 \longleftrightarrow \text{ord-res-4-final } s2$ 
    unfolding match-def
    using ord-res-3-final-iff-ord-res-4-final by metis
next
  show  $\forall i s1 s2. \text{match } i s1 s2 \longrightarrow$ 
    safe-state ord-res-3-step ord-res-3-final s1  $\wedge$  safe-state ord-res-4-step ord-res-4-final s2
    using ord-res-3-step-safe ord-res-4-step-safe
    by (simp add: safe-state-if-all-states-safe)
next
  show wfP ( $\lambda i'. \text{False}$ )
    by simp
next
  show  $\forall i s1 s2 s1'. \text{match } i s1 s2 \longrightarrow \text{ord-res-3-step } s1 s1' \longrightarrow$ 
     $(\exists i' s2'. \text{ord-res-4-step}^{++} s2 s2' \wedge \text{match } i' s1' s2') \vee (\exists i'. \text{match } i' s1' s2 \wedge$ 
    False)
    unfolding match-def
    using forward-simulation-between-3-and-4 by metis
qed
end

```

## 31 ORD-RES-5 (explicit model construction)

```

type-synonym 'f ord-res-5-state = 'f gclause fset  $\times$  'f gclause fset  $\times$  'f gclause fset  $\times$ 
  ('f gterm  $\Rightarrow$  'f gclause option)  $\times$  'f gclause option

```

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-4-matches-ord-res-5* :: 'f *ord-res-4-state*  $\Rightarrow$  'f *ord-res-5-state*  $\Rightarrow$   
*bool* **where**

*ord-res-5-invars*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \Longrightarrow$   
 $(\forall C. C = \text{Some } C \longleftrightarrow \text{is-least-false-clause } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) C) \Longrightarrow$   
*ord-res-4-matches-ord-res-5*  $(N, U_{er}, \mathcal{F}) (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$

**lemma** *ord-res-4-final-iff-ord-res-5-final*:

**assumes** *match*: *ord-res-4-matches-ord-res-5*  $S_4 S_5$

**shows** *ord-res-4-final*  $S_4 \longleftrightarrow$  *ord-res-5-final*  $S_5$

**using** *match*

**proof** (*cases*  $S_4 S_5$  *rule*: *ord-res-4-matches-ord-res-5.cases*)

**case** *match-hyps*:  $(1 N U_{er} \mathcal{F} \mathcal{M} \mathcal{C})$

**show** *?thesis*

**unfolding** *match-hyps*(1,2,3)

**proof** (*intro iffI* *ord-res-5-final.intros*)

**assume** *ord-res-4-final*  $(N, U_{er}, \mathcal{F})$

**hence**  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \vee \neg \text{ex-false-clause } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$

**by** (*simp add*: *ord-res-4-final.simps* *ord-res-final-def*)

**thus** *ord-res-5-final*  $(N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$

**proof** (*elim disjE*)

**assume**  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$

**hence** *is-least-false-clause*  $(\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) \{\#\}$

**using** *is-least-false-clause-empty* **by** *metis*

**hence**  $C = \text{Some } \{\#\}$

**by** (*smt* (*verit*) *all-smaller-clauses-true-wrt-respective-Interp-def* *is-least-false-clause-def*

*linorder-cls.is-least-in-filter-iff* *linorder-cls.le-imp-less-or-eq* *match-hyps*(3)

*empty-lesseq-cls* *ord-res-5-invars-def*)

**thus** *?thesis*

**using** *ord-res-5-final.contradiction-found* **by** *metis*

**next**

**assume**  $\neg \text{ex-false-clause } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$

**hence**  $C = \text{None}$

**using** *match-hyps*(2-)

**by** (*metis* *ex-false-clause-if-least-false-clause* *option.exhaust*)

**thus** *?thesis*

**using** *ord-res-5-final.model-found* **by** *metis*

**qed**

**next**

**assume** *ord-res-5-final*  $(N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$

**thus** *ord-res-4-final*  $(N, U_{er}, \mathcal{F})$

**proof** (*cases*  $(N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$  *rule*: *ord-res-5-final.cases*)

**case** *model-found*

**have** *all-smaller-clauses-true-wrt-respective-Interp*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$

**using**  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$

**unfolding** *ord-res-5-invars-def* **by** *metis*  
**hence**  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{ord-res-Interp } (\text{iefac } \mathcal{F} \mid \uparrow (\text{fset } N \cup \text{fset } U_{er})) C \models C$   
**by** (*simp add: model-found all-smaller-clauses-true-wrt-respective-Interp-def*)  
**hence**  $\neg \text{ex-false-clause } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   
**by** (*simp add: ex-false-clause-def*)  
**then show** *?thesis*  
**by** (*metis ord-res-4-final.intros ord-res-final-def*)  
**next**  
**case** *contradiction-found*  
**hence**  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
**using**  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$   
**by** (*metis next-clause-in-factorized-clause-def ord-res-5-invars-def*)  
**then show** *?thesis*  
**by** (*metis ord-res-4-final.intros ord-res-final-def*)  
**qed**  
**qed**  
**qed**

**lemma** *forward-simulation-between-4-and-5:*

**fixes**  $S_4 S_4' S_5$   
**assumes** *match: ord-res-4-matches-ord-res-5*  $S_4 S_5$  **and** *step: ord-res-4-step*  $S_4 S_4'$   
**shows**  $\exists S_5'. \text{ord-res-5-step}^{++} S_5 S_5' \wedge \text{ord-res-4-matches-ord-res-5 } S_4' S_5'$   
**using** *match*  
**proof** (*cases*  $S_4 S_5$  *rule: ord-res-4-matches-ord-res-5.cases*)  
**case** *match-hyps: (1 N U<sub>er</sub> F M C)*  
**hence**  
 $S_4\text{-def: } S_4 = (N, U_{er}, \mathcal{F})$  **and**  
 $S_5\text{-def: } S_5 = (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$   
**unfolding** *atomize-conj* **by** *metis*

**have** *dom-M-eq:  $\bigwedge C. \mathcal{C} = \text{Some } C \implies \text{dom } \mathcal{M} = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C$*   
**using** *match-hyps unfolding ord-res-5-invars-def model-eq-interp-upto-next-clause-def*  
**by** *simp*

**obtain**  $s_4'$  **where**  $S_4'\text{-def: } S_4' = (N, s_4')$  **and**  $\text{step': } \text{ord-res-4 } N (U_{er}, \mathcal{F}) s_4'$   
**using** *step unfolding S<sub>4</sub>-def* **by** (*auto simp: ord-res-4-step.simps*)

**show** *?thesis*  
**using** *step'*  
**proof** (*cases*  $N (U_{er}, \mathcal{F}) s_4'$  *rule: ord-res-4.cases*)  
**case** *step-hyps: (factoring NN C L F')*  
**have**  $\mathcal{C} = \text{Some } C$   
**using** *match-hyps(3-) step-hyps* **by** *metis*

**define**  $\mathcal{M}' :: 'f \text{ gterm} \Rightarrow 'f \text{ gterm literal multiset option}$  **where**  
 $\mathcal{M}' = (\lambda-. \text{None})$

**define**  $C' :: 'f$  *gclause option where*  
 $C' = \text{The-optional} (\text{linorder-cls.is-least-in-fset} (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})))$

**have** *ord-res-5-step*:  $\text{ord-res-5 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}', \mathcal{M}', C')$

**proof** (*rule ord-res-5.factoring*)  
**have** *is-least-false-clause* ( $\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ )  $C$   
**using** *step-hyps by argo*  
**then show**  $\neg \text{dom } \mathcal{M} \models C$   
**using** *dom-M-eq[OF <C = Some C>]*  
**by** (*metis (mono-tags, lifting) is-least-false-clause-def*  
*linorder-cls.is-least-in-filter-iff ord-res-Interp-entails-if-greatest-lit-is-pos*  
*unproductive-if-nex-strictly-maximal-pos-lit sup-bot.right-neutral*)

**next**  
**show** *ord-res.is-maximal-lit*  $L C$   
**using** *step-hyps by metis*

**next**  
**show** *is-pos*  $L$   
**using** *step-hyps by metis*

**next**  
**show**  $\neg \text{ord-res.is-strictly-maximal-lit } L C$   
**using** *step-hyps*  
**by** (*metis (no-types, lifting) is-least-false-clause-def literal.collapse(1)*  
*pos-lit-not-greatest-if-maximal-in-least-false-clause*)

**next**  
**show**  $\mathcal{F}' = \text{finsert } C \mathcal{F}$   
**using** *step-hyps by metis*

**qed** (*simp-all add: M'-def C'-def*)

**moreover have**  $\exists \mathcal{M}'' C''$ .  
 $(\text{ord-res-5 } N)^{**} (U_{er}, \mathcal{F}', \mathcal{M}', C') (U_{er}, \mathcal{F}', \mathcal{M}'', C'') \wedge$   
 $(\forall C. (C'' = \text{Some } C) \longleftrightarrow \text{is-least-false-clause} (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) C)$

**proof** (*rule ord-res-5-construct-model-upto-least-false-clause*)  
**show** *ord-res-5-invars*  $N (U_{er}, \mathcal{F}', \mathcal{M}', C')$   
**using** *ord-res-5-step <ord-res-5-invars N (U\_{er}, \mathcal{F}, \mathcal{M}, C)> <C = Some C>*  
**by** (*metis ord-res-5-preserves-invars*)

**qed**

**ultimately obtain**  $\mathcal{M}'' C''$  **where**  
*s5-steps*:  $(\text{ord-res-5 } N)^{++} (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}', \mathcal{M}'', C'')$  **and**  
*next-clause-least-false*:  
 $(\forall C. (C'' = \text{Some } C) \longleftrightarrow \text{is-least-false-clause} (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) C)$   
**by** (*meson rtranclp-into-tranclp2*)

**have** *ord-res-5-step<sup>++</sup>*  $S5 (N, U_{er}, \mathcal{F}', \mathcal{M}'', C'')$   
**unfolding** *S5-def <C = Some C>*  
**using** *s5-steps by (metis tranclp-ord-res-5-step-if-tranclp-ord-res-5)*

**moreover have** *ord-res-4-matches-ord-res-5*  $S4' (N, U_{er}, \mathcal{F}', \mathcal{M}'', C'')$

```

    unfolding  $S_4'$ -def  $\langle s_4' = (U_{er}, \mathcal{F}') \rangle$ 
  proof (intro ord-res-4-matches-ord-res-5.intros)
    show ord-res-5-invars  $N (U_{er}, \mathcal{F}', \mathcal{M}'', C'')$ 
      using s5-steps  $\langle C = \text{Some } C \rangle \langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, C) \rangle$ 
      by (smt (verit, best) ord-res-5-preserves-invars tranclp-induct)
  next
    show  $\forall C. (C'' = \text{Some } C) = \text{is-least-false-clause } (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) C$ 
      using next-clause-least-false .
  qed

  ultimately show ?thesis
    by metis
next
case step-hyps: (resolution NN C L D  $U_{er}'$ )
have  $C = \text{Some } C$ 
  using match-hyps(3-) step-hyps by metis

define  $\mathcal{M}' :: 'f \text{ gterm} \Rightarrow 'f \text{ gterm literal multiset option where}$ 
 $\mathcal{M}' = (\lambda \cdot \text{None})$ 

define  $C' :: 'f \text{ gclause option where}$ 
 $C' = \text{The-optional } (\text{linorder-clc.is-least-in-fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}')))$ 

have ord-res-5-step: ord-res-5  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}', \mathcal{F}, \mathcal{M}', C')$ 
proof (rule ord-res-5.resolution)
  have is-least-false-clause  $(\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) C$ 
    using step-hyps by argo
  then show  $\neg \text{dom } \mathcal{M} \models C$ 
    using dom-M-eq[OF  $\langle C = \text{Some } C \rangle$ ]
    by (metis (mono-tags, lifting) is-least-false-clause-def
      linorder-clc.is-least-in-filter-iff ord-res-Interp-entails-if-greatest-lit-is-pos
      unproductive-if-nex-strictly-maximal-pos-lit sup-bot.right-neutral)
next
  show ord-res.is-maximal-lit  $L C$ 
    using step-hyps by metis
next
  show is-neg  $L$ 
    using step-hyps by metis
next
  show  $\mathcal{M} (\text{atm-of } L) = \text{Some } D$ 
    using step-hyps
    by (smt (verit)  $\langle C = \text{Some } C \rangle$  all-produced-atoms-in-model-def insertI1
      match-hyps(3)
      ord-res-5-invars-def)
next
  show  $U_{er}' = \text{finsert } (\text{eres } D C) U_{er}$ 
    using step-hyps by metis
qed (simp-all add:  $\mathcal{M}'$ -def  $C'$ -def)

```

**moreover have**  $\exists \mathcal{M}'' \mathcal{C}''$ .  
 $(ord\text{-}res\text{-}5\ N)^{**} (U_{er}', \mathcal{F}, \mathcal{M}', \mathcal{C}') (U_{er}', \mathcal{F}, \mathcal{M}'', \mathcal{C}'') \wedge$   
 $(\forall C. (C'' = Some\ C) \longleftrightarrow is\text{-}least\text{-}false\text{-}clause\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}'))\ C)$   
**proof** (rule *ord-res-5-construct-model-upto-least-false-clause*)  
**show** *ord-res-5-invars*  $N\ (U_{er}', \mathcal{F}, \mathcal{M}', \mathcal{C}')$   
**using** *ord-res-5-step*  $\langle ord\text{-}res\text{-}5\text{-invars}\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle\ \langle C = Some\ C \rangle$   
**by** (*metis ord-res-5-preserves-invars*)  
**qed**

**ultimately obtain**  $\mathcal{M}'' \mathcal{C}''$  **where**  
*s5-steps*:  $(ord\text{-}res\text{-}5\ N)^{++} (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C) (U_{er}', \mathcal{F}, \mathcal{M}'', \mathcal{C}'')$  **and**  
*next-clause-least-false*:  
 $(\forall C. (C'' = Some\ C) \longleftrightarrow is\text{-}least\text{-}false\text{-}clause\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}'))\ C)$   
**by** (*meson rtranclp-into-tranclp2*)

**have** *ord-res-5-step*<sup>++</sup>  $S5\ (N, U_{er}', \mathcal{F}, \mathcal{M}'', \mathcal{C}'')$   
**unfolding** *S5-def*  $\langle C = Some\ C \rangle$   
**using** *s5-steps* **by** (*metis tranclp-ord-res-5-step-if-tranclp-ord-res-5*)

**moreover have** *ord-res-4-matches-ord-res-5*  $S4'\ (N, U_{er}', \mathcal{F}, \mathcal{M}'', \mathcal{C}'')$   
**unfolding** *S4'-def*  $\langle s4' = (U_{er}', \mathcal{F}) \rangle$   
**proof** (*intro ord-res-4-matches-ord-res-5.intros*)  
**show** *ord-res-5-invars*  $N\ (U_{er}', \mathcal{F}, \mathcal{M}'', \mathcal{C}'')$   
**using** *s5-steps*  $\langle C = Some\ C \rangle\ \langle ord\text{-}res\text{-}5\text{-invars}\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$   
**by** (*smt (verit, best) ord-res-5-preserves-invars tranclp-induct*)  
**next**  
**show**  $\forall C. (C'' = Some\ C) = is\text{-}least\text{-}false\text{-}clause\ (iefac\ \mathcal{F}\ |\uparrow\ (N\ |\cup\ U_{er}'))\ C$   
**using** *next-clause-least-false* .  
**qed**

**ultimately show** *?thesis*  
**by** *metis*

**qed**  
**qed**

**theorem** *bisimulation-ord-res-4-ord-res-5*:  
**defines** *match*  $\equiv \lambda-. ord\text{-}res\text{-}4\text{-matches-ord-res-5}$   
**shows**  $\exists (MATCH :: nat \times nat \Rightarrow 'f\ ord\text{-}res\text{-}4\text{-state} \Rightarrow 'f\ ord\text{-}res\text{-}5\text{-state} \Rightarrow bool)$   
 $\mathcal{R}_f\ \mathcal{R}_b$ .  
*bisimulation ord-res-4-step ord-res-4-final ord-res-5-step ord-res-5-final MATCH*  
 $\mathcal{R}_f\ \mathcal{R}_b$   
**proof** (rule *ex-bisimulation-from-forward-simulation*)  
**show** *right-unique ord-res-4-step*  
**using** *right-unique-ord-res-4-step* .  
**next**  
**show** *right-unique ord-res-5-step*  
**using** *right-unique-ord-res-5-step* .  
**next**  
**show**  $\forall s. ord\text{-}res\text{-}4\text{-final}\ s \longrightarrow (\exists s'. ord\text{-}res\text{-}4\text{-step}\ s\ s')$



```

    by (metis finished-def ord-res-4-semantics.final-finished)
next
show  $\forall s. \text{ord-res-5-final } s \longrightarrow (\nexists s'. \text{ord-res-5-step } s s')$ 
    by (metis finished-def ord-res-5-semantics.final-finished)
next
show  $\forall i s_4 s_5. \text{match } i s_4 s_5 \longrightarrow \text{ord-res-4-final } s_4 \longleftrightarrow \text{ord-res-5-final } s_5$ 
    unfolding match-def
    using ord-res-4-final-iff-ord-res-5-final by metis
next
show  $\forall i S_4 S_5. \text{match } i S_4 S_5 \longrightarrow$ 
    safe-state ord-res-4-step ord-res-4-final  $S_4 \wedge$  safe-state ord-res-5-step ord-res-5-final
     $S_5$ 
    proof (intro allI impI conjI)
      fix i  $S_4 S_5$ 
      show safe-state ord-res-4-step ord-res-4-final  $S_4$ 
        using ord-res-4-step-safe safe-state-if-all-states-safe by metis

      assume match i  $S_4 S_5$ 
      thus safe-state ord-res-5-step ord-res-5-final  $S_5$ 
        using <match i  $S_4 S_5$ >
        using ord-res-5-safe-state-if-invars
        using match-def ord-res-4-matches-ord-res-5.cases by metis
    qed
next
show wfp ( $\lambda -. \text{False}$ )
    by simp
next
show  $\forall i s_1 s_2 s_1'. \text{match } i s_1 s_2 \longrightarrow$ 
    ord-res-4-step  $s_1 s_1' \longrightarrow$ 
     $(\exists i' s_2'. \text{ord-res-5-step}^{++} s_2 s_2' \wedge \text{match } i' s_1' s_2') \vee (\exists i'. \text{match } i' s_1' s_2$ 
 $\wedge \text{False})$ 
    unfolding match-def
    using forward-simulation-between-4-and-5 by metis
qed
end

```

## 32 ORD-RES-6 (model backjump)

```

type-synonym 'f ord-res-6-state = 'f gclause fset  $\times$  'f gclause fset  $\times$  'f gclause
fset  $\times$ 
('f gterm  $\Rightarrow$  'f gclause option)  $\times$  'f gclause option

```

```

context simulation-SCLFOL-ground-ordered-resolution begin

```

```

inductive ord-res-5-matches-ord-res-6 :: 'f ord-res-5-state  $\Rightarrow$  'f ord-res-6-state  $\Rightarrow$ 
bool where
  ord-res-5-invars  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \Longrightarrow$ 

```

*ord-res-5-matches-ord-res-6*  $(N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$

**lemma** *ord-res-5-final-iff-ord-res-6-final*:  
fixes  $i S5 S6$   
assumes *match*: *ord-res-5-matches-ord-res-6*  $S5 S6$   
shows *ord-res-5-final*  $S5 \longleftrightarrow$  *ord-res-6-final*  $S6$   
using *match*  
**proof** (*cases*  $S5 S6$  *rule*: *ord-res-5-matches-ord-res-6.cases*)  
case  $(1 N U_{er} \mathcal{F} \mathcal{M} \mathcal{C})$   
thus ?thesis  
by (*metis* (*no-types, opaque-lifting*) *ord-res-5-final.simps ord-res-6-final.cases*  
*ord-res-6-final.contradiction-found ord-res-6-final.model-found*)  
**qed**

**lemma** *backward-simulation-between-5-and-6*:  
fixes  $S5 S6 S6'$   
assumes *match*: *ord-res-5-matches-ord-res-6*  $S5 S6$  **and** *step*: *ord-res-6-step*  $S6 S6'$   
shows  $\exists S5'. \text{ord-res-5-step}^{++} S5 S5' \wedge \text{ord-res-5-matches-ord-res-6} S5' S6'$   
using *match*  
**proof** (*cases*  $S5 S6$  *rule*: *ord-res-5-matches-ord-res-6.cases*)  
case *match-hyps*:  $(1 N U_{er} \mathcal{F} \mathcal{M} \mathcal{C})$   
hence *S5-def*:  $S5 = (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$  **and** *S6-def*:  $S6 = (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$   
by *metis+*  
  
**obtain**  $s6'$  **where** *S6'-def*:  $S6' = (N, s6')$  **and** *step'*: *ord-res-6*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) s6'$   
using *step unfolding S6-def*  
using *ord-res-6-step.simps* **by** *auto*

**show** ?thesis  
using *step'*  
**proof** (*cases*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) s6'$  *rule*: *ord-res-6.cases*)  
case *step-hyps*: (*skip*  $C C'$ )

**define**  $S5'$  **where**  
 $S5' = (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}')$

**show** ?thesis  
**proof** (*intro exI conjI*)  
have *step5*: *ord-res-5*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}')$   
using *ord-res-5.skip step-hyps* **by** *metis*  
hence *ord-res-5-step*  $S5 S5'$   
unfolding *S5-def S5'-def*  
by (*metis ord-res-5-step.simps step-hyps(1)*)  
thus *ord-res-5-step*<sup>++</sup>  $S5 S5'$   
by *simp*

have *ord-res-5-invars*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}')$

```

    using step5 match-hyps(3) ord-res-5-preserves-invars step-hyps(1) by metis
  thus ord-res-5-matches-ord-res-6 S5' S6'
    unfolding S5'-def S6'-def ⟨s6' = (Uer, F, M, C')⟩
    using ord-res-5-matches-ord-res-6.intros by metis
qed
next
case step-hyps: (production C L M' C')

define S5' where
  S5' = (N, Uer, F, M', C')

show ?thesis
proof (intro exI conjI)
  have step5: ord-res-5 N (Uer, F, M, Some C) (Uer, F, M', C')
    using ord-res-5.production step-hyps by metis
  hence ord-res-5-step S5 S5'
    unfolding S5-def S5'-def
    by (metis ord-res-5-step.simps step-hyps(1))
  thus ord-res-5-step++ S5 S5'
    by simp

  have ord-res-5-invars N (Uer, F, M', C')
    using step5 match-hyps(3) ord-res-5-preserves-invars step-hyps(1) by metis
  thus ord-res-5-matches-ord-res-6 S5' S6'
    unfolding S5'-def S6'-def ⟨s6' = (Uer, F, M', C')⟩
    using ord-res-5-matches-ord-res-6.intros by metis
qed
next
case step-hyps: (factoring D K F')

define S5' where
  S5' = (N, Uer, F', M, Some (efac D))

have D |∈| iefac F |∧| (N |∪| Uer)
  by (metis match-hyps(3) next-clause-in-factorized-clause-def ord-res-5-invars-def
step-hyps(1))
  hence iefac F' |∧| (N |∪| Uer) ≠ {||}
    by blast
  then obtain C where C-least: linorder-cls.is-least-in-fset (iefac F' |∧| (N |∪|
Uer)) C
    by (metis linorder-cls.ex1-least-in-fset)

have efac D ≠ D
  by (metis ex1-efac-eq-factoring-chain is-pos-def ex-ground-factoringI step-hyps(4,5,6))

show ?thesis
proof (intro exI conjI)
  have The-optional (linorder-cls.is-least-in-fset (iefac F' |∧| (N |∪| Uer))) =
Some C

```

**proof** (rule *The-optional-eq-SomeI*)  
**show**  $\exists_{\leq 1} x. \text{linorder-cls.is-least-in-fset} (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) x$   
**by** *blast*  
**next**  
**show**  $\text{linorder-cls.is-least-in-fset} (\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er})) C$   
**using** *C-least* .  
**qed**  
**hence** *step5: ord-res-5*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) (U_{er}, \mathcal{F}', \text{Map.empty}, \text{Some } C)$   
**using** *ord-res-5.factorizing step-hyps by metis*  
**moreover** **have**  $(\text{ord-res-5 } N)^{**} \dots (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } D))$   
**proof** (rule *full-rtranclp-ord-res-5-run-upto*)  
**show** *ord-res-6*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D) (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } D))$   
**using** *step' S6-def S6'-def*  $\langle s6' = (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } D)) \rangle \langle \mathcal{C} = \text{Some } D \rangle$  **by** *argo*  
**next**  
**show** *ord-res-5-invars*  $N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } D))$   
**using** *match-hyps(3) ord-res-6-preserves-invars step' step-hyps(2)* **by** *blast*  
**next**  
**have** *iefac*  $\mathcal{F} D = D$  **and**  $D \mid \in \mid N \mid \cup \mid U_{er}$   
**unfolding** *atomize-conj*  
**using**  $\langle \text{efac } D \neq D \rangle \langle D \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}) \rangle$  [*unfolded fimage-iff*]  
**unfolding** *iefac-def*  
**by** (*metis ex1-efac-eq-factorizing-chain factorizable-if-neq-efac*)  
  
**have** *iefac- $\mathcal{F}'$ -eq*:  $\text{iefac } \mathcal{F}' = (\text{iefac } \mathcal{F})(D := \text{efac } D)$   
**unfolding**  $\langle \mathcal{F}' = \text{finsert } D \mathcal{F} \rangle$  *iefac-def* **by** *auto*  
  
**have** *fimage-iefac- $\mathcal{F}'$ -eq*:  
 $\text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}) = \text{finsert } (\text{efac } D) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} - \{D\}))$   
**unfolding** *iefac- $\mathcal{F}'$ -eq*  
**unfolding** *fun-upd-fimage[of iefac  $\mathcal{F} D$  efac  $D$ ]*  $\langle D \mid \in \mid N \mid \cup \mid U_{er} \rangle$   
**using**  $\langle D \mid \in \mid N \mid \cup \mid U_{er} \rangle$  **by** *argo*  
  
**have**  $\{C \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \text{efac } D\} =$   
 $\{C \mid \in \mid \text{finsert } (\text{efac } D) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} - \{D\})). C \prec_c \text{efac } D\}$   
**unfolding** *fimage-iefac- $\mathcal{F}'$ -eq ..*  
  
**also** **have**  $\dots = \{C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er} - \{D\}). C \prec_c \text{efac } D\}$   
**by** *auto*  
  
**also** **have**  $\dots = \{C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \text{efac } D\}$   
**by** (*smt (verit, ccfv-SIG) iefac  $\mathcal{F} D = D$  efac-properties-if-not-ident(1)*  
*ffilter-eq-ffilter-minus-singleton fimage-finsert finsertI1 finsert-fminus1*  
*finsert-fminus-single linorder-cls.less-imp-not-less*)  
  
**finally** **have**  $\{C \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \text{efac } D\} =$   
 $\{C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c \text{efac } D\}$  .

**next**  
**have** *dom-M-eq*:  $\text{dom } \mathcal{M} = \text{ord-res.interp } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})))$   
*D*  
**using**  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle \langle \mathcal{C} = \text{Some } D \rangle$   
**unfolding** *ord-res-5-invars-def model-eq-interp-upto-next-clause-def*  
**by** *metis*  
  
**have** *atm-of K*  $\notin \text{dom } \mathcal{M}$   
**by** (*metis linorder-lit.is-maximal-in-mset-iff literal.collapse(1)*  
*pos-literal-in-imp-true-cls step-hyps(3) step-hyps(4) step-hyps(5)*)  
  
**have**  $A \prec_t \text{atm-of } K$  **if**  $A \in \text{dom } \mathcal{M}$  **for**  $A$   
**proof** –  
**obtain**  $C$  **where**  
 $C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**  
 $C \prec_c D$  **and**  
 $A \in \text{ord-res.production } (\text{fset } (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) C$   
**using**  $\langle A \in \text{dom } \mathcal{M} \rangle$  **unfolding** *dom-M-eq*  
**unfolding** *ord-res.interp-def UN-iff*  
**by** *blast*  
  
**hence** *ord-res.is-strictly-maximal-lit (Pos A) C*  
**using** *ord-res.mem-productionE* **by** *metis*  
  
**hence**  $\text{Pos } A \preceq_l K$   
**using**  $\langle \text{ord-res.is-maximal-lit } K D \rangle \langle C \prec_c D \rangle$   
**by** (*metis ord-res.asymp-less-lit ord-res.transp-less-lit linorder-cls.less-asymp*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset linorder-lit.leI*  
*linorder-lit.multipHO-if-maximal-less-that-maximal multip-eq-multipHO*)  
  
**hence**  $A \preceq_t \text{atm-of } K$   
**by** (*metis literal.collapse(1) literal.sel(1) ord-res.less-lit-simps(1) reflclp-iff*  
*step-hyps(5)*)  
  
**moreover have**  $A \neq \text{atm-of } K$   
**using**  $\langle \text{atm-of } K \notin \text{dom } \mathcal{M} \rangle \langle A \in \text{dom } \mathcal{M} \rangle$  **by** *metis*  
  
**ultimately show** *?thesis*  
**by** *order*  
**qed**  
**hence**  $\text{dom } \mathcal{M} \subseteq \{A. \exists K. \text{ord-res.is-maximal-lit } K (\text{efac } D) \wedge A \prec_t \text{atm-of } K\}$   
  
**using** *linorder-lit.is-maximal-in-mset-iff step-hyps(4)* **by** *auto*  
**thus**  $\mathcal{M} = \text{restrict-map } \mathcal{M} \{A. \exists K. \text{ord-res.is-maximal-lit } K (\text{efac } D) \wedge A \prec_t \text{atm-of } K\}$   
**using** *restrict-map-ident-if-dom-subset* **by** *fastforce*  
**next**  
**show** *linorder-cls.is-least-in-fset (iefac F' |↑ (N |∪| U<sub>er</sub>)) C*  
**using** *C-least* .

```

qed
ultimately have steps5: (ord-res-5 N)++ (Uer, F, M, Some D) (Uer, F',
M, Some (efac D))
  by simp
thus ord-res-5-step++ S5 S5'
  using S5'-def S5-def step-hyps(1) tranclp-ord-res-5-step-if-tranclp-ord-res-5
by metis

  have ord-res-5-invars N (Uer, F', M, Some (efac D))
  using steps5 match-hyps(3) tranclp-ord-res-5-preserves-invars step-hyps(1)
by metis
thus ord-res-5-matches-ord-res-6 S5' S6'
  unfolding S5'-def S6'-def ⟨s6' = (Uer, F', M, Some (efac D))⟩
  using ord-res-5-matches-ord-res-6.intros by metis
qed
next
case step-hyps: (resolution-bot C L D Uer' M')

  define S5' :: - × - × - × (f gterm ⇒ f gclause option) × f gclause option
where
  S5' = (N, Uer', F, M', Some {#})

  show ?thesis
proof (intro exI conjI)
  have {#} |∈| iefac F |∧| (N |∪| Uer')
  using ⟨Uer' = finsert (eres D C) Uer⟩ ⟨eres D C = {#}⟩
  using iefac-def by simp

  hence linorder-cls.is-least-in-fset (iefac F |∧| (N |∪| Uer')) {#}
  by (metis linorder-cls.is-minimal-in-fset-eq-is-least-in-fset
linorder-cls.is-minimal-in-fset-iff linorder-cls.leD mempty-lesseq-cls)

  hence The-optional (linorder-cls.is-least-in-fset (iefac F |∧| (N |∪| Uer'))) =
Some {#}
  by (metis linorder-cls.Uniq-is-least-in-fset The-optional-eq-SomeI)

  hence step5: ord-res-5 N (Uer, F, M, Some C) (Uer', F, M', Some {#})
  using ord-res-5.resolution step-hyps by metis

  thus ord-res-5-step++ S5 S5'
  unfolding S5-def ⟨C = Some C⟩ S5'-def
  by (simp only: ord-res-5-step.intros tranclp.r-into-trancl)

  show ord-res-5-matches-ord-res-6 S5' S6'
  using step5
  by (metis S5'-def S6'-def match-hyps(3) ord-res-5-matches-ord-res-6.intros
ord-res-5-preserves-invars step-hyps(1) step-hyps(2))
qed
next

```

**case** *step-hyps*: (*resolution-pos*  $E$   $L$   $D$   $U_{er'}$   $\mathcal{M}'$   $K$ )

**define**  $S5' :: - \times - \times - \times (f \text{ gterm} \Rightarrow f \text{ gclause option}) \times f \text{ gclause option}$

**where**

$S5' = (N, U_{er'}, \mathcal{F}, \mathcal{M}', \text{Some} (eres D E))$

**hence**  $iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er'}) \neq \{\mid\}$

**using**  $\langle U_{er'} = finsert (eres D E) U_{er} \rangle$  **by** *simp*

**then obtain**  $C$  **where**  $C$ -*least*: *linorder-cls.is-least-in-fset* ( $iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er'})$ )  $C$

**by** (*metis linorder-cls.ex1-least-in-fset*)

**show** *?thesis*

**proof** (*intro exI conjI*)

**have** *The-optional* (*linorder-cls.is-least-in-fset* ( $iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er'})$ )) = *Some C*

**proof** (*rule The-optional-eq-SomeI*)

**show**  $\exists_{\leq 1} x. \text{linorder-cls.is-least-in-fset} (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er'})) x$

**by** *blast*

**next**

**show** *linorder-cls.is-least-in-fset* ( $iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er'})$ )  $C$

**using**  $C$ -*least* .

**qed**

**hence** *step5*: *ord-res-5*  $N$  ( $U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E$ ) ( $U_{er'}, \mathcal{F}, \text{Map.empty}, \text{Some } C$ )

**using** *ord-res-5.resolution step-hyps* **by** *metis*

**moreover have** (*ord-res-5*  $N$ )<sup>\*\*</sup> ... ( $U_{er'}, \mathcal{F}, \mathcal{M}', \text{Some} (eres D E)$ )

**proof** (*rule full-rtranclp-ord-res-5-run-upto*)

**show** *ord-res-6*  $N$  ( $U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E$ ) ( $U_{er'}, \mathcal{F}, \mathcal{M}', \text{Some} (eres D E)$ )

**using** *step'  $\langle C = \text{Some } E \rangle$   $\langle s6' = (U_{er'}, \mathcal{F}, \mathcal{M}', \text{Some} (eres D E)) \rangle$*  **by**

*argo*

**next**

**show** *ord-res-5-invars*  $N$  ( $U_{er'}, \mathcal{F}, \mathcal{M}', \text{Some} (eres D E)$ )

**using** *match-hyps(3) ord-res-6-preserved-invars step' step-hyps(2)* **by** *blast*

**next**

**have**  $eres D E \neq E$

**using** *step-hyps* **by** (*metis linorder-lit.Uniq-is-maximal-in-mset the1-equality'*)

**moreover have**  $eres D E \preceq_c E$

**using** *eres-le* .

**ultimately have**  $eres D E \prec_c E$

**by** *order*

**have**  $\forall F. \text{is-least-false-clause} (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) F \longrightarrow E \preceq_c F$

**using** *ord-res-5-invars*  $N$  ( $U_{er}, \mathcal{F}, \mathcal{M}, C$ )

**unfolding** *ord-res-5-invars-def  $\langle C = \text{Some } E \rangle$*

```

using next-clause-lt-least-false-clause[of  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E)$ ]
by simp

have E-least-false: is-least-false-clause (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ )  $E$ 
unfolding is-least-false-clause-def linorder-cls.is-least-in-ffilter-iff
proof (intro conjI ballI impI)
  show  $E \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ 
    using  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$ 
    unfolding ord-res-5-invars-def  $\langle \mathcal{C} = \text{Some } E \rangle$ 
    by (metis next-clause-in-factorized-clause-def)
next
  have  $\neg \text{ord-res.interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) E \models E$ 
    using  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$ 
    unfolding ord-res-5-invars-def  $\langle \mathcal{C} = \text{Some } E \rangle$ 
    using  $\langle \neg \text{dom } \mathcal{M} \models E \rangle$  by (metis model-eq-interp-upto-next-clause-def)
moreover have ord-res.production (fset (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ ))  $E = \{\}$ 
proof –
  have  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-strictly-maximal-lit } L E$ 
    using  $\langle \text{ord-res.is-maximal-lit } L E \rangle \langle \text{is-neg } L \rangle$ 
    by (metis Uniq-D linorder-lit.Uniq-is-maximal-in-mset
      linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset)
    thus ?thesis
    using unproductive-if-nex-strictly-maximal-pos-lit by metis
qed
ultimately show  $\neg \text{ord-res.Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) E \models E$ 
by simp
next
fix  $F$ 
assume F-in:  $F \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  and  $F \neq E$  and
  F-false:  $\neg \text{ord-res.Interp (fset (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) F \models F$ 
have  $\neg F \prec_c E$ 
  using  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$ 
  unfolding ord-res-5-invars-def  $\langle \mathcal{C} = \text{Some } E \rangle$ 
  unfolding all-smaller-clauses-true-wrt-respective-Interp-def
  using F-in F-false
  by (metis option.inject)
thus  $E \prec_c F$ 
  using  $\langle F \neq E \rangle$  by order
qed

have D-prod: ord-res.production (fset (iefac  $\mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ ))  $D \neq \{\}$ 
  using  $\langle \text{ord-res-5-invars } N (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \rangle$ 
  unfolding ord-res-5-invars-def  $\langle \mathcal{C} = \text{Some } E \rangle$ 
  by (metis atoms-in-model-were-produced-def empty-iff step-hyps(6))

have iefac  $\mathcal{F}$  (eres  $D E$ ) = eres  $D E$ 
using E-least-false D-prod
by (smt (verit, ccfv-threshold)
   $\langle \forall F. \text{is-least-false-clause (iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) F \longrightarrow (\prec_c)^{==} E F \rangle$ )

```



$\langle \text{eres } D \ E \prec_c \ E \rangle$  *clause-true-if-resolved-true ex1-eres-eq-full-run-ground-resolution*  
*fmage-finsert finsert-absorb finsert-iff full-run-def funion-finsert-right*  
*is-least-false-clause-def is-least-false-clause-finsert-smaller-false-clause*  
*linorder-cls.is-least-in-fset-ffilterD(2) linorder-cls.leD match-hyps(3)*  
*next-clause-in-factorized-clause-def ord-res-5-invars-def ord-res-6-preserves-invars*  
*rtranclpD step' step-hyps(2) step-hyps(7)*

**hence**  $\{|C | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \rangle. C \prec_c \text{eres } D \ E|\} =$   
 $\{|C | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}) \rangle. C \prec_c \text{eres } D \ E|\}$   
**unfolding**  $\langle U_{er}' = \text{finsert } (\text{eres } D \ E) \ U_{er} \rangle$  **by** *auto*

**next**  
**show**  $\mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. \exists K. \text{ord-res.is-maximal-lit } K \ (\text{eres } D \ E)$   
 $\wedge A \prec_t \text{atm-of } K\}$   
**using**  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$   
**by** *(smt (verit, ccfv-SIG) Collect-cong linorder-lit.Uniq-is-maximal-in-mset*  
*step-hyps(10)*  
*the1-equality')*

**next**  
**show** *linorder-cls.is-least-in-fset (iefac  $\mathcal{F}$  |  $\uparrow (N | \cup | U_{er})$ )  $C$*   
**using** *C-least .*

**qed**

**ultimately have** *steps5: (ord-res-5  $N$ )<sup>++</sup> ( $U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E$ ) ( $U_{er}', \mathcal{F},$*   
 $\mathcal{M}', \text{Some } (\text{eres } D \ E)$ )

**by** *simp*

**thus** *ord-res-5-step<sup>++</sup>  $S5 \ S5'$*   
**unfolding**  *$S5$ -def  $\langle C = \text{Some } E \rangle \ S5'$ -def*  
**by** *(metis tranclp-ord-res-5-step-if-tranclp-ord-res-5)*

**show** *ord-res-5-matches-ord-res-6  $S5' \ S6'$*   
**unfolding**  *$S5'$ -def  $S6'$ -def  $\langle s6' = (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } (\text{eres } D \ E)) \rangle$*   
**using** *steps5*  
**using** *match-hyps(3) ord-res-5-matches-ord-res-6.intros ord-res-6-preserves-invars*  
*step'*  
*step-hyps(2) by metis*

**qed**

**next**  
**case** *step-hyps: (resolution-neg  $E \ L \ D \ U_{er}' \ \mathcal{M}' \ K \ C$ )*

**define**  *$S5' :: - \times - \times - \times (f \text{ gterm} \Rightarrow f \text{ gclause option}) \times f \text{ gclause option}$*   
**where**  
 $S5' = (N, U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)$

**hence** *iefac  $\mathcal{F}$  |  $\uparrow (N | \cup | U_{er}') \neq \{\}\}$*   
**using**  $\langle U_{er}' = \text{finsert } (\text{eres } D \ E) \ U_{er} \rangle$  **by** *simp*

**then obtain**  *$B$  where  $B$ -least: linorder-cls.is-least-in-fset (iefac  $\mathcal{F}$  |  $\uparrow (N | \cup |$*   
 $U_{er}') \rangle B$

**by** *(metis linorder-cls.ex1-least-in-fset)*

**show** *?thesis*  
**proof** (*intro exI conjI*)  
**have** *The-optional (linorder-cls.is-least-in-fset (iefac  $\mathcal{F}$  |<sup>q</sup> (N | $\cup$ |  $U_{er}'$ ))) =*  
*Some B*  
**proof** (*rule The-optional-eq-SomeI*)  
**show**  $\exists_{\leq 1} x. \text{linorder-cls.is-least-in-fset (iefac } \mathcal{F} \text{ |}^q \text{ (N |} \cup \text{| } U_{er}') \text{)) } x$   
**by** *blast*  
**next**  
**show** *linorder-cls.is-least-in-fset (iefac  $\mathcal{F}$  |<sup>q</sup> (N | $\cup$ |  $U_{er}'$ )) B*  
**using** *B-least .*  
**qed**

**hence** *step5: ord-res-5 N (U<sub>er</sub>,  $\mathcal{F}$ ,  $\mathcal{M}$ , Some E) (U<sub>er'</sub>,  $\mathcal{F}$ , Map.empty, Some*  
*B)*  
**using** *ord-res-5.resolution step-hyps by metis*

**moreover have** (*ord-res-5 N*)<sup>\*\*</sup> ... (*U<sub>er'</sub>,  $\mathcal{F}$ ,  $\mathcal{M}'$ , Some C*)  
**proof** (*rule full-rtranclp-ord-res-5-run-upto*)  
**show** *ord-res-6 N (U<sub>er</sub>,  $\mathcal{F}$ ,  $\mathcal{M}$ , Some E) (U<sub>er'</sub>,  $\mathcal{F}$ ,  $\mathcal{M}'$ , Some C)*  
**using** *step'  $\langle C = \text{Some } E \rangle \langle s6' = (U_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C) \rangle$  by argo*  
**next**  
**show** *ord-res-5-invars N (U<sub>er'</sub>,  $\mathcal{F}$ ,  $\mathcal{M}'$ , Some C)*  
**using** *match-hyps(3) ord-res-6-preserves-invars step' step-hyps(2) by blast*  
**next**  
**have** *ord-res.is-strictly-maximal-lit (Pos (atm-of K)) C*  
**using**  $\langle \mathcal{M} \text{ (atm-of } K) = \text{Some } C \rangle$   
 $\langle \text{ord-res-5-invars } N \text{ (} U_{er}, \mathcal{F}, \mathcal{M}, C) \rangle$  [*unfolded ord-res-5-invars-def*  
*atoms-in-model-were-produced-def, simplified*]  
**using** *ord-res.mem-productionE by blast*

**moreover have** *Pos (atm-of K)  $\prec_i$  K*  
**using**  $\langle \text{is-neg } K \rangle$  **by** (*cases K*) *simp-all*

**ultimately have** *C  $\prec_c$  eres D E*  
**using**  $\langle \text{ord-res.is-maximal-lit } K \text{ (eres } D \ E) \rangle$   
**by** (*metis ord-res.asymp-less-lit ord-res.transp-less-lit*  
*linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset*  
*linorder-lit.multip<sub>HO</sub>-if-maximal-less-that-maximal multp-eq-multp<sub>HO</sub>)*

**hence** *C  $\prec_c$  E*  
**using** *eres-le[of D E] by order*

**have** *C  $\prec_c$  efac (eres D E)*  
**by** (*metis Uniq-D  $\langle C \prec_c \text{eres } D \ E \rangle \text{efac-spec is-pos-def linorder-lit.Uniq-is-maximal-in-mset}$*   
*step-hyps(10) step-hyps(11))*

**moreover have** *efac (eres D E)  $\preceq_c$  eres D E*  
**by** (*metis efac-subset subset-implies-reflclp-multp*)

**ultimately have**  $C \prec_c \text{iefac } \mathcal{F} \text{ (eres } D \ E)$   
**unfolding** *iefac-def by auto*

**hence**  $\{|Ca | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}'). Ca \prec_c C|\} =$   
 $\{|Ca | \in | \text{iefac } \mathcal{F} | \uparrow (N | \cup | U_{er}). Ca \prec_c C|\}$   
**unfolding**  $\langle U_{er}' = \text{finsert (eres } D \ E) \ U_{er} \rangle$  **by auto**

**have**  $(\exists K. \text{ord-res.is-maximal-lit } K \ C \wedge A \prec_t \text{atm-of } K) \longleftrightarrow A \prec_t \text{atm-of } K$  **for } A**  
**using**  $\langle \text{ord-res.is-strictly-maximal-lit (Pos (atm-of } K)) \ C \rangle$   
**by**  $(\text{metis Uniq-def linorder-lit.Uniq-is-maximal-in-mset linorder-lit.is-maximal-in-mset-if-is-greatest-in-mset literal.sel(1)})$

**hence**  $\{A. \exists K. \text{ord-res.is-maximal-lit } K \ C \wedge A \prec_t \text{atm-of } K\} = \{A. A \prec_t \text{atm-of } K\}$   
**by** *metis*

**thus**  $\mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. \exists K. \text{ord-res.is-maximal-lit } K \ C \wedge A \prec_t \text{atm-of } K\}$   
**using**  $\langle \mathcal{M}' = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} \rangle$  **by argo**  
**next**  
**show** *linorder.cls.is-least-in-fset (iefac } \mathcal{F} | \uparrow (N | \cup | U\_{er}') \ B*  
**using** *B-least .*  
**qed**

**ultimately have** *steps5: (ord-res-5 N)<sup>++</sup> (U<sub>er</sub>, } \mathcal{F}, \mathcal{M}, \text{Some } E) (U\_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C)*  
**by** *simp*

**thus** *ord-res-5-step<sup>++</sup> S5 S5'*  
**unfolding** *S5-def } \langle C = \text{Some } E \rangle S5'-def*  
**by**  $(\text{metis tranclp-ord-res-5-step-if-tranclp-ord-res-5})$

**show** *ord-res-5-matches-ord-res-6 S5' S6'*  
**unfolding** *S5'-def S6'-def } \langle s6' = (U\_{er}', \mathcal{F}, \mathcal{M}', \text{Some } C) \rangle*  
**using** *steps5*  
**using** *match-hyps(3) ord-res-5-matches-ord-res-6.intros ord-res-6-preserves-invars step'*  
*step-hyps(2) by metis*  
**qed**  
**qed**  
**qed**

**theorem** *bisimulation-ord-res-5-ord-res-6:*  
**defines** *match*  $\equiv \lambda-. \text{ord-res-5-matches-ord-res-6}$   
**shows**  $\exists (\text{MATCH} :: \text{nat} \times \text{nat} \Rightarrow 'f \text{ord-res-5-state} \Rightarrow 'f \text{ord-res-6-state} \Rightarrow \text{bool})$   
 $\mathcal{R}_f \ \mathcal{R}_b.$   
*bisimulation ord-res-5-step ord-res-5-final ord-res-6-step ord-res-6-final MATCH*

```

 $\mathcal{R}_f \mathcal{R}_b$ 
proof (rule ex-bisimulation-from-backward-simulation)
  show right-unique ord-res-5-step
    using right-unique-ord-res-5-step .
next
  show right-unique ord-res-6-step
    using right-unique-ord-res-6-step .
next
  show  $\forall s. \text{ord-res-5-final } s \longrightarrow (\nexists s'. \text{ord-res-5-step } s s')$ 
    by (metis finished-def ord-res-5-semantics.final-finished)
next
  show  $\forall s. \text{ord-res-6-final } s \longrightarrow (\nexists s'. \text{ord-res-6-step } s s')$ 
    by (metis finished-def ord-res-6-semantics.final-finished)
next
  show  $\forall i S5 S6. \text{match } i S5 S6 \longrightarrow \text{ord-res-5-final } S5 \longleftrightarrow \text{ord-res-6-final } S6$ 
    unfolding match-def
    using ord-res-5-final-iff-ord-res-6-final by metis
next
  show  $\forall i S5 S6. \text{match } i S5 S6 \longrightarrow \text{safe-state ord-res-5-step ord-res-5-final } S5 \wedge \text{safe-state ord-res-6-step ord-res-6-final } S6$ 
proof (intro allI impI conjI)
  fix  $i S5 S6$ 
  assume match i S5 S6
  show safe-state ord-res-5-step ord-res-5-final S5
    using  $\langle \text{match } i S5 S6 \rangle$ 
    using ord-res-5-safe-state-if-invars
    using match-def ord-res-5-matches-ord-res-6.cases by metis
  show safe-state ord-res-6-step ord-res-6-final S6
    using  $\langle \text{match } i S5 S6 \rangle$ 
    using ord-res-6-safe-state-if-invars
    using match-def ord-res-5-matches-ord-res-6.cases by metis
  qed
next
  show wfp ( $\lambda - . \text{False}$ )
    by simp
next
  show  $\forall i S5 S6 S6'. \text{match } i S5 S6 \longrightarrow \text{ord-res-6-step } S6 S6' \longrightarrow (\exists i' S5'. \text{ord-res-5-step}^{++} S5 S5' \wedge \text{match } i' S5' S6') \vee (\exists i'. \text{match } i' S5 S6' \wedge \text{False})$ 
    unfolding match-def
    using backward-simulation-between-5-and-6 by metis
  qed
end

```

### 33 ORD-RES-7 (clause-guided literal trail construction)

**type-synonym** *'f ord-res-7-state* =  
*'f gclause fset* × *'f gclause fset* × *'f gclause fset* × (*'f gliteral* × *'f gclause option*)  
*list* ×  
*'f gclause option*

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-6-matches-ord-res-7* ::  
*'f gterm fset* ⇒ *'f ord-res-6-state* ⇒ *'f ord-res-7-state* ⇒ *bool* **where**  
*ord-res-5-invars* *N (U<sub>er</sub>, F, M, C)* ⇒  
*ord-res-7-invars* *N (U<sub>er</sub>, F, Γ, C)* ⇒  
(∀ *A C. M A = Some C* ⇔ *map-of Γ (Pos A) = Some (Some C)*) ⇒  
(∀ *A. M A = None* ⇔ *map-of Γ (Neg A) ≠ None* ∨ *A |∉| trail-atms Γ*) ⇒  
*i = atms-of-clss (N |∪| U<sub>er</sub>) - trail-atms Γ* ⇒  
*ord-res-6-matches-ord-res-7 i (N, U<sub>er</sub>, F, M, C) (N, U<sub>er</sub>, F, Γ, C)*

**lemma** *ord-res-6-final-iff-ord-res-7-final*:  
**fixes** *i S6 S7*  
**assumes** *match: ord-res-6-matches-ord-res-7 i S6 S7*  
**shows** *ord-res-6-final S6* ⇔ *ord-res-7-final S7*  
**using** *match*  
**proof** (*cases i S6 S7 rule: ord-res-6-matches-ord-res-7.cases*)  
**case** *match-hyps: (1 N U<sub>er</sub> F M C Γ)*

**show** *ord-res-6-final S6* ⇔ *ord-res-7-final S7*  
**proof** (*rule iffI*)  
**assume** *ord-res-6-final S6*  
**thus** *ord-res-7-final S7*  
**unfolding** *⟨S6 = (N, U<sub>er</sub>, F, M, C)⟩*  
**proof** (*cases (N, U<sub>er</sub>, F, M, C) rule: ord-res-6-final.cases*)  
**case** *model-found*  
**thus** *ord-res-7-final S7*  
**unfolding** *⟨S7 = (N, U<sub>er</sub>, F, Γ, C)⟩*  
**using** *ord-res-7-final.model-found*  
**by** *metis*  
**next**  
**case** *contradiction-found*  
**thus** *ord-res-7-final S7*  
**unfolding** *⟨S7 = (N, U<sub>er</sub>, F, Γ, C)⟩*  
**using** *ord-res-7-final.contradiction-found*  
**by** *metis*  
**qed**  
**next**  
**assume** *ord-res-7-final S7*  
**thus** *ord-res-6-final S6*  
**unfolding** *⟨S7 = (N, U<sub>er</sub>, F, Γ, C)⟩*

```

proof (cases (N, Uer, F, Γ, C) rule: ord-res-7-final.cases)
  case model-found
  thus ord-res-6-final S6
    unfolding ⟨S6 = (N, Uer, F, M, C)⟩
    using ord-res-6-final.model-found
    by metis
  next
  case contradiction-found
  thus ord-res-6-final S6
    unfolding ⟨S6 = (N, Uer, F, M, C)⟩
    using ord-res-6-final.contradiction-found
    by metis
  qed
qed
qed

lemma backward-simulation-between-6-and-7:
  fixes i S6 S7 S7'
  assumes match: ord-res-6-matches-ord-res-7 i S6 S7 and step: constant-context
  ord-res-7 S7 S7'
  shows
    (∃ i' S6'. ord-res-6-step++ S6 S6' ∧ ord-res-6-matches-ord-res-7 i' S6' S7') ∨
    (∃ i'. ord-res-6-matches-ord-res-7 i' S6 S7' ∧ i' |C| i)
  using match
proof (cases i S6 S7 rule: ord-res-6-matches-ord-res-7.cases)
  case match-hyps: (1 N Uer F M C Γ)

  note S6-def = ⟨S6 = (N, Uer, F, M, C)⟩
  note invars-6 = ⟨ord-res-5-invars N (Uer, F, M, C)⟩
  note invars-7 = ⟨ord-res-7-invars N (Uer, F, Γ, C)⟩[
    unfolded ord-res-7-invars-def, rule-format, OF refl]

  have Γ-sorted: sorted-wrt (λx y. atm-of (fst y) <t atm-of (fst x)) Γ
    using invars-7 by argo

  have Γ-consistent: trail-consistent Γ
    using invars-7 by (metis trail-consistent-if-sorted-wrt-atoms)

  hence Γ-distinct-atoms: distinct (map fst Γ)
    using distinct-lits-if-trail-consistent by iprover

  have clause-true-wrt-model-if-true-wrt-Γ: dom M ⊨ D
    if D-true: trail-true-cls Γ D for D
  proof –
    obtain L where L ∈# D and L-true: trail-true-lit Γ L
      using D-true unfolding trail-true-cls-def by auto

  have ∃C. (L, C) ∈ set Γ
    using L-true unfolding trail-true-lit-def by auto

```

```

show ?thesis
proof (cases L)
  case (Pos A)

    then obtain C where (Pos A, Some C) ∈ set Γ
      using invars-γ ⟨∃ C. (L, C) ∈ set Γ⟩
      by (metis fst-conv literal.disc(1) not-None-eq snd-conv)

    hence map-of Γ (Pos A) = Some (Some C)
      using Γ-distinct-atoms by (metis map-of-is-SomeI)

    hence M A = Some C
      using ⟨∀ A C. (M A = Some C) = (map-of Γ (Pos A) = Some (Some C))⟩
by metis

    hence A ∈ dom M
      by blast

    then show ?thesis
      using ⟨L ∈# D⟩ ⟨L = Pos A⟩ by blast
next
  case (Neg A)

    hence (Neg A, None) ∈ set Γ
      using invars-γ ⟨∃ C. (L, C) ∈ set Γ⟩
      by (metis fst-conv literal.disc(2) snd-conv)

    hence map-of Γ (Neg A) ≠ None
      by (simp add: weak-map-of-SomeI)

    hence M A = None
      using ⟨∀ A. (M A = None) = (map-of Γ (Neg A) ≠ None ∨ A |∉| trail-atms
Γ)⟩ by metis

    hence A ∉ dom M
      by blast

    then show ?thesis
      using ⟨L ∈# D⟩ ⟨L = Neg A⟩ by blast
qed
qed

have clause-false-wrt-model-if-false-wrt-Γ: ¬ dom M ≡ D
if D-false: trail-false-cls Γ D for D
unfolding true-cls-def
proof (intro notI , elim bexE)
  fix L :: 'f gterm literal
  assume L ∈# D and dom M ≡l L

```

```

have trail-false-lit  $\Gamma L$ 
  using  $\langle L \in \# D \rangle$  D-false unfolding trail-false-cls-def by metis

hence  $\neg$  trail-true-lit  $\Gamma L$  and trail-defined-lit  $\Gamma L$ 
  unfolding atomize-conj
  using  $\Gamma$ -consistent  $\langle L \in \# D \rangle$  not-trail-true-cls-and-trail-false-cls that
  trail-defined-lit-iff-true-or-false trail-true-cls-def by blast

show False
proof (cases L)
  case (Pos A)

    hence  $\mathcal{M} A \neq \text{None}$ 
      using  $\langle \text{dom } \mathcal{M} \models L \rangle$  by blast

    hence map-of  $\Gamma$  (Pos A)  $\neq \text{None}$ 
      using  $\langle \forall A C. (\mathcal{M} A = \text{Some } C) = (\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C)) \rangle$ 
by blast

    hence Pos A  $\in$  fst ' set  $\Gamma$ 
      by (simp add: map-of-eq-None-iff)

    hence trail-true-lit  $\Gamma$  (Pos A)
      unfolding trail-true-lit-def .

    moreover have  $\neg$  trail-true-lit  $\Gamma$  (Pos A)
      using  $\langle \neg$  trail-true-lit  $\Gamma L \rangle$   $\langle L = \text{Pos } A \rangle$  by argo

    ultimately show False
      by contradiction
  next
  case (Neg A)

    hence  $\mathcal{M} A = \text{None}$ 
      using  $\langle \text{dom } \mathcal{M} \models L \rangle$  by blast

    hence map-of  $\Gamma$  (Neg A)  $\neq \text{None} \vee A \notin$  trail-atms  $\Gamma$ 
      using  $\langle \forall A. (\mathcal{M} A = \text{None}) = (\text{map-of } \Gamma (\text{Neg } A) \neq \text{None} \vee A \notin$  trail-atms
 $\Gamma) \rangle$  by blast

    hence trail-true-lit  $\Gamma$  (Neg A)  $\vee \neg$  trail-defined-lit  $\Gamma$  (Neg A)
      unfolding map-of-eq-None-iff not-not
      unfolding trail-true-lit-def trail-defined-lit-iff-trail-defined-atm literal.sel
      .

    then show ?thesis
      using  $\langle \neg$  trail-true-lit  $\Gamma L \rangle$   $\langle$  trail-defined-lit  $\Gamma L \rangle$   $\langle L = \text{Neg } A \rangle$  by argo
qed

```



qed

obtain  $s\gamma'$  where

$S\gamma' = (N, s\gamma')$  and

$step'$ :  $ord-res-7\ N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})\ s\gamma'$

using **step unfolding**  $\langle S\gamma' = (N, U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle$

by (*auto elim: constant-context.cases*)

have  $invars-s\gamma'$ :  $ord-res-7-invars\ N\ s\gamma'$

using  $ord-res-7-preserves-invars[OF\ step'\ \langle ord-res-7-invars\ N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle]$

show *?thesis*

using  $step'$

proof (*cases*  $N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})\ s\gamma'$  *rule: ord-res-7.cases*)

case  $step-hyps$ : (*decide-neg*  $C\ L\ A\ \Gamma'$ )

define  $i'$  where

$i' = atms-of-clss\ (N\ |\cup|\ U_{er})\ |-|\ trail-atms\ \Gamma'$

have  $A\ |\in|\ atms-of-clss\ (N\ |\cup|\ U_{er})$  and  $A\ \prec_t\ atm-of\ L$  and  $A\ |\notin|\ trail-atms\ \Gamma$

using  $step-hyps\ unfolding\ atomize-conj\ linorder-trm.is-least-in-filter-iff$  by *argo*

have  $ord-res-6-matches-ord-res-7\ i'\ S6\ S\gamma'$

$unfolding\ S6-def\ \langle C = Some\ C \rangle\ \langle S\gamma' = (N, s\gamma') \rangle\ \langle s\gamma' = (U_{er}, \mathcal{F}, \Gamma', Some\ C) \rangle$

proof (*rule*  $ord-res-6-matches-ord-res-7.intros$ )

show  $ord-res-5-invars\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C)$

using  $invars-6\ unfolding\ \langle C = Some\ C \rangle$  .

next

show  $ord-res-7-invars\ N\ (U_{er}, \mathcal{F}, \Gamma', Some\ C)$

using  $invars-s\gamma'\ unfolding\ \langle s\gamma' = (U_{er}, \mathcal{F}, \Gamma', Some\ C) \rangle$  .

next

show  $\forall A\ C.\ (\mathcal{M}\ A = Some\ C) = (map-of\ \Gamma'\ (Pos\ A) = Some\ (Some\ C))$

using  $match-hyps\ unfolding\ \langle \Gamma' = (Neg\ A, None) \# \Gamma \rangle$  by *simp*

next

show  $\forall A.\ (\mathcal{M}\ A = None) = (map-of\ \Gamma'\ (Neg\ A) \neq None \vee A\ |\notin|\ trail-atms\ \Gamma')$

$unfolding\ \langle \Gamma' = (Neg\ A, None) \# \Gamma \rangle$

using  $match-hyps\ \langle A\ |\notin|\ trail-atms\ \Gamma \rangle$  by *force*

next

show  $i' = atms-of-clss\ (N\ |\cup|\ U_{er})\ |-|\ trail-atms\ \Gamma'$

$unfolding\ i'-def\ ..$

qed

moreover have  $i' |\subset|\ i$

proof –

```

have  $i = \text{finsert } A \ i'$ 
  unfolding  $\text{match-hyps } i'\text{-def}$ 
  using  $\langle A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle \langle A \mid \notin \mid \text{trail-atms } \Gamma \rangle \text{step-hyps}(6)$  by
force

moreover have  $A \mid \notin \mid i'$ 
  unfolding  $i'\text{-def}$ 
  using  $\text{step-hyps}(6)$  by fastforce

ultimately show  $?thesis$ 
  by auto
qed

ultimately show  $?thesis$ 
  by metis
next
case  $\text{step-hyps}: (\text{skip-defined } C \ L \ C')$ 

define  $S6'$  where
   $S6' = (N, U_{er}, \mathcal{F}, \mathcal{M}, C')$ 

have  $C\text{-almost-defined}: \text{trail-defined-cls } \Gamma \ \{\#x \in \# \ C. \ x \neq L\# \}$ 
  using  $\text{step-hyps}$  by (metis clause-almost-definedI invars-7)

hence  $C\text{-defined}: \text{trail-defined-cls } \Gamma \ C$ 
  using  $\text{step-hyps}$  unfolding  $\text{trail-defined-cls-def}$  by auto

hence  $C\text{-true}: \text{trail-true-cls } \Gamma \ C$ 
  using  $\text{step-hyps}$  by (metis trail-true-or-false-cls-if-defined)

have  $\text{step6}: \text{ord-res-6 } N \ (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) \ (U_{er}, \mathcal{F}, \mathcal{M}, C')$ 
proof (rule ord-res-6.skip)
  show  $\text{dom } \mathcal{M} \models C$ 
  using  $\text{clause-true-wrt-model-if-true-wrt-}\Gamma[\text{OF } C\text{-true}]$  .
next
  show  $C' = \text{The-optional } (\text{linorder-cls.is-least-in-fset}$ 
    (ffilter ( $\prec_c$ )  $C$ ) (iefac  $\mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$ )))
  using  $\text{step-hyps}$  by argo
qed

hence  $\text{ord-res-6-step}^{++} \ S6 \ S6'$ 
  using  $S6\text{-def } \langle C = \text{Some } C \rangle \ S6'\text{-def } \text{ord-res-6-step.intros}$  by blast

moreover have  $\text{ord-res-6-matches-ord-res-7 } i \ S6' \ S7'$ 
  unfolding  $S6'\text{-def } \langle S7' = (N, s7') \rangle \langle s7' = (U_{er}, \mathcal{F}, \Gamma, C') \rangle$ 
proof (rule ord-res-6-matches-ord-res-7.intros)
  show  $\text{ord-res-5-invars } N \ (U_{er}, \mathcal{F}, \mathcal{M}, C')$ 
  using  $\text{invars-6}$  unfolding  $\langle C = \text{Some } C \rangle$ 
  using  $\text{ord-res-6-preserves-invars}[\text{OF } \text{step6}]$  by argo

```

```

next
  show ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}')$ 
    using invars-s7' unfolding  $\langle s7' = (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}') \rangle$  .
next
  show  $\forall A C. (\mathcal{M} A = \text{Some } C) = (\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C))$ 
    using match-hyps by argo
next
  show  $\forall A. (\mathcal{M} A = \text{None}) = (\text{map-of } \Gamma (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms}$ 
 $\Gamma)$ 
    using match-hyps by argo
next
  show  $i = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \mid - \mid \text{trail-atms } \Gamma$ 
    using match-hyps by argo
qed

ultimately show ?thesis
  by metis
next
  case step-hyps:  $(\text{skip-undefined-neg } C L \Gamma' \mathcal{C}')$ 

define S6' where
   $S6' = (N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}')$ 

define i' where
   $i' = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \mid - \mid \text{trail-atms } \Gamma'$ 

have trail-true-lit  $\Gamma' L$ 
  unfolding  $\langle \Gamma' = (L, \text{None}) \# \Gamma \rangle$  by (simp add: trail-true-lit-def)

hence C-true: trail-true-cls  $\Gamma' C$ 
  using step-hyps unfolding linorder-lit.is-maximal-in-mset-iff trail-true-cls-def
  by metis

have step6: ord-res-6  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}')$ 
proof (rule ord-res-6.skip)
  show dom  $\mathcal{M} \models C$ 
    using C-true
  by (metis domIff linorder-lit.is-maximal-in-mset-iff literal.collapse(2) match-hyps(6)
    step-hyps(4) step-hyps(6) step-hyps(7) trail-defined-lit-iff-trail-defined-atm
    true-cls-def true-lit-simps(2))
next
  show  $\mathcal{C}' = \text{The-optional } (\text{linorder-cls.is-least-in-fset}$ 
 $(\text{ffilter } ((\prec_c) C) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))))$ 
    using step-hyps by argo
qed

hence ord-res-6-step++  $S6 S6'$ 
  using S6-def  $\langle C = \text{Some } C \rangle S6'$ -def ord-res-6-step.intros by blast

```

```

moreover have ord-res-6-matches-ord-res-7  $i' S6' S7'$ 
  unfolding  $S6'\text{-def}$   $\langle S7' = (N, s7') \rangle \langle s7' = (U_{er}, \mathcal{F}, \Gamma', C') \rangle$ 
proof (rule ord-res-6-matches-ord-res-7.intros)
  show ord-res-5-invars  $N (U_{er}, \mathcal{F}, \mathcal{M}, C')$ 
    using invars-6 unfolding  $\langle C = \text{Some } C \rangle$ 
    using ord-res-6-preserves-invars[OF step6] by argo
next
  show ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma', C')$ 
    using invars-s7' unfolding  $\langle s7' = (U_{er}, \mathcal{F}, \Gamma', C') \rangle$  .
next
  show  $\forall A C. (\mathcal{M} A = \text{Some } C) = (\text{map-of } \Gamma' (\text{Pos } A) = \text{Some } (\text{Some } C))$ 
    using match-hyps
    unfolding  $\langle \Gamma' = (L, \text{None}) \# \Gamma \rangle$ 
    by (metis literal.disc(1) map-of-Cons-code(2) step-hyps(7))
next
  show  $\forall A. (\mathcal{M} A = \text{None}) = (\text{map-of } \Gamma' (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma')$ 
    using match-hyps
    unfolding  $\langle \Gamma' = (L, \text{None}) \# \Gamma \rangle$ 
    by (metis finsert-iff literal.collapse(2) literal.sel(2) map-of-Cons-code(2))
option.discI
    prod.sel(1) step-hyps(6) step-hyps(7) trail-atms.simps(2)
    trail-defined-lit-iff-trail-defined-atm
next
  show  $i' = \text{atms-of-clss } (N \cup U_{er}) \text{ |- } \text{trail-atms } \Gamma'$ 
    using i'-def .
qed

ultimately show ?thesis
  by metis
next
case step-hyps: (skip-undefined-pos C L D)

define  $S6'$  where
   $S6' = (N, U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D)$ 

have trail-defined-cls  $\Gamma \{\#x \in \# C. x \neq L \wedge x \neq -L\# \}$ 
proof (rule clause-almost-almost-definedI)
  show  $C \in \text{iefac } \mathcal{F} \uparrow (N \cup U_{er})$ 
    using invars-7 step-hyps by metis
next
  show ord-res.is-maximal-lit  $L C$ 
    using step-hyps by argo
next
  show  $\neg (\exists A \in \text{atms-of-clss } (N \cup U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \text{trail-atms } \Gamma)$ 
    using step-hyps by argo
qed

moreover have  $-L \notin \# C$ 

```

**by** (*metis atm-of-uminus is-pos-def linorder-lit.is-maximal-in-mset-iff linorder-lit.neqE  
linorder-trm.less-irrefl literal.collapse(2) literal.sel(1) ord-res.less-lit-simps(4)  
step-hyps(4) step-hyps(7) uminus-not-id'*)

**ultimately have** *trail-defined-cls*  $\Gamma \{\#x \in \# C. x \neq L\# \}$   
**unfolding** *trail-defined-cls-def* **by** *auto*

**hence** *trail-true-cls*  $\Gamma \{\#x \in \# C. x \neq L\# \}$   
**using**  $\langle \neg \text{trail-false-cls } \Gamma \{\#K \in \# C. K \neq L\# \} \rangle$  **by** (*metis trail-true-or-false-cls-if-defined*)

**hence** *C-true*: *trail-true-cls*  $\Gamma C$   
**by** (*auto simp: trail-true-cls-def*)

**have** *step6*: *ord-res-6*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } C) (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D)$   
**proof** (*rule ord-res-6.skip*)  
**show**  $\text{dom } \mathcal{M} \models C$   
**using** *clause-true-wrt-model-if-true-wrt- $\Gamma$ [OF C-true]* .

**next**  
**show**  $\text{Some } D = \text{The-optional } (\text{linorder-cls.is-least-in-fset } (\text{ffilter } (\langle \prec_c \rangle C) (\text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))))$   
**using** *linorder-cls.Uniq-is-least-in-fset step-hyps(9) The-optional-eq-SomeI*  
**by** *fastforce*

**qed**

**hence** *ord-res-6-step*<sup>++</sup>  $S6 S6'$   
**using** *S6-def*  $\langle C = \text{Some } C \rangle$  *S6'-def ord-res-6-step.intros* **by** *blast*

**moreover have** *ord-res-6-matches-ord-res-7*  $i S6' S7'$   
**unfolding** *S6'-def*  $\langle S7' = (N, s7') \rangle \langle s7' = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$   
**proof** (*rule ord-res-6-matches-ord-res-7.intros*)  
**show** *ord-res-5-invars*  $N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } D)$   
**using** *invars-6 unfolding*  $\langle C = \text{Some } C \rangle$   
**using** *ord-res-6-preserves-invars[OF step6]* **by** *argo*

**next**  
**show** *ord-res-7-invars*  $N (U_{er}, \mathcal{F}, \Gamma, \text{Some } D)$   
**using** *invars-s7' unfolding*  $\langle s7' = (U_{er}, \mathcal{F}, \Gamma, \text{Some } D) \rangle$  .

**next**  
**show**  $\forall A C. (\mathcal{M} A = \text{Some } C) = (\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C))$   
**using** *match-hyps* **by** *argo*

**next**  
**show**  $\forall A. (\mathcal{M} A = \text{None}) = (\text{map-of } \Gamma (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma)$   
**using** *match-hyps* **by** *argo*

**next**  
**show**  $i = \text{atms-of-cls } (N \mid \cup \mid U_{er}) \mid - \mid \text{trail-atms } \Gamma$   
**using** *match-hyps* **by** *argo*

**qed**

**ultimately show** *?thesis*

by *metis*  
 next  
 case *step-hyps*: (*skip-undefined-pos-ultimate*  $C L \Gamma$ )  
  
 define *S6'* where  
 $S6' = (N, U_{er}, \mathcal{F}, \mathcal{M}, None :: 'f gclause option)$   
  
 define *i'* where  
 $i' = atms-of-cls (N \mid \cup \mid U_{er}) \mid - \mid trail-atms \Gamma'$   
  
 have *trail-defined-cls*  $\Gamma \{ \#x \in \# C. x \neq L \wedge x \neq - L \# \}$   
 proof (*rule clause-almost-almost-definedI*)  
 show  $C \mid \in \mid iefac \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er})$   
 using *invars-7 step-hyps* by *metis*  
 next  
 show *ord-res.is-maximal-lit*  $L C$   
 using *step-hyps* by *argo*  
 next  
 show  $\neg (\exists A \mid \in \mid atms-of-cls (N \mid \cup \mid U_{er}). A \prec_t atm-of L \wedge A \mid \notin \mid trail-atms \Gamma)$   
 using *step-hyps* by *argo*  
 qed  
  
 moreover have  $- L \notin \# C$   
 by (*metis atm-of-uminus is-pos-def linorder-lit.is-maximal-in-mset-iff linorder-lit.neqE*  
*linorder-trm.less-irrefl literal.collapse(2) literal.sel(1) ord-res.less-lit-simps(4)*  
*step-hyps(4) step-hyps(7) uminus-not-id')*  
  
 ultimately have *trail-defined-cls*  $\Gamma \{ \#x \in \# C. x \neq L \# \}$   
 unfolding *trail-defined-cls-def* by *auto*  
  
 hence *trail-true-cls*  $\Gamma \{ \#x \in \# C. x \neq L \# \}$   
 using  $\langle \neg trail-false-cls \Gamma \{ \#K \in \# C. K \neq L \# \} \rangle$  by (*metis trail-true-or-false-cls-if-defined*)  
  
 hence *C-true*: *trail-true-cls*  $\Gamma C$   
 by (*auto simp: trail-true-cls-def*)  
  
 have *step6*: *ord-res-6*  $N (U_{er}, \mathcal{F}, \mathcal{M}, Some C) (U_{er}, \mathcal{F}, \mathcal{M}, None)$   
 proof (*rule ord-res-6.skip*)  
 show *dom*  $\mathcal{M} \models C$   
 using *clause-true-wrt-model-if-true-wrt-Γ[OF C-true]* .  
 next  
 have  $\neg (\exists D. linorder-cls.is-least-in-fset (ffilter ((\prec_c) C) (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))) D)$   
 using  $\langle \neg fBex (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})) ((\prec_c) C) \rangle$   
 by (*meson linorder-cls.is-least-in-ffilter-iff*)  
  
 thus  $None = The-optional (linorder-cls.is-least-in-fset (ffilter ((\prec_c) C) (iefac \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}))))$   
 unfolding *The-optional-def* by *metis*

qed

hence  $ord\text{-}res\text{-}6\text{-}step^{++} S6 S6'$   
 using  $S6\text{-}def \langle C = Some C \rangle S6'\text{-}def ord\text{-}res\text{-}6\text{-}step.intros$  by blast

moreover have  $ord\text{-}res\text{-}6\text{-}matches\text{-}ord\text{-}res\text{-}7 i' S6' S7'$   
 unfolding  $S6'\text{-}def \langle S7' = (N, s7') \rangle \langle s7' = (U_{er}, \mathcal{F}, \Gamma', None) \rangle$   
 proof (rule  $ord\text{-}res\text{-}6\text{-}matches\text{-}ord\text{-}res\text{-}7.intros$ )  
 show  $ord\text{-}res\text{-}5\text{-}invars N (U_{er}, \mathcal{F}, \mathcal{M}, None)$   
 using  $invars\text{-}6$  unfolding  $\langle C = Some C \rangle$   
 using  $ord\text{-}res\text{-}6\text{-}preserves\text{-}invars[OF step6]$  by argo  
 next  
 show  $ord\text{-}res\text{-}7\text{-}invars N (U_{er}, \mathcal{F}, \Gamma', None)$   
 using  $invars\text{-}s7'$  unfolding  $\langle s7' = (U_{er}, \mathcal{F}, \Gamma', None) \rangle$  .  
 next  
 show  $\forall A C. (\mathcal{M} A = Some C) = (map\text{-}of \Gamma' (Pos A) = Some (Some C))$   
 using  $match\text{-}hyps(3-)$   
 unfolding  $\langle \Gamma' = (- L, None) \# \Gamma \rangle$   
 by ( $metis is\text{-}pos\text{-}neg\text{-}not\text{-}is\text{-}pos literal.disc(1) map\text{-}of\text{-}Cons\text{-}code(2) step\text{-}hyps(7)$ )  
 next  
 show  $\forall A. (\mathcal{M} A = None) = (map\text{-}of \Gamma' (Neg A) \neq None \vee A \notin trail\text{-}atms$   
 $\Gamma')$   
 using  $match\text{-}hyps(3-)$   
 unfolding  $\langle \Gamma' = (- L, None) \# \Gamma \rangle$   
 by ( $metis (no\text{-}types, opaque\text{-}lifting) atm\text{-}of\text{-}eq\text{-}atm\text{-}of eq\text{-}fst\text{-}iff fset\text{-}simps(2)$ )  
 $insertCI$   
 $insertE literal.discI(2) literal.sel(2) map\text{-}of\text{-}Cons\text{-}code(2) option.distinct(1)$   
 $trail\text{-}defined\text{-}lit\text{-}iff\text{-}trail\text{-}defined\text{-}atm step\text{-}hyps(6) step\text{-}hyps(7) trail\text{-}atms.simps(2)$ )  
 next  
 show  $i' = atms\text{-}of\text{-}clss (N \cup U_{er}) \text{-} trail\text{-}atms \Gamma'$   
 using  $i'\text{-}def$  .  
 qed

ultimately show  $?thesis$   
 by  $metis$

next  
 case  $step\text{-}hyps: (production C L \Gamma' C')$

define  $S6'$  where  
 $S6' = (N, U_{er}, \mathcal{F}, \mathcal{M}(atm\text{-}of L \mapsto C), C')$

define  $i'$  where  
 $i' = atms\text{-}of\text{-}clss (N \cup U_{er}) \text{-} trail\text{-}atms \Gamma'$

have  $L \in \# C$   
 using  $step\text{-}hyps$  unfolding  $linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\text{-}iff$  by argo

have  $step6: ord\text{-}res\text{-}6 N (U_{er}, \mathcal{F}, \mathcal{M}, Some C) (U_{er}, \mathcal{F}, \mathcal{M}(atm\text{-}of L \mapsto C), C')$

**proof** (*rule ord-res-6.production*)  
**have**  $atm\text{-of } L \notin trail\text{-atms } \Gamma$   
**using**  $\langle \neg trail\text{-defined-lit } \Gamma L \rangle$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm* .

**hence**  $\mathcal{M}(atm\text{-of } L) = None$   
**using** *match-hyps(3-)* **by** *metis*

**hence**  $atm\text{-of } L \notin dom \mathcal{M}$   
**unfolding** *dom-def* **by** *simp*

**hence**  $\neg dom \mathcal{M} \models_l L$   
**using**  $\langle is\text{-pos } L \rangle$  **unfolding** *true-lit-def* **by** *metis*

**moreover have**  $\neg dom \mathcal{M} \models \{\#K \in\# C. K \neq L\# \}$   
**using** *clause-false-wrt-model-if-false-wrt- $\Gamma$ [OF  $\langle trail\text{-false-cls } \Gamma \{\#K \in\# C. K \neq L\# \} \rangle$ ]* .

**ultimately show**  $\neg dom \mathcal{M} \models C$   
**using**  $\langle L \in\# C \rangle$   
**unfolding** *true-cls-def* **by** *auto*

**next**  
**show** *ord-res.is-maximal-lit*  $L C$   
**using** *step-hyps* **by** *argo*

**next**  
**show** *is-pos*  $L$   
**using** *step-hyps* **by** *argo*

**next**  
**show** *ord-res.is-strictly-maximal-lit*  $L C$   
**using** *step-hyps* **by** *argo*

**next**  
**show**  $\mathcal{M}(atm\text{-of } L \mapsto C) = \mathcal{M}(atm\text{-of } L \mapsto C) ..$

**next**  
**show**  $C' = The\text{-optional } (linorder\text{-cls.is-least-in-fset } (ffilter ((<_c) C) (iefac \mathcal{F} \mid \uparrow (N \mid \cup U_{er}))))$   
**using** *step-hyps* **by** *argo*

**qed**

**hence** *ord-res-6-step<sup>++</sup>*  $S6 S6'$   
**using** *S6-def*  $\langle C = Some C \rangle$  *S6'-def* *ord-res-6-step.intros* **by** *blast*

**moreover have** *ord-res-6-matches-ord-res-7*  $i' S6' S7'$   
**unfolding** *S6'-def*  $\langle S7' = (N, s7') \rangle \langle s7' = (U_{er}, \mathcal{F}, \Gamma', C') \rangle$

**proof** (*rule ord-res-6-matches-ord-res-7.intros*)  
**show** *ord-res-5-invars*  $N (U_{er}, \mathcal{F}, \mathcal{M}(atm\text{-of } L \mapsto C), C')$   
**using** *invars-6* **unfolding**  $\langle C = Some C \rangle$   
**using** *ord-res-6-preserves-invars[OF step6]* **by** *argo*

**next**  
**show** *ord-res-7-invars*  $N (U_{er}, \mathcal{F}, \Gamma', C')$



```

    using invars-s7' unfolding ⟨s7' = (Uer, F, Γ', C')⟩ .
  next
    show ∀ A D. ((M(atm-of L ↦ C)) A = Some D) = (map-of Γ' (Pos A) =
Some (Some D))
    using match-hyps(3-)
    unfolding ⟨Γ' = (L, Some C) # Γ⟩
    using step-hyps(7) by auto
  next
    show ∀ A. ((M(atm-of L ↦ C)) A = None) = (map-of Γ' (Neg A) ≠ None
∨ A |∉| trail-atms Γ')
    using match-hyps(3-)
    unfolding ⟨Γ' = (L, Some C) # Γ⟩
    by (metis (no-types, opaque-lifting) domI domIff finsert-iff fun-upd-apply
literal.collapse(1) literal.discI(2) map-of-Cons-code(2) map-of-eq-None-iff
prod.sel(1)
step-hyps(6) step-hyps(7) trail-atms.simps(2) trail-defined-lit-def umi-
nus-Pos)
  next
    show i' = atms-of-cls (N |∪| Uer) |-| trail-atms Γ'
    using i'-def .
  qed

ultimately show ?thesis
  by metis
next
case step-hyps: (factoring C L F')

define S6' where
  S6' = (N, Uer, F', M, Some (efac C))

have L ∈# C
  using step-hyps unfolding linorder-lit.is-maximal-in-mset-iff by argo

have step6: ord-res-6 N (Uer, F, M, Some C) (Uer, F', M, Some (efac C))
proof (rule ord-res-6.factoring)
  have atm-of L |∉| trail-atms Γ
    using ⟨¬ trail-defined-lit Γ L⟩
    unfolding trail-defined-lit-iff-trail-defined-atm .

  hence M (atm-of L) = None
    using match-hyps(3-) by metis

  hence atm-of L ∉ dom M
    unfolding dom-def by simp

  hence ¬ dom M ||l L
    using ⟨is-pos L⟩ unfolding true-lit-def by metis

  moreover have ¬ dom M || {#K ∈# C. K ≠ L#}

```

**using** *clause-false-wrt-model-if-false-wrt-Γ*[*OF*  $\langle \text{trail-false-cls } \Gamma \{ \#K \in \# C. K \neq L\# \} \rangle$ ].

**ultimately show**  $\neg \text{dom } \mathcal{M} \models C$   
**using**  $\langle L \in \# C \rangle$   
**unfolding** *true-cls-def* **by** *auto*  
**next**  
**show** *ord-res.is-maximal-lit*  $L C$   
**using** *step-hyps* **by** *argo*  
**next**  
**show** *is-pos*  $L$   
**using** *step-hyps* **by** *argo*  
**next**  
**show**  $\neg \text{ord-res.is-strictly-maximal-lit } L C$   
**using** *step-hyps* **by** *argo*  
**next**  
**show**  $\mathcal{F}' = \text{finsert } C \mathcal{F}$   
**using** *step-hyps* **by** *argo*  
**qed**

**hence** *ord-res-6-step*<sup>++</sup>  $S6 S6'$   
**using** *S6-def*  $\langle C = \text{Some } C \rangle$  *S6'-def* *ord-res-6-step.intros* **by** *blast*

**moreover have** *ord-res-6-matches-ord-res-7*  $i S6' S7'$   
**unfolding** *S6'-def*  $\langle S7' = (N, s7') \rangle$   $\langle s7' = (U_{er}, \mathcal{F}', \Gamma, \text{Some } (\text{efac } C)) \rangle$   
**proof** (*rule ord-res-6-matches-ord-res-7.intros*)  
**show** *ord-res-5-invars*  $N (U_{er}, \mathcal{F}', \mathcal{M}, \text{Some } (\text{efac } C))$   
**using** *invars-6* **unfolding**  $\langle C = \text{Some } C \rangle$   
**using** *ord-res-6-preserves-invars*[*OF step6*] **by** *argo*  
**next**  
**show** *ord-res-7-invars*  $N (U_{er}, \mathcal{F}', \Gamma, \text{Some } (\text{efac } C))$   
**using** *invars-s7'* **unfolding**  $\langle s7' = (U_{er}, \mathcal{F}', \Gamma, \text{Some } (\text{efac } C)) \rangle$ .

**next**  
**show**  $\forall A C. (\mathcal{M} A = \text{Some } C) = (\text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C))$   
**using** *match-hyps*( $\beta-$ ) **by** *argo*

**next**  
**show**  $\forall A. (\mathcal{M} A = \text{None}) = (\text{map-of } \Gamma (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma)$   
**using** *match-hyps*( $\beta-$ ) **by** *argo*

**next**  
**show**  $i = \text{atms-of-cls } (N \cup U_{er}) \mid - \mid \text{trail-atms } \Gamma$   
**using** *match-hyps*( $\beta-$ ) **by** *argo*  
**qed**

**ultimately show** *?thesis*  
**by** *metis*

**next**  
**case** *step-hyps*: (*resolution-bot*  $E L D U_{er}' \Gamma'$ )

```

define  $S6'$  where
   $S6' = (N, U_{er}', \mathcal{F}, (\lambda-. None) :: 'f gterm \Rightarrow 'f gclause option, Some (\{\#\} :: 'f gclause))$ 

define  $i'$  where
   $i' = atms-of-cls (N \cup U_{er}') \mid - \mid trail-atms \Gamma'$ 

have  $step6$ :  $ord-res-6 N (U_{er}, \mathcal{F}, \mathcal{M}, Some E) (U_{er}', \mathcal{F}, \lambda-. None, Some \{\#\})$ 
proof ( $rule\ ord-res-6.resolution-bot$ )
  show  $\neg dom \mathcal{M} \Vdash E$ 
    using  $clause-false-wrt-model-if-false-wrt-\Gamma[OF \langle trail-false-cls \Gamma E \rangle]$  .
next
  show  $ord-res.is-maximal-lit L E$ 
    using  $step-hyps$  by  $argo$ 
next
  show  $is-neg L$ 
    using  $step-hyps$  by  $argo$ 
next
  show  $\mathcal{M} (atm-of L) = Some D$ 
    by ( $metis\ literal.collapse(2)\ match-hyps(5)\ step-hyps(5)\ step-hyps(6)\ umi-nus-Neg$ )
next
  show  $U_{er}' = finsert (eres D E) U_{er}$ 
    using  $step-hyps$  by  $argo$ 
next
  show  $eres D E = \{\#\}$ 
    using  $step-hyps$  by  $argo$ 
next
  show  $((\lambda-. None)) = (\lambda-. None) ..$ 
qed

hence  $ord-res-6-step^{++} S6 S6'$ 
  using  $S6-def \langle C = Some E \rangle S6'-def\ ord-res-6-step.intros$  by  $blast$ 

moreover have  $ord-res-6-matches-ord-res-7 i' S6' S7'$ 
  unfolding  $S6'-def \langle S7' = (N, s7') \rangle \langle s7' = (U_{er}', \mathcal{F}, \Gamma', Some \{\#\}) \rangle$ 
proof ( $rule\ ord-res-6-matches-ord-res-7.intros$ )
  show  $ord-res-5-invars N (U_{er}', \mathcal{F}, \lambda-. None, Some \{\#\})$ 
    using  $invars-6$  unfolding  $\langle C = Some E \rangle$ 
    using  $ord-res-6-preserves-invars[OF\ step6]$  by  $argo$ 
next
  show  $ord-res-7-invars N (U_{er}', \mathcal{F}, \Gamma', Some \{\#\})$ 
    using  $invars-s7'$  unfolding  $\langle s7' = (U_{er}', \mathcal{F}, \Gamma', Some \{\#\}) \rangle$  .
next
  show  $\forall A C. (None = Some C) = (map-of \Gamma' (Pos A) = Some (Some C))$ 
    unfolding  $\langle \Gamma' = [] \rangle$  by  $simp$ 
next
  show  $\forall A. (None = None) = (map-of \Gamma' (Neg A) \neq None \vee A \notin trail-atms \Gamma')$ 

```

```

    unfolding ⟨ $\Gamma' = []$ ⟩ by simp
  next
    show  $i' = \text{atms-of-clss } (N \mid \cup \mid U_{er}') \mid - \mid \text{trail-atms } \Gamma'$ 
      using  $i'\text{-def}$  .
  qed

  ultimately show ?thesis
    by metis
  next
  case step-hyps: (resolution-pos E L D  $U_{er}'$   $\Gamma'$  K)

  define  $S6'$  where
     $S6' = (N, U_{er}', \mathcal{F}, \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\}, \text{Some } (\text{eres } D \ E))$ 

  define  $i'$  where
     $i' = \text{atms-of-clss } (N \mid \cup \mid U_{er}') \mid - \mid \text{trail-atms } \Gamma'$ 

  have mem-set- $\Gamma'$ -iff:  $\bigwedge x. (x \in \text{set } \Gamma') = (\text{atm-of } (\text{fst } x) \prec_t \text{atm-of } K \wedge x \in \text{set } \Gamma)$ 
  unfolding ⟨ $\Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma$ ⟩
  unfolding mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone[OF  $\Gamma$ -sorted mono-atms-lt]
  by auto

  have step6: ord-res-6 N ( $U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E$ ) ( $U_{er}', \mathcal{F}, \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\}, \text{Some } (\text{eres } D \ E))$ 
  proof (rule ord-res-6.resolution-pos)
    show  $\neg \text{dom } \mathcal{M} \Vdash E$ 
      using clause-false-wrt-model-if-false-wrt- $\Gamma$ [OF ⟨trail-false-clt  $\Gamma \ E$ ⟩] .
  next
  show ord-res.is-maximal-lit L E
    using step-hyps by argo
  next
  show is-neg L
    using step-hyps by argo
  next
  show  $\mathcal{M} (\text{atm-of } L) = \text{Some } D$ 
    by (metis literal.collapse(2) match-hyps(5) step-hyps(5) step-hyps(6) uminus-Neg)
  next
  show  $U_{er}' = \text{finsert } (\text{eres } D \ E) \ U_{er}$ 
    using step-hyps by argo
  next
  show  $\text{eres } D \ E \neq \{\#\}$ 
    using step-hyps by argo
  next
  show  $\text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\}$ 
    ..

```

```

next
  show ord-res.is-maximal-lit K (eres D E)
    using step-hyps by argo
next
  show is-pos K
    using step-hyps by argo
qed

hence ord-res-6-step++ S6 S6'
  using S6-def ⟨C = Some E⟩ S6'-def ord-res-6-step.intros by blast

moreover have ord-res-6-matches-ord-res-7 i' S6' S7'
  unfolding S6'-def ⟨S7' = (N, s7')⟩ ⟨s7' = (Uer', F, Γ', Some (eres D E))⟩
  proof (rule ord-res-6-matches-ord-res-7.intros)
    show ord-res-5-invars N (Uer', F, restrict-map M {A. A <t atm-of K},
Some (eres D E))
      using invars-6 unfolding ⟨C = Some E⟩
      using ord-res-6-preserves-invars[OF step6] by argo
    next
      show ord-res-7-invars N (Uer', F, Γ', Some (eres D E))
        using invars-s7' unfolding ⟨s7' = (Uer', F, Γ', Some (eres D E))⟩ .
    next
      show ∀ A C. (restrict-map M {A. A <t atm-of K} A = Some C) =
        (map-of Γ' (Pos A) = Some (Some C))
      proof (intro allI)
        fix A :: 'f gterm and C :: 'f gclause
        show restrict-map M {A. A <t atm-of K} A = Some C ↔ map-of Γ'
(Pos A) = Some (Some C)
          proof (cases A ∈ dom M ∧ A <t atm-of K)
            case True
              have restrict-map M {A. A <t atm-of K} A = Some C ↔ M A =
Some C
                using True by simp
          also have ... ↔ map-of Γ (Pos A) = Some (Some C)
            using match-hyps(3-) by metis
          also have ... ↔ map-of Γ' (Pos A) = Some (Some C)
          proof -
            have Pos A ∈ fst ' set Γ
              using True
              by (metis domIff map-of-eq-None-iff match-hyps(5) not-None-eq)
            hence ∃C. (Pos A, C) ∈ set Γ
              by fastforce
          hence ∃C. (Pos A, C) ∈ set Γ ∧ (Pos A, C) ∈ set Γ'
            using True unfolding mem-set-Γ'-iff prod.sel literal.sel by metis

```

```

moreover have distinct (map fst  $\Gamma'$ )
  using  $\Gamma$ -distinct-atoms
proof (rule distinct-suffix)
  show suffix (map fst  $\Gamma'$ ) (map fst  $\Gamma$ )
    using map-mono-suffix step-hyps(9) suffix-dropWhile by blast
qed

ultimately have map-of  $\Gamma$  (Pos A) = map-of  $\Gamma'$  (Pos A)
  using  $\Gamma$ -distinct-atoms by (auto dest: map-of-is-SomeI)

thus ?thesis
  by argo
qed

finally show ?thesis .
next
case False
have restrict-map  $\mathcal{M}$  {A. A  $\prec_t$  atm-of K} A = None
  using False unfolding restrict-map-def by auto

moreover have map-of  $\Gamma'$  (Pos A)  $\neq$  Some (Some C)
  using False unfolding de-Morgan-conj
proof (elim disjE)
  assume A  $\notin$  dom  $\mathcal{M}$ 

  hence  $\bigwedge C. (Pos A, C) \notin$  set  $\Gamma$ 
    using match-hyps(5)
    by (metis (no-types, opaque-lifting) domIff fst-eqD invars-7 is-pos-def
map-of-SomeD
not-None-eq snd-conv weak-map-of-SomeI)

  hence  $\bigwedge C. (Pos A, C) \notin$  set  $\Gamma'$ 
    unfolding mem-set- $\Gamma'$ -iff by simp

  then show map-of  $\Gamma'$  (Pos A)  $\neq$  Some (Some C)
    by (meson map-of-SomeD)
next
assume  $\neg$  A  $\prec_t$  atm-of K

  hence  $\bigwedge C. (Pos A, C) \notin$  set  $\Gamma'$ 
    unfolding mem-set- $\Gamma'$ -iff by simp

  then show map-of  $\Gamma'$  (Pos A)  $\neq$  Some (Some C)
    by (meson map-of-SomeD)
qed

ultimately show ?thesis
  by simp
qed

```

```

qed
next
show  $\forall A. (\text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} A = \text{None}) =$ 
   $(\text{map-of } \Gamma' (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma')$ 
proof (intro allI)
  fix A :: 'f gterm
  show  $(\text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} A = \text{None}) =$ 
     $(\text{map-of } \Gamma' (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma')$ 
  proof (cases A  $\prec_t$  atm-of K)
    case True

      have  $\text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} A = \text{None} \longleftrightarrow \mathcal{M} A = \text{None}$ 
        using True by simp

      also have  $\dots \longleftrightarrow \text{map-of } \Gamma (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma$ 
        using match-hyps(6) by metis

      also have  $\dots \longleftrightarrow \text{map-of } \Gamma' (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma$ 
        using True mem-set- $\Gamma'$ -iff
      by (metis eq-fst-iff literal.sel(2) map-of-SomeD not-None-eq weak-map-of-SomeI)

      also have  $\dots \longleftrightarrow \text{map-of } \Gamma' (\text{Neg } A) \neq \text{None} \vee A \notin \text{trail-atms } \Gamma'$ 
        using True mem-set- $\Gamma'$ -iff
        by (smt (verit, best) fset-trail-atms image-iff)

      finally show ?thesis .
    next
      case False

      have  $\text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} A = \text{None}$ 
        using False by simp

      moreover have  $A \notin \text{trail-atms } \Gamma'$ 
        using False mem-set- $\Gamma'$ -iff
        by (smt (verit, ccfv-threshold) fset-trail-atms image-iff)

      ultimately show ?thesis
        by metis
    qed
  qed
next
show  $i' = \text{atms-of-cls } (N \cup U_{er}') \mid\text{-} \text{trail-atms } \Gamma'$ 
  using  $i'$ -def .
qed

ultimately show ?thesis
  by metis
next
case step-hyps: (resolution-neg E L D  $U_{er}' \Gamma' K C$ )

```

**define**  $S6'$  **where**  
 $S6' = (N, U_{er}', \mathcal{F}, \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\}, \text{Some } C)$

**define**  $i'$  **where**  
 $i' = \text{atms-of-cls } (N \mid \cup \mid U_{er}') \mid - \mid \text{trail-atms } \Gamma'$

**have**  $\text{mem-set-}\Gamma'\text{-iff}$ :  $\bigwedge x. (x \in \text{set } \Gamma') = (\text{atm-of } (\text{fst } x) \prec_t \text{atm-of } K \wedge x \in \text{set } \Gamma)$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma \rangle$   
**unfolding**  $\text{mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone}[OF \Gamma\text{-sorted mono-atms-lt}]$   
**by** *auto*

**have**  $\text{step6}$ :  $\text{ord-res-6 } N (U_{er}, \mathcal{F}, \mathcal{M}, \text{Some } E) (U_{er}', \mathcal{F}, \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\}, \text{Some } C)$   
**proof** ( $\text{rule ord-res-6.resolution-neg}$ )  
**show**  $\neg \text{dom } \mathcal{M} \Vdash E$   
**using**  $\text{clause-false-wrt-model-if-false-wrt-}\Gamma[OF \langle \text{trail-false-cls } \Gamma E \rangle]$ .  
**next**  
**show**  $\text{ord-res.is-maximal-lit } L E$   
**using**  $\text{step-hyps}$  **by** *argo*  
**next**  
**show**  $\text{is-neg } L$   
**using**  $\text{step-hyps}$  **by** *argo*  
**next**  
**show**  $\mathcal{M} (\text{atm-of } L) = \text{Some } D$   
**by** ( $\text{metis literal.collapse}(2)$   $\text{match-hyps}(5)$   $\text{step-hyps}(5)$   $\text{step-hyps}(6)$   $\text{uminus-Neg}$ )  
**next**  
**show**  $U_{er}' = \text{finsert } (\text{eres } D E) U_{er}$   
**using**  $\text{step-hyps}$  **by** *argo*  
**next**  
**show**  $\text{eres } D E \neq \{\#\}$   
**using**  $\text{step-hyps}$  **by** *argo*  
**next**  
**show**  $\text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\} = \text{restrict-map } \mathcal{M} \{A. A \prec_t \text{atm-of } K\}$   
**next**  
**show**  $\text{ord-res.is-maximal-lit } K (\text{eres } D E)$   
**using**  $\text{step-hyps}$  **by** *argo*  
**next**  
**show**  $\text{is-neg } K$   
**using**  $\text{step-hyps}$  **by** *argo*  
**next**  
**show**  $\mathcal{M} (\text{atm-of } K) = \text{Some } C$   
**using**  $\text{match-hyps}(3-)$   
**by** ( $\text{metis } (\text{mono-tags, lifting}) \text{step-hyps}(11) \text{step-hyps}(12) \text{uminus-literal-def}$ )



qed

hence  $ord\text{-}res\text{-}6\text{-}step^{++} S6 S6'$   
**using**  $S6\text{-}def \langle C = Some E \rangle S6'\text{-}def ord\text{-}res\text{-}6\text{-}step.intros$  **by** *blast*

**moreover have**  $ord\text{-}res\text{-}6\text{-}matches\text{-}ord\text{-}res\text{-}7 i' S6' S7'$   
**unfolding**  $S6'\text{-}def \langle S7' = (N, s7') \rangle \langle s7' = (U_{er}', \mathcal{F}, \Gamma', Some C) \rangle$   
**proof** (*rule ord-res-6-matches-ord-res-7.intros*)  
**show**  $ord\text{-}res\text{-}5\text{-}invars N (U_{er}', \mathcal{F}, restrict\text{-}map \mathcal{M} \{A. A \prec_t atm\text{-}of K\},$   
*Some C*)  
**using** *invars-6 unfolding*  $\langle C = Some E \rangle$   
**using** *ord-res-6-preserves-invars[OF step6]* **by** *argo*

**next**  
**show**  $ord\text{-}res\text{-}7\text{-}invars N (U_{er}', \mathcal{F}, \Gamma', Some C)$   
**using** *invars-s7' unfolding*  $\langle s7' = (U_{er}', \mathcal{F}, \Gamma', Some C) \rangle$  .

**next**  
**show**  $\forall A C. (restrict\text{-}map \mathcal{M} \{A. A \prec_t atm\text{-}of K\} A = Some C) =$   
 $(map\text{-}of \Gamma' (Pos A) = Some (Some C))$   
**proof** (*intro allI*)  
**fix**  $A :: 'f gterm$  **and**  $C :: 'f gclause$   
**show**  $restrict\text{-}map \mathcal{M} \{A. A \prec_t atm\text{-}of K\} A = Some C \longleftrightarrow map\text{-}of \Gamma'$   
 $(Pos A) = Some (Some C)$   
**proof** (*cases*  $A \in dom \mathcal{M} \wedge A \prec_t atm\text{-}of K$ )  
**case** *True*  
**have**  $restrict\text{-}map \mathcal{M} \{A. A \prec_t atm\text{-}of K\} A = Some C \longleftrightarrow \mathcal{M} A =$   
*Some C*  
**using** *True by simp*

**also have**  $\dots \longleftrightarrow map\text{-}of \Gamma (Pos A) = Some (Some C)$   
**using** *match-hyps(3-)* **by** *metis*

**also have**  $\dots \longleftrightarrow map\text{-}of \Gamma' (Pos A) = Some (Some C)$   
**proof** –  
**have**  $Pos A \in fst \text{ ` } set \Gamma$   
**using** *True*  
**by** (*metis domIff map-of-eq-None-iff match-hyps(5) not-None-eq*)

**hence**  $\exists C. (Pos A, C) \in set \Gamma$   
**by** *fastforce*

**hence**  $\exists C. (Pos A, C) \in set \Gamma \wedge (Pos A, C) \in set \Gamma'$   
**using** *True unfolding mem-set-Γ'-iff prod.sel literal.sel* **by** *metis*

**moreover have**  $distinct (map fst \Gamma')$   
**using** *Γ-distinct-atoms*  
**proof** (*rule distinct-suffix*)  
**show**  $suffix (map fst \Gamma') (map fst \Gamma)$   
**using** *map-mono-suffix step-hyps(9) suffix-dropWhile* **by** *blast*

qed

```

ultimately have map-of  $\Gamma$  (Pos A) = map-of  $\Gamma'$  (Pos A)
  using  $\Gamma$ -distinct-atoms by (auto dest: map-of-is-SomeI)

thus ?thesis
  by argo
qed

finally show ?thesis .
next
case False
have restrict-map  $\mathcal{M}$  {A. A  $\prec_t$  atm-of K} A = None
  using False unfolding restrict-map-def by auto

moreover have map-of  $\Gamma'$  (Pos A)  $\neq$  Some (Some C)
  using False unfolding de-Morgan-conj
proof (elim disjE)
  assume A  $\notin$  dom  $\mathcal{M}$ 

  hence  $\bigwedge C. (Pos A, C) \notin set \Gamma$ 
    using match-hyps(5)
    by (metis (no-types, opaque-lifting) domIff fst-eqD invars-7 is-pos-def
map-of-SomeD
not-None-eq snd-conv weak-map-of-SomeI)

  hence  $\bigwedge C. (Pos A, C) \notin set \Gamma'$ 
    unfolding mem-set- $\Gamma'$ -iff by simp

  then show map-of  $\Gamma'$  (Pos A)  $\neq$  Some (Some C)
    by (meson map-of-SomeD)
  next
  assume  $\neg A \prec_t atm-of K$ 

  hence  $\bigwedge C. (Pos A, C) \notin set \Gamma'$ 
    unfolding mem-set- $\Gamma'$ -iff by simp

  then show map-of  $\Gamma'$  (Pos A)  $\neq$  Some (Some C)
    by (meson map-of-SomeD)
  qed

ultimately show ?thesis
  by simp
qed
qed
next
show  $\forall A. (restrict-map \mathcal{M} \{A. A \prec_t atm-of K\} A = None) =$ 
  (map-of  $\Gamma'$  (Neg A)  $\neq$  None  $\vee A \notin trail-atms \Gamma'$ )
proof (intro allI)
  fix A :: 'f gterm

```

```

show (restrict-map  $\mathcal{M}$   $\{A. A \prec_t \text{atm-of } K\}$   $A = \text{None}$ ) =
  (map-of  $\Gamma'$  ( $\text{Neg } A \neq \text{None} \vee A \notin \text{trail-atms } \Gamma'$ )
proof (cases  $A \prec_t \text{atm-of } K$ )
  case True

  have restrict-map  $\mathcal{M}$   $\{A. A \prec_t \text{atm-of } K\}$   $A = \text{None}$   $\longleftrightarrow$   $\mathcal{M} A = \text{None}$ 
    using True by simp

  also have  $\dots \longleftrightarrow$  map-of  $\Gamma$  ( $\text{Neg } A \neq \text{None} \vee A \notin \text{trail-atms } \Gamma$ )
    using match-hyps(6) by metis

  also have  $\dots \longleftrightarrow$  map-of  $\Gamma'$  ( $\text{Neg } A \neq \text{None} \vee A \notin \text{trail-atms } \Gamma$ )
    using True mem-set- $\Gamma'$ -iff
by (metis eq-fst-iff literal.sel(2) map-of-SomeD not-None-eq weak-map-of-SomeI)

  also have  $\dots \longleftrightarrow$  map-of  $\Gamma'$  ( $\text{Neg } A \neq \text{None} \vee A \notin \text{trail-atms } \Gamma'$ )
    using True mem-set- $\Gamma'$ -iff
    by (smt (verit, best) fset-trail-atms image-iff)

  finally show ?thesis .
next
  case False

  have restrict-map  $\mathcal{M}$   $\{A. A \prec_t \text{atm-of } K\}$   $A = \text{None}$ 
    using False by simp

  moreover have  $A \notin \text{trail-atms } \Gamma'$ 
    using False mem-set- $\Gamma'$ -iff
    by (smt (verit, ccfv-threshold) fset-trail-atms image-iff)

  ultimately show ?thesis
    by metis
  qed
qed
next
  show  $i' = \text{atms-of-clss } (N \cup U_{er}) \mid - \mid \text{trail-atms } \Gamma'$ 
    using i'-def .
  qed

  ultimately show ?thesis
    by metis
  qed
qed

theorem bisimulation-ord-res-6-ord-res-7:
  defines match  $\equiv$  ord-res-6-matches-ord-res-7
  shows  $\exists (\text{MATCH} :: \text{nat} \times \text{nat} \Rightarrow 'f \text{ord-res-6-state} \Rightarrow 'f \text{ord-res-7-state} \Rightarrow \text{bool})$ 
 $\mathcal{R}_f \mathcal{R}_b.$ 
  bisimulation ord-res-6-step ord-res-6-final (constant-context ord-res-7) ord-res-7-final

```

```

    MATCH  $\mathcal{R}_f \mathcal{R}_b$ 
  proof (rule ex-bisimulation-from-backward-simulation)
    show right-unique ord-res-6-step
      using right-unique-ord-res-6-step .
  next
    show right-unique (constant-context ord-res-7)
      using right-unique-constant-context right-unique-ord-res-7 by metis
  next
    show  $\forall S. \text{ord-res-6-final } S \longrightarrow (\exists S'. \text{ord-res-6-step } S S')$ 
      by (metis finished-def ord-res-6-semantics.final-finished)
  next
    show  $\forall S. \text{ord-res-7-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-7 } S S')$ 
      by (metis finished-def ord-res-7-semantics.final-finished)
  next
    show  $\forall i S6 S7. \text{match } i S6 S7 \longrightarrow \text{ord-res-6-final } S6 \longleftrightarrow \text{ord-res-7-final } S7$ 
      unfolding match-def
      using ord-res-6-final-iff-ord-res-7-final by metis
  next
    show  $\forall i S6 S7. \text{match } i S6 S7 \longrightarrow$ 
      safe-state ord-res-6-step ord-res-6-final  $S6 \wedge$ 
      safe-state (constant-context ord-res-7) ord-res-7-final  $S7$ 
    proof (intro allI impI conjI)
      fix  $i S6 S7$ 
      assume match  $i S6 S7$ 
      show safe-state ord-res-6-step ord-res-6-final  $S6$ 
        using  $\langle \text{match } i S6 S7 \rangle$ [unfolded match-def]
        using ord-res-6-safe-state-if-invars
        using ord-res-6-matches-ord-res-7.simps by auto
      show safe-state (constant-context ord-res-7) ord-res-7-final  $S7$ 
        using  $\langle \text{match } i S6 S7 \rangle$ [unfolded match-def]
        using ord-res-7-safe-state-if-invars
        using ord-res-6-matches-ord-res-7.simps by auto
    qed
  next
    show wfp ( $|\subset|$ )
      using wfp-pfssubset .
  next
    show  $\forall i S6 S7 S7'. \text{match } i S6 S7 \longrightarrow \text{constant-context ord-res-7 } S7 S7' \longrightarrow$ 
       $(\exists i' S6'. \text{ord-res-6-step}^{++} S6 S6' \wedge \text{match } i' S6' S7') \vee$ 
       $(\exists i'. \text{match } i' S6 S7' \wedge i' |\subset| i)$ 
      unfolding match-def
      using backward-simulation-between-6-and-7 by metis
  qed
end

```

## 34 ORD-RES-8 (atom-guided literal trail construction)

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-8-can-decide-neg* **where**

- $\neg$  *trail-false-cls*  $\Gamma C \implies$
- linorder-lit.is-maximal-in-mset*  $C L \implies$
- linorder-trm.is-least-in-fset*  $\{|A| \in| \text{atms-of-clss } (N \cup| U_{er}).$
- $A \prec_t \text{atm-of } L \wedge A \notin| \text{trail-atms } \Gamma\} A \implies$
- ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C$

**inductive** *ord-res-8-can-skip-undefined-neg* **where**

- $\neg$  *trail-false-cls*  $\Gamma C \implies$
- linorder-lit.is-maximal-in-mset*  $C L \implies$
- $\neg(\exists A \in| \text{atms-of-clss } (N \cup| U_{er}). A \prec_t \text{atm-of } L \wedge A \notin| \text{trail-atms } \Gamma) \implies$
- $\neg$  *trail-defined-lit*  $\Gamma L \implies$
- is-neg*  $L \implies$
- ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma C$

**inductive** *ord-res-8-can-skip-undefined-pos-ultimate* **where**

- $\neg$  *trail-false-cls*  $\Gamma C \implies$
- linorder-lit.is-maximal-in-mset*  $C L \implies$
- $\neg(\exists A \in| \text{atms-of-clss } (N \cup| U_{er}). A \prec_t \text{atm-of } L \wedge A \notin| \text{trail-atms } \Gamma) \implies$
- $\neg$  *trail-defined-lit*  $\Gamma L \implies$
- is-pos*  $L \implies$
- $\neg$  *trail-false-cls*  $\Gamma \{\#K \in\# C. K \neq L\# \} \implies$
- $\neg(\exists D \in| \text{iefac } \mathcal{F} \mid^i (N \cup| U_{er}). C \prec_c D) \implies$
- ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma C$

**inductive** *ord-res-8-can-produce* **where**

- $\neg$  *trail-false-cls*  $\Gamma C \implies$
- linorder-lit.is-maximal-in-mset*  $C L \implies$
- $\neg(\exists A \in| \text{atms-of-clss } (N \cup| U_{er}). A \prec_t \text{atm-of } L \wedge A \notin| \text{trail-atms } \Gamma) \implies$
- $\neg$  *trail-defined-lit*  $\Gamma L \implies$
- is-pos*  $L \implies$
- trail-false-cls*  $\Gamma \{\#K \in\# C. K \neq L\# \} \implies$
- linorder-lit.is-greatest-in-mset*  $C L \implies$
- ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C$

**inductive** *ord-res-8-can-factorize* **where**

- $\neg$  *trail-false-cls*  $\Gamma C \implies$
- linorder-lit.is-maximal-in-mset*  $C L \implies$
- $\neg(\exists A \in| \text{atms-of-clss } (N \cup| U_{er}). A \prec_t \text{atm-of } L \wedge A \notin| \text{trail-atms } \Gamma) \implies$
- $\neg$  *trail-defined-lit*  $\Gamma L \implies$
- is-pos*  $L \implies$
- trail-false-cls*  $\Gamma \{\#K \in\# C. K \neq L\# \} \implies$
- linorder-lit.is-greatest-in-mset*  $C L \implies$
- ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma C$

**definition** *is-least-nonskipped-clause* **where**

*is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma C \longleftrightarrow$   
*linorder-cls.is-least-in-fset*  $\{|C| \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})\}.$   
*trail-false-cl*  $\Gamma C \vee$   
*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma C\} C$

**lemma** *is-least-nonskipped-clause-mempty*:

**assumes** *bot-in*:  $\{\#\} \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$   
**shows** *is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma \{\#\}$   
**unfolding** *is-least-nonskipped-clause-def linorder-cls.is-least-in-filter-iff*

**proof** (*intro conjI ballI impI*)

**show**  $\{\#\} \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$   
**using** *bot-in* .

**next**

**show** *trail-false-cl*  $\Gamma \{\#\} \vee$   
*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma \{\#\} \vee$   
*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma \{\#\} \vee$   
*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma \{\#\} \vee$   
*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma \{\#\} \vee$  *ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma$   
 $\{\#\}$   
**by** *simp*

**next**

**fix**  $C :: 'f \text{ gclause}$   
**assume**  $C \neq \{\#\}$   
**thus**  $\{\#\} \prec_c C$   
**using** *mempty-lesseq-cl* **by** *blast*

**qed**

**lemma** *nex-is-least-nonskipped-clause-if*:

**assumes**  
*no-undef-atom*:  $\neg (\exists A \in |atms-of-clss (N \cup U_{er}). A \notin |trail-atms \Gamma)$  **and**  
*no-false-clause*:  $\neg fBex (iefac \mathcal{F} |^\dagger (N \cup U_{er})) (trail-false-cl \Gamma)$   
**shows**  $\nexists C. is-least-nonskipped-clause N U_{er} \mathcal{F} \Gamma C$

**unfolding** *not-ex*

**proof** (*intro allI notI*)

**fix**  $C :: 'f \text{ gclause}$   
**assume** *is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma C$

**hence** *C-in*:  $C \in |iefac \mathcal{F} |^\dagger (N \cup U_{er})$  **and**

*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma C$

**unfolding** *atomize-conj*  
**unfolding** *is-least-nonskipped-clause-def*  
**unfolding** *linorder-cls.is-least-in-filter-iff*  
**using** *no-false-clause* **by** *metis*

**hence**  $\exists L. \text{linorder-lit.is-maximal-in-mset } C L \wedge \neg \text{trail-defined-lit } \Gamma L$   
**proof** (*elim disjE*)

**assume** *ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C$

**then show** *?thesis*

**by** (*metis (mono-tags, lifting) assms(1) linorder-trm.is-least-in-filter-iff*  
*ord-res-8-can-decide-neg.cases*)

**qed** (*auto elim:*

*ord-res-8-can-skip-undefined-neg.cases*  
*ord-res-8-can-skip-undefined-pos-ultimate.cases*  
*ord-res-8-can-produce.cases*  
*ord-res-8-can-factorize.cases*)

**hence**  $\exists L. \text{atm-of } L \in | \text{atms-of-clss } (N \cup U_{er}) \wedge \text{atm-of } L \notin | \text{trail-atms } \Gamma$   
**using** *C-in*

**unfolding** *linorder-lit.is-maximal-in-mset-iff trail-defined-lit-iff-trail-defined-atm*  
**by** (*metis atm-of-in-atms-of-clssI*)

**thus** *False*

**using** *no-undef-atom* **by** *metis*

**qed**

**lemma** *MAGIC5*:

**assumes** *invars: ord-res-7-invars*  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$  **and**

*no-more-steps:  $\nexists C'. \text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}')$*

**shows**  $(\forall C. C = \text{Some } C \longleftrightarrow \text{is-least-nonskipped-clause } N U_{er} \mathcal{F} \Gamma C)$

**proof** (*cases C*)

**case** *None*

**have** *trail-atms*  $\Gamma = \text{atms-of-clss } (N \cup U_{er})$

**using** *None invars[unfolded ord-res-7-invars-def]* **by** *simp*

**have**  $\forall C \in | \text{iefac } \mathcal{F} | \wedge (N \cup U_{er}). \text{trail-true-cls } \Gamma C$

**using** *None invars[unfolded ord-res-7-invars-def]* **by** *simp*

**hence** *no-false*:  $\forall C \in | \text{iefac } \mathcal{F} | \wedge (N \cup U_{er}). \neg \text{trail-false-cls } \Gamma C$

**using** *invars[unfolded ord-res-7-invars-def]*

**by** (*meson invars not-trail-true-cls-and-trail-false-cls*  
*ord-res-7-invars-implies-trail-consistent*)

**have**  $\nexists C. \text{is-least-nonskipped-clause } N U_{er} \mathcal{F} \Gamma C$

**proof** (*rule notI, elim exE*)

**fix** *C*

**assume** *is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma C$

hence

*C-in*:  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**

*C-spec-disj*:

*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$

*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$

*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma C \vee$

*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C \vee$

*ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma C$

**unfolding** *atomize-conj*

**unfolding** *is-least-nonskipped-clause-def*

**unfolding** *linorder-cls.is-least-in-ffilter-iff*

**using** *no-false* **by** *metis*

**thus** *False*

**proof** (*elim disjE*)

**assume** *ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C$

**thus** *?thesis*

**using**  $\langle \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$

**using** *ord-res-8-can-decide-neg.cases*

**by** (*metis (no-types, lifting) linorder-trm.is-least-in-ffilter-iff*)

**next**

**assume** *ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma C$

**thus** *?thesis*

**using**  $\langle \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$

**using** *ord-res-8-can-skip-undefined-neg.cases*

**by** (*metis C-in atm-of-in-atms-of-clssI linorder-lit.is-maximal-in-mset-iff trail-defined-lit-iff-trail-defined-atm*)

**next**

**assume** *ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma C$

**thus** *?thesis*

**using**  $\langle \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$

**using** *ord-res-8-can-skip-undefined-pos-ultimate.cases*

**by** (*metis C-in atm-of-in-atms-of-clssI linorder-lit.is-maximal-in-mset-iff trail-defined-lit-iff-trail-defined-atm*)

**next**

**assume** *ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C$

**thus** *False*

**using**  $\langle \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$

**using** *ord-res-8-can-produce.cases*

**by** (*metis C-in atm-of-in-atms-of-clssI linorder-lit.is-maximal-in-mset-iff trail-defined-lit-iff-trail-defined-atm*)

**next**

**assume** *ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma C$

**thus** *False*

**using**  $\langle \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er}) \rangle$

**using** *ord-res-8-can-factorize.cases*

**by** (*metis C-in atm-of-in-atms-of-clssI linorder-lit.is-maximal-in-mset-iff trail-defined-lit-iff-trail-defined-atm*)

**qed**



qed

thus ?thesis  
 using None by simp

next  
 case (Some D)

have  $D\text{-in}: D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
 using Some invars[unfolded ord-res-7-invars-def] by simp

have  $\text{is-least-nonskipped-clause } N U_{er} \mathcal{F} \Gamma D$   
 proof (cases  $D = \{\#\}$ )  
 case True  
 thus ?thesis  
 using D-in is-least-nonskipped-clause-empty by metis

next  
 case False

then obtain  $L_D$  where  $D\text{-max-lit}: \text{linorder-lit.is-maximal-in-mset } D L_D$   
 using linorder-lit.ex-maximal-in-mset by presburger

show ?thesis  
 unfolding is-least-nonskipped-clause-def  
 unfolding linorder-cls.is-least-in-filter-iff  
 proof (intro conjI ballI impI)  
 show  $D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$   
 using D-in .

next  
 show  $\text{trail-false-cls } \Gamma D \vee$   
 $\text{ord-res-8-can-decide-neg } N U_{er} \mathcal{F} \Gamma D \vee$   
 $\text{ord-res-8-can-skip-undefined-neg } N U_{er} \mathcal{F} \Gamma D \vee$   
 $\text{ord-res-8-can-skip-undefined-pos-ultimate } N U_{er} \mathcal{F} \Gamma D \vee$   
 $\text{ord-res-8-can-produce } N U_{er} \mathcal{F} \Gamma D \vee \text{ord-res-8-can-factorize } N U_{er} \mathcal{F} \Gamma D$   
 proof (cases  $\text{trail-false-cls } \Gamma D$ )  
 case True  
 then show ?thesis  
 by metis

next  
 case  $D\text{-not-false}: \text{False}$

then obtain  $L$  where  $D\text{-max-lit}: \text{linorder-lit.is-maximal-in-mset } D L$   
 by (metis linorder-lit.ex-maximal-in-mset trail-false-cls-empty)

show ?thesis  
 proof (cases  $\exists A \in \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \prec_t \text{atm-of } L \wedge A \notin \text{trail-atms}$   
 $\Gamma$ )  
 case True  
 hence  $\text{ord-res-8-can-decide-neg } N U_{er} \mathcal{F} \Gamma D$

```

using ord-res-8-can-decide-neg.intros[OF D-not-false D-max-lit]
by (metis (no-types, lifting) equalsffemptyD fmember-filter linorder-trm.ex-least-in-fset)

thus ?thesis
by argo
next
case no-undef-atm: False
show ?thesis
proof (cases trail-defined-lit  $\Gamma$   $L$ )
  case L-defined: True

  hence  $\exists C'. \text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, C) (U_{er}, \mathcal{F}, \Gamma, C')$ 
    unfolding  $\langle C = \text{Some } D \rangle$ 
    using ord-res-7.skip-defined[OF D-not-false D-max-lit no-undef-atm]
    by metis

  hence False
    using no-more-steps by contradiction

  thus ?thesis ..
next
case L-undef: False
show ?thesis
proof (cases L)
  case (Pos A)

  hence L-pos: is-pos L
    by simp

  show ?thesis
proof (cases trail-false-cls  $\Gamma$   $\{\#K \in\# D. K \neq L\# \}$ )
  case D-almost-false: True
    thus ?thesis
      using ord-res-8-can-factorize.intros[
        OF D-not-false D-max-lit no-undef-atm L-undef L-pos]
      using ord-res-8-can-produce.intros[
        OF D-not-false D-max-lit no-undef-atm L-undef L-pos]
      by metis
    next
case D-not-flagrantly-false: False
show ?thesis
proof (cases  $\exists E \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). D \prec_c E$ )
  case True

  hence  $\exists C'. \text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, C) (U_{er}, \mathcal{F}, \Gamma, C')$ 
    unfolding  $\langle C = \text{Some } D \rangle$ 
    using ord-res-7.skip-undefined-pos[
      OF D-not-false D-max-lit no-undef-atm L-undef L-pos
      D-not-flagrantly-false]

```

```

    by (metis femptyE fmember-filter linorder-cls.ex-least-in-fset)

  hence False
    using no-more-steps by contradiction

  thus ?thesis ..
next
  case False

  hence ord-res-8-can-skip-undefined-pos-ultimate  $N U_{er} \mathcal{F} \Gamma D$ 
    using ord-res-8-can-skip-undefined-pos-ultimate.intros[
      OF D-not-false D-max-lit no-undef-atm L-undef L-pos
    ]
  D-not-flagrantly-false]
    by metis

  thus ?thesis
    by argo
  qed
next
  case (Neg A)
  hence L-neg: is-neg L
    by simp

  hence ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma D$ 
    unfolding ⟨C = Some D⟩
    using ord-res-8-can-skip-undefined-neg.intros[
      OF D-not-false D-max-lit no-undef-atm L-undef]
    by metis

  thus ?thesis
    by argo
  qed
qed
qed
qed

fix E :: 'f gterm literal multiset
assume
  E-in: E |∈| iefac  $\mathcal{F}$  |' (N |∪|  $U_{er}$ ) and
  E-neg: E ≠ D and
  E-spec: trail-false-cls  $\Gamma E \vee$ 
  ord-res-8-can-decide-neg  $N U_{er} \mathcal{F} \Gamma E \vee$ 
  ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma E \vee$ 
  ord-res-8-can-skip-undefined-pos-ultimate  $N U_{er} \mathcal{F} \Gamma E \vee$ 
  ord-res-8-can-produce  $N U_{er} \mathcal{F} \Gamma E \vee$ 
  ord-res-8-can-factorize  $N U_{er} \mathcal{F} \Gamma E$ 

have true-cls-if-lt-D:

```

$\forall C \in |iefac \mathcal{F} |^{\dagger} (N \cup U_{er}). C \prec_c D \longrightarrow \text{trail-true-cls } \Gamma C$   
**using** *invars[unfolded ord-res-7-invars-def] Some by simp*

**have**  $\Gamma$ -lower-set: *linorder-trm.is-lower-fset (trail-atms  $\Gamma$ ) (atms-of-cls (N  $\cup$   $U_{er}$ ))*  
**using** *invars[unfolded ord-res-7-invars-def] by simp*

**have** *FOO*:  
 $\forall C \in |iefac \mathcal{F} |^{\dagger} (N \cup U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{ord-res.is-maximal-lit } K C \longrightarrow$   
 $\neg (\exists A \in |atms-of-cls (N \cup U_{er}). A \prec_t \text{atm-of } K \wedge A \notin \text{trail-atms } \Gamma))$   
**using** *invars[unfolded ord-res-7-invars-def] Some E-in by simp*

**hence** *BAR*:  
 $\forall C \in |iefac \mathcal{F} |^{\dagger} (N \cup U_{er}). C \prec_c D \longrightarrow$   
 $(\forall K. \text{linorder-lit.is-maximal-in-mset } C K \longrightarrow$   
 $\neg \text{trail-defined-lit } \Gamma K \longrightarrow (\text{trail-true-cls } \Gamma \{ \#x \in \# C. x \neq K \# \} \wedge \text{is-pos}$   
 $K))$   
**using** *invars[unfolded ord-res-7-invars-def] Some by simp*

**show**  $D \prec_c E$   
**using** *E-spec*  
**proof** (*elim disjE*)  
**assume** *trail-false-cls  $\Gamma E$*   
**hence**  $\neg \text{trail-true-cls } \Gamma E$   
**using** *invars not-trail-true-cls-and-trail-false-cls*  
*ord-res-7-invars-implies-trail-consistent by blast*  
**thus**  $D \prec_c E$   
**using** *E-in E-neq true-cls-if-lt-D by force*

**next**  
**assume** *ord-res-8-can-decide-neg N  $U_{er}$   $\mathcal{F}$   $\Gamma E$*   
**thus**  $D \prec_c E$   
**proof** (*cases N  $U_{er}$   $\mathcal{F}$   $\Gamma E$  rule: ord-res-8-can-decide-neg.cases*)  
**case** (*1  $L_E A$* )  
**hence**  $\exists A \in |atms-of-cls (N \cup U_{er}). A \prec_t \text{atm-of } L_E \wedge A \notin \text{trail-atms}$   
 $\Gamma$   
**unfolding** *linorder-trm.is-least-in-ffilter-iff by metis*  
**thus** *?thesis*  
**using** *FOO[rule-format, OF E-in -  $\langle$ ord-res.is-maximal-lit  $L_E E$  $\rangle$ ] E-in*  
*E-neq*  
**by force**  
**qed**

**next**  
**assume** *ord-res-8-can-skip-undefined-neg N  $U_{er}$   $\mathcal{F}$   $\Gamma E$*   
**thus**  $D \prec_c E$   
**proof** (*cases N  $U_{er}$   $\mathcal{F}$   $\Gamma E$  rule: ord-res-8-can-skip-undefined-neg.cases*)  
**case** *hyps: (1  $L_E$ )*  
**thus** *?thesis*  
**using** *BAR[rule-format, OF E-in -  $\langle$ ord-res.is-maximal-lit  $L_E E$  $\rangle$ ]*

```

      using invars[unfolded ord-res-7-invars-def Some, rule-format, OF refl]
Some E-in
  using E-neq by fastforce
  qed
next
  assume ord-res-8-can-skip-undefined-pos-ultimate N Uer F Γ E
  thus D <c E
proof (cases N Uer F Γ E rule: ord-res-8-can-skip-undefined-pos-ultimate.cases)
  case (1 L)
  then show ?thesis using E-neq D-in by force
  qed
next
  assume ord-res-8-can-produce N Uer F Γ E
  hence ¬ trail-true-cls Γ E
proof (cases N Uer F Γ E rule: ord-res-8-can-produce.cases)
  case (1 L)
  then show ?thesis
    using invars[THEN ord-res-7-invars-implies-trail-consistent]
  by (smt (verit, ccfv-SIG) mem-Collect-eq not-trail-true-cls-and-trail-false-cls
      set-mset-filter trail-defined-lit-iff-true-or-false trail-true-cls-def)
  qed
  thus D <c E
    using E-in E-neq true-cls-if-lt-D by force
next
  assume ord-res-8-can-factorize N Uer F Γ E
  hence ¬ trail-true-cls Γ E
proof (cases N Uer F Γ E rule: ord-res-8-can-factorize.cases)
  case (1 L)
  then show ?thesis
    using invars[THEN ord-res-7-invars-implies-trail-consistent]
  by (smt (verit, ccfv-SIG) mem-Collect-eq not-trail-true-cls-and-trail-false-cls
      set-mset-filter trail-defined-lit-iff-true-or-false trail-true-cls-def)
  qed
  thus D <c E
    using E-in E-neq true-cls-if-lt-D by force
  qed
qed
qed
qed

moreover have Uniq (is-least-nonskipped-clause N Uer F Γ)
  unfolding is-least-nonskipped-clause-def
  using linorder-cls.Uniq-is-least-in-fset
  by simp

ultimately show ?thesis
  using Some
  by (metis (no-types) The-optional-eq-SomeD The-optional-eq-SomeI)
qed

```

**lemma** *MAGIC6*:

**assumes** *invars*: *ord-res-7-invars*  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$

**shows**  $\exists \mathcal{C}'. (ord-res-7\ N)^{**} (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}') \wedge$   
 $(\nexists \mathcal{C}''. ord-res-7\ N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}') (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}''))$

**proof** –

**define**  $R$  **where**

$R = (\lambda \mathcal{C} \mathcal{C}'. ord-res-7\ N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}'))$

**define**  $f :: 'f\ gclause\ option \Rightarrow 'f\ gclause\ fset$  **where**

$f = (\lambda \mathcal{C}. case\ \mathcal{C}\ of\ None \Rightarrow \{\}\ \mid\ Some\ C \Rightarrow \{|D\ |\in\| \ iefac\ \mathcal{F}\ |\ |\ (N\ |\cup\| \ U_{er}).$   
 $C \preceq_c D\ |\})$

**let**  $?less-f = (|\subset|)$

**have** *Wellfounded.wfp-on*  $\{x'. R^{**} \mathcal{C} x'\} R^{-1-1}$

**proof** (*rule wfp-on-if-convertible-to-wfp-on*)

**have** *wfp*  $(|\subset|)$

**by** *auto*

**thus** *Wellfounded.wfp-on*  $(f \text{ ` } \{x'. R^{**} \mathcal{C} x'\}) ?less-f$

**using** *Wellfounded.wfp-on-subset subset-UNIV* **by** *metis*

**next**

**fix**  $\mathcal{C}_x \mathcal{C}_y :: 'f\ gclause\ option$

**have** *rtranclp-with-constsD*:  $(\lambda y\ y'. R (x, y) (x, y'))^{**} y\ y' \Longrightarrow$   
 $R^{**} (x, y) (x, y')$  **for**  $R\ x\ y\ y'$

**proof** (*induction y arbitrary: rule: converse-rtranclp-induct*)

**case** *base*

**show**  $?case$

**by** *simp*

**next**

**case** (*step w*)

**thus**  $?case$

**by** *force*

**qed**

**assume**  $\mathcal{C}_x \in \{x'. R^{**} \mathcal{C} x'\}$  **and**  $\mathcal{C}_y \in \{x'. R^{**} \mathcal{C} x'\}$

**hence**

$(ord-res-7\ N)^{**} (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}_x)$  **and**

$(ord-res-7\ N)^{**} (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}_y)$

**unfolding** *atomize-conj mem-Collect-eq R-def*

**by** (*auto intro: rtranclp-with-constsD*)

**hence**

*x-invars*: *ord-res-7-invars*  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}_x)$  **and**

*y-invars*: *ord-res-7-invars*  $N (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}_y)$

**using** *ord-res-7-preserves-invars*

**using** *invars* **by** (*metis rtranclp-ord-res-7-preserves-ord-res-7-invars*) $+$

**have**  $\Gamma$ -*consistent*: *trail-consistent*  $\Gamma$

**using** *x-invars* **by** (*metis ord-res-7-invars-implies-trail-consistent*)

**have** *less-f-if*:  $?less-f (f \mathcal{C}_y) (f \mathcal{C}_x)$   
**if**  $\mathcal{C}_x = \text{Some } C$  **and**  
 $\mathcal{C}_y\text{-disj}$ :  $\mathcal{C}_y = \text{None} \vee \mathcal{C}_y = \text{Some } D \wedge C \prec_c D$  **and**  
 $C\text{-in}$ :  $C \in \text{iefac } \mathcal{F} \uparrow (N \cup U_{er})$   
**for**  $C D$   
**proof** –  
**have**  $f\text{-x}$ :  $f \mathcal{C}_x = \{D \mid D \in \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}), C \preceq_c D\}$   
**by** (*auto simp add*:  $\langle \mathcal{C}_x = \text{Some } C \rangle f\text{-def}$ )  
  
**moreover have**  $C \in f \mathcal{C}_x$   
**using**  $C\text{-in } f\text{-x}$  **by** *simp*  
  
**moreover have**  $C \notin f \mathcal{C}_y \wedge f \mathcal{C}_y \subseteq f \mathcal{C}_x$   
**using**  $\mathcal{C}_y\text{-disj}$   
**proof** (*elim disjE conjE; intro conjI*)  
**assume**  $\mathcal{C}_y = \text{None}$   
**thus**  $C \notin f \mathcal{C}_y$  **and**  $f \mathcal{C}_y \subseteq f \mathcal{C}_x$   
**unfolding**  $f\text{-x}$   
**by** (*simp-all add*:  $f\text{-def}$ )  
**next**  
**assume**  $\mathcal{C}_y = \text{Some } D$  **and**  $C \prec_c D$   
  
**have**  $\bigwedge x. D \preceq_c x \implies C \preceq_c x$   
**using**  $\langle C \prec_c D \rangle$  **by** *auto*  
  
**moreover have**  $f\text{st-}f\text{-y}$ :  $f \mathcal{C}_y = \{E \mid E \in \text{iefac } \mathcal{F} \uparrow (N \cup U_{er}), D \preceq_c E\}$   
**by** (*auto simp add*:  $\langle \mathcal{C}_y = \text{Some } D \rangle f\text{-def}$ )  
  
**ultimately show**  $f \mathcal{C}_y \subseteq f \mathcal{C}_x$   
**using**  $f\text{-x}$  **by** *auto*  
  
**show**  $C \notin f \mathcal{C}_y$   
**using**  $\langle C \prec_c D \rangle f\text{st-}f\text{-y}$  **by** *auto*  
**qed**  
  
**ultimately have**  $f \mathcal{C}_y \subseteq f \mathcal{C}_x$   
**by** *blast*  
  
**thus** *?thesis*  
**by** (*simp add*:  $\text{lex-prodp-def}$ )  
**qed**  
  
**have** *eres-not-in-if*:  $eres D E \notin U_{er}$   
**if**  $\mathcal{C}_x = \text{Some } E$  **and**  $E\text{-false}$ :  $\text{trail-false-cls } \Gamma E$  **and**  
 $E\text{-max-lit}$ :  $\text{ord-res.is-maximal-lit } L E$  **and**  $L\text{-neg}$ :  $\text{is-neg } L$   
 $\text{map-of } \Gamma (- L) = \text{Some } (\text{Some } D)$   
**for**  $D E L$   
**proof** –

**have**  
*clauses-lt-E-true:*  
 $\forall C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). C \prec_c E \longrightarrow \text{trail-true-cls } \Gamma C$  **and**  
 *$\Gamma$ -prop-greatest:*  
 $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{linorder-lit.is-greatest-in-mset } C$   
*(fst Ln)*  
**using** *x-invars unfolding*  $\langle C_x = \text{Some } E \rangle$  *ord-res-7-invars-def* **by** *simp-all*

**have**  $(- L, \text{Some } D) \in \text{set } \Gamma$   
**using**  $\langle \text{map-of } \Gamma (- L) = \text{Some } (\text{Some } D) \rangle$  **by** *(metis map-of-SomeD)*

**hence** *D-greatest-lit: linorder-lit.is-greatest-in-mset D (- L)*  
**using**  *$\Gamma$ -prop-greatest* **by** *fastforce*

**have** *eres D E  $\prec_c$  E*  
**using** *eres-lt-if*  
**using** *E-max-lit L-neg D-greatest-lit*  
**by** *metis*

**hence** *eres D E  $\neq$  E*  
**by** *order*

**have**  $L \in \# E$   
**using** *E-max-lit unfolding linorder-lit.is-maximal-in-mset-iff* **by** *metis*

**hence**  $- L \notin \# E$   
**using** *not-both-lit-and-comp-lit-in-false-clause-if-trail-consistent*  
**using**  *$\Gamma$ -consistent E-false* **by** *metis*

**hence**  $\forall K \in \# \text{eres } D E. \text{atm-of } K \prec_t \text{atm-of } L$   
**using** *lit-in-eres-lt-greatest-lit-in-greatest-resolvent[OF  $\langle \text{eres } D E \neq E \rangle$*   
*E-max-lit]*  
**by** *metis*

**hence**  $\forall K \in \# \text{eres } D E. K \neq L \wedge K \neq - L$   
**by** *fastforce*

**moreover have**  $\forall L \in \# \text{eres } D E. L \in \# D \vee L \in \# E$   
**using** *lit-in-one-of-resolvents-if-in-eres* **by** *metis*

**moreover have** *D-almost-false: trail-false-cls  $\Gamma \{ \#K \in \# D. K \neq - L \# \}$*   
**using** *ord-res-7-invars-implies-propagated-clause-almost-false*  
**using**  $\langle (- L, \text{Some } D) \in \text{set } \Gamma \rangle$  *x-invars*  
**by** *metis*

**ultimately have** *trail-false-cls  $\Gamma (\text{eres } D E)$*   
**using** *E-false unfolding trail-false-cls-def* **by** *fastforce*

**have** *eres D E  $\not\in$  N  $\mid \cup \mid U_{er}$*



```

    using eres-not-in-known-clauses-if-trail-false-cls
    using  $\Gamma$ -consistent-clauses-lt-E-true  $\langle$ eres  $D E \prec_c E \rangle$   $\langle$ trail-false-cls  $\Gamma$  (eres
D E) $\rangle$ 
    by metis

    thus eres  $D E \notin U_{er}$ 
    by blast
qed

assume  $R^{-1-1} C_y C_x$ 
hence ord-res-7  $N (U_{er}, \mathcal{F}, \Gamma, C_x) (U_{er}, \mathcal{F}, \Gamma, C_y)$ 
    unfolding conversesep-iff R-def .
thus ?less-f (f  $C_y$ ) (f  $C_x$ )
proof (cases  $N (U_{er}, \mathcal{F}, \Gamma, C_x) (U_{er}, \mathcal{F}, \Gamma, C_y)$  rule: ord-res-7.cases)
    case step-hyps: (decide-neg C L A)
    hence False by simp
    thus ?thesis ..
next
case step-hyps: (skip-defined C L)

have C  $\in$  iefac  $\mathcal{F} \uparrow (N \cup U_{er})$ 
    using ord-res-7-invars-def step-hyps(1) x-invars by presburger

moreover have  $\exists D. C_y = None \vee C_y = Some D \wedge C \prec_c D$ 
proof (cases  $C_y$ )
    case None
    thus ?thesis
    by simp
next
case (Some D)
    thus ?thesis
    using step-hyps
    by (metis linorder-cls.is-least-in-filter-iff Some-eq-The-optionalD)
qed

ultimately show ?thesis
    using less-f-if step-hyps by metis
next
case step-hyps: (skip-undefined-neg C L)
    hence False by simp
    thus ?thesis ..
next
case step-hyps: (skip-undefined-pos C L D)

have C  $\in$  iefac  $\mathcal{F} \uparrow (N \cup U_{er})$ 
    using ord-res-7-invars-def step-hyps(1) x-invars by presburger

moreover have  $\exists D. C_y = None \vee C_y = Some D \wedge C \prec_c D$ 
    using step-hyps by (metis linorder-cls.is-least-in-filter-iff)

```

```

ultimately show ?thesis
  using less-f-if step-hyps by metis
next
case step-hyps: (skip-undefined-pos-ultimate C L)
hence False by simp
thus ?thesis ..
next
case step-hyps: (production C L)
hence False by simp
thus ?thesis ..
next
case step-hyps: (factoring C L)

have C |∈| iefac  $\mathcal{F}$  | $\uparrow$ | (N | $\cup$ |  $U_{er}$ )
  using ord-res-7-invars-def step-hyps(1) x-invars by presburger

moreover have efac C  $\neq$  C
  using step-hyps by (metis greatest-literal-in-efacI)

ultimately have C | $\notin$ |  $\mathcal{F}$ 
by (smt (verit, ccfv-threshold) fimage-iff iefac-def ex1-efac-eq-factoring-chain
    factorizable-if-neq-efac)

hence False
  using  $\langle \mathcal{F} = \text{finsert } C \ \mathcal{F} \rangle$  by blast

thus ?thesis ..
next
case step-hyps: (resolution-bot E L D)
hence eres D E | $\notin$ |  $U_{er}$ 
  using eres-not-in-if by metis
hence False
  using  $\langle U_{er} = \text{finsert } (eres\ D\ E)\ U_{er} \rangle$  by blast
thus ?thesis ..
next
case (resolution-pos E L D K)
hence eres D E | $\notin$ |  $U_{er}$ 
  using eres-not-in-if by metis
hence False
  using  $\langle U_{er} = \text{finsert } (eres\ D\ E)\ U_{er} \rangle$  by blast
thus ?thesis ..
next
case (resolution-neg E L D K C)
hence eres D E | $\notin$ |  $U_{er}$ 
  using eres-not-in-if by metis
hence False
  using  $\langle U_{er} = \text{finsert } (eres\ D\ E)\ U_{er} \rangle$  by blast
thus ?thesis ..

```

```

qed
qed

then obtain  $C'$  where  $R^{**} C C'$  and  $\nexists z. R C' z$ 
  using ex-terminating-rtranclp-strong by metis

show ?thesis
proof (intro exI conjI)
  show  $(ord-res-7 N)^{**} (U_{er}, \mathcal{F}, \Gamma, C) (U_{er}, \mathcal{F}, \Gamma, C')$ 
    using  $\langle R^{**} C C' \rangle$ 
    by (induction C rule: converse-rtranclp-induct) (auto simp: R-def)
next
  show  $\nexists C''. ord-res-7 N (U_{er}, \mathcal{F}, \Gamma, C') (U_{er}, \mathcal{F}, \Gamma, C'')$ 
    using  $\langle \nexists z. R C' z \rangle$ 
    by (simp add: R-def)
qed
qed

inductive ord-res-7-matches-ord-res-8 :: 'f ord-res-7-state  $\Rightarrow$  'f ord-res-8-state  $\Rightarrow$ 
  bool where
  ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma, C) \Longrightarrow$ 
  ord-res-8-invars  $N (U_{er}, \mathcal{F}, \Gamma) \Longrightarrow$ 
   $(\forall C. C = Some C \longleftrightarrow is-least-nonskipped-clause N U_{er} \mathcal{F} \Gamma C) \Longrightarrow$ 
  ord-res-7-matches-ord-res-8  $(N, U_{er}, \mathcal{F}, \Gamma, C) (N, U_{er}, \mathcal{F}, \Gamma)$ 

lemma ord-res-7-final-iff-ord-res-8-final:
  fixes S7 S8
  assumes match: ord-res-7-matches-ord-res-8 S7 S8
  shows ord-res-7-final S7  $\longleftrightarrow$  ord-res-8-final S8
  using match
proof (cases S7 S8 rule: ord-res-7-matches-ord-res-8.cases)
  case match-hyps: (1 N Uer  $\mathcal{F}$   $\Gamma$  C)

  note invars7 =  $\langle ord-res-7-invars N (U_{er}, \mathcal{F}, \Gamma, C) \rangle [unfolded ord-res-7-invars-def,$ 
    rule-format, OF refl]

  have  $\Gamma$ -consistent: trail-consistent  $\Gamma$ 
    using invars7 by (metis trail-consistent-if-sorted-wrt-atoms)

  show ord-res-7-final S7  $\longleftrightarrow$  ord-res-8-final S8
proof (rule iffI)
  assume ord-res-7-final S7
  thus ord-res-8-final S8
    unfolding  $\langle S7 = (N, U_{er}, \mathcal{F}, \Gamma, C) \rangle$ 
  proof (cases  $(N, U_{er}, \mathcal{F}, \Gamma, C)$  rule: ord-res-7-final.cases)
    case model-found
    show ord-res-8-final S8
      unfolding  $\langle S8 = (N, U_{er}, \mathcal{F}, \Gamma) \rangle$ 
    proof (rule ord-res-8-final.model-found)

```

**have**  $C = \text{None} \longrightarrow \text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using** *invars7* **by** *argo*

**hence**  $\text{trail-atms } \Gamma = \text{atms-of-clss } (N \mid \cup \mid U_{er})$   
**using** *model-found* **by** *argo*

**thus**  $\neg (\exists A \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er}). A \mid \notin \mid \text{trail-atms } \Gamma)$   
**by** *metis*

**next**

**have**  $\forall C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-true-clc } \Gamma C$   
**using** *invars7* *model-found* **by** *simp*

**moreover have**  $\neg (\text{trail-true-clc } \Gamma C \wedge \text{trail-false-clc } \Gamma C)$  **for**  $C$   
**using** *not-trail-true-clc-and-trail-false-clc*[*OF*  $\Gamma$ -*consistent*] .

**ultimately show**  $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). \text{trail-false-clc } \Gamma C)$   
**by** *metis*

**qed**

**next**

**case** *contradiction-found*

**show** *ord-res-8-final*  $S8$

**unfolding**  $\langle S8 = (N, U_{er}, \mathcal{F}, \Gamma) \rangle$

**proof** (*rule ord-res-8-final.contradiction-found*)

**show**  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$

**using** *invars7*  $\langle C = \text{Some } \{\#\} \rangle$  **by** *metis*

**qed**

**qed**

**next**

**assume** *ord-res-8-final*  $S8$

**thus** *ord-res-7-final*  $S7$

**unfolding**  $\langle S8 = (N, U_{er}, \mathcal{F}, \Gamma) \rangle$

**proof** (*cases*  $(N, U_{er}, \mathcal{F}, \Gamma)$  *rule: ord-res-8-final.cases*)

**case** *model-found*

**hence**  $\nexists C. \text{is-least-nonskipped-clause } N U_{er} \mathcal{F} \Gamma C$   
**using** *nex-is-least-nonskipped-clause-if* **by** *metis*

**hence**  $C = \text{None}$   
**using** *match-hyps* **by** *simp*

**thus** *ord-res-7-final*  $S7$

**unfolding**  $\langle S7 = (N, U_{er}, \mathcal{F}, \Gamma, C) \rangle$

**using** *ord-res-7-final.model-found* **by** *metis*

**next**

**case** *contradiction-found*

**hence** *is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma \{\#\}$   
**using** *is-least-nonskipped-clause-mempty* **by** *metis*

**hence**  $C = \text{Some } \{\#\}$   
**using** *match-hyps* **by** *presburger*

**thus** *ord-res-7-final*  $S7$   
**unfolding**  $\langle S7 = (N, U_{er}, \mathcal{F}, \Gamma, C) \rangle$   
**using** *ord-res-7-final.contradiction-found* **by** *metis*

**qed**  
**qed**  
**qed**

**lemma** *backward-simulation-between-7-and-8*:  
**fixes**  $i S7 S8 S8'$   
**assumes** *match*: *ord-res-7-matches-ord-res-8*  $S7 S8$  **and** *step*: *constant-context*  
*ord-res-8*  $S8 S8'$   
**shows**  $\exists S7'. (\text{constant-context } \text{ord-res-7})^{++} S7 S7' \wedge \text{ord-res-7-matches-ord-res-8}$   
 $S7' S8'$   
**using** *match*

**proof** (*cases*  $S7 S8$  *rule*: *ord-res-7-matches-ord-res-8.cases*)  
**case** *match-hyps*:  $(1 N U_{er} \mathcal{F} \Gamma C)$

**note**  $S7\text{-def} = \langle S7 = (N, U_{er}, \mathcal{F}, \Gamma, C) \rangle$

**note**  $\text{invars7} = \langle \text{ord-res-7-invars } N (U_{er}, \mathcal{F}, \Gamma, C) \rangle [\text{unfolded } \text{ord-res-7-invars-def},$   
*rule-format, OF refl]*

**have**  $\Gamma\text{-sorted}$ : *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma$   
**using**  $\text{invars7}$  **by** *argo*

**have**  $\Gamma\text{-consistent}$ : *trail-consistent*  $\Gamma$   
**using** *trail-consistent-if-sorted-wrt-atoms* [*OF*  $\Gamma\text{-sorted}$ ] .

**have**  $\Gamma\text{-lower-set}$ : *linorder-trm.is-lower-fset*  $(\text{trail-atms } \Gamma) (\text{atms-of-cls } (N \mid \cup U_{er}))$   
**using**  $\text{invars7}$  **by** *argo*

**have**  $C\text{-eq-None-implies-all-atoms-defined}$ :  $C = \text{None} \longrightarrow \text{trail-atms } \Gamma = \text{atms-of-cls}$   
 $(N \mid \cup U_{er})$   
**using**  $\text{invars7}$  **by** *argo*

**obtain**  $s8'$  **where**  
 $S8'\text{-def}$ :  $S8' = (N, s8')$  **and**  
 $\text{step}'$ : *ord-res-8*  $N (U_{er}, \mathcal{F}, \Gamma) s8'$   
**using** *step* **unfolding**  $\langle S8 = (N, U_{er}, \mathcal{F}, \Gamma) \rangle$   
**by** (*auto elim*: *constant-context.cases*)

**have**  $\text{invars-s8}'$ : *ord-res-8-invars*  $N s8'$   
**using** *ord-res-8-preserves-invars* [*OF*  $\text{step}' \langle \text{ord-res-8-invars } N (U_{er}, \mathcal{F}, \Gamma) \rangle$ ] .

**show** *?thesis*

**using** *step'*  
**proof** (*cases*  $N (U_{er}, \mathcal{F}, \Gamma)$  *s8' rule: ord-res-8.cases*)  
**case** *step-hyps*: (*decide-neg*  $A \Gamma$ )

**have**  
*A-in*:  $A \in |atms-of-clss (N \cup U_{er})$  **and**  
*A-undef*:  $A \notin |trail-atms \Gamma$  **and**  
*A-least*:  $\forall y \in |atms-of-clss (N \cup U_{er}). y \neq A \longrightarrow (\forall A_1 \in |trail-atms \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$   
**using** *step-hyps*(3) **unfolding** *linorder-trm.is-least-in-fset-iff* **by** *auto*

**have**  $C \neq None$   
**using** *C-eq-None-implies-all-atoms-defined* *A-in* *A-undef* **by** *metis*

**then obtain**  $D :: 'f\ gclause$  **where**  $C = Some\ D$   
**by** *blast*

**hence** *D-in*:  $D \in |iefac \mathcal{F} | \uparrow (N \cup U_{er})$   
**by** (*metis*  $\langle C = Some\ D \rangle$  *invars7*)

**have** *is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma D$   
**using** *match-hyps*  $\langle C = Some\ D \rangle$  **by** *metis*

**moreover have** *D-not-false*:  $\neg trail-false-cls \Gamma D$   
**using** *D-in step-hyps* **by** *metis*

**moreover have**  $\neg ord-res-8-can-produce\ N\ U_{er}\ \mathcal{F}\ \Gamma\ D$   
**proof** (*rule notI*)  
**assume** *ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma D$   
**thus** *False*  
**proof** (*cases*  $N U_{er} \mathcal{F} \Gamma D$  *rule: ord-res-8-can-produce.cases*)  
**case** (1 *L'*)  
**thus** *?thesis*  
**by** (*metis* *A-in* *A-least* *A-undef* *D-in* *atm-of-in-atms-of-clssI*  
*atoms-of-trail-lt-atom-of-propagatable-literal* *clause-could-propagate-def*  
*invars7* *linorder-lit.is-maximal-in-mset-iff literal.collapse(1)* *step-hyps*(4))

**qed**  
**qed**

**moreover have**  $\neg ord-res-8-can-factorize\ N\ U_{er}\ \mathcal{F}\ \Gamma\ D$   
**proof** (*rule notI*)  
**assume** *ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma D$   
**thus** *False*  
**proof** (*cases*  $N U_{er} \mathcal{F} \Gamma D$  *rule: ord-res-8-can-factorize.cases*)  
**case** (1 *L'*)  
**thus** *False*  
**by** (*metis* *A-in* *A-least* *A-undef* *D-in* *atm-of-in-atms-of-clssI*  
*atoms-of-trail-lt-atom-of-propagatable-literal* *clause-could-propagate-def*)

```

invars7
  linorder-lit.is-maximal-in-mset-iff literal.collapse(1) step-hyps(4))
  qed
  qed

ultimately have ord-res-8-can-decide-neg N Uer  $\mathcal{F}$   $\Gamma$  D  $\vee$ 
  ord-res-8-can-skip-undefined-neg N Uer  $\mathcal{F}$   $\Gamma$  D  $\vee$ 
  ord-res-8-can-skip-undefined-pos-ultimate N Uer  $\mathcal{F}$   $\Gamma$  D
  unfolding is-least-nonskipped-clause-def
  unfolding linorder-cls.is-least-in-ffilter-iff
  by argo

then obtain C' where first-step7: ord-res-7 N (Uer,  $\mathcal{F}$ ,  $\Gamma$ , Some D) (Uer,  $\mathcal{F}$ ,
 $\Gamma'$ , C')
proof (elim disjE; atomize-elim)
  assume ord-res-8-can-decide-neg N Uer  $\mathcal{F}$   $\Gamma$  D
  thus  $\exists C'$ . ord-res-7 N (Uer,  $\mathcal{F}$ ,  $\Gamma$ , Some D) (Uer,  $\mathcal{F}$ ,  $\Gamma'$ , C')
  proof (cases N Uer  $\mathcal{F}$   $\Gamma$  D rule: ord-res-8-can-decide-neg.cases)
    case hyps: (1 L A')
    hence A' = A
    by (smt (verit, del-insts)  $\Gamma$ -lower-set linorder-trm.is-least-in-ffilter-iff
      linorder-trm.neg-iff linorder-trm.not-in-lower-setI linorder-trm.order.strict-trans
      step-hyps(3))
    thus ?thesis
    using hyps  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$ 
    using ord-res-7.decide-neg[of  $\Gamma$  D - N Uer A  $\Gamma'$   $\mathcal{F}$ ] by blast
  qed
next
  assume ord-res-8-can-skip-undefined-neg N Uer  $\mathcal{F}$   $\Gamma$  D
  thus  $\exists C'$ . ord-res-7 N (Uer,  $\mathcal{F}$ ,  $\Gamma$ , Some D) (Uer,  $\mathcal{F}$ ,  $\Gamma'$ , C')
  proof (cases N Uer  $\mathcal{F}$   $\Gamma$  D rule: ord-res-8-can-skip-undefined-neg.cases)
    case hyps: (1 L)
    hence L = Neg A
    by (smt (verit) A-in A-least A-undef D-in  $\Gamma$ -lower-set atm-of-in-atms-of-clsI
      linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3
      linorder-trm.not-in-lower-setI literal.disc(2) literal.expand literal.sel(2)
      trail-defined-lit-iff-trail-defined-atm)
    thus ?thesis
    using hyps  $\langle \Gamma' = (\text{Neg } A, \text{None}) \# \Gamma \rangle$ 
    using ord-res-7.skip-undefined-neg by blast
  qed
next
  assume ord-res-8-can-skip-undefined-pos-ultimate N Uer  $\mathcal{F}$   $\Gamma$  D
  thus  $\exists C'$ . ord-res-7 N (Uer,  $\mathcal{F}$ ,  $\Gamma$ , Some D) (Uer,  $\mathcal{F}$ ,  $\Gamma'$ , C')
  proof (cases N Uer  $\mathcal{F}$   $\Gamma$  D rule: ord-res-8-can-skip-undefined-pos-ultimate.cases)
    case hyps: (1 L)
    hence L = Pos A
    by (smt (verit, best) A-in A-least A-undef D-in atm-of-in-atms-of-clsI
      invars7

```

```

      linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3
      linorder-trm.not-in-lower-setI literal.disc(1) literal.expand literal.sel(1)
      trail-defined-lit-iff-trail-defined-atm)
    thus ?thesis
      using hyps ⟨ $\Gamma' = (\text{Neg } A, \text{None}) \# \Gamma$ ⟩
      using ord-res-7.skip-undefined-pos-ultimate by fastforce
  qed
qed

moreover have ord-res-7-invars N (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}'$ )
  using ⟨ $\mathcal{C} = \text{Some } D$ ⟩ first-step7 match-hyps(3) ord-res-7-preserves-invars by
blast

ultimately obtain  $\mathcal{C}''$  where
  following-steps7: (ord-res-7 N)** (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}'$ ) (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ ) and
  no-more-step7: (∄  $\mathcal{C}'''$ . ord-res-7 N (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ ) (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}'''$ ))
  using MAGIC6 by metis

show ?thesis
proof (intro exI conjI)
  have (ord-res-7 N)** (Uer,  $\mathcal{F}$ ,  $\Gamma$ ,  $\mathcal{C}$ ) (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ )
    unfolding ⟨ $\mathcal{C} = \text{Some } D$ ⟩
    using first-step7 following-steps7 by simp

  thus (constant-context ord-res-7)** S7 (N, Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ )
    unfolding S7-def by (simp add: tranclp-constant-context)

  show ord-res-7-matches-ord-res-8 (N, Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ ) S8'
    unfolding S8'-def ⟨ $s8' = (U_{er}, \mathcal{F}, \Gamma')$ ⟩
  proof (intro ord-res-7-matches-ord-res-8.intros allI)
    show ord-res-7-invars N (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ )
      using ⟨ord-res-7-invars N (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}'$ )⟩ following-steps7
      rtranclp-ord-res-7-preserves-ord-res-7-invars by blast

    show ord-res-8-invars N (Uer,  $\mathcal{F}$ ,  $\Gamma'$ )
      using invars-s8' step-hyps(1) by blast

  fix C :: 'f g clause
  show  $\mathcal{C}'' = \text{Some } C \iff \text{is-least-nonskipped-clause } N U_{er} \mathcal{F} \Gamma' C$ 
    using MAGIC5 ⟨ord-res-7-invars N (Uer,  $\mathcal{F}$ ,  $\Gamma'$ ,  $\mathcal{C}''$ )⟩ no-more-step7 by
metis
  qed
  qed
next
case step-hyps: (propagate A C  $\Gamma'$ )

have
  A-in: A |∈| atms-of-clss (N |∪| Uer) and
  A-undef: A |∉| trail-atms  $\Gamma$  and

```



$A$ -least:  $\forall y | \in | \text{atms-of-clss } (N \cup | U_{er}). y \neq A \longrightarrow (\forall A_1 | \in | \text{trail-atms } \Gamma. A_1 \prec_t y) \longrightarrow A \prec_t y$

**using** *step-hyps*(3) **unfolding** *linorder-trm.is-least-in-fset-iff* **by** *auto*

**have**

$C$ -in:  $C | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er})$  **and**

$C$ -can-prop: *clause-could-propagate*  $\Gamma C (Pos A)$  **and**

$C$ -least:  $\forall D | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er}).$

$D \neq C \longrightarrow \text{clause-could-propagate } \Gamma D (Pos A) \longrightarrow C \prec_c D$

**using** *step-hyps* **unfolding** *atomize-conj linorder-cls.is-least-in-ffilter-iff* **by**

*argo*

**hence**

$Pos A$ -undef:  $\neg \text{trail-defined-lit } \Gamma (Pos A)$  **and**

$C$ -max-lit: *linorder-lit.is-maximal-in-mset*  $C (Pos A)$  **and**

$C$ -almost-false: *trail-false-cls*  $\Gamma \{\#K \in \# C. K \neq Pos A \#\}$

**unfolding** *atomize-conj clause-could-propagate-def* **by** *argo*

**have** *is-least-nonskipped-clause*  $N U_{er} \mathcal{F} \Gamma C$

**unfolding** *is-least-nonskipped-clause-def*

**unfolding** *linorder-cls.is-least-in-ffilter-iff*

**proof** (*intro conjI ballI impI*)

**show**  $C | \in | \text{iefac } \mathcal{F} | \uparrow (N \cup | U_{er})$

**using**  $C$ -in .

**next**

**have** *ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C$

**proof** (*rule ord-res-8-can-produce.intros*)

**show**  $\neg \text{trail-false-cls } \Gamma C$

**using** *step-hyps*  $C$ -in **by** *metis*

**next**

**show** *ord-res.is-maximal-lit*  $(Pos A) C$

**using** *step-hyps* **by** *blast*

**next**

**show**  $\neg (\exists Aa | \in | \text{atms-of-clss } (N \cup | U_{er}). Aa \prec_t \text{atm-of } (Pos A) \wedge Aa \notin | \text{trail-atms } \Gamma)$

**unfolding** *literal.sel*

**using** *step-hyps*

**by** (*smt* (*verit*, *ccfv-threshold*)  $\Gamma$ -*lower-set linorder-trm.dual-order.asym*

*linorder-trm.is-least-in-ffilter-iff linorder-trm.is-lower-set-iff*

*linorder-trm.neq-iff*)

**next**

**show**  $\neg \text{trail-defined-lit } \Gamma (Pos A)$

**using**  $A$ -undef **unfolding** *trail-defined-lit-iff-trail-defined-atm literal.sel* .

**next**

**show** *is-pos*  $(Pos A)$

**by** *simp*

**next**

**show** *trail-false-cls*  $\Gamma \{\#K \in \# C. K \neq Pos A \#\}$

**using**  $\langle \text{clause-could-propagate } \Gamma C (Pos A) \rangle$

**unfolding** *clause-could-propagate-def* **by** *argo*  
**next**  
**show** *ord-res.is-strictly-maximal-lit* (*Pos A*) *C*  
**using** *step-hyps* **by** *argo*  
**qed**

**thus** *trail-false-cls*  $\Gamma C \vee$   
*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma C \vee$   
*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma C \vee$  *ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma C$   
**by** *argo*

**next**

**fix**  $D :: 'f gclause$

**assume**

$D-in: D \in |iefac \mathcal{F} |^{\uparrow} (N \cup U_{er})$  **and**

$D-neg: D \neq C$  **and**

$D-spec-disj: trail-false-cls \Gamma D \vee$

*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma D$

**hence**

*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma D \vee$

*ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma D$

**using** *step-hyps*  $D-in$  **by** *metis*

**thus**  $C \prec_c D$

**proof** (*elim disjE*)

**assume** *ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma D$

**thus**  $C \prec_c D$

**proof** (*cases*  $N U_{er} \mathcal{F} \Gamma D$  *rule: ord-res-8-can-decide-neg.cases*)

**case** *hyps: (1 L A')*

**hence**  $A = A'$

**using** *step-hyps*

**by** (*smt (verit, del-insts)*  $\Gamma$ -*lower-set linorder-trm.antisym-conv3*

*linorder-trm.dual-order.strict-implies-not-eq linorder-trm.dual-order.strict-trans*

*linorder-trm.is-least-in-ffilter-iff linorder-trm.not-in-lower-setI*)

**hence**  $A \prec_t atm-of L$

**using** *hyps*

**unfolding** *linorder-trm.is-least-in-ffilter-iff*

**by** *argo*

**hence**  $Pos A \prec_l L$

**by** (*cases L*) *simp-all*

**thus** *?thesis*  
**using** *C-max-lit*  $\langle \text{linorder-lit.is-maximal-in-mset } D \ L \rangle$   
**by** (*metis linorder-lit.multip-if-maximal-less-that-maximal*)  
**qed**  
**next**  
**assume** *ord-res-8-can-skip-undefined-neg*  $N \ U_{er} \ \mathcal{F} \ \Gamma \ D$   
**thus**  $C \prec_c D$   
**proof** (*cases N U<sub>er</sub> F Γ D rule: ord-res-8-can-skip-undefined-neg.cases*)  
**case** *hyps: (1 L)*  
**hence** *atm-of*  $L = A$   
**using** *step-hyps*  
**by** (*smt (verit, best)*  
*A-in A-least A-undef D-in Γ-lower-set atm-of-in-atms-of-clssI*  
*linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3*  
*linorder-trm.not-in-lower-setI trail-defined-lit-iff-trail-defined-atm*)  
**hence**  $Pos \ A \prec_l L$   
**using**  $\langle \text{is-neg } L \rangle$  **by** (*cases L*) *simp-all*  
**thus** *?thesis*  
**using** *C-max-lit*  $\langle \text{linorder-lit.is-maximal-in-mset } D \ L \rangle$   
**by** (*metis linorder-lit.multip-if-maximal-less-that-maximal*)  
**qed**  
**next**  
**assume** *ord-res-8-can-skip-undefined-pos-ultimate*  $N \ U_{er} \ \mathcal{F} \ \Gamma \ D$   
**thus**  $C \prec_c D$   
**proof** (*cases N U<sub>er</sub> F Γ D rule: ord-res-8-can-skip-undefined-pos-ultimate.cases*)  
**case** *hyps: (1 L)*  
**thus** *?thesis*  
**by** (*meson C-in D-neq linorder-cls.linorder-cases*)  
**qed**  
**next**  
**assume** *ord-res-8-can-produce*  $N \ U_{er} \ \mathcal{F} \ \Gamma \ D$   
  
**then obtain**  $L$  **where**  
*D-max-lit: ord-res.is-maximal-lit L D and*  
 $\neg (\exists A | \in | \text{atms-of-clss} (N \cup U_{er}). A \prec_t \text{atm-of } L \wedge A \notin | \text{trail-atms } \Gamma)$   
**and**  
*L-undef:  $\neg$  trail-defined-lit  $\Gamma \ L$  and*  
*D-almost-false: trail-false-cls  $\Gamma \ \{\#K \in \# D. K \neq L\#$  and*  
*is-pos L*  
**by** (*auto elim: ord-res-8-can-factorize.cases ord-res-8-can-produce.cases*)  
  
**hence** *atm-of*  $L = A$   
**using** *step-hyps*  
**by** (*smt (verit, ccfv-SIG)* *A-in A-least A-undef D-in Γ-lower-set*  
*linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3*  
*linorder-trm.not-in-lower-setI atm-of-in-atms-of-clssI*  
*trail-defined-lit-iff-trail-defined-atm*)  
  
**hence**  $L = Pos \ A$

**using**  $\langle is\text{-}pos\ L \rangle$  **by**  $(cases\ L)$  *simp-all*

**hence** *clause-could-propagate*  $\Gamma\ D\ (Pos\ A)$   
**unfolding** *clause-could-propagate-def*  
**using** *D-almost-false D-max-lit L-undef* **by** *metis*

**thus**  $C \prec_c D$   
**using** *D-in D-neq C-least* **by** *metis*

**next**  
**assume** *ord-res-8-can-factorize*  $N\ U_{er}\ \mathcal{F}\ \Gamma\ D$

**then obtain**  $L$  **where**  
*D-max-lit: ord-res.is-maximal-lit*  $L\ D$  **and**  
 $\neg (\exists A|\in|atms\text{-}of\text{-}class\ (N\ |\cup|\ U_{er}).\ A \prec_t atm\text{-}of\ L \wedge A \notin| trail\text{-}atms\ \Gamma)$

**and**  
*L-undef:  $\neg$  trail-defined-lit*  $\Gamma\ L$  **and**  
*D-almost-false: trail-false-cls*  $\Gamma\ \{\#K \in\# D.\ K \neq L\#\}$  **and**  
*is-pos*  $L$   
**by**  $(auto\ elim:\ ord\text{-}res\text{-}8\text{-}can\text{-}factorize.cases\ ord\text{-}res\text{-}8\text{-}can\text{-}produce.cases)$

**hence**  $atm\text{-}of\ L = A$   
**using** *step-hyps*  
**by**  $(smt\ (verit,\ ccfv\text{-}SIG)\ A\text{-}in\ A\text{-}least\ A\text{-}undef\ D\text{-}in\ \Gamma\text{-}lower\text{-}set$   
*linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3*  
*linorder-trm.not-in-lower-setI atm-of-in-atms-of-classI*  
*trail-defined-lit-iff-trail-defined-atm)*

**hence**  $L = Pos\ A$   
**using**  $\langle is\text{-}pos\ L \rangle$  **by**  $(cases\ L)$  *simp-all*

**hence** *clause-could-propagate*  $\Gamma\ D\ (Pos\ A)$   
**unfolding** *clause-could-propagate-def*  
**using** *D-almost-false D-max-lit L-undef* **by** *metis*

**thus**  $C \prec_c D$   
**using** *D-in D-neq C-least* **by** *metis*

**qed**  
**qed**

**hence**  $\mathcal{C} = Some\ C$   
**using** *match-hyps* **by** *metis*

**define**  $\mathcal{C}'$  **where**  
 $\mathcal{C}' = The\text{-}optional\ (linorder\text{-}cls.is\text{-}least\text{-}in\text{-}fset$   
 $(ffilter\ ((\prec_c)\ C)\ (iefac\ \mathcal{F}\ |'|\ (N\ |\cup|\ U_{er}))))$

**have** *first-step7: ord-res-7*  $N\ (U_{er},\ \mathcal{F},\ \Gamma,\ Some\ C)\ (U_{er},\ \mathcal{F},\ \Gamma',\ \mathcal{C}')$   
**proof**  $(rule\ ord\text{-}res\text{-}7.production)$   
**show**  $\neg trail\text{-}false\text{-}cls\ \Gamma\ C$

```

    using C-in step-hyps(2) by blast
  next
    show ord-res.is-maximal-lit (Pos A) C
      using C-max-lit by force
  next
    show  $\neg (\exists Aa \in |atms-of-cls (N \cup U_{er}). Aa \prec_t atm-of (Pos A) \wedge Aa \notin |$ 
trail-atms  $\Gamma)$ 
      by (metis A-least  $\Gamma$ -lower-set linorder-trm.dual-order.asym linorder-trm.neq-iff
          linorder-trm.not-in-lower-setI literal.sel(1))
  next
    show  $\neg$  trail-defined-lit  $\Gamma$  (Pos A)
      using Pos-A-undef .
  next
    show is-pos (Pos A)
      by simp
  next
    show trail-false-cls  $\Gamma$   $\{\#K \in \# C. K \neq Pos A\#$ 
      using C-almost-false .
  next
    show ord-res.is-strictly-maximal-lit (Pos A) C
      using step-hyps by argo
  next
    show  $\Gamma' = (Pos A, Some C) \# \Gamma$ 
      using step-hyps by argo
  next
    show  $C' = The-optional (linorder-cls.is-least-in-fset (ffilter (( $\prec_c$ ) C) (iefac$ 
 $\mathcal{F} \upharpoonright (N \cup U_{er})))$ 
      using C'-def .
  qed

  moreover have ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma', C')$ 
    using  $\langle C = Some C \rangle$  first-step7 match-hyps(3) ord-res-7-preserves-invars by
  blast

  ultimately obtain  $C''$  where
    following-steps7: (ord-res-7  $N$ )**  $(U_{er}, \mathcal{F}, \Gamma', C') (U_{er}, \mathcal{F}, \Gamma', C'')$  and
    no-more-step7: ( $\nexists C'''$ . ord-res-7  $N (U_{er}, \mathcal{F}, \Gamma', C'') (U_{er}, \mathcal{F}, \Gamma', C''')$ )
    using MAGIC6 by metis

  show ?thesis
  proof (intro exI conjI)
    have (ord-res-7  $N$ )**  $(U_{er}, \mathcal{F}, \Gamma, C) (U_{er}, \mathcal{F}, \Gamma', C'')$ 
      unfolding  $\langle C = Some C \rangle$ 
      using first-step7 following-steps7 by simp

    thus (constant-context ord-res-7)**  $S7 (N, U_{er}, \mathcal{F}, \Gamma', C'')$ 
      unfolding S7-def by (simp add: tranclp-constant-context)

    show ord-res-7-matches-ord-res-8  $(N, U_{er}, \mathcal{F}, \Gamma', C'') S8'$ 

```

**unfolding**  $S8'$ -def  $\langle s8' = (U_{er}, \mathcal{F}, \Gamma) \rangle$   
**proof** (intro ord-res-7-matches-ord-res-8.intros allI)  
**show** ord-res-7-invars  $N (U_{er}, \mathcal{F}, \Gamma', C'')$   
**using**  $\langle \text{ord-res-7-invars } N (U_{er}, \mathcal{F}, \Gamma', C') \rangle$  following-steps7  
by blast  
  
**show** ord-res-8-invars  $N (U_{er}, \mathcal{F}, \Gamma')$   
**using** invars-s8' step-hyps(1) **by** blast  
  
**fix**  $C :: 'f$  gclause  
**show**  $C'' = \text{Some } C \longleftrightarrow \text{is-least-nonskipped-clause } N U_{er} \mathcal{F} \Gamma' C$   
**using** MAGIC5  $\langle \text{ord-res-7-invars } N (U_{er}, \mathcal{F}, \Gamma', C'') \rangle$  no-more-step7 **by**  
metis  
**qed**  
**qed**  
**next**  
**case** step-hyps: (factorize  $A C \mathcal{F}'$ )  
  
**have**  
 $A$ -in:  $A \in | \text{atms-of-clss } (N \cup | U_{er})$  **and**  
 $A$ -undef:  $A \notin | \text{trail-atms } \Gamma$  **and**  
 $A$ -least:  $\forall y \in | \text{atms-of-clss } (N \cup | U_{er}). y \neq A \longrightarrow (\forall A_1 \in | \text{trail-atms } \Gamma. A_1$   
 $\prec_t y) \longrightarrow A \prec_t y$   
**using** step-hyps(3) **unfolding** linorder-trm.is-least-in-fset-iff **by** auto  
  
**have**  
 $C$ -in:  $C \in | \text{iefac } \mathcal{F} \upharpoonright | (N \cup | U_{er})$  **and**  
 $C$ -can-prop: clause-could-propagate  $\Gamma C (Pos A)$  **and**  
 $C$ -least:  $\forall D \in | \text{iefac } \mathcal{F} \upharpoonright | (N \cup | U_{er}).$   
 $D \neq C \longrightarrow \text{clause-could-propagate } \Gamma D (Pos A) \longrightarrow C \prec_c D$   
**using** step-hyps **unfolding** atomize-conj linorder-cls.is-least-in-filter-iff **by**  
argo  
  
**hence**  
 $Pos$ - $A$ -undef:  $\neg \text{trail-defined-lit } \Gamma (Pos A)$  **and**  
 $C$ -max-lit: linorder-lit.is-maximal-in-mset  $C (Pos A)$  **and**  
 $C$ -almost-false: trail-false-cls  $\Gamma \{ \#K \in \# C. K \neq Pos A \# \}$   
**unfolding** atomize-conj clause-could-propagate-def **by** argo  
  
**have**  $C$ -not-false:  $\neg \text{trail-false-cls } \Gamma C$   
**using**  $C$ -in step-hyps **by** metis  
  
**have** no-undef-atm-lt- $A$ :  
 $\neg (\exists Aa \in | \text{atms-of-clss } (N \cup | U_{er}). Aa \prec_t A \wedge Aa \notin | \text{trail-atms } \Gamma)$   
**by** (metis  $A$ -least  $\Gamma$ -lower-set linorder-trm.dual-order.asym linorder-trm.neq-iff  
linorder-trm.not-in-lower-setI)  
  
**have** is-least-nonskipped-clause  $N U_{er} \mathcal{F} \Gamma C$   
**unfolding** is-least-nonskipped-clause-def

```

unfolding linorder-cls.is-least-in-filter-iff
proof (intro conjI ballI impI)
  show  $C \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$ 
    using C-in .
next
  have ord-res-8-can-factorize  $N U_{er} \mathcal{F} \Gamma C$ 
  proof (intro ord-res-8-can-factorize.intros)
    show  $\neg \text{trail-false-cls } \Gamma C$ 
      using C-not-false .
  next
    show ord-res.is-maximal-lit  $(Pos A) C$ 
      using C-max-lit .
  next
    show  $\neg (\exists Aa \in \text{atms-of-clss } (N \mid \cup \mid U_{er}). Aa \prec_t \text{atm-of } (Pos A) \wedge Aa \notin \text{trail-atms } \Gamma)$ 
      using no-undef-atm-lt-A by simp
  next
    show  $\neg \text{trail-defined-lit } \Gamma (Pos A)$ 
      using Pos-A-undef .
  next
    show is-pos  $(Pos A)$ 
      by simp
  next
    show trail-false-cls  $\Gamma \{\#K \in \# C. K \neq Pos A\# \}$ 
      using C-almost-false .
  next
    show  $\neg \text{ord-res.is-strictly-maximal-lit } (Pos A) C$ 
      using step-hyps by argo
qed

thus trail-false-cls  $\Gamma C \vee$ 
  ord-res-8-can-decide-neg  $N U_{er} \mathcal{F} \Gamma C \vee$ 
  ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma C \vee$ 
  ord-res-8-can-skip-undefined-pos-ultimate  $N U_{er} \mathcal{F} \Gamma C \vee$ 
  ord-res-8-can-produce  $N U_{er} \mathcal{F} \Gamma C \vee$ 
  ord-res-8-can-factorize  $N U_{er} \mathcal{F} \Gamma C$ 
  using  $\langle \text{ord-res-8-can-factorize } N U_{er} \mathcal{F} \Gamma C \rangle$  by argo
next
fix  $D :: 'f \text{ gclause}$ 
assume
   $D\text{-in}: D \in \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  and
   $D\text{-neg}: D \neq C$  and
   $D\text{-spec-disj}: \text{trail-false-cls } \Gamma D \vee$ 
  ord-res-8-can-decide-neg  $N U_{er} \mathcal{F} \Gamma D \vee$ 
  ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma D \vee$ 
  ord-res-8-can-skip-undefined-pos-ultimate  $N U_{er} \mathcal{F} \Gamma D \vee$ 
  ord-res-8-can-produce  $N U_{er} \mathcal{F} \Gamma D \vee$ 
  ord-res-8-can-factorize  $N U_{er} \mathcal{F} \Gamma D$ 

```

hence

*ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma D \vee$   
*ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma D \vee$   
*ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma D \vee$   
*ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma D \vee$   
*ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma D$   
using *step-hyps* *D-in* by *metis*

thus  $C \prec_c D$

proof (*elim disjE*)

assume *ord-res-8-can-decide-neg*  $N U_{er} \mathcal{F} \Gamma D$

thus  $C \prec_c D$

proof (*cases*  $N U_{er} \mathcal{F} \Gamma D$  rule: *ord-res-8-can-decide-neg.cases*)

case *hyps*: (1  $L A'$ )

hence  $A = A'$

using *step-hyps*

by (*smt* (*verit*, *del-insts*)  $\Gamma$ -*lower-set* *linorder-trm.antisym-conv3*

*linorder-trm.dual-order.strict-implies-not-eq* *linorder-trm.dual-order.strict-trans*

*linorder-trm.is-least-in-ffilter-iff* *linorder-trm.not-in-lower-setI*)

hence  $A \prec_t \text{atm-of } L$

using *hyps*

unfolding *linorder-trm.is-least-in-ffilter-iff*

by *argo*

hence  $\text{Pos } A \prec_l L$

by (*cases*  $L$ ) *simp-all*

thus ?thesis

using *C-max-lit*  $\langle \text{linorder-lit.is-maximal-in-mset } D L \rangle$

by (*metis* *linorder-lit.multip-if-maximal-less-that-maximal*)

qed

next

assume *ord-res-8-can-skip-undefined-neg*  $N U_{er} \mathcal{F} \Gamma D$

thus  $C \prec_c D$

proof (*cases*  $N U_{er} \mathcal{F} \Gamma D$  rule: *ord-res-8-can-skip-undefined-neg.cases*)

case *hyps*: (1  $L$ )

hence  $\text{atm-of } L = A$

using *step-hyps*

by (*smt* (*verit*, *best*)

*A-in A-least A-undef D-in*  $\Gamma$ -*lower-set* *atm-of-in-atms-of-clsI*

*linorder-lit.is-maximal-in-mset-iff* *linorder-trm.antisym-conv3*

*linorder-trm.not-in-lower-setI* *trail-defined-lit-iff-trail-defined-atm*)

hence  $\text{Pos } A \prec_l L$

using  $\langle \text{is-neg } L \rangle$  by (*cases*  $L$ ) *simp-all*

thus ?thesis

using *C-max-lit*  $\langle \text{linorder-lit.is-maximal-in-mset } D L \rangle$

by (*metis* *linorder-lit.multip-if-maximal-less-that-maximal*)

qed

next

assume *ord-res-8-can-skip-undefined-pos-ultimate*  $N U_{er} \mathcal{F} \Gamma D$

thus  $C \prec_c D$



**proof** (*cases*  $N U_{er} \mathcal{F} \Gamma D$  *rule: ord-res-8-can-skip-undefined-pos-ultimate.cases*)  
**case** *hypos*: (1  $L$ )  
**thus** *?thesis*  
**by** (*meson*  $C$ -in  $D$ -neq *linorder-cls.linorder-cases*)  
**qed**  
**next**  
**assume** *ord-res-8-can-produce*  $N U_{er} \mathcal{F} \Gamma D$

**then obtain**  $L$  **where**  
*D-max-lit*: *ord-res.is-maximal-lit*  $L D$  **and**  
 $\neg (\exists A | \in | \text{atms-of-clss} (N \cup U_{er}). A \prec_t \text{atm-of } L \wedge A \notin | \text{trail-atms } \Gamma)$

**and**  
*L-undef*:  $\neg \text{trail-defined-lit } \Gamma L$  **and**  
*D-almost-false*: *trail-false-cls*  $\Gamma \{\#K \in \# D. K \neq L\# \}$  **and**  
*is-pos*  $L$   
**by** (*auto elim*: *ord-res-8-can-factorize.cases ord-res-8-can-produce.cases*)

**hence** *atm-of*  $L = A$   
**using** *step-hyps*  
**by** (*smt* (*verit*, *ccfv-SIG*)  $A$ -in  $A$ -least  $A$ -undef  $D$ -in  $\Gamma$ -lower-set  
*linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3*  
*linorder-trm.not-in-lower-setI atm-of-in-atms-of-clssI*  
*trail-defined-lit-iff-trail-defined-atm*)

**hence**  $L = \text{Pos } A$   
**using**  $\langle \text{is-pos } L \rangle$  **by** (*cases*  $L$ ) *simp-all*

**hence** *clause-could-propagate*  $\Gamma D$  ( $\text{Pos } A$ )  
**unfolding** *clause-could-propagate-def*  
**using** *D-almost-false D-max-lit L-undef* **by** *metis*

**thus**  $C \prec_c D$   
**using**  $D$ -in  $D$ -neq  $C$ -least **by** *metis*

**next**  
**assume** *ord-res-8-can-factorize*  $N U_{er} \mathcal{F} \Gamma D$

**then obtain**  $L$  **where**  
*D-max-lit*: *ord-res.is-maximal-lit*  $L D$  **and**  
 $\neg (\exists A | \in | \text{atms-of-clss} (N \cup U_{er}). A \prec_t \text{atm-of } L \wedge A \notin | \text{trail-atms } \Gamma)$

**and**  
*L-undef*:  $\neg \text{trail-defined-lit } \Gamma L$  **and**  
*D-almost-false*: *trail-false-cls*  $\Gamma \{\#K \in \# D. K \neq L\# \}$  **and**  
*is-pos*  $L$   
**by** (*auto elim*: *ord-res-8-can-factorize.cases ord-res-8-can-produce.cases*)

**hence** *atm-of*  $L = A$   
**using** *step-hyps*  
**by** (*smt* (*verit*, *ccfv-SIG*)  $A$ -in  $A$ -least  $A$ -undef  $D$ -in  $\Gamma$ -lower-set  
*linorder-lit.is-maximal-in-mset-iff linorder-trm.antisym-conv3*)

*linorder-trm.not-in-lower-setI atm-of-in-atms-of-clsI  
trail-defined-lit-iff-trail-defined-atm)*

**hence**  $L = \text{Pos } A$   
**using**  $\langle \text{is-pos } L \rangle$  **by**  $(\text{cases } L)$  *simp-all*

**hence** *clause-could-propagate*  $\Gamma D (\text{Pos } A)$   
**unfolding** *clause-could-propagate-def*  
**using** *D-almost-false D-max-lit L-undef* **by** *metis*

**thus**  $C \prec_c D$   
**using** *D-in D-neq C-least* **by** *metis*

**qed**  
**qed**

**hence**  $C = \text{Some } C$   
**using** *match-hyps* **by** *metis*

**define**  $C'$  **where**  
 $C' = \text{Some } (\text{efac } C)$

**have** *first-step7: ord-res-7 N*  $(U_{er}, \mathcal{F}, \Gamma, \text{Some } C) (U_{er}, \mathcal{F}', \Gamma, C')$   
**unfolding** *C'-def*

**proof** (*rule ord-res-7.factoring*)  
**show**  $\neg \text{trail-false-cls } \Gamma C$   
**using** *C-not-false* .

**next**  
**show** *ord-res.is-maximal-lit*  $(\text{Pos } A) C$   
**using** *C-max-lit* .

**next**  
**show**  $\neg (\exists Aa | \in | \text{atms-of-cls } (N \cup U_{er}). Aa \prec_t \text{atm-of } (\text{Pos } A) \wedge Aa \notin | \text{trail-atms } \Gamma)$   
**using** *no-undef-atm-lt-A* **by** *simp*

**next**  
**show**  $\neg \text{trail-defined-lit } \Gamma (\text{Pos } A)$   
**using** *Pos-A-undef* .

**next**  
**show** *is-pos*  $(\text{Pos } A)$   
**by** *simp*

**next**  
**show** *trail-false-cls*  $\Gamma \{\#K \in \# C. K \neq \text{Pos } A\# \}$   
**using** *C-almost-false* .

**next**  
**show**  $\neg \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) C$   
**using** *step-hyps* **by** *argo*

**next**  
**show**  $\mathcal{F}' = \text{finsert } C \mathcal{F}$   
**using** *step-hyps* **by** *argo*

**qed**

**moreover have** *ord-res-7-invars*  $N (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}')$   
**using**  $\langle \mathcal{C} = \text{Some } C \rangle$  *first-step7 match-hyps(3) ord-res-7-preserves-invars* **by**  
*blast*

**ultimately obtain**  $\mathcal{C}''$  **where**

*following-steps7*:  $(\text{ord-res-7 } N)^{**} (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}') (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'')$  **and**  
*no-more-step7*:  $(\nexists \mathcal{C}'''. \text{ord-res-7 } N (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'') (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'''))$   
**using** *MAGIC6* **by** *metis*

**show** *?thesis*

**proof** (*intro exI conjI*)

**have**  $(\text{ord-res-7 } N)^{++} (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'')$   
**unfolding**  $\langle \mathcal{C} = \text{Some } C \rangle$   
**using** *first-step7 following-steps7* **by** *simp*

**thus**  $(\text{constant-context ord-res-7})^{++} S7 (N, U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'')$   
**unfolding** *S7-def* **by** (*simp add: tranclp-constant-context*)

**show** *ord-res-7-matches-ord-res-8*  $(N, U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'') S8'$   
**unfolding** *S8'-def*  $\langle s8' = (U_{er}, \mathcal{F}', \Gamma) \rangle$

**proof** (*intro ord-res-7-matches-ord-res-8.intros allI*)

**show** *ord-res-7-invars*  $N (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'')$   
**using**  $\langle \text{ord-res-7-invars } N (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}') \rangle$  *following-steps7*  
*rtranclp-ord-res-7-preserves-ord-res-7-invars* **by** *blast*

**show** *ord-res-8-invars*  $N (U_{er}, \mathcal{F}', \Gamma)$   
**using** *invars-s8' step-hyps(1)* **by** *blast*

**fix**  $C :: !f \text{ gclause}$

**show**  $\mathcal{C}'' = \text{Some } C \longleftrightarrow \text{is-least-nonskipped-clause } N U_{er} \mathcal{F}' \Gamma C$

**using** *MAGIC5*  $\langle \text{ord-res-7-invars } N (U_{er}, \mathcal{F}', \Gamma, \mathcal{C}'') \rangle$  *no-more-step7* **by**

*metis*

**qed**

**qed**

**next**

**case** *step-hyps*: (*resolution*  $E A D U_{er}' \Gamma'$ )

**note**  $E\text{-max-lit} = \langle \text{ord-res.is-maximal-lit } (Neg A) E \rangle$

**have**

$E\text{-in}$ :  $E \in | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er})$  **and**

$E\text{-false}$ : *trail-false-cls*  $\Gamma E$  **and**

$E\text{-least}$ :  $\forall F \in | \in | \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er}). F \neq E \longrightarrow \text{trail-false-cls } \Gamma F \longrightarrow E$

$\prec_c F$

**using** *step-hyps*

**unfolding** *atomize-conj*

**unfolding** *linorder-cls.is-least-in-filter-iff*

**by** *argo*

```

have is-least-nonskipped-clause  $N U_{er} \mathcal{F} \Gamma E$ 
  unfolding is-least-nonskipped-clause-def
  unfolding linorder-cls.is-least-in-filter-iff
proof (intro conjI ballI impI)
  show  $E \in \text{iefac } \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er})$ 
    using E-in .
next
  show trail-false-cl  $\Gamma E \vee$ 
    ord-res-8-can-decide-neg  $N U_{er} \mathcal{F} \Gamma E \vee$ 
    ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma E \vee$ 
    ord-res-8-can-skip-undefined-pos-ultimate  $N U_{er} \mathcal{F} \Gamma E \vee$ 
    ord-res-8-can-produce  $N U_{er} \mathcal{F} \Gamma E \vee$ 
    ord-res-8-can-factorize  $N U_{er} \mathcal{F} \Gamma E$ 
    using E-false by argo
next
fix  $F :: 'f \text{ gclause}$ 
assume
   $F\text{-in}: F \in \text{iefac } \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er})$  and
   $F\text{-neg}: F \neq E$  and
   $D\text{-spec-disj}: \text{trail-false-cl } \Gamma F \vee$ 
    ord-res-8-can-decide-neg  $N U_{er} \mathcal{F} \Gamma F \vee$ 
    ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma F \vee$ 
    ord-res-8-can-skip-undefined-pos-ultimate  $N U_{er} \mathcal{F} \Gamma F \vee$ 
    ord-res-8-can-produce  $N U_{er} \mathcal{F} \Gamma F \vee$ 
    ord-res-8-can-factorize  $N U_{er} \mathcal{F} \Gamma F$ 

show  $E \prec_c F$ 
  using D-spec-disj
proof (elim disjE)
  assume trail-false-cl  $\Gamma F$ 
  thus  $E \prec_c F$ 
    using E-least F-in F-neg by metis
next
  assume ord-res-8-can-decide-neg  $N U_{er} \mathcal{F} \Gamma F$ 
  thus  $E \prec_c F$ 
proof (cases  $N U_{er} \mathcal{F} \Gamma F$  rule: ord-res-8-can-decide-neg.cases)
  case hyps: (1 L' A')
  thus ?thesis
    using no-undef-atom-le-max-lit-if-lt-false-clause[
       $OF \Gamma\text{-lower-set } E\text{-in } E\text{-false } E\text{-max-lit } F\text{-in } \langle \text{ord-res.is-maximal-lit } L' \rangle$ 
    ]
    by (metis (no-types, lifting) F-neg linorder-cls.neg-iff
      linorder-trm.is-least-in-filter-iff reflclp-iff)
qed
next
  assume ord-res-8-can-skip-undefined-neg  $N U_{er} \mathcal{F} \Gamma F$ 
  thus  $E \prec_c F$ 
proof (cases  $N U_{er} \mathcal{F} \Gamma F$  rule: ord-res-8-can-skip-undefined-neg.cases)

```

```

    case (1 L')
    thus ?thesis
    using no-undef-atom-le-max-lit-if-lt-false-clause[
      OF  $\Gamma$ -lower-set E-in E-false E-max-lit F-in  $\langle$ ord-res.is-maximal-lit L'
F>]
    by (metis F-in F-neq atm-of-in-atms-of-clsI linorder-cls.not-less-iff-gr-or-eq
      linorder-lit.is-maximal-in-mset-iff reflclp-iff
      trail-defined-lit-iff-trail-defined-atm)
    qed
  next
  assume ord-res-8-can-skip-undefined-pos-ultimate N Uer  $\mathcal{F}$   $\Gamma$  F
  thus E  $\prec_c$  F
  proof (cases N Uer  $\mathcal{F}$   $\Gamma$  F rule: ord-res-8-can-skip-undefined-pos-ultimate.cases)
    case (1 L')
    thus ?thesis
    using no-undef-atom-le-max-lit-if-lt-false-clause[
      OF  $\Gamma$ -lower-set E-in E-false E-max-lit F-in  $\langle$ ord-res.is-maximal-lit L'
F>]
    by (metis F-in F-neq atm-of-in-atms-of-clsI linorder-cls.not-less-iff-gr-or-eq
      linorder-lit.is-maximal-in-mset-iff reflclp-iff
      trail-defined-lit-iff-trail-defined-atm)
    qed
  next
  assume ord-res-8-can-produce N Uer  $\mathcal{F}$   $\Gamma$  F
  thus E  $\prec_c$  F
  proof (cases N Uer  $\mathcal{F}$   $\Gamma$  F rule: ord-res-8-can-produce.cases)
    case (1 L')
    thus ?thesis
    using no-undef-atom-le-max-lit-if-lt-false-clause[
      OF  $\Gamma$ -lower-set E-in E-false E-max-lit F-in  $\langle$ ord-res.is-maximal-lit L'
F>]
    by (metis F-in F-neq atm-of-in-atms-of-clsI linorder-cls.not-less-iff-gr-or-eq
      linorder-lit.is-maximal-in-mset-iff reflclp-iff
      trail-defined-lit-iff-trail-defined-atm)
    qed
  next
  assume ord-res-8-can-factorize N Uer  $\mathcal{F}$   $\Gamma$  F
  thus E  $\prec_c$  F
  proof (cases N Uer  $\mathcal{F}$   $\Gamma$  F rule: ord-res-8-can-factorize.cases)
    case (1 L')
    thus ?thesis
    using no-undef-atom-le-max-lit-if-lt-false-clause[
      OF  $\Gamma$ -lower-set E-in E-false E-max-lit F-in  $\langle$ ord-res.is-maximal-lit L'
F>]
    by (metis F-in F-neq atm-of-in-atms-of-clsI linorder-cls.not-less-iff-gr-or-eq
      linorder-lit.is-maximal-in-mset-iff reflclp-iff
      trail-defined-lit-iff-trail-defined-atm)
    qed
  qed
  qed

```

qed

hence  $C = \text{Some } E$   
**using** *match-hyps* **by** *metis*

**obtain**  $C'$  **where** *first-step7*:  $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) (U_{er}', \mathcal{F}, \Gamma', C')$

**proof** *atomize-elim*  
**have**  $(\text{Pos } A, \text{Some } D) \in \text{set } \Gamma$   
**using**  $\langle \text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } D) \rangle$  **by**  $(\text{metis } \text{map-of-SomeD})$

hence  $D$ -almost-false:  $\text{trail-false-clc } \Gamma \{ \#K \in \# D. K \neq \text{Pos } A \# \}$   
**using** *ord-res-7-invars-implies-propagated-clause-almost-false*  
 $\langle \text{ord-res-7-invars } N (U_{er}, \mathcal{F}, \Gamma, C) \rangle$  **by** *metis*

**have**  $\text{eres-false}$ :  $\text{trail-false-clc } \Gamma (\text{eres } D E)$   
**unfolding** *trail-false-clc-def*

**proof**  $(\text{intro ballI})$   
**fix**  $K :: 'f \text{ gliteral}$   
**assume**  $K \in \# \text{eres } D E$   
**hence**  $K \in \# D \wedge K \neq \text{Pos } A \vee K \in \# E$   
**using** *strong-lit-in-one-of-resolvents-if-in-eres[OF E-max-lit]* **by** *simp*

**thus**  $\text{trail-false-lit } \Gamma K$   
**proof**  $(\text{elim } \text{disjE } \text{conjE})$   
**assume**  $K \in \# D$  **and**  $K \neq \text{Pos } A$   
**thus**  $\text{trail-false-lit } \Gamma K$   
**using**  $D$ -almost-false **unfolding** *trail-false-clc-def* **by** *simp*

**next**  
**assume**  $K \in \# E$   
**thus**  $\text{trail-false-lit } \Gamma K$   
**using**  $E$ -false **unfolding** *trail-false-clc-def* **by** *simp*

qed

qed

**show**  $\exists C'. \text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) (U_{er}', \mathcal{F}, \Gamma', C')$

**proof**  $(\text{cases } \text{eres } D E = \{ \# \})$   
**case** *True*  
**hence**  $\bigwedge Ln. \forall K. \text{ord-res.is-maximal-lit } K (\text{eres } D E) \longrightarrow \text{atm-of } K \preceq_t$   
 $\text{atm-of } (\text{fst } Ln)$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**by** *simp*  
**hence**  $\Gamma' = \text{dropWhile } (\lambda Ln. \text{True}) \Gamma$   
**using** *step-hyps* **by** *meson*  
**hence**  $\Gamma' = []$   
**by** *simp*

**show** *?thesis*

**proof**  $(\text{intro exI})$   
**show**  $\text{ord-res-7 } N (U_{er}, \mathcal{F}, \Gamma, \text{Some } E) (U_{er}', \mathcal{F}, \Gamma', \text{Some } \{ \# \})$   
**using** *ord-res-7.resolution-bot[OF E-false E-max-lit]*

$\langle \text{map-of } \Gamma \text{ (Pos } A) = \text{Some (Some } D) \rangle \langle U_{er}' = \text{finsert (eres } D \ E) \ U_{er} \rangle$   
*True*  $\langle \Gamma' = [] \rangle$   
**by** *simp*  
**qed**  
**next**  
**case** *False*  
**then obtain** *K* **where** *eres-max-lit: ord-res.is-maximal-lit K (eres D E)*  
**using** *linorder-lit.ex-maximal-in-mset by presburger*  
**hence**  $\bigwedge Ln. (\forall K. \text{ord-res.is-maximal-lit } K \text{ (eres } D \ E) \longrightarrow \text{atm-of } K \preceq_t$   
*atm-of (fst Ln))*  $\longleftrightarrow$   
*atm-of } K \preceq\_t \text{atm-of (fst Ln))*  
**by** *(metis linorder-lit.Uniq-is-maximal-in-mset the1-equality')*  
**hence**  $\Gamma'$ -*eq:  $\Gamma' = \text{dropWhile } (\lambda Ln. \text{atm-of } K \preceq_t \text{atm-of (fst Ln)) } \Gamma$*   
**using** *step-hyps by meson*  
**show** *?thesis*  
**proof** *(cases K)*  
**case** *(Pos A<sub>K</sub>)*  
**hence** *K-pos: is-pos K*  
**by** *simp*  
  
**then show** *?thesis*  
**using** *ord-res-7.resolution-pos[OF E-false E-max-lit - - - False  $\Gamma'$ -eq*  
*eres-max-lit K-pos]*  
**using** *step-hyps by fastforce*  
**next**  
**case** *(Neg A<sub>K</sub>)*  
  
**hence** *K-neg: is-neg K*  
**by** *simp*  
  
**have** *trail-false-lit  $\Gamma$  K*  
**using** *eres-false eres-max-lit*  
**unfolding** *linorder-lit.is-maximal-in-mset-iff trail-false-cls-def by metis*  
  
**hence**  $\exists \text{opt. } (- K, \text{opt}) \in \text{set } \Gamma$   
**unfolding** *trail-false-lit-def by auto*  
  
**moreover have**  $\forall Ln \in \text{set } \Gamma. \text{is-neg (fst Ln)} = (\text{snd Ln} = \text{None})$   
**using** *invars7 by argo*  
  
**ultimately obtain** *C* **where**  $(- K, \text{Some } C) \in \text{set } \Gamma$   
**unfolding** *Neg uminus-Neg by fastforce*  
  
**hence** *map-of  $\Gamma$  (- K) = Some (Some C)*  
**proof** *(rule map-of-is-SomeI[rotated])*  
**show** *distinct (map fst  $\Gamma$ )*  
**using**  *$\Gamma$ -consistent*  
**by** *(metis distinct-lits-if-trail-consistent)*  
**qed**

```

then show ?thesis
  using ord-res-7.resolution-neg[OF E-false E-max-lit - - - False  $\Gamma'$ -eq
    eres-max-lit K-neg]
  using step-hyps by fastforce
qed
qed
qed

moreover have ord-res-7-invars  $N (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}')$ 
  using  $\langle C = \text{Some } E \rangle$  first-step7 match-hyps(3) ord-res-7-preserves-invars by
blast

ultimately obtain  $C''$  where
  following-steps7:  $(\text{ord-res-7 } N)^{**} (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}') (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'')$  and
  no-more-step7:  $(\nexists C'''. \text{ord-res-7 } N (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'') (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'''))$ 
  using MAGIC6 by metis

show ?thesis
proof (intro exI conjI)
  have  $(\text{ord-res-7 } N)^{++} (U_{er'}, \mathcal{F}, \Gamma, \mathcal{C}) (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'')$ 
    unfolding  $\langle C = \text{Some } E \rangle$ 
    using first-step7 following-steps7 by simp

  thus  $(\text{constant-context ord-res-7})^{++} S7 (N, U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'')$ 
    unfolding S7-def by (simp add: tranclp-constant-context)

  show ord-res-7-matches-ord-res-8  $(N, U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'') S8'$ 
    unfolding S8'-def  $\langle s8' = (U_{er'}, \mathcal{F}, \Gamma') \rangle$ 
  proof (intro ord-res-7-matches-ord-res-8.intros allI)
    show ord-res-7-invars  $N (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'')$ 
      using  $\langle \text{ord-res-7-invars } N (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}') \rangle$  following-steps7
      rtranclp-ord-res-7-preserves-ord-res-7-invars by blast

    show ord-res-8-invars  $N (U_{er'}, \mathcal{F}, \Gamma')$ 
      using invars-s8' step-hyps(1) by blast

  fix  $C :: 'f$  gclause
  show  $C'' = \text{Some } C \iff \text{is-least-nonskipped-clause } N U_{er'} \mathcal{F} \Gamma' C$ 
    using MAGIC5  $\langle \text{ord-res-7-invars } N (U_{er'}, \mathcal{F}, \Gamma', \mathcal{C}'') \rangle$  no-more-step7 by
metis
  qed
  qed
  qed
qed

theorem bisimulation-ord-res-7-ord-res-8:
  defines match  $\equiv \lambda-. \text{ord-res-7-matches-ord-res-8}$ 
  shows  $\exists (\text{MATCH} :: \text{nat} \times \text{nat} \Rightarrow 'f \text{ord-res-7-state} \Rightarrow 'f \text{ord-res-8-state} \Rightarrow \text{bool})$ 

```



```

 $\mathcal{R}_f \mathcal{R}_b.$ 
  bisimulation
    (constant-context ord-res-7) ord-res-7-final
    (constant-context ord-res-8) ord-res-8-final
    MATCH  $\mathcal{R}_f \mathcal{R}_b$ 
proof (rule ex-bisimulation-from-backward-simulation)
  show right-unique (constant-context ord-res-7)
    using right-unique-constant-context right-unique-ord-res-7 by metis
next
  show right-unique (constant-context ord-res-8)
    using right-unique-constant-context right-unique-ord-res-8 by metis
next
  show  $\forall S. \text{ord-res-7-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-7 } S S')$ 
    by (metis finished-def ord-res-7-semantics.final-finished)
next
  show  $\forall S. \text{ord-res-8-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-8 } S S')$ 
    by (metis finished-def ord-res-8-semantics.final-finished)
next
  show  $\forall i S7 S8. \text{match } i S7 S8 \longrightarrow \text{ord-res-7-final } S7 \longleftrightarrow \text{ord-res-8-final } S8$ 
    unfolding match-def
    using ord-res-7-final-iff-ord-res-8-final by metis
next
  show  $\forall i S7 S8. \text{match } i S7 S8 \longrightarrow$ 
    safe-state (constant-context ord-res-7) ord-res-7-final  $S7 \wedge$ 
    safe-state (constant-context ord-res-8) ord-res-8-final  $S8$ 
proof (intro allI impI conjI)
  fix  $i S7 S8$ 
  assume match: match  $i S7 S8$ 
  show safe-state (constant-context ord-res-7) ord-res-7-final  $S7$ 
    using match[unfolded match-def]
    using ord-res-7-safe-state-if-invars
    using ord-res-7-matches-ord-res-8.simps by auto

  show safe-state (constant-context ord-res-8) ord-res-8-final  $S8$ 
    using match[unfolded match-def]
    using ord-res-8-safe-state-if-invars
    using ord-res-7-matches-ord-res-8.simps by auto
qed
next
  show wfp ( $\lambda - . \text{False}$ )
    by simp
next
  show  $\forall i S7 S8 S8'. \text{match } i S7 S8 \longrightarrow \text{constant-context ord-res-8 } S8 S8' \longrightarrow$ 
     $(\exists i' S7'. (\text{constant-context ord-res-7})^{++} S7 S7' \wedge \text{match } i' S7' S8') \vee$ 
     $(\exists i'. \text{match } i' S7 S8' \wedge \text{False})$ 
    unfolding match-def
    using backward-simulation-between-7-and-8 by metis
qed

```

end

### 35 ORD-RES-9 (factorize when propagating)

**type-synonym** 'f ord-res-9-state =  
'f gclause fset × 'f gclause fset × 'f gclause fset × ('f gliteral × 'f gclause option)  
list

**context** simulation-SCLFOL-ground-ordered-resolution **begin**

**inductive** ord-res-8-matches-ord-res-9 :: 'f ord-res-8-state ⇒ 'f ord-res-9-state ⇒  
bool **where**

ord-res-8-invars  $N (U_{er}, \mathcal{F}, \Gamma) \implies$   
ord-res-8-matches-ord-res-9  $(N, U_{er}, \mathcal{F}, \Gamma) (N, U_{er}, \mathcal{F}, \Gamma)$

**lemma** ord-res-8-final-iff-ord-res-9-final:

**fixes** S8 S9

**assumes** match: ord-res-8-matches-ord-res-9 S8 S9

**shows** ord-res-8-final S8  $\longleftrightarrow$  ord-res-8-final S9

**using** match

**proof** (cases S8 S9 rule: ord-res-8-matches-ord-res-9.cases)

**case** (1 N U<sub>er</sub>  $\mathcal{F}$   $\Gamma$ )

**then show** ?thesis

**by** argo

qed

**lemma** backward-simulation-between-8-and-9:

**fixes** S8 S9 S9'

**assumes** match: ord-res-8-matches-ord-res-9 S8 S9 **and** step: constant-context  
ord-res-9 S9 S9'

**shows**  $\exists S8'. (\text{constant-context ord-res-8})^{++} S8 S8' \wedge \text{ord-res-8-matches-ord-res-9}$   
 $S8' S9'$

**using** match

**proof** (cases S8 S9 rule: ord-res-8-matches-ord-res-9.cases)

**case** match-hyps: (1 N U<sub>er</sub>  $\mathcal{F}$   $\Gamma$ )

**note** S8-def =  $\langle S8 = (N, U_{er}, \mathcal{F}, \Gamma) \rangle$

**note** S9-def =  $\langle S9 = (N, U_{er}, \mathcal{F}, \Gamma) \rangle$

**note** invars =  $\langle \text{ord-res-8-invars } N (U_{er}, \mathcal{F}, \Gamma) \rangle$

**obtain** s9' **where** S9'-def: S9' =  $(N, s9')$  **and** step': ord-res-9 N  $(U_{er}, \mathcal{F}, \Gamma)$   
s9'

**using** step unfolding S9-def

**using** constant-context.cases **by** blast

**have** ord-res-8 N  $(U_{er}, \mathcal{F}, \Gamma)$  s9'  $\vee$  (ord-res-8 N OO ord-res-8 N)  $(U_{er}, \mathcal{F}, \Gamma)$   
s9'

**using** step' ord-res-9-is-one-or-two-ord-res-9-steps **by** metis

```

hence steps8: (ord-res-8 N)++ (Uer, F, Γ) s9'
  by auto

show ?thesis
proof (intro exI conjI)
  show (constant-context ord-res-8)++ S8 (N, s9')
    unfolding S8-def
    using steps8 by (simp add: tranclp-constant-context)
next
  have ord-res-8-invars N s9'
    using invars steps8 tranclp-ord-res-8-preserves-invars by metis

  thus ord-res-8-matches-ord-res-9 (N, s9') S9'
    unfolding S9'-def
    by (metis ord-res-8-matches-ord-res-9.intros prod-cases3)
qed
qed

theorem bisimulation-ord-res-8-ord-res-9:
  defines match ≡ λ-. ord-res-8-matches-ord-res-9
  shows ∃ (MATCH :: nat × nat ⇒ 'f ord-res-8-state ⇒ 'f ord-res-9-state ⇒ bool)
  Rf Rb.
    bisimulation
      (constant-context ord-res-8) ord-res-8-final
      (constant-context ord-res-9) ord-res-8-final
      MATCH Rf Rb
proof (rule ex-bisimulation-from-backward-simulation)
  show right-unique (constant-context ord-res-8)
    using right-unique-constant-context right-unique-ord-res-8 by metis
next
  show right-unique (constant-context ord-res-9)
    using right-unique-constant-context right-unique-ord-res-9 by metis
next
  show ∀ S. ord-res-8-final S → (∃ S'. constant-context ord-res-8 S S')
    by (metis finished-def ord-res-8-semantics.final-finished)
next
  show ∀ S. ord-res-8-final S → (∃ S'. constant-context ord-res-9 S S')
    by (metis finished-def ord-res-9-semantics.final-finished)
next
  show ∀ i S8 S9. match i S8 S9 → ord-res-8-final S8 ↔ ord-res-8-final S9
    unfolding match-def
    using ord-res-8-final-iff-ord-res-9-final by metis
next
  show ∀ i S8 S9. match i S8 S9 →
    safe-state (constant-context ord-res-8) ord-res-8-final S8 ∧
    safe-state (constant-context ord-res-9) ord-res-8-final S9
proof (intro allI impI conjI)
  fix i S8 S9
  assume match: match i S8 S9

```

```

show safe-state (constant-context ord-res-8) ord-res-8-final S8
  using match[unfolded match-def]
  using ord-res-8-safe-state-if-invars
  using ord-res-8-matches-ord-res-9.simps by auto

show safe-state (constant-context ord-res-9) ord-res-8-final S9
  using match[unfolded match-def]
  using ord-res-9-safe-state-if-invars
  using ord-res-8-matches-ord-res-9.simps by auto
qed
next
  show wfp ( $\lambda - . \text{False}$ )
    by simp
next
  show  $\forall i S8 S9 S9'. \text{match } i S8 S9 \longrightarrow \text{constant-context ord-res-9 } S9 S9' \longrightarrow$ 
     $(\exists i' S8'. (\text{constant-context ord-res-8})^{++} S8 S8' \wedge \text{match } i' S8' S9') \vee$ 
     $(\exists i'. \text{match } i' S8 S9' \wedge \text{False})$ 
    unfolding match-def
    using backward-simulation-between-8-and-9 by metis
qed
end

```

## 36 ORD-RES-10 (propagate iff a conflict is produced)

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**inductive** *ord-res-9-matches-ord-res-10* :: '*f ord-res-9-state*  $\Rightarrow$  '*f ord-res-10-state*  
 $\Rightarrow$  *bool* **where**

```

  ord-res-8-invars  $N (U_{er}, \mathcal{F}, \Gamma_9) \Longrightarrow$ 
  ord-res-10-invars  $N (U_{er}, \mathcal{F}, \Gamma_{10}) \Longrightarrow$ 
  list-all2  $(\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \Longrightarrow$ 
  list-all2  $(\lambda x y. \text{snd } y \neq \text{None} \longrightarrow x = y) \Gamma_9 \Gamma_{10} \Longrightarrow$ 
  ord-res-9-matches-ord-res-10  $(N, U_{er}, \mathcal{F}, \Gamma_9) (N, U_{er}, \mathcal{F}, \Gamma_{10})$ 

```

**lemma** *ord-res-9-final-iff-ord-res-10-final*:

**fixes** *S9 S10*

**assumes** *match*: *ord-res-9-matches-ord-res-10 S9 S10*

**shows** *ord-res-8-final S9*  $\longleftrightarrow$  *ord-res-8-final S10*

**using** *match*

**proof** (*cases S9 S10 rule: ord-res-9-matches-ord-res-10.cases*)

**case** *match-hyps*:  $(1 N U_{er} \mathcal{F} \Gamma_9 \Gamma_{10})$

**then show** *?thesis*

**using** *trail-atms-eq-trail-atms-if-same-lits*[*OF*  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$ ]

**using** *trail-false-cls-eq-trail-false-cls-if-same-lits*[*OF*  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$ ]

**unfolding** *ord-res-8-final.simps*  
**by** *simp*  
**qed**

**lemma** *backward-simulation-between-9-and-10*:  
**fixes** *S9 S10 S10'*  
**assumes**  
*match: ord-res-9-matches-ord-res-10 S9 S10 and*  
*step: constant-context ord-res-10 S10 S10'*  
**shows**  $\exists S9'. (\text{constant-context } \text{ord-res-9})^{++} S9 S9' \wedge \text{ord-res-9-matches-ord-res-10 } S9' S10'$   
**using** *match*  
**proof** (*cases S9 S10 rule: ord-res-9-matches-ord-res-10.cases*)  
**case** *match-hyps: (1 N U<sub>er</sub>  $\mathcal{F}$   $\Gamma_9$   $\Gamma_{10}$ )*

**note** *S9-def* =  $\langle S9 = (N, U_{er}, \mathcal{F}, \Gamma_9) \rangle$   
**note** *S10-def* =  $\langle S10 = (N, U_{er}, \mathcal{F}, \Gamma_{10}) \rangle$   
**note** *invars9* =  $\langle \text{ord-res-8-invars } N (U_{er}, \mathcal{F}, \Gamma_9) \rangle$   
**note** *invars10* =  $\langle \text{ord-res-10-invars } N (U_{er}, \mathcal{F}, \Gamma_{10}) \rangle$

**have** *trail-atms*  $\Gamma_9 = \text{trail-atms } \Gamma_{10}$   
**using**  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$  *trail-atms-eq-trail-atms-if-same-lits*  
**by** *metis*

**have** *trail-false-lit*  $\Gamma_9 = \text{trail-false-lit } \Gamma_{10}$   
**using**  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$  *trail-false-lit-eq-trail-false-lit-if-same-lits*  
**by** *metis*

**have** *trail-false-cls*  $\Gamma_9 = \text{trail-false-cls } \Gamma_{10}$   
**using**  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$  *trail-false-cls-eq-trail-false-cls-if-same-lits*  
**by** *metis*

**have** *trail-defined-lit*  $\Gamma_9 = \text{trail-defined-lit } \Gamma_{10}$   
**using**  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$   
*trail-defined-lit-eq-trail-defined-lit-if-same-lits* **by** *metis*

**have** *trail-defined-cls*  $\Gamma_9 = \text{trail-defined-cls } \Gamma_{10}$   
**using**  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$   
*trail-defined-cls-eq-trail-defined-cls-if-same-lits* **by** *metis*

**have** *clause-could-propagate*  $\Gamma_9 = \text{clause-could-propagate } \Gamma_{10}$   
**unfolding** *clause-could-propagate-def*  
**unfolding**  $\langle \text{trail-defined-lit } \Gamma_9 = \text{trail-defined-lit } \Gamma_{10} \rangle$   
**unfolding**  $\langle \text{trail-false-cls } \Gamma_9 = \text{trail-false-cls } \Gamma_{10} \rangle$  ..

**have**  $\Gamma_9$ -*sorted*: *sorted-wrt*  $(\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)) \Gamma_9$   
**using** *invars9[unfolded ord-res-8-invars-def trail-is-sorted-def, simplified]* **by**  
*argo*

```

obtain  $s10'$  where  $S10' = (N, s10')$  and  $step10: ord-res-10\ N\ (U_{er}, \mathcal{F}, \Gamma_{10})$ 
 $s10'$ 
  using  $step\ unfolding\ S10-def\ by\ (auto\ elim: constant-context.cases)$ 

show  $?thesis$ 
  using  $step10$ 
proof  $(cases\ N\ (U_{er}, \mathcal{F}, \Gamma_{10})\ s10'\ rule: ord-res-10.cases)$ 
  case  $step-hyps: (decide-neg\ A\ \Gamma_{10}')$ 

define  $\Gamma_9'$  where
   $\Gamma_9' = (Neg\ A, None) \# \Gamma_9$ 

show  $?thesis$ 
proof  $(intro\ exI\ conjI)$ 
  have  $step9: ord-res-9\ N\ (U_{er}, \mathcal{F}, \Gamma_9)\ (U_{er}, \mathcal{F}, \Gamma_9')$ 
  proof  $(rule\ ord-res-9.decide-neg)$ 
    show  $\neg (\exists C | \in |iefac\ \mathcal{F}\ | \uparrow (N\ |\cup| U_{er}). trail-false-cls\ \Gamma_9\ C)$ 
    using  $step-hyps\ \langle trail-false-cls\ \Gamma_9 = trail-false-cls\ \Gamma_{10} \rangle$  by  $argo$ 
  next
    show  $linorder-trm.is-least-in-fset$ 
     $\{ |A_2 | \in | atms-of-cls (N\ |\cup| U_{er}). \forall A_1 | \in | trail-atms\ \Gamma_9. A_1 \prec_t A_2 | \}$   $A$ 
    using  $step-hyps\ \langle trail-atms\ \Gamma_9 = trail-atms\ \Gamma_{10} \rangle$  by  $metis$ 
  next
    show  $\neg (\exists C | \in |iefac\ \mathcal{F}\ | \uparrow (N\ |\cup| U_{er}). clause-could-propagate\ \Gamma_9\ C\ (Pos\ A))$ 
    using  $step-hyps\ \langle clause-could-propagate\ \Gamma_9 = clause-could-propagate\ \Gamma_{10} \rangle$ 
by  $metis$ 
  next
    show  $\Gamma_9' = (Neg\ A, None) \# \Gamma_9$ 
    using  $\Gamma_9'-def$  .
  qed

thus  $(constant-context\ ord-res-9)^{++}\ S9\ (N, U_{er}, \mathcal{F}, \Gamma_9')$ 
  unfolding  $S9-def\ by\ (auto\ intro: constant-context.intros)$ 

show  $ord-res-9-matches-ord-res-10\ (N, U_{er}, \mathcal{F}, \Gamma_9')\ S10'$ 
  unfolding  $\langle S10' = (N, s10') \rangle \langle s10' = (U_{er}, \mathcal{F}, \Gamma_{10}') \rangle$ 
proof  $(rule\ ord-res-9-matches-ord-res-10.intros)$ 
  show  $ord-res-8-invars\ N\ (U_{er}, \mathcal{F}, \Gamma_9')$ 
  using  $invars9\ step9\ ord-res-9-preserves-invars\ by\ metis$ 
next
  show  $ord-res-10-invars\ N\ (U_{er}, \mathcal{F}, \Gamma_{10}')$ 
  using  $invars10\ step10\ ord-res-10-preserves-invars\ \langle s10' = (U_{er}, \mathcal{F}, \Gamma_{10}') \rangle$ 
by  $metis$ 
next
  show  $list-all2\ (\lambda x\ y. fst\ x = fst\ y)\ \Gamma_9'\ \Gamma_{10}'$ 
  unfolding  $\langle \Gamma_9' = (Neg\ A, None) \# \Gamma_9 \rangle \langle \Gamma_{10}' = (Neg\ A, None) \# \Gamma_{10} \rangle$ 
  using  $\langle list-all2\ (\lambda x\ y. fst\ x = fst\ y)\ \Gamma_9\ \Gamma_{10} \rangle$  by  $simp$ 
next

```

```

    show list-all2 ( $\lambda x y. \text{snd } y \neq \text{None} \longrightarrow x = y$ )  $\Gamma_9' \Gamma_{10}'$ 
      unfolding  $\langle \Gamma_9' = (\text{Neg } A, \text{None}) \# \Gamma_9 \rangle \langle \Gamma_{10}' = (\text{Neg } A, \text{None}) \# \Gamma_{10} \rangle$ 
      using  $\langle \text{list-all2 } (\lambda x y. \text{snd } y \neq \text{None} \longrightarrow x = y) \Gamma_9 \Gamma_{10} \rangle$ 
      by simp
  qed
next
case step-hyps: (decide-pos  $A \ C \ \Gamma_{10}' \ \mathcal{F}'$ )

define  $\Gamma_9'$  where
   $\Gamma_9' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma_9$ 

show ?thesis
proof (intro exI conjI)
  have step9: ord-res-9  $N \ (U_{er}, \mathcal{F}, \Gamma_9) \ (U_{er}, \mathcal{F}', \Gamma_9')$ 
  proof (rule ord-res-9.propagate)
    show  $\neg (\exists C \in |\text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). \text{trail-false-cls } \Gamma_9 \ C)$ 
      using step-hyps  $\langle \text{trail-false-cls } \Gamma_9 = \text{trail-false-cls } \Gamma_{10} \rangle$  by argo
  next
  show linorder-trm.is-least-in-fset
     $\{ | A_2 \in | \text{atms-of-clss } (N \ | \cup \ | U_{er}). \forall A_1 \in | \text{trail-atms } \Gamma_9. A_1 \prec_t A_2 | \} A$ 
    using step-hyps  $\langle \text{trail-atms } \Gamma_9 = \text{trail-atms } \Gamma_{10} \rangle$  by metis
  next
  show linorder-cls.is-least-in-fset
     $\{ | C \in | \text{iefac } \mathcal{F} \ | \uparrow (N \ | \cup \ | U_{er}). \text{clause-could-propagate } \Gamma_9 \ C \ (\text{Pos } A) | \} C$ 
    using step-hyps  $\langle \text{clause-could-propagate } \Gamma_9 = \text{clause-could-propagate } \Gamma_{10} \rangle$ 
by metis
  next
  show  $\Gamma_9' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma_9$ 
    using  $\Gamma_9'$ -def .
  next
  show  $\mathcal{F}' = (\text{if } \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) \ C \ \text{then } \mathcal{F} \ \text{else } \text{finsert } C \ \mathcal{F})$ 
    using step-hyps by argo
  qed

thus (constant-context ord-res-9)++  $S9 \ (N, U_{er}, \mathcal{F}', \Gamma_9')$ 
  unfolding  $S9$ -def by (auto intro: constant-context.intros)

show ord-res-9-matches-ord-res-10  $(N, U_{er}, \mathcal{F}', \Gamma_9') \ S10'$ 
  unfolding  $\langle S10' = (N, s10') \rangle \langle s10' = (U_{er}, \mathcal{F}', \Gamma_{10}') \rangle$ 
  proof (rule ord-res-9-matches-ord-res-10.intros)
    show ord-res-8-invars  $N \ (U_{er}, \mathcal{F}', \Gamma_9')$ 
      using invars9 step9 ord-res-9-preserves-invars by metis
  next
  show ord-res-10-invars  $N \ (U_{er}, \mathcal{F}', \Gamma_{10}')$ 
    using invars10 step10 ord-res-10-preserves-invars  $\langle s10' = (U_{er}, \mathcal{F}', \Gamma_{10}') \rangle$ 
by metis
  next

```

```

    show list-all2 (λx y. fst x = fst y) Γ9' Γ10'
      unfolding ⟨Γ9' = (Pos A, Some (efac C)) # Γ9⟩ ⟨Γ10' = (Pos A, None)
# Γ10⟩
      using ⟨list-all2 (λx y. fst x = fst y) Γ9 Γ10⟩ by simp
    next
      show list-all2 (λx y. snd y ≠ None → x = y) Γ9' Γ10'
      unfolding ⟨Γ9' = (Pos A, Some (efac C)) # Γ9⟩ ⟨Γ10' = (Pos A, None)
# Γ10⟩
      using ⟨list-all2 (λx y. snd y ≠ None → x = y) Γ9 Γ10⟩
      by simp
    qed
  qed
next
case step-hyps: (propagate A C Γ10' F')

define Γ9' where
  Γ9' = (Pos A, Some (efac C)) # Γ9

show ?thesis
proof (intro exI conjI)
  have step9: ord-res-9 N (Uer, F, Γ9) (Uer, F', Γ9')
  proof (rule ord-res-9.propagate)
    show ¬ (∃ C |∈| iefac F |↑| (N |∪| Uer). trail-false-cls Γ9 C)
      using step-hyps ⟨trail-false-cls Γ9 = trail-false-cls Γ10⟩ by argo
    next
      show linorder-trm.is-least-in-fset
        { |A2 |∈| atms-of-cls (N |∪| Uer). ∀ A1 |∈| trail-atms Γ9. A1 <t A2 |} A
        using step-hyps ⟨trail-atms Γ9 = trail-atms Γ10⟩ by metis
    next
      show linorder-cls.is-least-in-fset
        { |C |∈| iefac F |↑| (N |∪| Uer). clause-could-propagate Γ9 C (Pos A) |} C
        using step-hyps ⟨clause-could-propagate Γ9 = clause-could-propagate Γ10⟩
  by metis
  next
    show Γ9' = (Pos A, Some (efac C)) # Γ9
      using Γ9'-def .
  next
    show F' = (if ord-res.is-strictly-maximal-lit (Pos A) C then F else finsert
C F)
      using step-hyps by argo
  qed

thus (constant-context ord-res-9)++ S9 (N, Uer, F', Γ9')
  unfolding S9-def by (auto intro: constant-context.intros)

show ord-res-9-matches-ord-res-10 (N, Uer, F', Γ9') S10'
  unfolding ⟨S10' = (N, s10')⟩ ⟨s10' = (Uer, F', Γ10')⟩
  proof (rule ord-res-9-matches-ord-res-10.intros)
    show ord-res-8-invars N (Uer, F', Γ9')

```



```

    using invars9 step9 ord-res-9-preserves-invars by metis
  next
    show ord-res-10-invars N (Uer, F', Γ10')
    using invars10 step10 ord-res-10-preserves-invars ⟨s10' = (Uer, F', Γ10')⟩
  by metis
  next
    show list-all2 (λx y. fst x = fst y) Γ9' Γ10'
    unfolding ⟨Γ9' = (Pos A, Some (efac C)) # Γ9⟩ ⟨Γ10' = (Pos A, Some
(efac C)) # Γ10⟩
    using ⟨list-all2 (λx y. fst x = fst y) Γ9 Γ10⟩ by simp
  next
    show list-all2 (λx y. snd y ≠ None → x = y) Γ9' Γ10'
    unfolding ⟨Γ9' = (Pos A, Some (efac C)) # Γ9⟩ ⟨Γ10' = (Pos A, Some
(efac C)) # Γ10⟩
    using ⟨list-all2 (λx y. snd y ≠ None → x = y) Γ9 Γ10⟩
    by simp
  qed
  qed
  next
  case step-hyps: (resolution D A C Uer' Γ10')

  have ∀ Ln Γ'. Γ10 = Ln # Γ' →
    (snd Ln ≠ None) = fBex (iefac F |↑ (N |∪| Uer)) (trail-false-cls Γ10) ∧
    (∀ x∈set Γ'. snd x = None)
  using invars10 by (simp add: ord-res-10-invars.simps)

  then obtain Γ10'' where Γ10 = (Pos A, Some C) # Γ10''
  using ⟨map-of Γ10 (Pos A) = Some (Some C)⟩
  by (metis list.set-cases map-of-SomeD not-Some-eq snd-conv)

  then obtain Γ9'' where Γ9 = (Pos A, Some C) # Γ9''
  using ⟨list-all2 (λx y. snd y ≠ None → x = y) Γ9 Γ10⟩
  by (smt (verit, best) list-all2-Cons2 option.discI snd-conv)

  define Γ9' where
    Γ9' = dropWhile (λLn. ∀ K. ord-res.is-maximal-lit K (eres C D) →
    atm-of K ≼t atm-of (fst Ln)) Γ9

  show ?thesis
  proof (intro exI conjI)
    have step9: ord-res-9 N (Uer, F, Γ9) (Uer', F, Γ9')
    proof (rule ord-res-9.resolution)
      show linorder-cls.is-least-in-fset (ffilter (trail-false-cls Γ9) (iefac F |↑ (N
|∪| Uer))) D
      using step-hyps ⟨trail-false-cls Γ9 = trail-false-cls Γ10⟩ by argo
    next
      show ord-res.is-maximal-lit (Neg A) D
      using step-hyps by argo
    next
  end

```

```

show map-of  $\Gamma_9$  (Pos A) = Some (Some C)
  unfolding  $\langle \Gamma_9 = (\text{Pos } A, \text{Some } C) \# \Gamma_9'' \rangle$  by simp
next
show  $U_{er}' = \text{finsert } (\text{eres } C D) U_{er}$ 
  using step-hyps by argo
next
show  $\Gamma_9' = \text{dropWhile } (\lambda Ln. \forall K. \text{ord-res.is-maximal-lit } K (\text{eres } C D) \longrightarrow$ 
   $\text{atm-of } K \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma_9$ 
  using  $\Gamma_9'$ -def .
qed

thus (constant-context ord-res-9)++ S9 (N,  $U_{er}'$ ,  $\mathcal{F}$ ,  $\Gamma_9'$ )
  unfolding S9-def by (auto intro: constant-context.intros)

show ord-res-9-matches-ord-res-10 (N,  $U_{er}'$ ,  $\mathcal{F}$ ,  $\Gamma_9'$ ) S10'
  unfolding  $\langle S10' = (N, s10') \rangle$   $\langle s10' = (U_{er}', \mathcal{F}, \Gamma_{10}') \rangle$ 
proof (rule ord-res-9-matches-ord-res-10.intros)
  show ord-res-8-invars N ( $U_{er}'$ ,  $\mathcal{F}$ ,  $\Gamma_9'$ )
    using invars9 step9 ord-res-9-preserves-invars by metis
next
show ord-res-10-invars N ( $U_{er}'$ ,  $\mathcal{F}$ ,  $\Gamma_{10}'$ )
  using invars10 step10 ord-res-10-preserves-invars  $\langle s10' = (U_{er}', \mathcal{F}, \Gamma_{10}') \rangle$ 
by metis
next
define P :: 'f gterm literal  $\times$  'f gterm literal multiset option  $\Rightarrow$  bool where
  P  $\equiv$   $\lambda Ln. \forall K. \text{ord-res.is-maximal-lit } K (\text{eres } C D) \longrightarrow \text{atm-of } K \preceq_t \text{atm-of}$ 
  (fst Ln)

have length (takeWhile P  $\Gamma_9$ ) = length (takeWhile P  $\Gamma_{10}$ )
  using  $\langle \text{list-all2 } (\lambda x y. \text{fst } x = \text{fst } y) \Gamma_9 \Gamma_{10} \rangle$ 
proof (induction  $\Gamma_9 \Gamma_{10}$  rule: list.rel-induct)
  case Nil
  show ?case
    by simp
next
case (Cons x xs y ys)
  then show ?case
    by (simp add: P-def)
qed

moreover have  $\Gamma_9 = \text{takeWhile } P \Gamma_9 @ \Gamma_9'$ 
  unfolding takeWhile-dropWhile-id
  unfolding P-def  $\langle \Gamma_9' = \text{dropWhile } - \Gamma_9 \rangle$  by simp

moreover have  $\Gamma_{10} = \text{takeWhile } P \Gamma_{10} @ \Gamma_{10}'$ 
  unfolding takeWhile-dropWhile-id
  unfolding P-def  $\langle \Gamma_{10}' = \text{dropWhile } - \Gamma_{10} \rangle$  by simp

ultimately have  $\bigwedge Q. \text{list-all2 } Q \Gamma_9 \Gamma_{10} \longleftrightarrow$ 

```

(*list-all2*  $Q$  (*takeWhile*  $P$   $\Gamma_9$ ) (*takeWhile*  $P$   $\Gamma_{10}$ )  $\wedge$  *list-all2*  $Q$   $\Gamma_9'$   $\Gamma_{10}'$ )  
**using** *list-all2-append* **by** *metis*

**thus**

*list-all2* ( $\lambda x y. \text{fst } x = \text{fst } y$ )  $\Gamma_9' \Gamma_{10}'$   
*list-all2* ( $\lambda x y. \text{snd } y \neq \text{None} \longrightarrow x = y$ )  $\Gamma_9' \Gamma_{10}'$   
**unfolding** *atomize-conj*  
**using**  $\langle$ *list-all2* ( $\lambda x y. \text{fst } x = \text{fst } y$ )  $\Gamma_9 \Gamma_{10}$  $\rangle$   
**using**  $\langle$ *list-all2* ( $\lambda x y. \text{snd } y \neq \text{None} \longrightarrow x = y$ )  $\Gamma_9 \Gamma_{10}$  $\rangle$   
**by** (*simp only*.)

**qed**

**qed**

**qed**

**qed**

**theorem** *bisimulation-ord-res-9-ord-res-10*:

**defines** *match*  $\equiv \lambda \cdot. \text{ord-res-9-matches-ord-res-10}$

**shows**  $\exists$  (*MATCH* ::  $\text{nat} \times \text{nat} \Rightarrow 'f \text{ord-res-8-state} \Rightarrow 'f \text{ord-res-9-state} \Rightarrow \text{bool}$ )  
 $\mathcal{R}_f \mathcal{R}_b$ .

*bisimulation*

(*constant-context ord-res-9*) *ord-res-8-final*

(*constant-context ord-res-10*) *ord-res-8-final*

*MATCH*  $\mathcal{R}_f \mathcal{R}_b$

**proof** (*rule ex-bisimulation-from-backward-simulation*)

**show** *right-unique* (*constant-context ord-res-9*)

**using** *right-unique-constant-context right-unique-ord-res-9* **by** *metis*

**next**

**show** *right-unique* (*constant-context ord-res-10*)

**using** *right-unique-constant-context right-unique-ord-res-10* **by** *metis*

**next**

**show**  $\forall S. \text{ord-res-8-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-9 } S S')$

**by** (*metis finished-def ord-res-9-semantics.final-finished*)

**next**

**show**  $\forall S. \text{ord-res-8-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-10 } S S')$

**by** (*metis finished-def ord-res-10-semantics.final-finished*)

**next**

**show**  $\forall i S9 S10. \text{match } i S9 S10 \longrightarrow \text{ord-res-8-final } S9 \longleftrightarrow \text{ord-res-8-final } S10$

**unfolding** *match-def*

**using** *ord-res-9-final-iff-ord-res-10-final* **by** *metis*

**next**

**show**  $\forall i S9 S10. \text{match } i S9 S10 \longrightarrow$

*safe-state* (*constant-context ord-res-9*) *ord-res-8-final*  $S9 \wedge$

*safe-state* (*constant-context ord-res-10*) *ord-res-8-final*  $S10$

**proof** (*intro allI impI conjI*)

**fix**  $i S9 S10$

**assume** *match*: *match*  $i S9 S10$

**show** *safe-state* (*constant-context ord-res-9*) *ord-res-8-final*  $S9$

**using** *match[unfolded match-def]*

**using** *ord-res-9-safe-state-if-invars*

```

using ord-res-9-matches-ord-res-10.simps by auto

show safe-state (constant-context ord-res-10) ord-res-8-final S10
using match[unfolded match-def]
using ord-res-10-safe-state-if-invars
using ord-res-9-matches-ord-res-10.simps by auto
qed
next
show wfp (λ- -. False)
by simp
next
show ∀ i S9 S10 S10'. match i S9 S10 → constant-context ord-res-10 S10 S10'
→
(∃ i' S9'. (constant-context ord-res-9)++ S9 S9' ∧ match i' S9' S10') ∨
(∃ i'. match i' S9 S10' ∧ False)
unfolding match-def
using backward-simulation-between-9-and-10 by metis
qed

end

```

### 37 ORD-RES-11 (SCL strategy)

**context** simulation-SCLFOL-ground-ordered-resolution **begin**

```

inductive ord-res-10-matches-ord-res-11 :: 'f ord-res-10-state ⇒ 'f ord-res-11-state
⇒ bool where
ord-res-10-invars N (Uer10, F, Γ) ⇒
ord-res-11-invars N (Uer11, F, Γ, C) ⇒
Uer11 = Uer10 - {|#}| ⇒
if {#} |∈| iefac F |! (N |∪| Uer10) then Γ = [] ∧ C = Some {#} else C =
None ⇒
ord-res-10-matches-ord-res-11 (N, Uer10, F, Γ) (N, Uer11, F, Γ, C)

```

**lemma** ord-res-10-final-iff-ord-res-11-final:

```

fixes S10 S11
assumes match: ord-res-10-matches-ord-res-11 S10 S11
shows ord-res-8-final S10 ↔ ord-res-11-final S11
using match
proof (cases S10 S11 rule: ord-res-10-matches-ord-res-11.cases)
case match-hyps: (1 N Uer10 F Γ Uer11 C)
show ?thesis
proof (rule iffI)
assume ord-res-8-final S10
thus ord-res-11-final S11
unfolding ⟨S10 = →⟩
proof (cases (N, Uer10, F, Γ) rule: ord-res-8-final.cases)
case model-found
hence {#} |∉| iefac F |! (N |∪| Uer10)

```

```

    using trail-false-cls-mempty by blast
  hence C = None
    using match-hyps by argo
  moreover have  $U_{er11} = U_{er10}$ 
    using match-hyps
    by (metis ‹{#} |∉| iefac F |∧| (N |∪| Uer10)› fimage-eqI finsert-fminus1
finsert-iff
    fminus-finsert-absorb funionI2 iefac-mempty)
  ultimately show ?thesis
    unfolding ‹S11 = -›
    using model-found
    using ord-res-11-final.model-found
    by metis
next
  case contradiction-found
  hence  $\Gamma = [] \wedge C = \text{Some } \{ \# \}$ 
    using match-hyps by argo
  thus ?thesis
    unfolding ‹S11 = -›
    using ord-res-11-final.contradiction-found by metis
qed
next
  assume ord-res-11-final S11
  thus ord-res-8-final S10
    unfolding ‹S11 = -›
  proof (cases (N, Uer11, F,  $\Gamma$ , C) rule: ord-res-11-final.cases)
  case model-found
  show ?thesis
    unfolding ‹S10 = -›
  proof (rule ord-res-8-final.model-found)
  show  $\neg (\exists A | \in | \text{atms-of-cls } (N | \cup | U_{er10}). A | \notin | \text{trail-atms } \Gamma)$ 
    by (metis (no-types, lifting) fimage-iff fminus-finsert-absorb fminus-idemp
funionCI
    iefac-mempty local.model-found(1,2) match-hyps(5,6) option.simps(3))
  next
  show  $\neg (\exists C | \in | \text{iefac } \mathcal{F} | \wedge | (N | \cup | U_{er10}). \text{trail-false-cls } \Gamma C)$ 
    by (metis finsertCI finsert-fminus1 fminus-finsert-absorb funionI2 iefac-mempty
    local.model-found(1,3) match-hyps(5,6) option.simps(3) rev-fimage-eqI)
  qed
next
  case contradiction-found
  show ?thesis
    unfolding ‹S10 = -›
  proof (rule ord-res-8-final.contradiction-found)
  show {#} | ∈ | iefac F | ∧ | (N | ∪ | Uer10)
    using match-hyps contradiction-found
    by auto
  qed
qed

```

qed  
qed

**lemma** *forward-simulation-between-10-and-11*:

**fixes**  $S10\ S11\ S10'$

**assumes**

*match*: *ord-res-10-matches-ord-res-11*  $S10\ S11$  **and**

*step*: *constant-context ord-res-10*  $S10\ S10'$

**shows**  $\exists S11'. (\text{constant-context ord-res-11})^{++}\ S11\ S11' \wedge \text{ord-res-10-matches-ord-res-11}\ S10'\ S11'$

**using** *match*

**proof** (*cases S10 S11 rule: ord-res-10-matches-ord-res-11.cases*)

**case** *match-hyps*:  $(1\ N\ U_{er10}\ \mathcal{F}\ \Gamma\ U_{er11}\ \mathcal{C})$

**note**  $S10\text{-def} = \langle S10 = (N, U_{er10}, \mathcal{F}, \Gamma) \rangle$

**note**  $S11\text{-def} = \langle S11 = (N, U_{er11}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle$

**note**  $\text{invars10} = \langle \text{ord-res-10-invars}\ N\ (U_{er10}, \mathcal{F}, \Gamma) \rangle$

**note**  $\text{invars11} = \langle \text{ord-res-11-invars}\ N\ (U_{er11}, \mathcal{F}, \Gamma, \mathcal{C}) \rangle$

**have** *mempty-not-in-if-no-false-cls*:  $\{\#\} \mid \notin \mid \text{iefac}\ \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})$

**if**  $\neg \text{fBex} (\text{iefac}\ \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})) (\text{trail-false-cls}\ \Gamma)$

**using** *that by force*

**have** *C-eq-None-if-no-false-cls*:  $\mathcal{C} = \text{None}$

**if**  $\neg \text{fBex} (\text{iefac}\ \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})) (\text{trail-false-cls}\ \Gamma)$

**using** *match-hyps mempty-not-in-if-no-false-cls[OF that] by argo*

**have** *mempty-not-in-if*:  $\{\#\} \mid \notin \mid N \mid \cup \mid U_{er10}$

**if**  $\neg \text{fBex} (\text{iefac}\ \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})) (\text{trail-false-cls}\ \Gamma)$

**using** *that*

**by** (*metis (no-types, opaque-lifting) fimageI iefac-mempty trail-false-cls-mempty*)

**have** *U<sub>er11</sub>-eq-U<sub>er10</sub>-if*:  $U_{er11} = U_{er10}$

**if**  $\neg \text{fBex} (\text{iefac}\ \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})) (\text{trail-false-cls}\ \Gamma)$

**using** *mempty-not-in-if[OF that]  $\langle U_{er11} = U_{er10} \mid - \mid \{\{\#\}\} \rangle$*

**by** (*metis (no-types, opaque-lifting) finsertI1 finsert-ident fminusD2 funionCI funion-fempty-right funion-finsert-right funion-fminus-cancel2*)

**obtain**  $s10'$  **where**  $S10' = (N, s10')$  **and** *step10*: *ord-res-10*  $N\ (U_{er10}, \mathcal{F}, \Gamma)$

**using** *step unfolding S10-def by (auto elim: constant-context.cases)*

**show** *?thesis*

**using** *step10*

**proof** (*cases N (U<sub>er10</sub>, F, Γ) s10' rule: ord-res-10.cases*)

**case** *step-hyps*: (*decide-neg A Γ'*)

**have**  $\mathcal{C} = \text{None}$

**using** *step-hyps C-eq-None-if-no-false-cls by argo*

```

have {#} | $\notin$ |  $N \mid \cup \mid U_{er10}$ 
  using step-hyps mempty-not-in-if by argo

have  $U_{er11} = U_{er10}$ 
  using step-hyps U_{er11}-eq-U_{er10}-if by argo

show ?thesis
proof (intro exI conjI)
  have step11: ord-res-11  $N (U_{er11}, \mathcal{F}, \Gamma, None) (U_{er11}, \mathcal{F}, \Gamma', None)$ 
  proof (rule ord-res-11.decide-neg)
    show  $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er11}). \text{trail-false-cls } \Gamma C)$ 
      using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show linorder-trm.is-least-in-fset
         $\{ \mid A_2 \mid \in \mid \text{atms-of-clss } (N \mid \cup \mid U_{er11}). \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2 \mid \} A$ 
        using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show  $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er11}). \text{clause-could-propagate } \Gamma C (Pos$ 
A))
        using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show  $\Gamma' = (Neg A, None) \# \Gamma$ 
        using step-hyps by argo
  qed

thus (constant-context ord-res-11)++ S11  $(N, U_{er11}, \mathcal{F}, \Gamma', None)$ 
  unfolding S11-def  $\langle C = None \rangle$  by (auto intro: constant-context.intros)

show ord-res-10-matches-ord-res-11 S10'  $(N, U_{er11}, \mathcal{F}, \Gamma', None)$ 
  unfolding  $\langle S10' = (N, s10') \rangle \langle s10' = \rightarrow \rangle$ 
proof (rule ord-res-10-matches-ord-res-11.intros)
  show ord-res-10-invars  $N (U_{er10}, \mathcal{F}, \Gamma')$ 
    using step10  $\langle s10' = \rightarrow \rangle$  invars10 ord-res-10-preserves-invars by metis
  next
    show ord-res-11-invars  $N (U_{er11}, \mathcal{F}, \Gamma', None)$ 
      using step11 invars11  $\langle C = None \rangle$  ord-res-11-preserves-invars by metis
  next
    show  $U_{er11} = U_{er10} \mid - \mid \{ \mid \{ \# \} \mid \}$ 
      unfolding  $\langle U_{er11} = U_{er10} \rangle$ 
      using  $\langle \{ \# \} \mid \notin \mid N \mid \cup \mid U_{er10} \rangle$  by simp
  next
    have  $\{ \# \} \mid \notin \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})$ 
      using  $\langle \{ \# \} \mid \notin \mid N \mid \cup \mid U_{er10} \rangle$  by (simp add: iefac-def)
    thus if  $\{ \# \} \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow (N \mid \cup \mid U_{er10})$  then  $\Gamma' = [] \wedge None = Some \{ \# \}$ 
else  $None = None$ 
      by argo
  qed
qed

```

```

next
  case step-hyps: (decide-pos A C Γ' F')

  have C = None
    using step-hyps C-eq-None-if-no-false-cl by argo

  have {#} |≠| N |∪| Uer10
    using step-hyps mempty-not-in-if by argo

  have Uer11 = Uer10
    using step-hyps Uer11-eq-Uer10-if by argo

  show ?thesis
  proof (intro exI conjI)
    have step11: ord-res-11 N (Uer11, F, Γ, None) (Uer11, F', Γ', None)
    proof (rule ord-res-11.decide-pos)
      show ¬ (∃ C |∈| iefac F |' (N |∪| Uer11). trail-false-cl Γ C)
        using step-hyps unfolding ⟨Uer11 = Uer10⟩ by argo
    next
      show linorder-trm.is-least-in-fset
        { |A2 |∈| atms-of-clss (N |∪| Uer11). ∀ A1 |∈| trail-atms Γ. A1 <t A2 } A
        using step-hyps unfolding ⟨Uer11 = Uer10⟩ by argo
    next
      show linorder-cls.is-least-in-fset
        { |C |∈| iefac F |' (N |∪| Uer11). clause-could-propagate Γ C (Pos A) } C
        using step-hyps unfolding ⟨Uer11 = Uer10⟩ by argo
    next
      show Γ' = (Pos A, None) # Γ
        using step-hyps by argo
    next
      show ¬ (∃ C |∈| iefac F |' (N |∪| Uer11). trail-false-cl Γ' C)
        using step-hyps unfolding ⟨Uer11 = Uer10⟩ by argo
    next
      show F' = (if ord-res.is-strictly-maximal-lit (Pos A) C then F else finsert
C F)
        using step-hyps by argo
    qed

  thus (constant-context ord-res-11)++ S11 (N, Uer11, F', Γ', None)
    unfolding S11-def ⟨C = None⟩ by (auto intro: constant-context.intros)

  show ord-res-10-matches-ord-res-11 S10' (N, Uer11, F', Γ', None)
    unfolding ⟨S10' = (N, s10')⟩ ⟨s10' = ->⟩
  proof (rule ord-res-10-matches-ord-res-11.intros)
    show ord-res-10-invars N (Uer10, F', Γ')
      using step10 ⟨s10' = ->⟩ invars10 ord-res-10-preserves-invars by metis
    next
      show ord-res-11-invars N (Uer11, F', Γ', None)
        using step11 invars11 ⟨C = None⟩ ord-res-11-preserves-invars by metis

```



```

next
  show  $U_{er11} = U_{er10} \mid - \mid \{\#\}$ 
    unfolding  $\langle U_{er11} = U_{er10} \rangle$ 
    using  $\langle \#\} \mid \notin \mid N \mid \cup \mid U_{er10} \rangle$  by simp
next
  have  $\{\#\} \mid \notin \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er10})$ 
    using  $\langle \#\} \mid \notin \mid N \mid \cup \mid U_{er10} \rangle$  by (simp add: iefac-def)
    thus if  $\{\#\} \mid \in \mid \text{iefac } \mathcal{F}' \mid \uparrow \mid (N \mid \cup \mid U_{er10})$  then  $\Gamma' = [] \wedge \text{None} = \text{Some}$ 
 $\{\#\}$  else  $\text{None} = \text{None}$ 
    by argo
  qed
next
  case step-hyps: (propagate A C  $\Gamma'$   $\mathcal{F}'$ )

  have C = None
    using step-hyps C-eq-None-if-no-false-cls by argo

  have  $\{\#\} \mid \notin \mid N \mid \cup \mid U_{er10}$ 
    using step-hyps mempty-not-in-if by argo

  have  $U_{er11} = U_{er10}$ 
    using step-hyps  $U_{er11}$ -eq- $U_{er10}$ -if by argo

  show ?thesis
  proof (intro exI conjI)
    have step11: ord-res-11 N ( $U_{er11}$ ,  $\mathcal{F}$ ,  $\Gamma$ , None) ( $U_{er11}$ ,  $\mathcal{F}'$ ,  $\Gamma'$ , None)
    proof (rule ord-res-11.propagate)
      show  $\neg (\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er11}). \text{trail-false-cls } \Gamma C)$ 
        using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show linorder-trm.is-least-in-fset
         $\{|A_2 \mid \in \mid \text{atms-of-cls } (N \mid \cup \mid U_{er11}). \forall A_1 \mid \in \mid \text{trail-atms } \Gamma. A_1 \prec_t A_2\} A$ 
        using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show linorder-cls.is-least-in-fset
         $\{|C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er11}). \text{clause-could-propagate } \Gamma C (\text{Pos } A)\} C$ 
        using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show  $\Gamma' = (\text{Pos } A, \text{Some } (\text{efac } C)) \# \Gamma$ 
        using step-hyps by argo
    next
      show  $\exists C \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N \mid \cup \mid U_{er11}). \text{trail-false-cls } \Gamma' C$ 
        using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
    next
      show  $\mathcal{F}' = (\text{if } \text{ord-res.is-strictly-maximal-lit } (\text{Pos } A) C \text{ then } \mathcal{F} \text{ else } \text{finsert}$ 
      C  $\mathcal{F})$ 
        using step-hyps by argo
    qed
  end

```

```

thus (constant-context ord-res-11)++ S11 (N, Uer11,  $\mathcal{F}'$ ,  $\Gamma'$ , None)
  unfolding S11-def  $\langle C = None \rangle$  by (auto intro: constant-context.intros)

show ord-res-10-matches-ord-res-11 S10' (N, Uer11,  $\mathcal{F}'$ ,  $\Gamma'$ , None)
  unfolding  $\langle S10' = (N, s10') \rangle \langle s10' = \rightarrow \rangle$ 
proof (rule ord-res-10-matches-ord-res-11.intros)
  show ord-res-10-invars N (Uer10,  $\mathcal{F}'$ ,  $\Gamma'$ )
    using step10  $\langle s10' = \rightarrow \rangle$  invars10 ord-res-10-preserves-invars by metis
next
  show ord-res-11-invars N (Uer11,  $\mathcal{F}'$ ,  $\Gamma'$ , None)
    using step11 invars11  $\langle C = None \rangle$  ord-res-11-preserves-invars by metis
next
  show Uer11 = Uer10  $\mid$ -|  $\{\{\#\}\}$ 
    unfolding  $\langle U_{er11} = U_{er10} \rangle$ 
    using  $\langle \{\#\} \mid \notin N \mid \cup U_{er10} \rangle$  by simp
next
  have  $\{\#\} \mid \notin$  iefac  $\mathcal{F}' \mid \uparrow$  (N  $\mid \cup$  Uer10)
    using  $\langle \{\#\} \mid \notin N \mid \cup U_{er10} \rangle$  by (simp add: iefac-def)
    thus if  $\{\#\} \mid \in$  iefac  $\mathcal{F}' \mid \uparrow$  (N  $\mid \cup$  Uer10) then  $\Gamma' = [] \wedge None = Some$ 
 $\{\#\}$  else None = None
    by argo
  qed
qed
next
  case step-hyps: (resolution D A C Uer10'  $\Gamma'$ )

note D-max-lit =  $\langle$ ord-res.is-maximal-lit (Neg A) D $\rangle$ 

have  $\{\#\} \mid \notin$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup$  Uer10)
  using  $\langle$ linorder-cls.is-least-in-fset - D $\rangle \langle$ linorder-lit.is-maximal-in-mset D  $\rightarrow$ 
unfolding linorder-cls.is-least-in-ffilter-iff linorder-lit.is-maximal-in-mset-iff
    by (metis (no-types, lifting) empty-iff linorder-cls.leD mempty-lesseq-cls
set-mset-empty
  trail-false-cls-mempty)

have C = None
  using match-hyps  $\langle \{\#\} \mid \notin$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup$  Uer10) $\rangle$  by argo

have Uer11 = Uer10
  using match-hyps  $\langle \{\#\} \mid \notin$  iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup$  Uer10) $\rangle$  by force

have step11-conf: ord-res-11 N (Uer11,  $\mathcal{F}$ ,  $\Gamma$ , None) (Uer11,  $\mathcal{F}$ ,  $\Gamma$ , Some D)
proof (rule ord-res-11.conflict)
  show linorder-cls.is-least-in-fset
    (ffilter (trail-false-cls  $\Gamma$ ) (iefac  $\mathcal{F} \mid \uparrow$  (N  $\mid \cup$  Uer11))) D
  using step-hyps unfolding  $\langle U_{er11} = U_{er10} \rangle$  by argo
qed

```

```

have  $\Gamma$ -spec:  $\forall Ln \Gamma'. \Gamma = Ln \# \Gamma' \longrightarrow$ 
  (snd Ln  $\neq$  None) = fBex (iefac  $\mathcal{F}$  | $\uparrow$  (N | $\cup$ | Uer10)) (trail-false-cls  $\Gamma$ )  $\wedge$ 
  ( $\forall x \in \text{set } \Gamma'. \text{snd } x = \text{None}$ )
using invars10 by (simp add: ord-res-10-invars.simps)

then obtain  $\Gamma'''$  where  $\Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma'''$ 
using  $\langle \text{map-of } \Gamma (\text{Pos } A) = \text{Some } (\text{Some } C) \rangle$ 
by (metis list.set-cases map-of-SomeD not-Some-eq snd-conv)

have C-max-lit: linorder-lit.is-greatest-in-mset C (Pos A)
using invars10 by (simp add: ord-res-10-invars.simps  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma''' \rangle$ )

have ord-res.ground-resolution D C ((D - {#Neg A#}) + (C - {#Pos A#}))
proof (rule ord-res.ground-resolutionI)
  show D = add-mset (Neg A) (D - {#Neg A#})
    using D-max-lit unfolding linorder-lit.is-maximal-in-mset-iff by simp
next
  show C = add-mset (Pos A) (C - {#Pos A#})
    using C-max-lit unfolding linorder-lit.is-greatest-in-mset-iff by simp
next
  show C  $\prec_c$  D
    using C-max-lit D-max-lit
    by (simp add: clause-lt-clause-if-max-lit-comp
      linorder-lit.is-greatest-in-mset-iff-is-maximal-and-count-eq-one)
next
  show {#} = {#}  $\wedge$  ord-res.is-maximal-lit (Neg A) D  $\vee$  Neg A  $\in$  {#}
    using D-max-lit by argo
next
  show {#} = {#}
    by argo
next
  show ord-res.is-strictly-maximal-lit (Pos A) C
    using C-max-lit .
next
  show remove1-mset (Neg A) D + remove1-mset (Pos A) C = (D - {#Neg
A#}) + (C - {#Pos A#})
  ..
qed

hence eres C D  $\neq$  D
unfolding eres-ident-iff not-not ground-resolution-def by metis

have D-false: trail-false-cls  $\Gamma$  D
using step-hyps unfolding linorder-cls.is-least-in-ffilter-iff by argo

have clause-could-propagate  $\Gamma'''$  C (Pos A)
using invars10  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma''' \rangle$  by (simp add: ord-res-10-invars.simps)

```

**hence** *trail-false-cls*  $\Gamma''' \{ \#L \in \# C. L \neq \text{Pos } A \# \}$   
**unfolding** *clause-could-propagate-def* **by** *argo*

**hence** *C-almost-false*: *trail-false-cls*  $\Gamma \{ \#L \in \# C. L \neq \text{Pos } A \# \}$   
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma''' \rangle$   
**by** (*meson suffix-ConsD suffix-order.dual-order.refl trail-false-cls-if-trail-false-suffix*)

**have** *eres-false*: *trail-false-cls*  $\Gamma$  (*eres*  $C$   $D$ )  
**unfolding** *trail-false-cls-def*  
**proof** (*intro ballI*)  
**fix**  $K :: 'f$  *gliteral*  
**assume**  $K \in \#$  *eres*  $C$   $D$   
**hence**  $K \in \# C \wedge K \neq \text{Pos } A \vee K \in \# D$   
**using** *strong-lit-in-one-of-resolvents-if-in-eres[OF D-max-lit]* **by** *simp*  
**thus** *trail-false-lit*  $\Gamma$   $K$   
**proof** (*elim disjE conjE*)  
**assume**  $K \in \# C$  **and**  $K \neq \text{Pos } A$   
**thus** *trail-false-lit*  $\Gamma$   $K$   
**using** *C-almost-false unfolding trail-false-cls-def* **by** *simp*  
**next**  
**assume**  $K \in \# D$   
**thus** *trail-false-lit*  $\Gamma$   $K$   
**using** *D-false unfolding trail-false-cls-def* **by** *simp*  
**qed**  
**qed**

**have**  $\forall Ln \in \text{set } \Gamma. \forall C. \text{snd } Ln = \text{Some } C \longrightarrow \text{ord-res.is-strictly-maximal-lit}$   
(*fst*  $Ln$ )  $C$   
**using** *invars10* **by** (*simp add: ord-res-10-invars.simps*)

**hence** *C-max-lit*: *ord-res.is-strictly-maximal-lit* ( $\text{Pos } A$ )  $C$   
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma''' \rangle$  **by** *simp*

**have** *steps11-reso*: (*ord-res-11*  $N$ )\*\* ( $U_{er11}, \mathcal{F}, \Gamma, \text{Some } D$ ) ( $U_{er11}, \mathcal{F}, \Gamma, \text{Some}$   
(*eres*  $C$   $D$ ))  
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma''' \rangle$   
**using** *C-max-lit rtrancl-ord-res-11-all-resolution-steps* **by** *metis*

**define** *strict-P* :: '*f* *gterm literal*  $\times$  '*f* *gterm literal multiset option*  $\Rightarrow$  *bool* **where**  
*strict-P*  $\equiv \lambda Ln. \forall K. \text{ord-res.is-maximal-lit } K$  (*eres*  $C$   $D$ )  $\longrightarrow \text{atm-of } K \prec_t$   
*atm-of* (*fst*  $Ln$ )

**have**  $\forall K \in \#$  *eres*  $C$   $D. - K \in \text{fst ' set } \Gamma$   
**using** *eres-false unfolding trail-false-cls-def trail-false-lit-def* .

**moreover** **have**  $\Gamma$ -*sorted*: *sorted-wrt* ( $\lambda x y. \text{atm-of } (\text{fst } y) \prec_t \text{atm-of } (\text{fst } x)$ )  $\Gamma$   
**using** *invars10* **by** (*simp add: ord-res-10-invars.simps*)

**ultimately** **have** *dropWhile strict-P*  $\Gamma = \text{dropWhile } (\lambda Ln. - \text{fst } Ln \notin \#$  *eres*

```

C D)  $\Gamma$ 
  proof (induction  $\Gamma$ )
    case Nil
      show ?case by simp
    next
      case (Cons Ln  $\Gamma$ )

      show ?case
      proof (cases eres C D = {#})
        case True
          thus ?thesis
            unfolding strict-P-def linorder-lit.is-maximal-in-mset-iff by simp
        next
          case False

            then obtain L where eres-max-lit: ord-res.is-maximal-lit L (eres C D)
              using linorder-lit.ex-maximal-in-mset by metis

            hence strict-P-Ln-iff: strict-P Ln  $\longleftrightarrow$  atm-of L  $\prec_t$  atm-of (fst Ln)
              unfolding strict-P-def
              by (metis linorder-lit.Uniq-is-maximal-in-mset the1-equality')

            show ?thesis
            proof (cases atm-of L  $\prec_t$  atm-of (fst Ln))
              case True

                moreover have  $\text{fst Ln} \notin \#$  eres C D
                  using True
                by (smt (verit, best) atm-of-uminus eres-max-lit linorder-lit.dual-order.strict-trans
                    linorder-lit.is-maximal-in-mset-iff linorder-lit.neq-iff linorder-trm.order.irrefl
                    literal.exhaust-sel ord-res.less-lit-simps(4))

                moreover have dropWhile strict-P  $\Gamma =$  dropWhile ( $\lambda$ Ln.  $\text{fst Ln} \notin \#$ 
eres C D)  $\Gamma$ 
                  proof (rule Cons.IH)
                    show  $\forall K \in \#$ eres C D.  $\text{fst Ln} \notin \#$  eres C D
                      using Cons.prem True  $\prec$   $\text{fst Ln} \notin \#$  eres C D
                      using image-iff by fastforce
                    next
                      show sorted-wrt ( $\lambda$ x y. atm-of (fst y)  $\prec_t$  atm-of (fst x))  $\Gamma$ 
                        using Cons.prem by simp
                  qed

                ultimately show ?thesis
                  unfolding dropWhile.simps
                  unfolding strict-P-Ln-iff
                  by simp
              next
                case False
            end
          end
        case False
      end
    end
  end

```

**hence**  $\text{atm-of } (fst Ln) \preceq_t \text{atm-of } L$   
**by** *order*

**hence**  $\text{atm-of } (fst Ln) = \text{atm-of } L$   
**using** *Cons.prem*s  
**using** *atm-of-eq-atm-of eres-max-lit linorder-lit.is-maximal-in-mset-iff*  
**by** *fastforce*

**hence**  $L = fst Ln \vee L = -fst Ln$   
**by** (*metis atm-of-eq-atm-of*)

**moreover have**  $-fst Ln \notin \text{fst } \langle \text{set } \Gamma \rangle$   
**using**  $\langle \text{sorted-wrt } (\lambda x y. \text{atm-of } (fst y) \prec_t \text{atm-of } (fst x)) (Ln \# \Gamma) \rangle$   
**by** *fastforce*

**moreover have**  $-L \in \text{fst } \langle \text{set } (Ln \# \Gamma) \rangle$   
**using** *Cons.prem*s(1) *eres-max-lit linorder-lit.is-maximal-in-mset-iff* **by**  
*blast*

**ultimately have**  $-fst Ln \in \# \text{eres } C D$   
**using** *eres-max-lit linorder-lit.is-maximal-in-mset-iff* **by** *auto*

**then show** *?thesis*  
**unfolding** *dropWhile.simps*  
**unfolding** *strict-P-Ln-iff*  
**using** *False*  
**by** *arg*o

**qed**  
**qed**  
**qed**

**hence** *steps11-skip*:  $(\text{ord-res-11 } N)^{**}$   
 $(U_{er11}, \mathcal{F}, \Gamma, \text{Some } (\text{eres } C D))$   
 $(U_{er11}, \mathcal{F}, \text{dropWhile } \text{strict-P } \Gamma, \text{Some } (\text{eres } C D))$   
**using** *rtrancl-ord-res-11-all-skip-steps* **by** *metis*

**have** *most-steps11*:  $(\text{ord-res-11 } N)^{++}$   $(U_{er11}, \mathcal{F}, \Gamma, \text{None})$   
 $(U_{er11}, \mathcal{F}, \text{dropWhile } \text{strict-P } \Gamma, \text{Some } (\text{eres } C D))$   
**using** *step11-conf steps11-reso steps11-skip* **by** *simp*

**show** *?thesis*  
**proof** (*cases eres C D = {#}*)  
**case** *True*

**hence** *dropWhile strict-P*  $\Gamma = []$   
**unfolding** *strict-P-def*  $\langle \text{eres } C D = \{\#\} \rangle$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**by** *simp*

```

have  $\Gamma' = []$ 
  unfolding  $\langle \Gamma' = \text{dropWhile } - \Gamma \rangle \langle \text{eres } C D = \{\#\} \rangle$ 
  unfolding linorder-lit.is-maximal-in-mset-iff
  by simp

show ?thesis
proof (intro exI conjI)
  show (constant-context ord-res-11)++ S11 ( $N, U_{er11}, \mathcal{F}, [], \text{Some } \{\#\}$ )
    unfolding S11-def  $\langle C = \text{None} \rangle$ 
    using most-steps11[unfolded  $\langle \text{dropWhile strict-P } \Gamma = [] \rangle \langle \text{eres } C D =$ 
 $\{\#\} \rangle$ ]
    using tranclp-constant-context by metis

show ord-res-10-matches-ord-res-11 S10' ( $N, U_{er11}, \mathcal{F}, [], \text{Some } \{\#\}$ )
  unfolding  $\langle S10' = (N, s10') \rangle \langle s10' = - \rangle \langle \Gamma' = [] \rangle$ 
proof (rule ord-res-10-matches-ord-res-11.intros)
  show ord-res-10-invars  $N (U_{er10'}, \mathcal{F}, [])$ 
    using step10  $\langle s10' = - \rangle \langle \Gamma' = [] \rangle$  invars10 ord-res-10-preserves-invars
by metis
  next
    show ord-res-11-invars  $N (U_{er11}, \mathcal{F}, [], \text{Some } \{\#\})$ 
    using most-steps11 invars11
    unfolding  $\langle C = \text{None} \rangle \langle \text{dropWhile strict-P } \Gamma = [] \rangle \langle \text{eres } C D = \{\#\} \rangle$ 
    by (metis tranclp-ord-res-11-preserves-invars)
  next
    show  $U_{er11} = U_{er10'} \mid - \mid \{\{\#\}\}$ 
    unfolding  $\langle U_{er11} = U_{er10'} \rangle \langle U_{er10}' = \text{finsert } (\text{eres } C D) U_{er10} \rangle \langle \text{eres } C D = \{\#\} \rangle$ 
    using  $\langle \{\#\} \notin \text{iefac } \mathcal{F} \mid \mid (N \mid \cup \mid U_{er10}) \rangle$ 
    by force
  next
    show if  $\{\#\} \in \mid \text{iefac } \mathcal{F} \mid \mid (N \mid \cup \mid U_{er10}')$  then
       $[] = [] \wedge \text{Some } \{\#\} = \text{Some } \{\#\}$  else  $\text{Some } \{\#\} = \text{None}$ 
    unfolding  $\langle U_{er10}' = \text{finsert } (\text{eres } C D) U_{er10} \rangle \langle \text{eres } C D = \{\#\} \rangle$ 
    by simp
  qed
qed
next
  case False

then obtain  $L$  where eres-max-lit: ord-res.is-maximal-lit  $L$  (eres  $C D$ )
  using linorder-lit.ex-maximal-in-mset by metis

hence  $L \in \#$  eres  $C D$ 
  unfolding linorder-lit.is-maximal-in-mset-iff by argo

hence  $L \in \# C \wedge L \neq \text{Pos } A \vee L \in \# D \wedge L \neq \text{Neg } A$ 
  using stronger-lit-in-one-of-resolvents-if-in-eres  $\langle \text{eres } C D \neq D \rangle$  D-max-lit

```

**by** (*metis uminus-Neg*)

**hence**  $L \prec_l \text{Neg } A$   
**using** *C-max-lit D-max-lit*

**unfolding** *linorder-lit.is-greatest-in-mset-iff linorder-lit.is-maximal-in-mset-iff*  
**by** (*metis C-max-lit D-max-lit*  $\langle L \in \# \text{eres } C \ D \rangle$  *eres-lt-if ord-res.asymp-less-lit*  
*ord-res.less-lit-simps(3,4) ord-res.transp-less-lit in-remove1-mset-neq*  
*linorder-lit.less-than-maximal-if-multp<sub>HO</sub> linorder-lit.order.not-eq-order-implies-strict*  
*literal.disc(2) multp-eq-multp<sub>HO</sub> uminus-Neg*)

**have** *dropWhile strict-P*  $\Gamma = \text{dropWhile } (\lambda Ln. \text{atm-of } L \prec_t \text{atm-of } (\text{fst } Ln))$

$\Gamma$

**unfolding** *strict-P-def*  
**using** *eres-max-lit*  
**by** (*metis (no-types) Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

**also have**  $\dots = (- L, \text{None}) \# \text{dropWhile } (\lambda Ln. \text{atm-of } L \preceq_t \text{atm-of } (\text{fst } Ln))$   $\Gamma$

**proof** –

**have**  $- L \neq \text{Pos } A$   
**using**  $\langle L \prec_l \text{Neg } A \rangle$  **by** (*cases L*) *simp-all*

**hence**  $- L \in \text{fst 'set } \Gamma'''$   
**using** *eres-false*  $\langle L \in \# \text{eres } C \ D \rangle$   
**unfolding** *trail-false-cls-def trail-false-lit-def*  
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma'''$   
**by** *auto*

**hence**  $(- L, \text{None}) \in \text{set } \Gamma'''$   
**using**  $\Gamma$ -*spec* **unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma'''$  **by** *auto*

**then obtain**  $\Gamma_0 \ \Gamma_1$  **where**  $\Gamma'''' = \Gamma_1 \ @ \ (- L, \text{None}) \ # \ \Gamma_0$   
**by** (*meson split-list*)

**hence**  $\Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_1 \ @ \ (- L, \text{None}) \ # \ \Gamma_0$   
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma''''$  **by** (*simp only:*)

**have** *AAA*:  $\forall Ln \in \text{set } ((\text{Pos } A, \text{Some } C) \# \Gamma_1). \text{atm-of } L \prec_t \text{atm-of } (\text{fst } Ln)$

**using**  $\Gamma$ -*sorted* **unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_1 \ @ \ (- L, \text{None}) \ # \ \Gamma_0 \rangle$   
**by** (*simp add: sorted-wrt-append*)

**hence** *BBB*:  $\forall Ln \in \text{set } ((\text{Pos } A, \text{Some } C) \# \Gamma_1 \ @ \ [(- L, \text{None})]). \text{atm-of } L \preceq_t \text{atm-of } (\text{fst } Ln)$   
**by** *simp*

**have**  $\forall Ln \in \text{set } \Gamma_0. \text{atm-of } (\text{fst } Ln) \prec_t \text{atm-of } L$   
**using**  $\Gamma$ -*sorted* **unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_1 \ @ \ (- L, \text{None})$



$\# \Gamma_0 \rangle$   
**by** (*simp add: sorted-wrt-append*)

**hence** *CCC*:  $\forall Ln \in \text{set } \Gamma_0. \neg \text{atm-of } L \preceq_t \text{atm-of } (\text{fst } Ln)$   
**using** *linorder-trm.leD* **by** *blast*

**have** *dropWhile* ( $\lambda Ln. \text{atm-of } L \prec_t \text{atm-of } (\text{fst } Ln)$ )  $\Gamma = (- L, \text{None}) \# \Gamma_0$   
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_1 @ (- L, \text{None}) \# \Gamma_0 \rangle$   
**using** *dropWhile-append2 AAA* **by** *simp*

**also have**  $\dots = (- L, \text{None}) \# \text{dropWhile } (\lambda Ln. \text{atm-of } L \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma_0$   
**using** *CCC* **by** *simp*

**also have**  $\dots = (- L, \text{None}) \# \text{dropWhile } (\lambda Ln. \text{atm-of } L \preceq_t \text{atm-of } (\text{fst } Ln)) \Gamma$   
**unfolding**  $\langle \Gamma = (\text{Pos } A, \text{Some } C) \# \Gamma_1 @ (- L, \text{None}) \# \Gamma_0 \rangle$   
**using** *dropWhile-append2 BBB* **by** *simp*

**finally show** *?thesis* .  
**qed**

**also have**  $\dots = (- L, \text{None}) \# \Gamma'$   
**unfolding**  $\langle \Gamma' = \text{dropWhile } - \Gamma \rangle$   
**using** *eres-max-lit*  
**by** (*metis (no-types) Uniq-D linorder-lit.Uniq-is-maximal-in-mset*)

**finally have** *dropWhile strict-P*  $\Gamma = (- L, \text{None}) \# \Gamma'$  .

**have** *step10-back: ord-res-11 N*  
 $(U_{er11}, \mathcal{F}, \text{dropWhile strict-P } \Gamma, \text{Some } (\text{eres } C D))$   
 $(\text{finsert } (\text{eres } C D) U_{er11}, \mathcal{F}, \Gamma', \text{None})$   
**unfolding**  $\langle \text{dropWhile strict-P } \Gamma = (- L, \text{None}) \# \Gamma' \rangle$   
**proof** (*rule ord-res-11.backtrack*)  
**show**  $(- L, \text{None}) \# \Gamma' = (- L, \text{None}) \# \Gamma' ..$   
**next**  
**show**  $(- L) \in \# \text{eres } C D$   
**unfolding** *uminus-of-uminus-id*  
**using** *eres-max-lit*  
**unfolding** *linorder-lit.is-maximal-in-mset-iff* **by** *argo*  
**qed**

**hence** *all-steps11*:  $(\text{ord-res-11 } N)^{++} (U_{er11}, \mathcal{F}, \Gamma, \text{None})$   
 $(\text{finsert } (\text{eres } C D) U_{er11}, \mathcal{F}, \Gamma', \text{None})$   
**using** *most-steps11* **by** *simp*

**show** *?thesis*  
**proof** (*intro exI conjI*)  
**show** (*constant-context ord-res-11*) $^{++}$  *S11* ( $N, \text{finsert } (\text{eres } C D) U_{er11},$

```

 $\mathcal{F}, \Gamma', None)$ 
  unfolding  $S11-def$   $\langle C = None \rangle$ 
  using  $all-steps11$   $tranclp-constant-context$  by  $metis$ 

  have  $\{\#\} \notin iefac \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er10})$ 
    by  $(smt (verit, del-insts) False \langle \{\#\} \notin iefac \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er10}) \rangle$ 
 $fimage.rep-eq$ 
 $fimageE$   $fimageI$   $finsertE$   $funion-iff$   $iefac-def$   $mempty-in-image-efac-iff$ 
 $step-hyps(5))$ 

  show  $ord-res-10-matches-ord-res-11$   $S10' (N, finsert (eres C D) U_{er11}, \mathcal{F},$ 
 $\Gamma', None)$ 
    unfolding  $\langle S10' = (N, s10') \rangle$   $\langle s10' = - \rangle$ 
    proof  $(rule ord-res-10-matches-ord-res-11.intros)$ 
    show  $ord-res-10-invars$   $N (U_{er10}', \mathcal{F}, \Gamma')$ 
      using  $step10$   $\langle s10' = - \rangle$   $invars10$   $ord-res-10-preserves-invars$  by  $metis$ 
    next
    show  $ord-res-11-invars$   $N (finsert (eres C D) U_{er11}, \mathcal{F}, \Gamma', None)$ 
      using  $all-steps11$   $invars11$ 
      unfolding  $\langle C = None \rangle$ 
      by  $(metis tranclp-ord-res-11-preserves-invars)$ 
    next
    show  $finsert (eres C D) U_{er11} = U_{er10}' \mid - \mid \{\{\#\}\}$ 
      unfolding  $\langle U_{er11} = U_{er10} \rangle$   $\langle U_{er10}' = finsert (eres C D) U_{er10} \rangle$ 
      using  $\langle \{\#\} \notin iefac \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er10}') \rangle$ 
      using  $False$   $\langle U_{er11} = U_{er10} \rangle$   $\langle U_{er11} = U_{er10} \mid - \mid \{\{\#\}\} \rangle$  by  $auto$ 
    next
    show if  $\{\#\} \in iefac \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er10}')$  then
       $\Gamma' = [] \wedge None = Some \{\#\}$  else  $None = None$ 
      using  $\langle \{\#\} \notin iefac \mathcal{F} \mid^{\dagger} (N \mid \cup \mid U_{er10}') \rangle$  by  $simp$ 
    qed
  qed
qed
qed
qed

```

**theorem**  $bisimulation-ord-res-10-ord-res-11$ :

```

  defines  $match \equiv \lambda-. ord-res-10-matches-ord-res-11$ 
  shows  $\exists (MATCH :: nat \times nat \Rightarrow 'f ord-res-10-state \Rightarrow 'f ord-res-11-state \Rightarrow$ 
 $bool) \mathcal{R}_f \mathcal{R}_b.$ 
     $bisimulation$ 
     $(constant-context ord-res-10) ord-res-8-final$ 
     $(constant-context ord-res-11) ord-res-11-final$ 
     $MATCH \mathcal{R}_f \mathcal{R}_b$ 
proof  $(rule ex-bisimulation-from-forward-simulation)$ 
  show  $right-unique$   $(constant-context ord-res-10)$ 
    using  $right-unique-constant-context$   $right-unique-ord-res-10$  by  $metis$ 
  next
  show  $right-unique$   $(constant-context ord-res-11)$ 

```

```

    using right-unique-constant-context right-unique-ord-res-11 by metis
next
show  $\forall S. \text{ord-res-8-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-10 } S S')$ 
  by (metis finished-def ord-res-10-semantics.final-finished)
next
show  $\forall S. \text{ord-res-11-final } S \longrightarrow (\exists S'. \text{constant-context ord-res-11 } S S')$ 
  by (metis finished-def ord-res-11-semantics.final-finished)
next
show  $\forall i S10 S11. \text{match } i S10 S11 \longrightarrow \text{ord-res-8-final } S10 \longleftrightarrow \text{ord-res-11-final } S11$ 
  unfolding match-def
  using ord-res-10-final-iff-ord-res-11-final by metis
next
show  $\forall i S10 S11. \text{match } i S10 S11 \longrightarrow$ 
  safe-state (constant-context ord-res-10) ord-res-8-final S10  $\wedge$ 
  safe-state (constant-context ord-res-11) ord-res-11-final S11
proof (intro allI impI conjI)
  fix i S10 S11
  assume match: match i S10 S11
  show safe-state (constant-context ord-res-10) ord-res-8-final S10
    using match[unfolded match-def]
    using ord-res-10-safe-state-if-invars
    unfolding ord-res-10-matches-ord-res-11.simps by auto

  show safe-state (constant-context ord-res-11) ord-res-11-final S11
    using match[unfolded match-def]
    using ord-res-11-safe-state-if-invars
    using ord-res-10-matches-ord-res-11.simps by auto
qed
next
show wfp ( $\lambda - . \text{False}$ )
  by simp
next
show  $\forall i S10 S11 S10'. \text{match } i S10 S11 \longrightarrow \text{constant-context ord-res-10 } S10 S10' \longrightarrow$ 
  ( $\exists i' S11'. (\text{constant-context ord-res-11})^{++} S11 S11' \wedge \text{match } i' S10' S11'$ )  $\vee$ 
  ( $\exists i'. \text{match } i' S10' S11 \wedge \text{False}$ )
  unfolding match-def
  using forward-simulation-between-10-and-11 by metis
qed

end

type-synonym bisim-index-1-2 = nat  $\times$  nat
type-synonym bisim-index-1-3 = bisim-index-1-2  $\times$  (nat  $\times$  nat)
type-synonym bisim-index-1-4 = bisim-index-1-3  $\times$  (nat  $\times$  nat)
type-synonym bisim-index-1-5 = bisim-index-1-4  $\times$  (nat  $\times$  nat)
type-synonym bisim-index-1-6 = bisim-index-1-5  $\times$  (nat  $\times$  nat)
type-synonym bisim-index-1-7 = bisim-index-1-6  $\times$  (nat  $\times$  nat)

```

**type-synonym** *bisim-index-1-8* = *bisim-index-1-7* × (*nat* × *nat*)  
**type-synonym** *bisim-index-1-9* = *bisim-index-1-8* × (*nat* × *nat*)  
**type-synonym** *bisim-index-1-10* = *bisim-index-1-9* × (*nat* × *nat*)  
**type-synonym** *bisim-index-1-11* = *bisim-index-1-10* × (*nat* × *nat*)

**context** *simulation-SCLFOL-ground-ordered-resolution* **begin**

**theorem** *bisimulation-ord-res-1-ord-res-11*:

**obtains**

*MATCH* :: *bisim-index-1-11* ⇒ 'f *ord-res-1-state* ⇒ 'f *ord-res-11-state* ⇒ *bool*

**and**

$\mathcal{R}_f \mathcal{R}_b$  :: *bisim-index-1-11* ⇒ *bisim-index-1-11* ⇒ *bool*

**where**

*bisimulation*

*ord-res-1 ord-res-1-final*

(*constant-context ord-res-11*) *ord-res-11-final*

*MATCH*  $\mathcal{R}_f \mathcal{R}_b$

**apply** *atomize-elim*

**using** *bisimulation-ord-res-1-ord-res-2*

*bisimulation-ord-res-2-ord-res-3*

*bisimulation-ord-res-3-ord-res-4*

*bisimulation-ord-res-4-ord-res-5*

*bisimulation-ord-res-5-ord-res-6*

*bisimulation-ord-res-6-ord-res-7*

*bisimulation-ord-res-7-ord-res-8*

*bisimulation-ord-res-8-ord-res-9*

*bisimulation-ord-res-9-ord-res-10*

*bisimulation-ord-res-10-ord-res-11*

**using** *bisimulation-composition* **by** *meson*

**theorem**

**obtains**

*MATCH* :: *bisim-index-1-11* ⇒ 'f *ord-res-1-state* ⇒ 'f *ord-res-11-state* ⇒ *bool*

**and**

$\mathcal{R}_f \mathcal{R}_b$  :: *bisim-index-1-11* ⇒ *bisim-index-1-11* ⇒ *bool*

**where**

*bisimulation*

*ord-res-1 ord-res-1-final*

(*constant-context ord-res-11*) *ord-res-11-final*

*MATCH*  $\mathcal{R}_f \mathcal{R}_b$  **and**

$\bigwedge_j S1 S11. \text{MATCH } j S1 S11 \implies \text{ord-res-1-final } S1 \longleftrightarrow \text{ord-res-11-final } S11$

**using** *bisimulation-ord-res-1-ord-res-11* *bisimulation.agree-on-final*

**by** (*metis (no-types, opaque-lifting)*)

## 38 ORD-RES-11 is a regular SCL strategy

**definition** *gtrailelem-of-trailelem* **where**

*gtrailelem-of-trailelem* ≡ λ(*L, opt*).

(*lit-of-glit L, map-option* (λ*C. (cls-of-gcls {#K ∈# C. K ≠ L#}, lit-of-glit L,*

*Var*)) *opt*)

**fun** *state-of-gstate* :: -  $\Rightarrow$  ('f, 'v) *SCL-FOL.state* **where**  
*state-of-gstate* (*U<sub>G</sub>*, -,  $\Gamma_G$ ,  $\mathcal{C}_G$ ) =  
 (let  
    $\Gamma$  = *map gtrailelem-of-trailelem*  $\Gamma_G$ ;  
   *U* = *cls-of-gcls* |<sup>1</sup> *U<sub>G</sub>*;  
   *C* = *map-option* ( $\lambda C_G. (cls-of-gcls C_G, Var)$ )  $\mathcal{C}_G$   
 in ( $\Gamma$ , *U*, *C*))

**lemma** *fst-case-prod-simp*: *fst* (*case p of* (*x*, *y*)  $\Rightarrow$  (*f x*, *g x y*)) = *f* (*fst p*)  
**by** (*cases p*) *simp*

**lemma** *trail-false-cls-nonground-iff-trail-false-cls-ground*:

**fixes**  $\Gamma_G$  **and** *D<sub>G</sub>* :: 'f *gclause*

**fixes**  $\Gamma$  :: ('f, 'v) *SCL-FOL.trail* **and** *D* :: ('f, 'v) *term clause*

**defines**  $\Gamma \equiv \text{map gtrailelem-of-trailelem } \Gamma_G$  **and** *D*  $\equiv \text{cls-of-gcls } D_G$

**shows** *trail-false-cls*  $\Gamma$  *D*  $\longleftrightarrow$  *trail-false-cls*  $\Gamma_G$  *D<sub>G</sub>*

**proof** -

**have** *trail-false-cls*  $\Gamma$  *D*  $\longleftrightarrow$  ( $\forall L \in \# D. \text{trail-false-lit } \Gamma L$ )

**unfolding** *trail-false-cls-def* ..

**also have** ...  $\longleftrightarrow$  ( $\forall L_G \in \# D_G. \text{trail-false-lit } \Gamma (\text{lit-of-glit } L_G)$ )

**unfolding** *D-def cls-of-gcls-def* **by** *simp*

**also have** ...  $\longleftrightarrow$  ( $\forall L_G \in \# D_G. \text{trail-false-lit } \Gamma_G L_G$ )

**proof** -

**have** *trail-false-lit*  $\Gamma (\text{lit-of-glit } L_G)$   $\longleftrightarrow$  *trail-false-lit*  $\Gamma_G L_G$

**for** *L<sub>G</sub>* :: 'f *gterm literal*

**proof** -

**have** *trail-false-lit*  $\Gamma (\text{lit-of-glit } L_G)$   $\longleftrightarrow$  - *lit-of-glit* *L<sub>G</sub>*  $\in \text{fst ' set } \Gamma$

**unfolding** *trail-false-lit-def* ..

**also have** ...  $\longleftrightarrow$

- (*lit-of-glit* *L<sub>G</sub>* :: ('f, 'v) *term literal*)  $\in \text{set} (\text{map } (\lambda x. \text{lit-of-glit } (\text{fst } x))$

$\Gamma_G$ )

**unfolding**  $\Gamma$ -*def image-set list.map-comp*

**unfolding** *gtrailelem-of-trailelem-def*

**unfolding** *list.map-comp*

**unfolding** *comp-def fst-case-prod-simp* ..

**also have** ...  $\longleftrightarrow$  (*lit-of-glit* (- *L<sub>G</sub>*) :: ('f, 'v) *term literal*)  $\in \text{lit-of-glit ' fst$

*' set*  $\Gamma_G$

**by** (*cases L<sub>G</sub>*) (*auto simp: lit-of-glit-def*)

**also have** ...  $\longleftrightarrow$  - *L<sub>G</sub>*  $\in \text{fst ' set } \Gamma_G$

**using** *inj-image-mem-iff inj-lit-of-glit* **by** *metis*

**also have** ...  $\longleftrightarrow$  *trail-false-lit*  $\Gamma_G L_G$

**unfolding** *trail-false-lit-def* ..

**finally show** *trail-false-lit*  $\Gamma (\text{lit-of-glit } L_G)$  = *trail-false-lit*  $\Gamma_G L_G$  .

**qed**

**thus** *?thesis* **by** *metis*

**qed**

**also have**  $\dots \longleftrightarrow$  *trail-false-cls*  $\Gamma_G D_G$   
**unfolding** *trail-false-cls-def* ..  
**finally show** *?thesis* .  
**qed**

**theorem** *ord-res-11-is-strategy-for-regular-scl*:  
**fixes**  
 $N_G :: 'f$  *gclause fset* **and**  
 $N :: ('f, 'v)$  *term clause fset* **and**  
 $\beta_G :: 'f$  *gterm* **and**  
 $\beta :: ('f, 'v)$  *term* **and**  
 $S_G S_G' :: 'f$  *gclause fset*  $\times$  *'f gclause fset*  $\times$  *('f gliteral*  $\times$  *'f gclause option)* *list*  
 $\times$  *'f gclause option* **and**  
 $S S' :: ('f, 'v)$  *SCL-FOL.state*

**defines**  
 $N \equiv$  *cls-of-gcls*  $|^{\dagger}$   $N_G$  **and**  
 $\beta \equiv$  *term-of-gterm*  $\beta_G$  **and**  
 $S \equiv$  *state-of-gstate*  $S_G$  **and**  
 $S' \equiv$  *state-of-gstate*  $S_G'$

**assumes**  
*ball-le-beta\_G*:  $\forall A_G \in |$  *atms-of-cls*  $N_G$ .  $A_G \preceq_t \beta_G$  **and**  
*run*:  $(ord-res-11 N_G)^{**}$   $(\{\|\}, \{\|\}, \|\, , None)$   $S_G$  **and**  
*step*: *ord-res-11*  $N_G S_G S_G'$

**shows**  
*scl-fol.regular-scl*  $N \beta S S'$

**proof** –  
**have** *ord-res-11-invars*  $N_G (\{\|\}, \{\|\}, \|\, , None)$   
**using** *ord-res-11-invars-initial-state* .

**hence** *ord-res-11-invars*  $N_G S_G$   
**using** *run rtranclp-ord-res-11-preserves-invars* **by** *metis*

**obtain**  $U_G \mathcal{F} \Gamma_G \mathcal{C}_G$  **where** *S\_G-def*:  $S_G = (U_G, \mathcal{F}, \Gamma_G, \mathcal{C}_G)$   
**by** *(metis surj-pair)*

**obtain**  $\Gamma U \mathcal{C}$  **where** *S-def*:  $S = (\Gamma, U, \mathcal{C})$   
**by** *(metis surj-pair)*

**have**  $\Gamma$ -*def*:  $\Gamma =$  *map gtrailelem-of-trailelem*  $\Gamma_G$   
**using** *S-def S\_G-def*  $\langle S \equiv$  *state-of-gstate*  $S_G \rangle$  **by** *simp*

**have**  $U$ -*def*:  $U =$  *cls-of-gcls*  $|^{\dagger}$   $U_G$   
**using** *S-def S\_G-def*  $\langle S \equiv$  *state-of-gstate*  $S_G \rangle$  **by** *simp*

**have**  $\mathcal{C}$ -*def*:  $\mathcal{C} =$  *map-option*  $(\lambda C_G. (cls-of-gcls C_G, Var)) \mathcal{C}_G$   
**using** *S-def S\_G-def*  $\langle S \equiv$  *state-of-gstate*  $S_G \rangle$  **by** *simp*

**obtain**  $\mathcal{F}' U_G' :: 'f$  *gclause fset* **and**  $\Gamma_G' :: -$  *list* **and**  $\mathcal{C}_G' :: -$  *option* **where**  
*S\_G'-def*:  $S_G' = (U_G', \mathcal{F}', \Gamma_G', \mathcal{C}_G')$

**by** (*metis surj-pair*)

**obtain**  $\Gamma' :: - \text{list}$  **and**  $U' :: - \text{fset}$  **and**  $C' :: - \text{option}$  **where**  
*S'-def*:  $S' = (\Gamma', U', C')$   
**by** (*metis surj-pair*)

**have**  $\Gamma'$ -*def*:  $\Gamma' = \text{map } \text{gtrailelem-of-trailelem } \Gamma_G'$   
**using**  $S'$ -*def*  $S_G'$ -*def*  $\langle S' \equiv \text{state-of-gstate } S_G' \rangle$  **by** *simp*

**have**  $U'$ -*def*:  $U' = \text{cls-of-gcls } | \uparrow U_G'$   
**using**  $S'$ -*def*  $S_G'$ -*def*  $\langle S' \equiv \text{state-of-gstate } S_G' \rangle$  **by** *simp*

**have**  $C'$ -*def*:  $C' = \text{map-option } (\lambda C_G. (\text{cls-of-gcls } C_G, \text{Var})) C_G'$   
**using**  $S'$ -*def*  $S_G'$ -*def*  $\langle S' \equiv \text{state-of-gstate } S_G' \rangle$  **by** *simp*

**have** *atms-of-clss*  $U_G \mid \subseteq \mid$  *atms-of-clss*  $N_G$   
**using**  $\langle \text{ord-res-11-invars } N_G S_G \rangle [\text{unfolded } S_G\text{-def}]$   
**unfolding** *ord-res-11-invars.simps* **by** *simp*

**have** *atms-of-clss*  $(N_G \mid \cup \mid U_G) = \text{atms-of-clss } N_G \mid \cup \mid \text{atms-of-clss } U_G$   
**by** (*simp add: atms-of-clss-def fimage-funion*)

**also have**  $\dots = \text{atms-of-clss } N_G$   
**using**  $\langle \text{atms-of-clss } U_G \mid \subseteq \mid \text{atms-of-clss } N_G \rangle$  **by** *auto*

**finally have** *atms-of-clss*  $(N_G \mid \cup \mid U_G) = \text{atms-of-clss } N_G$  .

**have** *clauses-in-F-have-pos-max-lit*:  $\forall C \in \mid \mathcal{F}. \exists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L C$   
**using**  $\langle \text{ord-res-11-invars } N_G S_G \rangle [\text{unfolded } S_G\text{-def } \text{ord-res-11-invars.simps}]$   
**by** *simp*

**have** *nex-conflict-if-nbex-trail-false*:  
 $\neg \text{fBex } (\text{iefac } \mathcal{F} \mid \uparrow (N_G \mid \cup \mid U_G)) (\text{trail-false-cls } \Gamma_G) \implies \neg \text{Ex } (\text{scl-fol.conflict } N \beta S)$

**proof** (*elim contrapos-nn exE*)  
**fix**  $x :: ('f, 'v) \text{state}$   
**assume** *scl-fol.conflict*  $N \beta S x$   
**hence** *fBex*  $(N_G \mid \cup \mid U_G) (\text{trail-false-cls } \Gamma_G)$   
**unfolding** *S-def*  
**proof** (*cases*  $N \beta (\Gamma, U, C) x$  *rule: scl-fol.conflict.cases*)  
**case** (*conflictI*  $D \gamma$ )

**obtain**  $D_G$  **where**  $D_G \mid \in \mid N_G \mid \cup \mid U_G$  **and**  $D$ -*def*:  $D = \text{cls-of-gcls } D_G$   
**using**  $\langle D \mid \in \mid N \mid \cup \mid U \rangle$   
**unfolding**  $N$ -*def*  $U$ -*def* **by** *blast*

**moreover have** *trail-false-cls*  $\Gamma_G D_G$   
**proof** –

```

have is-ground-cls  $D$ 
  using  $\langle D = \text{cls-of-gcls } D_G \rangle$  by simp
hence  $D \cdot \gamma = D$ 
  by simp
hence trail-false-cls  $\Gamma$   $D$ 
  using conflictI
  unfolding SCL-FOL.trail-false-cls-def trail-false-cls-def
  unfolding SCL-FOL.trail-false-lit-def trail-false-lit-def
  by argo

thus ?thesis
  unfolding  $\Gamma$ -def  $D$ -def
  unfolding trail-false-cls-nonground-iff-trail-false-cls-ground .
qed
ultimately show ?thesis by metis
qed

```

```

thus fBex (iefac  $\mathcal{F}$   $| \uparrow$  ( $N_G$   $| \cup$   $U_G$ )) (trail-false-cls  $\Gamma_G$ )
  unfolding bex-trail-false-cls-simp .
qed

```

```

have nex-conflict-if-alread-in-conflict:  $C_G = \text{Some } C_G \implies \neg \text{Ex } (\text{scl-fol.conflict } N \beta S)$  for  $C_G$ 
  unfolding  $S$ -def  $C$ -def by (simp add: scl-fol.conflict.simps)

```

```

have nex-conflict-if-no-clause-could-propagate-comp:
   $\neg \text{Ex } (\text{scl-fol.conflict } N \beta ((\text{lit-of-glit } L_G, \text{None}) \# \Gamma, U, C))$ 
  if
    nex-false-clause-wrt- $\Gamma_G$ :  $\neg \text{fBex } (\text{iefac } \mathcal{F} | \uparrow (N_G | \cup U_G)) (\text{trail-false-cls } \Gamma_G)$ 
and
    ball-lit-atm- $L_G$ :  $\forall x | \in | \text{trail-atms } \Gamma_G. x \prec_t \text{atm-of } L_G$  and
    nex-clause-that-propagate:  $\neg (\exists C | \in | \text{iefac } \mathcal{F} | \uparrow (N_G | \cup U_G)).$ 
    clause-could-propagate  $\Gamma_G$   $C$  ( $- L_G$ )
  for  $L_G$ 
proof (intro notI, elim exE)
  fix  $S'' :: ('f, 'v) \text{SCL-FOL.state}$ 
  assume scl-fol.conflict  $N \beta ((\text{lit-of-glit } L_G, \text{None}) \# \Gamma, U, C)$   $S''$ 
  thus False
proof (cases N  $\beta$  ((lit-of-glit } L_G, \text{None}) \# \Gamma, U, C) S'' rule: scl-fol.conflict.cases)
  case (conflictI  $D$   $\gamma$ )

```

```

obtain  $D_G$  where  $D_G | \in | N_G | \cup U_G$  and  $D$ -def:  $D = \text{cls-of-gcls } D_G$ 
  using  $\langle D | \in | N | \cup U \rangle$   $N$ -def  $U$ -def by blast

```

```

have (lit-of-glit  $L_G :: ('f, 'v) \text{term literal, None}$ )  $\# \Gamma =$ 
  (map gtrailelem-of-trailelem (( $L_G, \text{None}$ )  $\# \Gamma_G$ ) :: ('f, 'v) SCL-FOL.trail)
  by (simp add:  $\Gamma$ -def gtrailelem-of-trailelem-def)

```

```

moreover have  $D \cdot \gamma = \text{cls-of-gcls } D_G$ 

```



**unfolding**  $D$ -def by *simp*

**ultimately have** *trail-false-cls*  
 $(\text{map } g\text{trailelem-of-trailelem } ((L_G, \text{None}) \# \Gamma_G) :: (f, 'v) \text{ SCL-FOL.trail})$   
 $(\text{cls-of-gcls } D_G)$   
**using**  $\langle \text{SCL-FOL.trail-false-cls } ((\text{lit-of-glit } L_G, \text{None}) \# \Gamma) (D \cdot \gamma) \rangle$   
**unfolding** *SCL-FOL.trail-false-cls-def trail-false-cls-def*  
**unfolding** *SCL-FOL.trail-false-lit-def trail-false-lit-def*  
**by** *metis*

**hence** *trail-false-cls*  $((L_G, \text{None}) \# \Gamma_G) D_G$   
**using** *trail-false-cls-nonground-iff-trail-false-cls-ground* by *blast*

**hence** *trail-false-cls*  $\Gamma_G \{\#K_G \in\# D_G. K_G \neq - L_G\# \}$   
**unfolding** *trail-false-cls-def trail-false-lit-def*  
**by** *auto*

**moreover have** *ord-res.is-maximal-lit*  $(- L_G) D_G$   
**unfolding** *linorder-lit.is-maximal-in-mset-iff*  
**proof**  $(\text{intro } \text{conjI } \text{ballI } \text{impI})$   
**show**  $- L_G \in\# D_G$   
**using**  $\langle D_G \mid \in \mid N_G \mid \cup \mid U_G \rangle \langle \text{trail-false-cls } ((L_G, \text{None}) \# \Gamma_G) D_G \rangle$   
*subtrail-falseI*  
*nex-false-clause-wrt- $\Gamma_G$*   
**unfolding** *bex-trail-false-cls-simp*  
**by** *blast*

**next**  
**fix**  $K_G$  **assume**  $K_G \in\# D_G$  **and**  $K_G \neq - L_G$   
**hence** *trail-false-lit*  $\Gamma_G K_G$   
**using**  $\langle \text{trail-false-cls } \Gamma_G \{\#K_G \in\# D_G. K_G \neq - L_G\# \} \rangle$   
**unfolding** *trail-false-cls-def* by *simp*  
**hence** *trail-defined-lit*  $\Gamma_G K_G$   
**by**  $(\text{simp add: } \text{trail-defined-lit-iff-true-or-false})$   
**hence** *atm-of*  $K_G \mid \in \mid \text{trail-atms } \Gamma_G$   
**unfolding** *trail-defined-lit-iff-trail-defined-atm* .  
**hence** *atm-of*  $K_G \prec_t \text{atm-of } L_G$   
**using** *ball-lt-atm- $L_G$*  by *metis*  
**hence**  $K_G \prec_l - L_G$   
**by**  $(\text{cases } L_G; \text{cases } K_G) \text{ simp-all}$   
**thus**  $\neg - L_G \prec_l K_G$   
**by** *order*

**qed**

**moreover have**  $\neg \text{trail-defined-lit } \Gamma_G (- L_G)$   
**by**  $(\text{metis } \text{atm-of-uminus } \text{linorder-trm.less-irrefl } \text{that}(2) \text{ trail-defined-lit-iff-trail-defined-atm})$

**ultimately have** *clause-could-propagate*  $\Gamma_G D_G (- L_G)$   
**unfolding** *clause-could-propagate-def* by *argo*

```

hence  $\exists C \in |N_G \cup U_G$ . clause-could-propagate  $\Gamma_G C (- L_G)$ 
using  $\langle D_G \in |N_G \cup U_G \rangle$  by metis

hence False
using nex-clause-that-propagate
unfolding bex-clause-could-propagate-simp
by contradiction

thus ?thesis .
qed
qed

show ?thesis
using step unfolding SG-def SG'-def
proof (cases NG (UG, F, ΓG, CG) (UG', F', ΓG', CG') rule: ord-res-11.cases)
case step-hyps: (decide-neg AG)

define  $A :: ('f, 'v)$  term where
 $A = \text{term-of-gterm } A_G$ 

let  $?f = \text{gtrailelem-of-trailelem}$ 
have  $\Gamma' = \text{map } ?f \Gamma_G'$ 
unfolding  $\Gamma'$ -def ..
also have  $\dots = \text{map } ?f ((\text{Neg } A_G, \text{None}) \# \Gamma_G)$ 
unfolding  $\langle \Gamma_G' = (\text{Neg } A_G, \text{None}) \# \Gamma_G \rangle$  ..
also have  $\dots = ?f (\text{Neg } A_G, \text{None}) \# \text{map } ?f \Gamma_G$ 
unfolding list.map ..
also have  $\dots = ?f (\text{Neg } A_G, \text{None}) \# \Gamma$ 
unfolding  $\Gamma$ -def ..
also have  $\dots = (\text{lit-of-glit } (\text{Neg } A_G), \text{None}) \# \Gamma$ 
unfolding gtrailelem-of-trailelem-def prod.case option.map ..
also have  $\dots = (\text{Neg } (\text{term-of-gterm } A_G), \text{None}) \# \Gamma$ 
unfolding lit-of-glit-def literal.map ..
also have  $\dots = (\text{Neg } A, \text{None}) \# \Gamma$ 
unfolding A-def ..
finally have  $\Gamma' = \text{decide-lit } (\text{Neg } A) \# \Gamma$ 
unfolding decide-lit-def .

have  $U' = U$ 
unfolding  $U'$ -def  $\langle U_G' = U_G \rangle$   $U$ -def ..

have  $\neg \text{Ex } (\text{scl-fol.conflict } N \beta S)$ 
using  $\langle \neg \text{fBex } (\text{iefac } \mathcal{F} \mid (N_G \cup U_G)) (\text{trail-false-cls } \Gamma_G) \rangle$  nex-conflict-if-nbex-trail-false
by metis

moreover have scl-fol.reasonable-scl N β S S'
unfolding scl-fol.reasonable-scl-def
proof (intro conjI impI notI ; (elim exE) ?)

```

```

have scl-fol.decide  $N \beta S S'$ 
  unfolding  $S\text{-def } S'\text{-def } \langle U' = U \rangle C\text{-def } C'\text{-def } \langle C_G = \text{None} \rangle \langle C_{G'} = \text{None} \rangle$ 
option.map
proof (rule scl-fol.decideI')
  show is-ground-lit (Neg A ·l Var)
    by (simp add: A-def)
next
  have  $\forall x \in | \text{trail-atms } \Gamma_G. x \prec_t A_G$ 
    using step-hyps linorder-trm.is-least-in-ffilter-iff by simp
  hence  $A_G \notin | \text{trail-atms } \Gamma_G$ 
    by blast
  hence  $A_G \notin \text{atm-of } \text{'fst' } \text{'set } \Gamma_G$ 
    unfolding fset-trail-atms .
  hence  $\text{term-of-gterm } A_G \notin \text{term-of-gterm } \text{'atm-of } \text{'fst' } \text{'set } \Gamma_G$ 
    using inj-image-mem-iff inj-term-of-gterm by metis
  hence  $\text{term-of-gterm } A_G \notin \text{set } (\text{map } (\lambda x. \text{term-of-gterm } (\text{atm-of } (\text{fst } x))))$ 
 $\Gamma_G$ )
    unfolding image-set list.map-comp comp-def .
  hence  $A \notin \text{set } (\text{map } (\lambda x. \text{atm-of } (\text{lit-of-glit } (\text{fst } x)))) \Gamma_G$ 
    unfolding A-def atm-of-lit-of-glit-conv .
  hence  $A \notin \text{atm-of } \text{'fst' } \text{'set } \Gamma$ 
unfolding image-set list.map-comp comp-def  $\Gamma\text{-def } \text{gtrailelem-of-trailelem-def}$ 
   $\text{fst-case-prod-simp}$  .
  hence  $A \notin | \text{trail-atms } \Gamma$ 
    unfolding fset-trail-atms .
  hence  $\neg \text{trail-defined-lit } \Gamma$  (Neg A ·l Var)
    by (simp add: trail-defined-lit-iff-trail-defined-atm)
  thus  $\neg \text{SCL-FOL.trail-defined-lit } \Gamma$  (Neg A ·l Var)
    by (simp add: SCL-FOL.trail-defined-lit-def trail-defined-lit-def)
next
  have  $A_G \in | \text{atms-of-clss } (N_G \cup U_G)$ 
    using step-hyps linorder-trm.is-least-in-ffilter-iff by blast
  hence  $A_G \in | \text{atms-of-clss } N_G$ 
    unfolding  $\langle \text{atms-of-clss } (N_G \cup U_G) = \text{atms-of-clss } N_G \rangle$  .
  hence  $A_G \preceq_t \beta_G$ 
    using ball-le- $\beta_G$  by metis
  moreover have  $\text{gterm-of-term } A = A_G$ 
    by (simp add: A-def)
  moreover have  $\text{gterm-of-term } \beta = \beta_G$ 
    by (simp add:  $\beta\text{-def}$ )
  ultimately have  $\text{gterm-of-term } A \preceq_t \text{gterm-of-term } \beta$ 
    by argo
  thus  $\text{less-B}^{==} (\text{atm-of } (\text{Neg } A) \cdot a \text{ Var}) \beta$ 
    using inj-term-of-gterm[THEN injD]
    by (auto simp: less-B-def A-def  $\beta\text{-def}$ )
next
  show  $\Gamma' = \text{trail-decide } \Gamma$  (Neg A ·l Var)
    using  $\langle \Gamma' = \text{decide-lit } (\text{Neg } A) \# \Gamma \rangle$ 
    unfolding subst-lit-id-subst .

```

```

qed

thus scl-fol.scl  $N \beta S S'$ 
  unfolding scl-fol.scl-def by argo
next
fix  $S'' :: ('f, 'v) SCL-FOL.state$ 
assume scl-fol.conflict  $N \beta S' S''$ 

moreover have  $\nexists S''. scl-fol.conflict N \beta S' S''$ 
proof –
  have  $\neg Ex (scl-fol.conflict N \beta ((lit-of-glit (Neg A_G), None) \# \Gamma, U, C))$ 
  proof (rule nex-conflict-if-no-clause-could-propagate-comp)
    show  $\neg fBex (iefac \mathcal{F} \mid \uparrow (N_G \mid \cup \mid U_G)) (trail-false-cls \Gamma_G)$ 
    using step-hyps by argo
  next
    show  $\forall x \mid \in \mid trail-atms \Gamma_G. x \prec_t atm-of (Neg A_G)$ 
    unfolding literal.sel
    using step-hyps linorder-trm.is-least-in-fset-iff by simp
  next
    show  $\neg (\exists C \mid \in \mid iefac \mathcal{F} \mid \uparrow (N_G \mid \cup \mid U_G). clause-could-propagate \Gamma_G C$ 
    ( $- Neg A_G$ ))
    using step-hyps by simp
  qed
  moreover have  $lit-of-glit (Neg A_G) = Neg A$ 
    unfolding A-def lit-of-glit-def literal.map ..
  ultimately show ?thesis
    unfolding S'-def  $\langle \Gamma' = decide-lit (Neg A) \# \Gamma \rangle decide-lit-def$ 
    using C'-def C-def  $\langle U' = U \rangle step-hyps(1,4)$  by argo
qed

ultimately show False
  by metis
qed

ultimately show ?thesis
  unfolding scl-fol.regular-scl-def by argo
next
case step-hyps: (decide-pos A_G)

define  $A :: ('f, 'v) term$  where
   $A = term-of-gterm A_G$ 

let  $?f = gtrailelem-of-trailelem$ 
have  $\Gamma' = map ?f \Gamma_G'$ 
  unfolding  $\Gamma'-def ..$ 
also have  $... = map ?f ((Pos A_G, None) \# \Gamma_G)$ 
  unfolding  $\langle \Gamma_G' = (Pos A_G, None) \# \Gamma_G \rangle ..$ 
also have  $... = ?f (Pos A_G, None) \# map ?f \Gamma_G$ 
  unfolding list.map ..

```

```

also have ... = ?f (Pos AG, None) # Γ
  unfolding Γ-def ..
also have ... = (lit-of-glit (Pos AG), None) # Γ
  unfolding prod.case option.map gtrailelem-of-trailelem-def ..
also have ... = (Pos (term-of-gterm AG), None) # Γ
  unfolding lit-of-glit-def literal.map ..
also have ... = (Pos A, None) # Γ
  unfolding A-def ..
finally have Γ' = decide-lit (Pos A) # Γ
  unfolding decide-lit-def .

have U' = U
  unfolding U'-def ⟨UG' = UG⟩ U-def ..

have ¬ Ex (scl-fol.conflict N β S)
  using step-hyps nex-conflict-if-nbex-trail-false by metis

moreover have scl-fol.reasonable-scl N β S S'
  unfolding scl-fol.reasonable-scl-def
proof (intro conjI impI notI ; (elim exE) ?)
  have scl-fol.decide N β S S'
    unfolding S-def S'-def ⟨U' = U⟩ C-def C'-def ⟨CG = None⟩ ⟨CG' = None⟩
option.map
  proof (rule scl-fol.decideI')
    show is-ground-lit (Pos A ·l Var)
      by (simp add: A-def)
  next
    have ∀ x |∈| trail-atms ΓG. x <t AG
      using step-hyps linorder-trm.is-least-in-ffilter-iff by simp
    hence AG |∉| trail-atms ΓG
      by blast
    hence AG ∉ atm-of 'fst ' set ΓG
      unfolding fset-trail-atms .
    hence term-of-gterm AG ∉ term-of-gterm 'atm-of 'fst ' set ΓG
      using inj-image-mem-iff inj-term-of-gterm by metis
    hence term-of-gterm AG ∉ set (map (λx. term-of-gterm (atm-of (fst x))))
ΓG)
      unfolding image-set list.map-comp comp-def .
    hence A ∉ set (map (λx. atm-of (lit-of-glit (fst x))) ΓG)
      unfolding A-def atm-of-lit-of-glit-conv .
    hence A ∉ atm-of 'fst ' set Γ
  unfolding image-set list.map-comp comp-def Γ-def gtrailelem-of-trailelem-def
    fst-case-prod-simp .
  hence A |∉| trail-atms Γ
    unfolding fset-trail-atms .
  hence ¬ trail-defined-lit Γ (Pos A ·l Var)
    by (simp add: trail-defined-lit-iff-trail-defined-atm)
  thus ¬ SCL-FOL.trail-defined-lit Γ (Pos A ·l Var)
    by (simp add: SCL-FOL.trail-defined-lit-def trail-defined-lit-def)

```

```

next
  have  $A_G \in | \text{atms-of-clss } (N_G \cup U_G)$ 
    using step-hyps linorder-trm.is-least-in-filter-iff by simp
  hence  $A_G \in | \text{atms-of-clss } N_G$ 
    unfolding  $\langle \text{atms-of-clss } (N_G \cup U_G) = \text{atms-of-clss } N_G \rangle$  .
  hence  $A_G \preceq_t \beta_G$ 
    using ball-le-beta_G by metis
  moreover have gterm-of-term  $A = A_G$ 
    by (simp add: A-def)
  moreover have gterm-of-term  $\beta = \beta_G$ 
    by (simp add: beta-def)
  ultimately have gterm-of-term  $A \preceq_t$  gterm-of-term  $\beta$ 
    by argo
  thus  $\text{less-B} = (\text{atm-of } (\text{Pos } A) \cdot a \text{ Var}) \beta$ 
    using inj-term-of-gterm[THEN injD]
    by (auto simp: less-B-def A-def beta-def)
next
  show  $\Gamma' = \text{trail-decide } \Gamma (\text{Pos } A \cdot l \text{ Var})$ 
    using  $\Gamma' = \text{decide-lit } (\text{Pos } A) \# \Gamma$ 
    unfolding subst-lit-id-subst .
qed

thus scl-fol.scl  $N \beta S S'$ 
  unfolding scl-fol.scl-def by argo
next
fix  $S'' :: ('f, 'v) \text{SCL-FOL.state}$ 
assume scl-fol.conflict  $N \beta S' S''$ 

moreover have  $\nexists S''. \text{scl-fol.conflict } N \beta S' S''$ 
proof -
  have  $\neg \text{Ex } (\text{scl-fol.conflict } N \beta ((\text{lit-of-glit } (\text{Pos } A_G), \text{None}) \# \Gamma, U, C))$ 
  proof (rule nex-conflict-if-no-clause-could-propagate-comp)
    show  $\neg \text{fBex } (\text{iefac } \mathcal{F} \mid \uparrow (N_G \cup U_G)) (\text{trail-false-cls } \Gamma_G)$ 
      using step-hyps by argo
  next
    show  $\forall x \in | \text{trail-atms } \Gamma_G. x \prec_t \text{atm-of } (\text{Pos } A_G)$ 
      unfolding literal.sel
      using step-hyps linorder-trm.is-least-in-filter-iff by simp
  next
    have clause-could-propagate  $\Gamma_G C (\text{Neg } A_G) \implies \text{trail-false-cls } \Gamma_G' C$  for
  C
    unfolding  $\langle \Gamma_G' = (\text{Pos } A_G, \text{None}) \# \Gamma_G \rangle$ 
    using trail-false-if-could-have-propagated by fastforce

  thus  $\neg (\exists C \in | \text{iefac } \mathcal{F} \mid \uparrow (N_G \cup U_G). \text{clause-could-propagate } \Gamma_G C (\text{Neg } A_G))$ 
    unfolding uminus-Pos
    using step-hyps by metis
qed

```

```

moreover have lit-of-glit (Pos AG) = Pos A
  unfolding A-def lit-of-glit-def literal.map ..
ultimately show ?thesis
  unfolding S'-def ⟨Γ' = decide-lit (Pos A) # Γ⟩ decide-lit-def
  using C'-def C-def ⟨U' = U⟩ step-hyps(1) step-hyps(3) by argo
qed

ultimately show False by metis
qed

ultimately show ?thesis
  unfolding scl-fol.regular-scl-def by argo
next
  case step-hyps: (propagate AG CG)

  have CG |∈| iefac F |' (NG |∪| UG) and CG-prop: clause-could-propagate ΓG
  CG (Pos AG)
  using step-hyps linorder-cls.is-least-in-fset-iff by simp-all

  define A :: ('f, 'v) term where
    A = term-of-gterm AG

  define C :: ('f, 'v) term clause where
    C = cls-of-gcls CG

  have ord-res.is-maximal-lit (Pos AG) CG and trail-false-cls ΓG {#K ∈# CG.
  K ≠ Pos AG#}
  using ⟨clause-could-propagate ΓG CG (Pos AG)⟩
  unfolding clause-could-propagate-def by metis+

  then obtain CG' where CG = add-mset (Pos AG) CG'
  by (metis linorder-lit.is-maximal-in-mset-iff mset-add)

  define C' :: ('f, 'v) term clause where
    C' = cls-of-gcls CG'

  let ?f = gtrailelem-of-trailelem
  have Γ' = map ?f ΓG'
  unfolding Γ'-def ..
  also have ... = map ?f ((Pos AG, Some (efac CG)) # ΓG)
  unfolding ⟨ΓG' = (Pos AG, Some -) # ΓG⟩ ..
  also have ... = ?f (Pos AG, Some (efac CG)) # map ?f ΓG
  unfolding list.map ..
  also have ... = ?f (Pos AG, Some (efac CG)) # Γ
  unfolding Γ-def ..
  also have ... = (lit-of-glit (Pos AG),
  Some (cls-of-gcls {#K ∈# iefac CG. K ≠ Pos AG#}, lit-of-glit (Pos AG),
  Var)) # Γ
  unfolding gtrailelem-of-trailelem-def prod.case option.map ..

```

**also have** ... = (*lit-of-glit* (*Pos A<sub>G</sub>*),  
*Some (cls-of-gcls {#K ∈# add-mset (Pos A<sub>G</sub>) {#K ∈# C<sub>G</sub>. K ≠ Pos A<sub>G</sub>#}.  
K ≠ Pos A<sub>G</sub>#},*  
*lit-of-glit (Pos A<sub>G</sub>), Var)) # Γ*  
**proof** –  
**have** *is-pos (Pos A<sub>G</sub>)*  
**by** *simp*  
  
**moreover have** *linorder-lit.is-maximal-in-mset C<sub>G</sub> (Pos A<sub>G</sub>)*  
**using** *C<sub>G</sub>-prop unfolding clause-could-propagate-def by argo*  
  
**ultimately show** *?thesis*  
**using** *efac-spec-if-pos-lit-is-maximal by metis*  
**qed**  
**also have** ... = (*lit-of-glit (Pos A<sub>G</sub>),*  
*Some (cls-of-gcls {#K ∈# C<sub>G</sub>. K ≠ Pos A<sub>G</sub>#}, lit-of-glit (Pos A<sub>G</sub>), Var))*  
# Γ  
**by** (*simp add: filter-filter-mset*)  
**also have** ... = (*Pos A, Some (cls-of-gcls {#K ∈# C<sub>G</sub>. K ≠ Pos A<sub>G</sub>#}, Pos*  
*A, Var)) # Γ*  
**by** (*simp add: A-def lit-of-glit-def*)  
**also have** ... = (*Pos A, Some (cls-of-gcls {#L ∈# C<sub>G</sub>. lit-of-glit L ≠ Pos*  
*A#}, Pos A, Var)) # Γ*  
**by** (*metis A-def glit-of-lit-lit-of-glit lit-of-glit-def literal.simps(9)*)  
**also have** ... = (*Pos A, Some ({#L ∈# cls-of-gcls C<sub>G</sub>. L ≠ Pos A#}, Pos A,*  
*Var)) # Γ*  
**unfolding** *cls-of-gcls-def*  
**unfolding** *image-mset-filter-mset-swap[of lit-of-glit λL. L ≠ Pos A C<sub>G</sub>]*  
**unfolding** *cls-of-gcls-def[symmetric] ..*  
**also have** ... = (*Pos A ·l Var, Some ({#L ∈# cls-of-gcls C<sub>G</sub>. L ≠ Pos A#},*  
*Pos A, Var)) # Γ*  
**by** *simp*  
**also have** ... = (*Pos A ·l Var, Some ({#L ∈# C. L ≠ Pos A#}, Pos A, Var))*  
# Γ  
**unfolding** *C-def ..*  
**finally have**  $\Gamma' = \text{propagate-lit } (Pos A) \{ \#L \in \# C. L \neq Pos A \# \} Var \# \Gamma$   
**unfolding** *propagate-lit-def .*  
  
**have**  $U' = U$   
**unfolding** *U'-def ⟨U<sub>G</sub>' = U<sub>G</sub>⟩ U-def ..*  
  
**obtain**  $C_{G0}$  **where**  $C_{G0} \mid \in \mid N_G \mid \cup \mid U_G$  **and**  $C_G = \text{iefac } \mathcal{F} C_{G0}$   
**using**  $\langle C_G \mid \in \mid \text{iefac } \mathcal{F} \mid \uparrow \mid (N_G \mid \cup \mid U_G) \rangle$  **by** *blast*  
  
**define**  $C_0 :: ('f, 'v)$  *term clause* **where**  
 $C_0 = \text{cls-of-gcls } C_{G0}$   
  
**have** *ord-res.is-maximal-lit (Pos A<sub>G</sub>) C<sub>G0</sub>*  
**using**  $\langle \text{ord-res.is-maximal-lit } (Pos A_G) C_G \rangle \langle C_G = \text{iefac } \mathcal{F} C_{G0} \rangle$



```

by (metis iefac-def linorder-lit.is-maximal-in-mset-iff set-mset-efac)

have  $\neg \text{Ex } (scl\text{-fol.conflict } N \beta S)$ 
  using step-hyps nex-conflict-if-nbex-trail-false by metis

moreover have scl-fol.reasonable-scl  $N \beta S S'$ 
  unfolding scl-fol.reasonable-scl-def
  proof (intro conjI impI notI ; (elim exE) ?)
    have scl-fol.propagate  $N \beta S S'$ 
      unfolding S-def S'-def  $\langle U' = U \rangle$  C-def C'-def  $\langle C_G = \text{None} \rangle$   $\langle C_G' = \text{None} \rangle$ 
    option.map
    proof (rule scl-fol.propagateI')
      show  $C_0 \in N \cup U$ 
        unfolding C0-def N-def U-def
        using  $\langle C_{G0} \in N_G \cup U_G \rangle$ 
        by blast
    next
      show is-ground-cls  $(C_0 \cdot \text{Var})$ 
        by (simp add: C0-def)

    have Pos  $A \in\# C_0$ 
      unfolding A-def C0-def
      by (metis (no-types, lifting)  $\langle C_G = \text{iefac } \mathcal{F} C_{G0} \rangle$  ord-res.is-maximal-lit
        (Pos AG) CG)
      cls-of-gcls-def iefac-def image-eqI linorder-lit.is-maximal-in-mset-iff
      lit-of-glit-def literal.map(1) multiset.set-map set-mset-efac)

  then show  $C_0 = \text{add-mset } (Pos A) (C_0 - \{\#Pos A\})$ 
    by simp

  have  $A_G \in \text{atms-of-cls } (N_G \cup U_G)$ 
    using step-hyps linorder-trm.is-least-in-filter-iff by simp
  hence  $A_G \in \text{atms-of-cls } N_G$ 
    unfolding  $\langle \text{atms-of-cls } (N_G \cup U_G) = \text{atms-of-cls } N_G \rangle$  .
  hence  $A_G \preceq_t \beta_G$ 
    using ball-le-βG by metis
  moreover have gterm-of-term  $A = A_G$ 
    by (simp add: A-def)
  moreover have gterm-of-term  $\beta = \beta_G$ 
    by (simp add: β-def)
  ultimately have gterm-of-term  $A \preceq_t$  gterm-of-term  $\beta$ 
    by argo
  hence less-B==  $A \beta$ 
    by (auto simp: less-B-def A-def β-def)

  show  $\forall K \in\# C_0 \cdot \text{Var}. \text{less-B}^{\text{==}} (\text{atm-of } K) \beta$ 
    unfolding subst-cls-id-subst
  proof (intro ballI)
    fix  $K :: (f, 'v) \text{Term.term literal}$ 

```

**assume**  $K \in\# C_0$   
**then obtain**  $K_G$  **where**  $K_G \in\# C_{G0}$  **and**  $K\text{-def}: K = \text{lit-of-glit } K_G$   
**unfolding**  $C_0\text{-def } \text{cls-of-gcls-def}$  **by** *blast*

**have**  $K_G \preceq_l \text{Pos } A_G$   
**using**  $\langle \text{ord-res.is-maximal-lit } (\text{Pos } A_G) C_{G0} \rangle \langle K_G \in\# C_{G0} \rangle$   
**unfolding**  $\text{linorder-lit.is-maximal-in-mset-iff}$  **by** *fastforce*

**hence**  $\text{atm-of } K_G \preceq_t A_G$   
**by**  $(\text{metis literal.collapse}(1) \text{ literal.collapse}(2) \text{ literal.sel}(1) \text{ ord-res.less-lit-simps}(1) \text{ ord-res.less-lit-simps}(4) \text{ reflclp-iff})$

**hence**  $\text{less-B}^{==} (\text{atm-of } K) A$   
**by**  $(\text{auto simp: less-B-def } K\text{-def } A\text{-def } \text{atm-of-lit-of-glit-conv})$

**then show**  $\text{less-B}^{==} (\text{atm-of } K) \beta$   
**using**  $\langle \text{less-B}^{==} A \beta \rangle$  **by** *order*

**qed**

**show**  $\{\#K \in\# C_0. K \neq \text{Pos } A\#\} = \{\#K \in\# \text{remove1-mset } (\text{Pos } A) C_0. K \cdot l \text{Var} \neq \text{Pos } A \cdot l \text{Var}\#\}$   
**by** *simp*

**show**  $\{\#K \in\# \text{remove1-mset } (\text{Pos } A) C_0. K = \text{Pos } A\#\} = \{\#K \in\# \text{remove1-mset } (\text{Pos } A) C_0. K \cdot l \text{Var} = \text{Pos } A \cdot l \text{Var}\#\}$   
**by** *simp*

**have**  $\text{trail-false-cls } \Gamma_G (\{\#K_G \in\# C_{G0}. K_G \neq \text{Pos } A_G\#\})$   
**by**  $(\text{smt } (\text{verit}, \text{ccfv-threshold}) \langle C_G = \text{iefac } \mathcal{F} C_{G0} \rangle \langle \text{trail-false-cls } \Gamma_G \{\#K \in\# C_G. K \neq \text{Pos } A_G\#\} \rangle \text{iefac-def mem-Collect-eq set-mset-efac set-mset-filter trail-false-cls-def})$

**moreover have**  $(\text{cls-of-gcls } \{\#K_G \in\# C_{G0}. K_G \neq \text{Pos } A_G\#\} :: ('f, 'v) \text{term clause}) = \{\#K \in\# C_0. K \neq \text{Pos } A\#\}$   
**by**  $(\text{smt } (\text{verit}) A\text{-def } C_0\text{-def } \text{cls-of-gcls-def } \text{filter-mset-cong0 } \text{glit-of-lit-lit-of-glit } \text{image-mset-filter-mset-swap } \text{lit-of-glit-def } \text{literal.map}(1))$

**ultimately have**  $\text{trail-false-cls } \Gamma (\{\#K \in\# C_0. K \neq \text{Pos } A\#\} \cdot \text{Var})$   
**unfolding** *subst-cls-id-subst*  
**using**  $\text{trail-false-cls-nonground-iff-trail-false-cls-ground}[THEN \text{iffD2}]$   
**by**  $(\text{metis } \Gamma\text{-def})$

**thus**  $\text{SCL-FOL.trail-false-cls } \Gamma (\{\#K \in\# C_0. K \neq \text{Pos } A\#\} \cdot \text{Var})$   
**unfolding**  $\text{SCL-FOL.trail-false-cls-def } \text{trail-false-cls-def}$   
**unfolding**  $\text{SCL-FOL.trail-false-lit-def } \text{trail-false-lit-def}$   
**by** *argo*

**have**  $\forall x \mid \in \mid$  *trail-atms*  $\Gamma_G. x \prec_t A_G$   
**using** *step-hyps linorder-trm.is-least-in-filter-iff* **by** *simp*  
**hence**  $A_G \mid \notin \mid$  *trail-atms*  $\Gamma_G$   
**by** *blast*  
**hence**  $A_G \notin$  *atm-of* ‘*fst*’ *set*  $\Gamma_G$   
**unfolding** *fset-trail-atms* .  
**hence** *term-of-gterm*  $A_G \notin$  *term-of-gterm* ‘*atm-of*’ ‘*fst*’ *set*  $\Gamma_G$   
**using** *inj-image-mem-iff inj-term-of-gterm* **by** *metis*  
**hence** *term-of-gterm*  $A_G \notin$  *set* (*map* ( $\lambda x. \text{term-of-gterm} (\text{atm-of} (\text{fst } x))$ ))  
 $\Gamma_G$ )  
**unfolding** *image-set list.map-comp comp-def* .  
**hence**  $A \notin$  *set* (*map* ( $\lambda x. \text{atm-of} (\text{lit-of-glit} (\text{fst } x))$ ))  $\Gamma_G$ )  
**unfolding** *A-def atm-of-lit-of-glit-conv* .  
**hence**  $A \notin$  *atm-of* ‘*fst*’ *set*  $\Gamma$   
**unfolding** *image-set list.map-comp comp-def*  $\Gamma$ -*def gtrailelem-of-trailelem-def*  
*fst-case-prod-simp* .  
**hence**  $A \mid \notin \mid$  *trail-atms*  $\Gamma$   
**unfolding** *fset-trail-atms* .  
**hence**  $\neg$  *trail-defined-lit*  $\Gamma$  (*Pos*  $A \cdot l$  *Var*)  
**by** (*simp add: trail-defined-lit-iff-trail-defined-atm*)  
**thus**  $\neg$  *SCL-FOL.trail-defined-lit*  $\Gamma$  (*Pos*  $A \cdot l$  *Var*)  
**by** (*simp add: SCL-FOL.trail-defined-lit-def trail-defined-lit-def*)  
  
**have** *set-mset* (*add-mset* (*Pos*  $A$ )  $\{\#K \in \# \text{remove1-mset} (\text{Pos } A) C_0. K$   
 $= \text{Pos } A\#\}$ ) =  
 $\{\text{Pos } A\}$   
**by** (*smt (verit) Collect-cong insert-compr mem-Collect-eq set-mset-add-mset-insert*  
*set-mset-filter singletonD*)  
**hence** *is-unifier* *Var* (*atm-of* ‘*set-mset*  
(*add-mset* (*Pos*  $A$ )  $\{\#K \in \# \text{remove1-mset} (\text{Pos } A) C_0. K = \text{Pos } A\#\}$ ))  
**by** (*metis (no-types, lifting) finite-imageI finite-set-mset image-empty*  
*image-insert*  
*is-unifier-alt singletonD*)  
**hence** *is-unifiers* *Var*  $\{\text{atm-of}$  ‘*set-mset*  
(*add-mset* (*Pos*  $A$ )  $\{\#K \in \# \text{remove1-mset} (\text{Pos } A) C_0. K = \text{Pos } A\#\}$ )\}  
**unfolding** *SCL-FOL.is-unifiers-def* **by** *simp*  
**thus** *SCL-FOL.is-ingu* *Var*  $\{\text{atm-of}$  ‘*set-mset*  
(*add-mset* (*Pos*  $A$ )  $\{\#K \in \# \text{remove1-mset} (\text{Pos } A) C_0. K = \text{Pos } A\#\}$ )\}  
**unfolding** *SCL-FOL.is-ingu-def* **by** *simp*  
  
**have**  $\{\#K \in \# C_{G_0}. K \neq \text{Pos } A_G\# \} = \{\#K \in \# C_G. K \neq \text{Pos } A_G\# \}$   
**using**  $\langle \text{ord-res.is-maximal-lit} (\text{Pos } A_G) C_G \rangle$   
**using**  $\langle \text{ord-res.is-maximal-lit} (\text{Pos } A_G) C_{G_0} \rangle$   
**unfolding**  $\langle C_G = \text{iefac } \mathcal{F} C_{G_0} \rangle$   
**by** (*metis add-mset-remove-trivial efac-spec-if-pos-lit-is-maximal*  
*ex1-efac-eq-factoring-chain factorizable-if-neq-efac iefac-def literal.disc(1)*)  
  
**hence**  $\{\#K \in \# C_0. K \neq \text{Pos } A\# \} = \{\#K \in \# C. K \neq \text{Pos } A\# \}$   
**unfolding**  $C_0$ -*def*  $C$ -*def*

```

by (smt (verit, ccfv-SIG) A-def cls-of-gcls-def filter-mset-cong0 glit-of-lit-lit-of-glit
  image-mset-filter-mset-swap lit-of-glit-def literal.map(1))

thus  $\Gamma' = \text{trail-propagate } \Gamma \text{ (Pos } A \cdot l \text{ Var) } (\{\#K \in\# C_0. K \neq \text{Pos } A\# \} \cdot$ 
Var) Var
  unfolding subst-cls-id-subst subst-lit-id-subst
  using  $\langle \Gamma' = \text{propagate-lit (Pos } A) \{\#L \in\# C. L \neq \text{Pos } A\# \} \text{ Var } \# \Gamma \rangle$ 
  by argo
qed

thus scl-fol.scl N  $\beta$  S S'
  unfolding scl-fol.scl-def by argo
next
fix S'' :: ('f, 'v) SCL-FOL.state
assume scl-fol.decide N  $\beta$  S S'
thus False
  unfolding S-def S'-def
proof (cases N  $\beta$  ( $\Gamma, U, C$ ) ( $\Gamma', U', C'$ ) rule: scl-fol.decide.cases)
  case (decideI L  $\gamma$ )
  show False
    using  $\langle \Gamma' = \text{decide-lit (L } \cdot l \text{ } \gamma) \# \Gamma \rangle$ 
    using  $\langle \Gamma' = \text{propagate-lit (Pos } A) \{\#L \in\# C. L \neq \text{Pos } A\# \} \text{ Var } \# \Gamma \rangle$ 
    unfolding decide-lit-def propagate-lit-def
    by blast
  qed
qed

ultimately show ?thesis
  unfolding scl-fol.regular-scl-def by argo
next
case step-hyps: (conflict CG)

have  $\Gamma' = \Gamma$ 
  unfolding  $\Gamma'$ -def  $\langle \Gamma_{G'} = \Gamma_G \rangle$   $\Gamma$ -def ..

have  $U' = U$ 
  unfolding  $U'$ -def  $\langle U_{G'} = U_G \rangle$   $U$ -def ..

have
  CG-in: CG  $\in$  | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$ | (NG  $\cup$  UG) and
  CG-false: trail-false-cls  $\Gamma_G$  CG and
  CG-lt:  $\forall E_G \in$  | $\in$ | iefac  $\mathcal{F}$  | $\uparrow$ | (NG  $\cup$  UG). EG  $\neq$  CG  $\longrightarrow$  trail-false-cls  $\Gamma_G$  EG
 $\longrightarrow$  CG  $\prec_c$  EG
  using  $\langle \text{linorder-cls.is-least-in-fset} - C_G \rangle$ 
  unfolding atomize-conj linorder-cls.is-least-in-filter-iff by argo

have  $\nexists L. \text{is-pos } L \wedge \text{ord-res.is-maximal-lit } L \text{ } C_G$ 
proof (rule notI , elim exE conjE)
  fix L :: 'f gliteral

```

```

assume is-pos  $L$  and  $C_G$ -max-lit: ord-res.is-maximal-lit  $L$   $C_G$ 

have  $\{\#\}$   $|\notin|$  iefac  $\mathcal{F}$   $|\uparrow|$   $(N_G \mid\cup\mid U_G)$ 
  using  $C_G$ -lt
  by (metis (full-types)  $C_G$ -max-lit bot-fset.rep-eq fBex-fempty linorder-cls.leD
    linorder-lit.is-maximal-in-mset-iff mempty-lesseq-cls set-mset-empty
    trail-false-cls-mempty)

have trail-false-lit  $\Gamma_G$   $L$ 
  using  $C_G$ -max-lit  $C_G$ -false
  unfolding linorder-lit.is-maximal-in-mset-iff trail-false-cls-def
  by metis

then obtain  $Ln$   $\Gamma_{G0}$  where  $\Gamma_G = Ln \# \Gamma_{G0}$ 
  unfolding trail-false-lit-def
  by (metis (no-types) List.insert-def image-iff insert-Nil neq-Nil-conv)

moreover have
  AAA:  $\forall x$   $xs$ .  $\Gamma_G = x \# xs \longrightarrow$ 
     $((snd\ x \neq None) \longleftrightarrow fBex\ (iefac\ \mathcal{F}\ |\uparrow|\ (N_G\ \mid\cup\mid\ U_G))\ (trail-false-cls\ (x\ \#$ 
 $xs))) \wedge$ 
     $(snd\ x \neq None \longrightarrow is-pos\ (fst\ x)) \wedge$ 
     $(\forall x \in set\ xs.\ snd\ x = None)$  and
  BBB:  $(\forall \Gamma_1\ Ln\ \Gamma_0.$   $\Gamma_G = \Gamma_1 @ Ln \# \Gamma_0 \longrightarrow snd\ Ln = None \longrightarrow$ 
     $\neg fBex\ (iefac\ \mathcal{F}\ |\uparrow|\ (N_G\ \mid\cup\mid\ U_G))\ (trail-false-cls\ (Ln\ \# \Gamma_0)))$  and
   $\Gamma_G$ -sorted: sorted-wrt  $(\lambda x\ y.$  atm-of  $(fst\ y) \prec_t$  atm-of  $(fst\ x))$   $\Gamma_G$ 
  using  $\langle ord-res-11-invars\ N_G\ S_G \rangle$  [unfolded  $S_G$ -def ord-res-11-invars.simps
    ord-res-10-invars.simps]
  by simp-all

moreover have  $fBex\ (iefac\ \mathcal{F}\ |\uparrow|\ (N_G\ \mid\cup\mid\ U_G))\ (trail-false-cls\ \Gamma_G)$ 
  using  $C_G$ -in  $C_G$ -false by metis

ultimately have  $snd\ Ln \neq None$  and is-pos  $(fst\ Ln)$  and  $\forall x \in set\ \Gamma_{G0}.$   $snd$ 
 $x = None$ 
  unfolding atomize-conj by metis

have  $\neg fBex\ (iefac\ \mathcal{F}\ |\uparrow|\ (N_G\ \mid\cup\mid\ U_G))\ (trail-false-cls\ \Gamma_{G0})$ 
proof (cases  $\Gamma_{G0}$ )
  case Nil
  then show ?thesis
    using  $\langle \{\#\} |\notin|$  iefac  $\mathcal{F}$   $|\uparrow|$   $(N_G \mid\cup\mid U_G) \rangle$ 
    unfolding trail-false-cls-def trail-false-lit-def
    by simp
  next
  case (Cons  $x$   $xs$ )
  then show ?thesis
    using  $\langle \Gamma_G = Ln \# \Gamma_{G0} \rangle$ 
    using  $\langle \forall x \in set\ \Gamma_{G0}.$   $snd\ x = None \rangle$ 

```

```

    using BBB[rule-format, of [Ln], unfolded append-Cons append-Nil]
    by simp
qed

hence  $\neg$  trail-false-cls  $\Gamma_{G0}$   $C_G$ 
  using  $C_G$ -in by metis

hence fst  $Ln = - L$ 
using  $C_G$ -false  $C_G$ -max-lit  $\Gamma_G$ -sorted[unfolded  $\langle \Gamma_G = Ln \# \Gamma_{G0} \rangle$  sorted-wrt.simps]
  by (smt (verit, ccfv-SIG) Neg-atm-of-iff  $\langle \Gamma_G = Ln \# \Gamma_{G0} \rangle$  atm-of-uminus
      ord-res.less-lit-simps(4) imageE image-insert insertE
      linorder-lit.dual-order.strict-trans linorder-lit.is-maximal-in-mset-iff
      linorder-lit.neq-iff linorder-trm.order.irrefl list.simps(15) literal.collapse(1)
      ord-res.ground-ordered-resolution-calculus-axioms trail-false-cls-def
      trail-false-lit-def)

hence  $\neg$  is-pos  $L$ 
  using  $\langle$ is-pos (fst  $Ln$ ) $\rangle$ 
  by (simp add: is-pos-neg-not-is-pos)

thus False
  using  $\langle$ is-pos  $L$  $\rangle$  by contradiction
qed

hence  $C_G \in | N_G \cup U_G$ 
proof -
  obtain  $C$  where  $C \in | N_G \cup U_G$  and  $C_G = \text{iefac } \mathcal{F} C$ 
  using  $C_G$ -in by blast

hence  $C_G = C$ 
  using  $\langle$  $\#$  $L$ . is-pos  $L \wedge$  ord-res.is-maximal-lit  $L C_G$  $\rangle$ 
  by (metis clauses-in- $\mathcal{F}$ -have-pos-max-lit ex1-efac-eq-factoring-chain iefac-def
      ord-res.ground-factorings-preserves-maximal-literal)

thus ?thesis
  using  $\langle C \in | N_G \cup U_G \rangle$  by simp
qed

have scl-fol.conflict  $N \beta S S'$ 
  unfolding  $S$ -def  $S'$ -def  $\langle \Gamma' = \Gamma \rangle$   $\langle U' = U \rangle$   $\mathcal{C}$ -def  $\mathcal{C}'$ -def  $\langle C_G = \text{None} \rangle$   $\langle C_G' = \text{Some } C_G \rangle$  option.map
= Some  $C_G$  option.map
proof (rule scl-fol.conflictI)
  show cls-of-gcls  $C_G \in | N \cup U$ 
  unfolding  $N$ -def  $U$ -def
  using  $\langle C_G \in | N_G \cup U_G \rangle$  by auto
next
  show is-ground-cls (cls-of-gcls  $C_G \cdot (\text{Var}::'v \Rightarrow (f, -) \text{Term.term})$ )
  by simp
next

```

```

have trail-false-cls  $\Gamma_G C_G$ 
  using  $\langle \text{trail-false-cls } \Gamma_G C_G \rangle$  .

hence trail-false-cls  $\Gamma$  (cls-of-gcls  $C_G \cdot \text{Var}$ )
  unfolding  $\Gamma$ -def subst-cls-id-subst
  using trail-false-cls-nonground-iff-trail-false-cls-ground by metis

thus SCL-FOL.trail-false-cls  $\Gamma$  (cls-of-gcls  $C_G \cdot \text{Var}$ )
  unfolding SCL-FOL.trail-false-cls-def trail-false-cls-def
  unfolding SCL-FOL.trail-false-lit-def trail-false-lit-def
  by argo
qed

thus ?thesis
  unfolding scl-fol.regular-scl-def by argo
next
case step-hyps: (skip  $L_G C_G n_G$ )

have  $\Gamma = \text{gtrailelem-of-trailelem } (L_G, n_G) \# \Gamma'$ 
  unfolding  $\Gamma$ -def  $\Gamma'$ -def  $\langle \Gamma_G = (L_G, n_G) \# \Gamma_G' \rangle$  by simp

have  $U' = U$ 
  unfolding  $U'$ -def  $\langle U_G' = U_G \rangle$   $U$ -def ..

have  $\neg \text{Ex } (\text{scl-fol.conflict } N \beta S)$ 
  using  $\langle C_G = \text{Some } C_G \rangle$  nex-conflict-if-alread-in-conflict by metis

moreover have scl-fol.reasonable-scl  $N \beta S S'$ 
  unfolding scl-fol.reasonable-scl-def
proof (intro conjI impI notI ; (elim exE) ?)
  have scl-fol.skip  $N \beta S S'$ 
    unfolding  $S$ -def  $S'$ -def  $\langle U' = U \rangle$   $C$ -def  $C'$ -def  $\langle C_G = \text{Some } C_G \rangle$   $\langle C_G' = \text{Some } C_G \rangle$  option.map
    unfolding  $\langle \Gamma = - \# \Gamma' \rangle$  gtrailelem-of-trailelem-def prod.case
proof (rule scl-fol.skipI)
  have  $- \text{lit-of-glit } L_G = \text{lit-of-glit } (- L_G)$ 
    by (cases  $L_G$ ) (simp-all add: lit-of-glit-def)
  show  $- \text{lit-of-glit } L_G \notin \# \text{cls-of-gcls } C_G \cdot \text{Var}$ 
    unfolding subst-cls-id-subst
    unfolding  $\langle - \text{lit-of-glit } L_G = \text{lit-of-glit } (- L_G) \rangle$ 
    unfolding cls-of-gcls-def
    using  $\langle - L_G \notin \# C_G \rangle$  inj-image-mset-mem-iff[OF inj-lit-of-glit]
    by metis
qed

thus scl-fol.scl  $N \beta S S'$ 
  unfolding scl-fol.scl-def by argo
next
fix  $S'' :: ('f, 'v) \text{SCL-FOL.state}$ 

```

```

assume scl-fol.conflict  $N \beta S' S''$ 

moreover have  $\nexists S''$ . scl-fol.conflict  $N \beta S' S''$ 
unfolding  $S'$ -def  $C'$ -def  $\langle C_G' = \text{Some } C_G \rangle$  by (simp add: scl-fol.conflict.simps)

ultimately show False
  by metis
qed

ultimately show ?thesis
  unfolding scl-fol.regular-scl-def by argo
next
case step-hyps: (resolution  $L_G C_G \Gamma_G'' D_G$ )

have  $\Gamma' = \Gamma$ 
  unfolding  $\Gamma'$ -def  $\langle \Gamma_G' = \Gamma_G \rangle$   $\Gamma$ -def ..

have  $U' = U$ 
  unfolding  $U'$ -def  $\langle U_G' = U_G \rangle$   $U$ -def ..

have  $C = \text{Some } (\text{cls-of-gcls } D_G, \text{Var})$ 
  unfolding  $C$ -def  $\langle C_G = \text{Some } D_G \rangle$  option.map ..

hence  $C$ -eq:  $C = \text{Some}$ 
  (add-mset ( $-\text{ lit-of-glit } L_G$ ) (remove1-mset ( $-\text{ lit-of-glit } L_G$ ) (cls-of-gcls  $D_G$ )),
  Var)
by (smt (verit, best) add-mset-remove-trivial atm-of-eq-atm-of atm-of-lit-of-glit-conv
  cls-of-gcls-def glit-of-lit-lit-of-glit image-mset-add-mset insert-DiffM step-hyps ( $\gamma$ )
  uminus-not-id')

have  $C' = \text{Some}$  (
  remove1-mset ( $-\text{ (lit-of-glit } L_G)$ ) (cls-of-gcls  $D_G$ ) +
  remove1-mset (lit-of-glit  $L_G$ ) (cls-of-gcls  $C_G$ ), Var)
  unfolding  $C'$ -def  $\langle C_G' = \text{Some } \rightarrow \text{option.map}$ 
  apply (simp add: cls-of-gcls-def)
by (smt (verit, ccfv-threshold) add-diff-cancel-right' add-mset-add-single atm-of-eq-atm-of
  atm-of-lit-of-glit-conv diff-single-trivial glit-of-lit-lit-of-glit
  image-mset-remove1-mset-if insert-DiffM is-pos-neg-not-is-pos msed-map-invR)
hence  $C'$ -eq:  $C' = \text{Some}$  (
  (remove1-mset ( $-\text{ (lit-of-glit } L_G)$ ) (cls-of-gcls  $D_G$ )  $\cdot$  Var +
  remove1-mset (lit-of-glit  $L_G$ ) (cls-of-gcls  $C_G$ )  $\cdot$  Var)  $\cdot$  Var, Var)
  by simp

have linorder-lit.is-greatest-in-mset  $C_G L_G$ 
using  $\langle \text{ord-res-11-invars } N_G S_G \rangle$  [unfolded  $S_G$ -def  $\langle \Gamma_G = (L_G, \text{Some } C_G) \#$ 
 $\Gamma_G'' \rangle$ ]
unfolding ord-res-11-invars.simps ord-res-10-invars.simps
by simp

```



**hence**  $\text{add-mset } L_G \{ \#y \in \# C_G. y \neq L_G \# \} = C_G$   
**using**  $\text{linorder-lit.explode-greatest-in-mset}$  **by**  $\text{metis}$

**hence**  $C_G - \{ \#L_G \# \} = \{ \#K \in \# C_G. K \neq L_G \# \}$   
**by**  $(\text{metis add-mset-remove-trivial})$

**hence**  $\text{cls-of-gcls } (C_G - \{ \#L_G \# \}) = \text{cls-of-gcls } \{ \#K \in \# C_G. K \neq L_G \# \}$   
**by**  $\text{argo}$

**moreover have**  $\text{cls-of-gcls } (C_G - \{ \#L_G \# \}) = \text{cls-of-gcls } C_G - \text{cls-of-gcls } \{ \#L_G \# \}$   
**unfolding**  $\text{cls-of-gcls-def}$   
**proof**  $(\text{rule image-mset-Diff})$   
**show**  $\{ \#L_G \# \} \subseteq \# C_G$   
**by**  $(\text{metis } \langle \text{ord-res.is-strictly-maximal-lit } L_G \ C_G \rangle \text{linorder-lit.is-greatest-in-mset-iff single-subset-iff})$

**qed**

**ultimately have**  $\text{cls-of-gcls } C_G - \{ \# \text{lit-of-glit } L_G \# \} = \text{cls-of-gcls } \{ \#K \in \# C_G. K \neq L_G \# \}$   
**by**  $(\text{metis } \langle \text{add-mset } L_G \{ \#y \in \# C_G. y \neq L_G \# \} = C_G \rangle \text{cls-of-gcls-def image-mset-remove1-mset-if union-single-eq-member})$

**have**  $\neg \text{Ex } (\text{scl-fol.conflict } N \ \beta \ S)$   
**using**  $\langle C_G = \text{Some } \rightarrow \text{nex-conflict-if-alread-in-conflict} \rangle$  **by**  $\text{metis}$

**moreover have**  $\text{scl-fol.reasonable-scl } N \ \beta \ S \ S'$   
**unfolding**  $\text{scl-fol.reasonable-scl-def}$   
**proof**  $(\text{intro conjI impI notI ; } (\text{elim exE}) \ ?)$   
**have**  $\text{scl-fol.resolve } N \ \beta \ S \ S'$   
**unfolding**  $S\text{-def } S'\text{-def } \langle \Gamma' = \Gamma \rangle \langle U' = U \rangle$   
**unfolding**  $C\text{-eq } C'\text{-eq}$   
**proof**  $(\text{rule scl-fol.resolveI})$   
**show**  $\Gamma = \text{trail-propagate } (\text{map } \text{gtrailelem-of-trailelem } \Gamma_G'')$   
 $(\text{lit-of-glit } L_G) (\text{remove1-mset } (\text{lit-of-glit } L_G) (\text{cls-of-gcls } C_G)) \text{Var}$   
**unfolding**  $\Gamma\text{-def } \langle \Gamma_G = (L_G, \text{Some } C_G) \# \Gamma_G'' \rangle \text{gtrailelem-of-trailelem-def list.map prod.case}$   
**unfolding**  $\text{propagate-lit-def subst-lit-id-subst option.map}$   
**unfolding**  $\langle \text{remove1-mset } (\text{lit-of-glit } L_G) (\text{cls-of-gcls } C_G) = \text{cls-of-gcls } \{ \#K \in \# C_G. K \neq L_G \# \} \rangle$   
**by**  $\text{argo}$

**next**  
**show**  $\text{lit-of-glit } L_G \cdot l \ \text{Var} = - (\text{lit-of-glit } L_G \cdot l \ \text{Var})$   
**by**  $\text{simp}$

**next**  
**show**  $\text{SCL-FOL.is-renaming } \text{Var}$   
**by**  $\text{simp}$

**next**

```

show SCL-FOL.is-renaming Var
  by simp
next
show
  vars-cls (add-mset ( $-$  lit-of-glit  $L_G$ )
    (remove1-mset ( $-$  lit-of-glit  $L_G$ ) (cls-of-gcls  $D_G$ ))  $\cdot$  Var)  $\cap$ 
  vars-cls (add-mset (lit-of-glit  $L_G$ )
    (remove1-mset (lit-of-glit  $L_G$ ) (cls-of-gcls  $C_G$ ))  $\cdot$  Var) =
  {}
  by (metis (no-types, lifting) boolean-algebra.conj-zero-right cls-of-gcls-def
    diff-single-trivial image-mset-add-mset insert-DiffM subst-cls-id-subst
    vars-cls-cls-of-gcls)
next
show SCL-FOL.is-ingu Var
  {{atm-of ( $-$  lit-of-glit  $L_G$ )  $\cdot$  a Var, atm-of (lit-of-glit  $L_G$ )  $\cdot$  a Var}}
by (simp add: SCL-FOL.is-ingu-def SCL-FOL.is-unifiers-def SCL-FOL.is-unifier-def)
next
show is-grounding-merge Var
  (vars-cls
    (add-mset ( $-$  lit-of-glit  $L_G$ ) (remove1-mset ( $-$  lit-of-glit  $L_G$ ) (cls-of-gcls
D_G))  $\cdot$  Var))
  (rename-subst-domain Var Var)
  (vars-cls
    (add-mset (lit-of-glit  $L_G$ ) (remove1-mset (lit-of-glit  $L_G$ ) (cls-of-gcls  $C_G$ ))
 $\cdot$  Var))
  (rename-subst-domain Var Var)
  by (simp add: is-grounding-merge-def)
qed

thus scl-fol.scl  $N$   $\beta$   $S$   $S'$ 
  unfolding scl-fol.scl-def by argo
next
fix  $S''$  :: ( $'f$ ,  $'v$ ) SCL-FOL.state
assume scl-fol.conflict  $N$   $\beta$   $S'$   $S''$ 

moreover have  $\nexists S''$ . scl-fol.conflict  $N$   $\beta$   $S'$   $S''$ 
unfolding  $S'$ -def  $C'$ -def  $\langle C_G' = \text{Some } \rightarrow \rangle$  by (simp add: scl-fol.conflict.simps)

ultimately show False
  by metis
qed

ultimately show ?thesis
  unfolding scl-fol.regular-scl-def by argo
next
case step-hyps: (backtrack  $L_G$   $C_G$ )

define  $K$  :: ( $'f$ ,  $'v$ ) term literal where
   $K = -$  lit-of-glit  $L_G$ 

```

```

define  $D :: ('f, 'v)$  term clause where
   $D = \text{cls-of-gcls } C_G - \{\#K\# \}$ 

have  $\text{add-mset } K \ D = \text{cls-of-gcls } C_G$ 
  by (smt (verit, best)  $D\text{-def } K\text{-def } \text{add-mset-remove-trivial } \text{atm-of-eq-atm-of}$ 
     $\text{atm-of-lit-of-glit-conv } \text{cls-of-gcls-def } \text{glit-of-lit-lit-of-glit } \text{image-mset-add-mset}$ 
     $\text{insert-DiffM } \text{step-hyps}(6) \ \text{uminus-not-id}'$ )

have  $U' = \text{finsert } (\text{add-mset } K \ D) \ U$ 
  unfolding  $U\text{-def } U'\text{-def } \langle U_G' = \text{finsert } C_G \ U_G \rangle$ 
  by (smt (verit, ccfv-SIG)  $D\text{-def } K\text{-def } \text{add-mset-remove-trivial } \text{atm-of-eq-atm-of}$ 
     $\text{atm-of-lit-of-glit-conv } \text{cls-of-gcls-def } \text{fimage-finsert } \text{glit-of-lit-lit-of-glit}$ 
     $\text{image-mset-add-mset } \text{insert-DiffM } \text{step-hyps}(6) \ \text{uminus-not-id}'$ )

have  $C = \text{Some } (\text{add-mset } K \ D, \ \text{Var})$ 
  by (smt (verit)  $D\text{-def } K\text{-def } C\text{-def } \text{add-mset-remove-trivial } \text{atm-of-eq-atm-of}$ 
     $\text{atm-of-lit-of-glit-conv } \text{cls-of-gcls-def } \text{glit-of-lit-lit-of-glit } \text{image-mset-add-mset}$ 
     $\text{insert-DiffM } \text{option.map}(2) \ \text{step-hyps}(1,6) \ \text{uminus-not-id}'$ )

have  $C' = \text{None}$ 
  unfolding  $C'\text{-def } \langle C_G' = \text{None} \rangle \ \text{option.map} \ ..$ 

have  $\neg \text{Ex } (\text{scl-fol.conflict } N \ \beta \ S)$ 
  using  $\langle C_G = \text{Some } \rightarrow \text{nex-conflict-if-alread-in-conflict} \ \text{by } \text{metis} \rangle$ 

moreover have  $\text{scl-fol.reasonable-scl } N \ \beta \ S \ S'$ 
  unfolding  $\text{scl-fol.reasonable-scl-def}$ 
proof (intro conjI impI notI ; (elim exE) ?)
  have  $\text{scl-fol.backtrack } N \ \beta \ S \ S'$ 
  unfolding  $S\text{-def } S'\text{-def}$ 
  unfolding  $\langle U' = \text{finsert } (\text{add-mset } K \ D) \ U \rangle \ \langle C = \text{Some } (\text{add-mset } K \ D,$ 
 $\text{Var}) \rangle \ \langle C' = \text{None} \rangle$ 
proof (rule  $\text{scl-fol.backtrackI}$ )
  show  $\Gamma = \text{trail-decide } ([] \ @ \ \Gamma') \ (\text{lit-of-glit } L_G)$ 
  unfolding append-Nil
  unfolding decide-lit-def
  unfolding  $\Gamma\text{-def } \langle \Gamma_G = - \ # \ \rightarrow \ \text{list.map } \Gamma'\text{-def}[\text{symmetric}] \rangle$ 
  unfolding  $\text{gtrailelem-of-trailelem-def } \text{prod.case } \text{option.map}$ 
  ..
next
  show  $\text{lit-of-glit } L_G = - \ (K \ \cdot l \ \text{Var})$ 
  unfolding  $K\text{-def}$  by simp
next
  have  $\text{sorted-wrt } (\lambda x \ y. \ \text{atm-of } (\text{fst } y) \ \prec_t \ \text{atm-of } (\text{fst } x)) \ \Gamma_G$ 
  using  $\langle \text{ord-res-11-invars } N_G \ S_G \rangle [\text{unfolded } S_G\text{-def}]$ 
  unfolding  $\text{ord-res-11-invars.simps } \text{ord-res-10-invars.simps}$ 
by fast

```

```

hence trail-consistent  $\Gamma_G$ 
  using trail-consistent-if-sorted-wrt-atoms by metis

hence  $\neg$  trail-defined-lit  $\Gamma_{G'} (- L_G)$ 
  by (metis trail-consistent.cases atm-of-eq-atm-of list.distinct(1) list.inject
    prod.sel(1) step-hyps(5) trail-defined-lit-iff-trail-defined-atm)

hence  $\neg$  trail-false-lit  $\Gamma_{G'} (- L_G)$ 
  using trail-defined-lit-iff-true-or-false by metis

hence  $\neg$  trail-false-cls  $\Gamma_{G'} C_G$ 
  using  $\langle - L_G \in\# C_G \rangle$ 
  unfolding trail-false-cls-def by metis

hence  $\neg$  trail-false-cls  $\Gamma' (add-mset K D)$ 
  unfolding  $\Gamma'$ -def  $\langle add-mset K D = cls-of-gcls C_G \rangle$ 
  unfolding trail-false-cls-nonground-iff-trail-false-cls-ground .

moreover have is-ground-cls  $(add-mset K D)$ 
  using  $\mathcal{C}$ -def  $\langle \mathcal{C} = Some (add-mset K D, Var) \rangle$  step-hyps(1) by auto

  ultimately have  $\nexists \gamma. is-ground-cls (add-mset K D \cdot \gamma) \wedge trail-false-cls \Gamma'$ 
  (add-mset K D \cdot \gamma)
  by simp

  thus  $\nexists \gamma. is-ground-cls (add-mset K D \cdot \gamma) \wedge SCL-FOL.trail-false-cls \Gamma'$ 
  (add-mset K D \cdot \gamma)
  unfolding SCL-FOL.trail-false-cls-def trail-false-cls-def
  unfolding SCL-FOL.trail-false-lit-def trail-false-lit-def
  by argo
qed

thus scl-fol.scl  $N \beta S S'$ 
  unfolding scl-fol.scl-def by argo
next
fix  $S'' :: ('f, 'v) SCL-FOL.state$ 
assume scl-fol.decide  $N \beta S S'$ 
thus False
  unfolding  $S$ -def  $\langle \mathcal{C} = Some - \rangle$ 
  by (auto elim!: scl-fol.decide.cases)
qed

ultimately show ?thesis
unfolding scl-fol.regular-scl-def by argo
qed
qed
end

```

**lemma** *wfp-on-antimono-stronger*:  
**fixes**  
 $A :: 'a \text{ set}$  **and**  $B :: 'b \text{ set}$  **and**  
 $f :: 'a \Rightarrow 'b$  **and**  
 $R :: 'b \Rightarrow 'b \Rightarrow \text{bool}$  **and**  $Q :: 'a \Rightarrow 'a \Rightarrow \text{bool}$   
**assumes**  
 $wf: wfp\text{-on } B \ R$  **and**  
 $sub: f \text{ ' } A \subseteq B$  **and**  
 $mono: \bigwedge x \ y. x \in A \Longrightarrow y \in A \Longrightarrow Q \ x \ y \Longrightarrow R \ (f \ x) \ (f \ y)$   
**shows**  $wfp\text{-on } A \ Q$   
**unfolding** *wfp-on-iff-ex-minimal*  
**proof** (*intro allI impI*)  
**fix**  $A' :: 'a \text{ set}$   
**assume**  $A' \subseteq A$  **and**  $A' \neq \{\}$   
**have**  $f \text{ ' } A' \subseteq B$   
**using**  $\langle A' \subseteq A \rangle \ sub$  **by** *blast*  
**moreover** **have**  $f \text{ ' } A' \neq \{\}$   
**using**  $\langle A' \neq \{\} \rangle$  **by** *blast*  
**ultimately** **have**  $\exists z \in f \text{ ' } A'. \forall y. R \ y \ z \longrightarrow y \notin f \text{ ' } A'$   
**using**  $wf \ wfp\text{-on-iff-ex-minimal}$  **by** *blast*  
**hence**  $\exists z \in A'. \forall y. R \ (f \ y) \ (f \ z) \longrightarrow y \notin A'$   
**by** *blast*  
**thus**  $\exists z \in A'. \forall y. Q \ y \ z \longrightarrow y \notin A'$   
**using**  $\langle A' \subseteq A \rangle \ mono$  **by** *blast*  
**qed**

For AFP-devel, delete  $\llbracket wfp\text{-on } ?B \ ?R; ?f \text{ ' } ?A \subseteq ?B; \bigwedge x \ y. \llbracket x \in ?A; y \in ?A; ?Q \ x \ y \rrbracket \Longrightarrow ?R \ (?f \ x) \ (?f \ y) \rrbracket \Longrightarrow wfp\text{-on } ?A \ ?Q$  as it is available in *HOL.Wellfounded*.

**corollary** (**in** *scl-fol-calculus*) *termination-projectable-strategy*:

**fixes**  
 $N :: ('f, 'v) \text{ Term.term clause fset}$  **and**  
 $\beta :: ('f, 'v) \text{ Term.term}$  **and**  
 $strategy$  **and**  $strategy\text{-init}$  **and**  $proj$   
**assumes** *strategy-restricts-regular-scl*:  
 $\bigwedge S \ S'. \text{strategy}^{**} \ strategy\text{-init } S \Longrightarrow \text{strategy } S \ S' \Longrightarrow \text{regular-scl } N \ \beta \ (proj \ S)$   
 $(proj \ S')$  **and**  
 $initial\text{-state}: proj \ strategy\text{-init} = initial\text{-state}$   
**shows**  $wfp\text{-on } \{S. \text{strategy}^{**} \ strategy\text{-init } S\} \ \text{strategy}^{-1-1}$   
**proof** (*rule wfp-on-antimono-stronger*)  
**show**  $wfp\text{-on } \{proj \ S \mid S. \text{strategy}^{**} \ strategy\text{-init } S\} \ (\text{regular-scl } N \ \beta)^{-1-1}$   
**proof** (*rule wfp-on-subset*)  
**show**  $wfp\text{-on } \{S. (\text{regular-scl } N \ \beta)^{**} \ initial\text{-state } S\} \ (\text{regular-scl } N \ \beta)^{-1-1}$   
**using** *termination-regular-scl* **by** *metis*  
**next**  
**show**  $\{proj \ S \mid S. \text{strategy}^{**} \ strategy\text{-init } S\} \subseteq \{S. (\text{regular-scl } N \ \beta)^{**} \ initial\text{-state } S\}$   
**proof** (*intro Collect-mono impI, elim exE conjE*)  
**fix**  $s \ S$  **assume**  $s = proj \ S$  **and**  $\text{strategy}^{**} \ strategy\text{-init } S$

```

show (regular-scl N β)** initial-state s
  unfolding ⟨s = proj S⟩
  using ⟨strategy** strategy-init S⟩
proof (induction S rule: rtranclp-induct)
  case base
  thus ?case
    unfolding initial-state by simp
next
  case (step y z)
  thus ?case
    using strategy-restricts-regular-scl
    by (meson rtranclp.simps)
qed
qed
qed
next
show proj ‘ {S. strategy** strategy-init S} ⊆ {proj S | S. strategy** strategy-init S}
  by blast
next
show  $\bigwedge S' S. S \in \{S. \text{strategy}^{**} \text{strategy-init } S\} \implies \text{strategy}^{-1-1} S' S \implies$ 
  (regular-scl N β)-1-1 (proj S') (proj S)
  using strategy-restricts-regular-scl by simp
qed

```

For AFP-devel, delete  $\llbracket \text{scl-fol-calculus } ?\text{renaming-vars } ?\text{less-B}; \bigwedge S S'. \llbracket ?\text{strategy}^{**} ?\text{strategy-init } S; ?\text{strategy } S S' \rrbracket \implies \text{scl-fol-calculus.regular-scl } ?\text{less-B } ?N ?\beta (?proj S) (?proj S'); ?proj ?\text{strategy-init} = \text{initial-state} \rrbracket \implies \text{wfp-on } \{S. ?\text{strategy}^{**} ?\text{strategy-init } S\} ?\text{strategy}^{-1-1}$  as it is available in *Simple-Clause-Learning.Termination*.

**corollary** (in simulation-SCLFOL-ground-ordered-resolution) ord-res-11-termination:

```

fixes N :: 'f gclause fset
shows wfp-on {S. (ord-res-11 N)** ({||}, {||}, [], None) S} (ord-res-11 N)-1-1
proof (rule scl-fol.termination-projectable-strategy)
fix S S'
assume run: (ord-res-11 N)** ({||}, {||}, [], None) S and step: ord-res-11 N S S'

```

**define** β :: 'f gterm **where**

β = (THE A. linorder-trm.is-greatest-in-fset (atms-of-cls N) A)

```

show scl-fol.regular-scl (cls-of-gcls |4 N) (term-of-gterm β) (state-of-gstate S)
(state-of-gstate S')

```

**proof** (rule ord-res-11-is-strategy-for-regular-scl)

**show**  $\forall A_G |\in| \text{atms-of-cls } N. A_G \preceq_t \beta$

**proof** (cases atms-of-cls N = {||})

**case** True

**thus** ?thesis

**by** simp

```

next
  case False
  then show ?thesis
    unfolding β-def
    by (metis (full-types) linorder-trm.Uniq-is-greatest-in-fset
      linorder-trm.ex-greatest-in-fset linorder-trm.is-greatest-in-fset-iff sup2CI
      the1-equality^)
qed
next
show (ord-res-11 N)** ({||}, {||}, [], None) S
  using run .
next
show ord-res-11 N S S'
  using step .
qed
next
show state-of-gstate ({||}, {||}, [], None) = SCL-FOL.initial-state
  by simp
qed

corollary (in scl-fol-calculus) static-non-subsumption-projectable-strategy:
  fixes strategy and strategy-init and proj
  assumes
    run: strategy** strategy-init S and
    step: backtrack N β (proj S) S' and
    strategy-restricts-regular-scl:
       $\bigwedge S S'. \text{strategy}^{**} \text{strategy-init } S \implies \text{strategy } S S' \implies \text{regular-scl } N \beta (\text{proj } S) (\text{proj } S')$ 
  and
    initial-state: proj strategy-init = initial-state
  defines
    U  $\equiv$  state-learned (proj S)
  shows  $\exists C \gamma. \text{state-conflict } (\text{proj } S) = \text{Some } (C, \gamma) \wedge \neg (\exists D |\in| N |\cup| U. \text{subsumes } D C)$ 
  proof –
  have (regular-scl N β)** initial-state (proj S)
    using run
  proof (induction S rule: rtranclp-induct)
  case base
  thus ?case
    unfolding initial-state by simp
  next
  case (step y z)
  thus ?case
    using strategy-restricts-regular-scl
    by (meson rtranclp.simps)
  qed

moreover have backtrack N β (proj S) S'
  using step by simp

```

**ultimately show** *?thesis*  
**unfolding** *U-def*  
**using** *static-non-subsumption-regular-scl*  
**by** *simp*  
**qed**

For AFP-devel, delete  $\llbracket scl\text{-fol}\text{-calculus } ?renaming\text{-vars } ?less\text{-B}; ?strategy^{**} ?strategy\text{-init } ?S; scl\text{-fol}\text{-calculus.backtrack } ?N ?\beta ( ?proj ?S) ?S'; \bigwedge S S'. \llbracket ?strategy^{**} ?strategy\text{-init } S; ?strategy S S' \rrbracket \implies scl\text{-fol}\text{-calculus.regular}\text{-scl } ?less\text{-B } ?N ?\beta ( ?proj S) ( ?proj S'); ?proj ?strategy\text{-init} = initial\text{-state} \rrbracket \implies \exists C \gamma. state\text{-conflict } ( ?proj ?S) = Some (C, \gamma) \wedge \neg (\exists D | \in | ?N | \cup | state\text{-learned } ( ?proj ?S). subsumes D C)$  as it is available in *Simple-Clause-Learning.Non-Redundancy*.

**corollary** (in *simulation-SCLFOL-ground-ordered-resolution*) *ord-res-11-non-subsumption*:

**fixes**  $N_G :: 'f gclause fset$  **and**  $s :: - \times - \times - \times -$   
**defines**  
 $\beta \equiv (THE A. linorder\text{-trm.is-greatest-in-fset } (atms\text{-of-cls } N_G) A)$   
**assumes**  
 $run: (ord\text{-res-11 } N_G)^{**} (\{\|\}, \{\|\}, [], None) s$  **and**  
 $step: scl\text{-fol.backtrack } (cls\text{-of-gcls } |' N_G) (term\text{-of-gterm } \beta) (state\text{-of-gstate } s)$   
 $s'$   
**shows**  $\exists U_{er} \mathcal{F} \Gamma D. s = (U_{er}, \mathcal{F}, \Gamma, Some D) \wedge \neg (\exists C | \in | N_G | \cup | U_{er}. C \subseteq \# D)$   
**proof** –  
**have**  $\exists C \gamma. state\text{-conflict } (state\text{-of-gstate } s) = Some (C, \gamma) \wedge \neg (\exists D | \in | cls\text{-of-gcls } |' N_G | \cup | state\text{-learned } (state\text{-of-gstate } s). subsumes D C)$   
**proof** (*rule scl-fol.static-non-subsumption-projectable-strategy[ of ord-res-11 N\_G - - - state-of-gstate , OF run step]*)  
**fix**  $S S'$   
**assume**  $run: (ord\text{-res-11 } N_G)^{**} (\{\|\}, \{\|\}, [], None) S$  **and**  $step: ord\text{-res-11 } N_G S S'$   
**show**  $scl\text{-fol.regular}\text{-scl } (cls\text{-of-gcls } |' N_G) (term\text{-of-gterm } \beta) (state\text{-of-gstate } S) (state\text{-of-gstate } S')$   
**proof** (*intro ord-res-11-is-strategy-for-regular-scl ballI*)  
**fix**  $A_G :: 'f gterm$   
**assume**  $A_G | \in | atms\text{-of-cls } N_G$   
**show**  $A_G \preceq_t \beta$   
**proof** (*cases atms-of-cls N\_G = {\|\}*)  
**case** *True*  
**thus** *?thesis*  
**using**  $\langle A_G | \in | atms\text{-of-cls } N_G \rangle$   
**by** *simp*  
**next**  
**case** *False*  
**then show** *?thesis*  
**using**  $\langle A_G | \in | atms\text{-of-cls } N_G \rangle$   
**unfolding**  $\beta\text{-def}$   
**by** (*metis (full-types) linorder-trm.Uniq-is-greatest-in-fset*)



```

      linorder-trm.ex-greatest-in-fset linorder-trm.is-greatest-in-fset-iff sup2CI
      the1-equality')
    qed
  next
  show (ord-res-11  $N_G$ )** ( $\{\|\}, \{\|\}, [], None$ )  $S$ 
    using run .
  next
  show ord-res-11  $N_G$   $S$   $S'$ 
    using step .
  qed
next
show state-of-gstate ( $\{\|\}, \{\|\}, [], None$ ) = initial-state
  by simp
qed

moreover obtain  $U_G \mathcal{F} \Gamma D$  where  $s = (U_G, \mathcal{F}, \Gamma, Some D)$ 
proof atomize-elim
  obtain  $U_G \mathcal{F} \Gamma \mathcal{C}$  where  $s = (U_G, \mathcal{F}, \Gamma, \mathcal{C})$ 
    by (metis prod.exhaust)

  moreover obtain  $D$  where  $\mathcal{C} = Some D$ 
    using step
  by (auto simp:  $\langle s = (U_G, \mathcal{F}, \Gamma, \mathcal{C}) \rangle$  elim: scl-fol.backtrack.cases)

  ultimately show  $\exists U_{er} \mathcal{F} \Gamma D. s = (U_{er}, \mathcal{F}, \Gamma, Some D)$ 
    by metis
qed

ultimately have  $\neg (\exists C | \in | N_G | \cup | U_G.
  subsumes (cls-of-gcls C :: ('f, 'v) term clause) (cls-of-gcls D))$ 
  by auto

hence  $\neg (\exists C | \in | N_G | \cup | U_G. (cls-of-gcls C :: ('f, 'v) term clause) \subseteq\# (cls-of-gcls D))$ 
  by (simp add: subsumes-def)

hence  $\neg (\exists C | \in | N_G | \cup | U_G. C \subseteq\# D)$ 
  by (metis cls-of-gcls-def image-mset-subseteq-mono)

thus ?thesis
  unfolding  $\langle s = (U_G, \mathcal{F}, \Gamma, Some D) \rangle$  by metis
qed
end

```

## References

- [1] M. Bromberger, C. Jain, and C. Weidenbach. SCL(FOL) can simulate non-redundant superposition clause learning. In B. Pientka and C. Tinelli, editors, *Automated Deduction – CADE 29*, pages 134–152, Cham, 2023. Springer Nature Switzerland.