

Decreasing-Diagrams

Harald Zankl

August 7, 2022

Abstract

This theory contains a formalization of decreasing diagrams showing that any locally decreasing abstract rewrite system is confluent. We consider the valley (van Oostrom, TCS 1994) and the conversion version (van Oostrom, RTA 2008) and closely follow the original proofs. As an application we prove Newman’s lemma.

A description of this formalization is available in [3].

Contents

1	Decreasing Diagrams	1
1.1	Valley Version	1
1.1.1	Appendix	1
1.1.2	Multisets	3
1.1.3	Lexicographic maximum measure	12
1.1.4	Labeled Rewriting	20
1.1.5	Application: Newman’s Lemma	27
1.2	Conversion Version	29

1 Decreasing Diagrams

```
theory Decreasing-Diagrams imports HOL-Library.Multiset Abstract-Rewriting.Abstract-Rewriting  
begin
```

1.1 Valley Version

This section follows [1].

1.1.1 Appendix

interaction of multisets with sets

```
definition diff :: 'a multiset  $\Rightarrow$  'a set  $\Rightarrow$  'a multiset  
where diff M S = filter-mset ( $\lambda x. x \notin S$ ) M
```

definition *intersect* :: 'a multiset \Rightarrow 'a set \Rightarrow 'a multiset
where *intersect* $M\ S = \text{filter-mset } (\lambda x. x \in S)\ M$

notation

diff (infixl -s 800) **and**
intersect (infixl \cap s 800)

lemma *count-diff* [simp]:

count ($M -s\ A$) $a = \text{count } M\ a * \text{of-bool } (a \notin A)$
by (*simp* add: *diff-def*)

lemma *set-mset-diff* [simp]:

set-mset ($M -s\ A$) = *set-mset* $M - A$
by (*auto simp* add: *diff-def*)

lemma *diff-eq-singleton-imp*:

$M -s\ A = \{\#a\#\} \implies a \in (\text{set-mset } M - A)$
unfolding *diff-def filter-mset-eq-conv* **by** *auto*

lemma *count-intersect* [simp]:

count ($M \cap s\ A$) $a = \text{count } M\ a * \text{of-bool } (a \in A)$
by (*simp* add: *intersect-def*)

lemma *set-mset-intersect* [simp]:

set-mset ($M \cap s\ A$) = *set-mset* $M \cap A$
by (*auto simp* add: *intersect-def*)

lemma *diff-from-empty*: $\{\#\} -s\ S = \{\#\}$ **unfolding** *diff-def* **by** *auto*

lemma *diff-empty*: $M -s\ \{\} = M$ **unfolding** *diff-def* **by** (*rule multiset-eqI*) *simp*

lemma *submultiset-implies-subset*: **assumes** $M \subseteq\# N$ **shows** *set-mset* $M \subseteq \text{set-mset } N$

using *assms mset-subset-eqD* **by** *auto*

lemma *subset-implies-remove-empty*: **assumes** *set-mset* $M \subseteq S$ **shows** $M -s\ S = \{\#\}$

unfolding *diff-def* **using** *assms* **by** (*induct* M) *auto*

lemma *remove-empty-implies-subset*: **assumes** $M -s\ S = \{\#\}$ **shows** *set-mset* $M \subseteq S$ **proof**

fix x **assume** $A: x \in \text{set-mset } M$

have $x \notin \text{set-mset } (M -s\ S)$ **using** *assms* **by** *auto*

thus $x \in S$ **using** A **unfolding** *diff-def* **by** *auto*

qed

lemma *lemmaA-3-8*: $(M + N) -s\ S = (M -s\ S) + (N -s\ S)$ **unfolding** *diff-def*
by (*rule multiset-eqI*) *simp*

lemma lemmaA-3-9: $(M -s S) -s T = M -s (S \cup T)$ **unfolding** *diff-def* **by** *(rule multiset-eqI) simp*

lemma lemmaA-3-10: $M = (M \cap s S) + (M -s S)$ **unfolding** *diff-def intersect-def* **by** *auto*

lemma lemmaA-3-11: $(M -s T) \cap s S = (M \cap s S) -s T$ **unfolding** *diff-def intersect-def* **by** *(rule multiset-eqI) simp*

1.1.2 Multisets

Definition 2.5(1)

definition $ds :: 'a\ rel \Rightarrow 'a\ set \Rightarrow 'a\ set$
where $ds\ r\ S = \{y . \exists x \in S. (y,x) \in r\}$

definition $dm :: 'a\ rel \Rightarrow 'a\ multiset \Rightarrow 'a\ set$
where $dm\ r\ M = ds\ r\ (set-mset\ M)$

definition $dl :: 'a\ rel \Rightarrow 'a\ list \Rightarrow 'a\ set$
where $dl\ r\ \sigma = ds\ r\ (set\ \sigma)$

notation

ds (**infixl** $\downarrow s$ 900) **and**
 dm (**infixl** $\downarrow m$ 900) **and**
 dl (**infixl** $\downarrow l$ 900)

missing but useful

lemma ds-ds-subseteq-ds: **assumes** $t: trans\ r$ **shows** $ds\ r\ (ds\ r\ S) \subseteq ds\ r\ S$ **proof**
fix x **assume** $A: x \in ds\ r\ (ds\ r\ S)$ **show** $x \in ds\ r\ S$ **proof** –
from A **obtain** $y\ z$ **where** $(x,y) \in r$ **and** $(y,z) \in r$ **and** $mem: z \in S$ **unfolding**
 $ds-def$ **by** *auto*
thus *?thesis* **using** $mem\ t\ trans-def$ **unfolding** $ds-def$ **by** *fast*
qed
qed

from PhD thesis of van Oostrom

lemma ds-monotone: **assumes** $S \subseteq T$ **shows** $ds\ r\ S \subseteq ds\ r\ T$ **using** *assms*
unfolding $ds-def$ **by** *auto*

lemma subset-imp-ds-subset: **assumes** $trans\ r$ **and** $S \subseteq ds\ r\ T$ **shows** $ds\ r\ S \subseteq ds\ r\ T$
using *assms ds-monotone ds-ds-subseteq-ds* **by** *blast*

Definition 2.5(2)

strict order (mult) is used from Multiset.thy

definition $mult-eq :: 'a\ rel \Rightarrow 'a\ multiset\ rel$ **where**
 $mult-eq\ r = (mult1\ r)^*$

definition $mul :: 'a\ rel \Rightarrow 'a\ multiset\ rel$ **where**

$mul\ r = \{(M,N).\exists I\ J\ K. M = I + K \wedge N = I + J \wedge set\text{-mset}\ K \subseteq dm\ r\ J \wedge J \neq \{\#\}\}$

definition $mul\text{-}eq :: 'a\ rel \Rightarrow 'a\ multiset\ rel$ **where**

$mul\text{-}eq\ r = \{(M,N).\exists I\ J\ K. M = I + K \wedge N = I + J \wedge set\text{-mset}\ K \subseteq dm\ r\ J\}$

lemma $in\text{-}mul\text{-}eqI$:

assumes $M = I + K\ N = I + J\ set\text{-mset}\ K \subseteq r\ \downarrow m\ J$

shows $(M, N) \in mul\text{-}eq\ r$

using $assms$ **by** $(auto\ simp\ add: mul\text{-}eq\text{-}def)$

lemma $downset\text{-}intro$:

assumes $\forall k \in set\text{-mset}\ K. \exists j \in set\text{-mset}\ J. (k,j) \in r$ **shows** $set\text{-mset}\ K \subseteq dm\ r\ J$

proof

fix x **assume** $x \in set\text{-mset}\ K$ **thus** $x \in dm\ r\ J$ **using** $assms$ **unfolding** $dm\text{-}def$ $ds\text{-}def$ **by** $fast$

qed

lemma $downset\text{-}elim$:

assumes $set\text{-mset}\ K \subseteq dm\ r\ J$ **shows** $\forall k \in set\text{-mset}\ K. \exists j \in set\text{-mset}\ J. (k,j) \in r$

proof

fix k **assume** $k \in set\text{-mset}\ K$ **thus** $\exists j \in set\text{-mset}\ J. (k,j) \in r$ **using** $assms$ **unfolding** $dm\text{-}def$ $ds\text{-}def$ **by** $fast$

qed

to closure-free representation

lemma $mult\text{-}eq\text{-}implies\text{-}one\text{-}or\text{-}zero\text{-}step$:

assumes $trans\ r$ **and** $(M,N) \in mult\text{-}eq\ r$ **shows** $\exists I\ J\ K. N = I + J \wedge M = I + K \wedge set\text{-mset}\ K \subseteq dm\ r\ J$

proof $(cases\ (M,N) \in mult\ r)$

case $True$ **thus** $?thesis$ **using** $mult\text{-}implies\text{-}one\text{-}step[OF\ assms(1)]$ $downset\text{-}intro$ **by** $blast$

next

case $False$ **hence** $A: M = N$ **using** $assms\ rtrancl\text{-}eq\text{-}or\text{-}trancl$ **unfolding** $mult\text{-}eq\text{-}def$ $mult\text{-}def$ **by** $metis$

hence $N = N + \{\#\} \wedge M = M + \{\#\} \wedge set\text{-mset}\ \{\#\} \subseteq dm\ r\ \{\#\}$ **by** $auto$

thus $?thesis$ **unfolding** A **by** $fast$

qed

from closure-free representation

lemma $one\text{-}step\text{-}implies\text{-}mult\text{-}eq$: **assumes** $trans\ r$ **and** $set\text{-mset}\ K \subseteq dm\ r\ J$

shows $(I+K, I+J) \in mult\text{-}eq\ r$

proof $(cases\ set\text{-mset}\ J = \{\})$

case $True$ **hence** $set\text{-mset}\ K = \{\}$ **using** $assms\ downset\text{-}elim$ **by** $(metis\ all\text{-}not\text{-}in\text{-}conv\ emptyE)$

thus $?thesis$ **using** $True$ **unfolding** $mult\text{-}eq\text{-}def$ **by** $auto$

next

case $False$ **hence** $h: J \neq \{\}$ **using** $set\text{-mset}\text{-}eq\text{-}empty\text{-}iff$ **by** $auto$

hence $(I+K, I+J) \in mult\ r$ **using** $set\text{-mset}\text{-}eq\text{-}empty\text{-}iff$ $assms\ one\text{-}step\text{-}implies\text{-}mult$

```

downset-elim
  by auto blast
  thus ?thesis unfolding mult-eq-def mult-def by auto
qed

lemma mult-is-mul: assumes trans r shows mult r = mul r proof
show mult r  $\subseteq$  mul r proof
  fix N M assume A: (N,M)  $\in$  mult r show (N,M)  $\in$  mul r proof –
  obtain I J K where M = I + J and N = I + K and J  $\neq$  {#} and set-mset
K  $\subseteq$  dm r J
  using mult-implies-one-step[OF assms A] downset-intro by metis
  thus ?thesis unfolding mul-def by auto
qed
qed
next
show mul r  $\subseteq$  mult r proof
  fix N M assume A: (N,M)  $\in$  mul r show (N,M)  $\in$  mult r proof –
  obtain I J K where M = I + J and N = I + K and J  $\neq$  {#} and set-mset
K  $\subseteq$  dm r J
  using A unfolding mul-def by auto
  thus ?thesis using one-step-implies-mult assms downset-elim by metis
qed
qed
qed

lemma mult-eq-is-mul-eq: assumes trans r shows mult-eq r = mul-eq r proof
show mult-eq r  $\subseteq$  mul-eq r proof
  fix N M assume A: (N,M)  $\in$  mult-eq r show (N,M)  $\in$  mul-eq r proof (cases
(N,M)  $\in$  mult r)
  case True thus ?thesis unfolding mult-is-mul[OF assms] mul-def mul-eq-def
by auto
  next
  case False hence eq: N = M using A rtranclD unfolding mult-def mult-eq-def
by metis
  hence M = M + {#}  $\wedge$  N = N + {#}  $\wedge$  set-mset {#}  $\subseteq$  dm r {#} by auto
  thus ?thesis unfolding eq unfolding mul-eq-def by fast
qed
qed
show mul-eq r  $\subseteq$  mult-eq r using one-step-implies-mult-eq[OF assms] unfolding
mul-eq-def by auto
qed

lemma mul-eq r = (mul r)= proof
show mul-eq r  $\subseteq$  (mul r)= proof
  fix M N assume A:(M,N)  $\in$  mul-eq r show (M,N)  $\in$  (mul r)= proof –
  from A obtain I J K where 1: M = I + K and 2: N = I + J and 3:
set-mset K  $\subseteq$  dm r J unfolding mul-eq-def by auto
  show ?thesis proof (cases J = {#})
  case True hence K = {#} using 3 unfolding dm-def ds-def by auto

```

```

    hence  $M = N$  using True 1 2 by auto
    thus ?thesis by auto
  next
    case False thus ?thesis using 1 2 3 unfolding mul-def mul-eq-def by auto
  qed
qed
show  $mul\text{-}eq\ r \supseteq (mul\ r)^=$  proof
  fix  $M\ N$  assume  $A:(M,N) \in (mul\ r)^=$  show  $(M,N) \in mul\text{-}eq\ r$ 
  proof (cases  $M = N$ )
    case True hence  $M = M + \{\#\}$  and  $N = M + \{\#\}$  and  $set\text{-}mset\ \{\#\} \subseteq$ 
     $dm\ r\ \{\#\}$  by auto
    thus ?thesis unfolding mul-eq-def by fast
  next
    case False hence  $(M,N) \in mul\ r$  using  $A$  by auto
    thus ?thesis unfolding mul-def mul-eq-def by auto
  qed
qed
qed

```

useful properties on multisets

```

lemma mul-eq-reflexive:  $(M,M) \in mul\text{-}eq\ r$  proof -
  have  $M = M + \{\#\}$  and  $set\text{-}mset\ \{\#\} \subseteq dm\ r\ \{\#\}$  by auto
  thus ?thesis unfolding mul-eq-def by fast
qed

```

```

lemma mul-eq-trans: assumes  $trans\ r$  and  $(M,N) \in mul\text{-}eq\ r$  and  $(N,P) \in mul\text{-}eq\ r$ 
shows  $(M,P) \in mul\text{-}eq\ r$ 
using assms unfolding mult-eq-is-mul-eq[symmetric,OF assms(1)] mult-eq-def
by auto

```

```

lemma mul-eq-singleton: assumes  $(M, \{\#\alpha\#\}) \in mul\text{-}eq\ r$  shows  $M = \{\#\alpha\#\}$ 
 $\vee set\text{-}mset\ M \subseteq dm\ r\ \{\#\alpha\#\}$  proof -
  from assms obtain  $I\ J\ K$  where  $1:M = I + K$  and  $2:\{\#\alpha\#\} = I + J$  and
 $3:set\text{-}mset\ K \subseteq dm\ r\ J$  unfolding mul-eq-def by auto
  thus ?thesis proof (cases  $I = \{\#\}$ )
    case True hence  $J = \{\#\alpha\#\}$  using  $2$  by auto
    thus ?thesis using  $1\ 3\ True$  by auto
  next
    case False hence  $i: I = \{\#\alpha\#\}$  using  $2$  union-is-single by metis
    hence  $J = \{\#\}$  using  $2$  union-is-single by metis
    thus ?thesis using  $1\ i\ 3$  unfolding dm-def ds-def by auto
  qed
qed

```

```

lemma mul-and-mul-eq-imp-mul: assumes  $trans\ r$  and  $(M,N) \in mul\ r$  and
 $(N,P) \in mul\text{-}eq\ r$  shows  $(M,P) \in mul\ r$ 
using assms unfolding mult-is-mul[symmetric,OF assms(1)] mult-eq-is-mul-eq[symmetric,OF
assms(1)] mult-def mult-eq-def by auto

```

lemma *mul-eq-and-mul-imp-mul*: **assumes** *trans r* **and** $(M,N) \in \text{mul-eq } r$ **and** $(N,P) \in \text{mul } r$ **shows** $(M,P) \in \text{mul } r$
using *assms unfolding mult-is-mul[symmetric,OF assms(1)] mult-eq-is-mul-eq[symmetric,OF assms(1)] mult-def mul-eq-def* **by auto**

lemma *wf-mul*: **assumes** *trans r* **and** *wf r* **shows** *wf (mul r)*
unfolding *mult-is-mul[symmetric,OF assms(1)]* **using** *wf-mult[OF assms(2)]* **by auto**

lemma *remove-is-empty-imp-mul*: **assumes** $M -s dm r \{\#\alpha\# \} = \{\#\}$ **shows** $(M,\{\#\alpha\# \}) \in \text{mul } r$ **proof** –
from *assms* **have** $C: \text{set-mset } M \subseteq dm r \{\#\alpha\# \}$ **by** (*metis remove-empty-implies-subset*)
have $M = \{\#\} + M$ **and** $\{\#\alpha\# \} = \{\#\} + \{\#\alpha\# \}$ **and** $\{\#\alpha\# \} \neq \{\#\}$ **by auto**
thus *?thesis* **using** C **unfolding** *mul-def* **by fast**
qed

Lemma 2.6

lemma *lemma2-6-1-set*: $ds r (S \cup T) = ds r S \cup ds r T$
unfolding *set-mset-union ds-def* **by auto**

lemma *lemma2-6-1-list*: $dl r (\sigma @ \tau) = dl r \sigma \cup dl r \tau$
unfolding *dl-def ds-def set-append* **by auto**

lemma *lemma2-6-1-multiset*: $dm r (M + N) = dm r M \cup dm r N$
unfolding *dm-def set-mset-union ds-def* **by auto**

lemma *lemma2-6-1-diff*: $(dm r M) - ds r S \subseteq dm r (M -s S)$
unfolding *diff-def dm-def ds-def* **by** (*rule subsetI*) *auto*

missing but useful

lemma *dl-monotone*: $dl r (\sigma @ \tau) \subseteq dl r (\sigma @ \tau' @ \tau)$ **unfolding** *lemma2-6-1-list* **by auto**

Lemma 2.6.2

lemma *lemma2-6-2-a*: **assumes** $t: \text{trans } r$ **and** $M \subseteq_{\#} N$ **shows** $(M,N) \in \text{mul-eq } r$ **proof** –
from *assms(2)* **obtain** J **where** $N = M + J$ **by** (*metis assms(2) mset-subset-eq-exists-conv*)
hence $M = M + \{\#\}$ **and** $N = M + J$ **and** $\text{set-mset } \{\#\} \subseteq dm r J$ **by auto**
thus *?thesis* **unfolding** *mul-eq-def* **by fast**
qed

lemma *mul-eq-not-equal-imp-elt*:
assumes $(M,N) \in \text{mul-eq } r$ **and** $y \in \text{set-mset } M - \text{set-mset } N$ **shows** $\exists z \in \text{set-mset } N. (y,z) \in r$ **proof** –
from *assms* **obtain** $I J K$ **where** $N = I + J$ **and** $M = I + K$ **and** $F3: \text{set-mset } K \subseteq dm r J$ **unfolding** *mul-eq-def* **by auto**
thus *?thesis* **using** *assms(2) downset-elim[OF F3]* **by auto**
qed

lemma lemma2-6-2-b: *assumes trans r and $(M,N) \in \text{mul-eq } r$ shows $dm \ r \ M \subseteq dm \ r \ N$ proof*
fix x assume A: $x \in dm \ r \ M$ **show $x \in dm \ r \ N$ proof –**
from A obtain y where $F2:y \in \text{set-mset } M$ and $F3:(x,y) \in r$ unfolding dm-def ds-def by auto
hence $\exists z \in \text{set-mset } N. (x,z) \in r$ proof (cases y \in set-mset N)
case True thus ?thesis using F3 unfolding ds-def by auto
next
case False thus ?thesis using mul-eq-not-equal-imp-elt assms F2 F3 trans-def by fast
qed
thus ?thesis unfolding dm-def ds-def by auto
qed
qed

Lemma 2.6.3

lemma ds-trans-contrapos: *assumes t: trans r and $x \notin ds \ r \ S$ and $(x,y) \in r$ shows $y \notin ds \ r \ S$ using assms unfolding ds-def trans-def by fast*

lemma dm-max-elt: *assumes i: irrefl r and t: trans r shows $x \in dm \ r \ M \implies \exists y \in \text{set-mset } (M -s dm \ r \ M). (x,y) \in r$ proof (induct M arbitrary: x) case empty thus ?case unfolding dm-def ds-def by auto next case (add p P) hence mem: $x \in (dm \ r \ P \cup dm \ r \ \{\#p\#})$ unfolding dm-def ds-def by auto from i t have not-mem-dm: $p \notin dm \ r \ \{\#p\#}$ unfolding dm-def ds-def irrefl-def by auto thus ?case proof (cases $x \in dm \ r \ P$) case False hence relp: $(x,p) \in r$ using mem unfolding dm-def ds-def by auto show ?thesis proof (cases $p \in dm \ r \ P$) case True thus ?thesis using relp t ds-trans-contrapos False unfolding dm-def by fast next case False thus ?thesis using not-mem-dm relp unfolding dm-def ds-def diff-def by auto qed next case True obtain y where key: $y \in \text{set-mset } P$ $y \notin dm \ r \ P$ $(x,y) \in r$ using add(1)[OF True] unfolding diff-def by auto thus ?thesis proof (cases $y \in dm \ r \ \{\#p\#}$) case True hence rely: $(y,p) \in r$ unfolding dm-def ds-def by auto hence relp: $(x,p) \in r$ using rely t key trans-def by metis have not-memp: $p \notin \text{set-mset } P$ using rely key unfolding dm-def ds-def by auto*

have $memp: p \in \text{set-mset } (P + \{\#p\#})$ **by** *auto*
have $p \notin dm\ r\ P$ **using** *ds-trans-contrapos[OF t] key(2) rely unfolding dm-def*
by *auto*
hence $p \notin dm\ r\ (P + \{\#p\#})$ **using** *not-mem-dm unfolding dm-def ds-def*
by *auto*
thus *?thesis* **using** *relp unfolding diff-def* **by** *auto*
next
case *False* **thus** *?thesis* **using** *key unfolding dm-def ds-def diff-def* **by** *auto*
qed
qed
qed

lemma *dm-subset*: **assumes** *i:irrefl r* **and** *t: trans r* **shows** $dm\ r\ M \subseteq dm\ r\ (M -s\ dm\ r\ M)$
using *assms dm-max-elt unfolding dm-def ds-def* **by** *fast*

lemma *dm-eq*: **assumes** *i:irrefl r* **and** *t: trans r* **shows** $dm\ r\ M = dm\ r\ (M -s\ dm\ r\ M)$
using *dm-subset[OF assms] unfolding dm-def ds-def diff-def* **by** *auto*

lemma *lemma2-6-3*: **assumes** *t:trans r* **and** *i:irrefl r* **and** $(M,N) \in \text{mul-eq } r$
shows $\exists I' J' K'. N = I' + J' \wedge M = I' + K' \wedge J' \cap\# K' = \{\#\} \wedge \text{set-mset } K' \subseteq dm\ r\ J'$

proof –

from *assms* **obtain** $I J K$ **where** $1:N = I + J$ **and** $2:M = I + K$ **and** $3:\text{set-mset } K \subseteq dm\ r\ J$ **unfolding** *mul-eq-def* **by** *auto*

have $\text{set-mset } (J \cap\# K) \subseteq r \downarrow_m J$ **using** 3 **by** *auto*

then obtain A **where** $r \downarrow_m J = \text{set-mset } (J \cap\# K) \cup A$

by *blast*

then have *key*: $\text{set-mset } (J -s\ dm\ r\ J) \subseteq \text{set-mset } (J - (J \cap\# K))$

by *clarsimp (metis Multiset.count-diff add.left-neutral add-diff-cancel-left' mem-Collect-eq not-gr0 set-mset-def)*

from $1\ 2\ 3$ **have** $N = (I + (J \cap\# K)) + (J - (J \cap\# K))$

by *(metis diff-union-cancelL subset-mset.inf-le2 multiset-diff-union-assoc multiset-inter-commute union-commute union-lcomm)*

moreover have $M = (I + (J \cap\# K)) + (K - (J \cap\# K))$

by *(rule multiset-eqI) (simp add: 2)*

moreover have $\text{set-mset } (K - (J \cap\# K)) \subseteq dm\ r\ (J - (J \cap\# K))$

proof –

have $\text{set-mset } (K - (J \cap\# K)) \subseteq dm\ r\ J$ **using** 3

by *(meson Multiset.diff-subset-eq-self mset-subset-eqD subset-eq)*

moreover have $\dots = dm\ r\ (J -s\ dm\ r\ J)$ **using** *dm-eq[OF i t]* **by** *auto*

moreover have $\dots \subseteq dm\ r\ (J - (J \cap\# K))$ **using** *ds-monotone[OF key] unfolding dm-def* **by** *auto*

ultimately show *?thesis* **by** *auto*

qed

moreover have $(J - (J \cap\# K)) \cap\# (K - (J \cap\# K)) = \{\#\}$ **by** *(rule multiset-eqI) auto*

ultimately show *?thesis* **by** *auto*

qed

Lemma 2.6.4

lemma lemma2-6-4: *assumes* $t: \text{trans } r$ **and** $N \neq \{\#\}$ **and** *set-mset* $M \subseteq \text{dm } r$ N **shows** $(M, N) \in \text{mul } r$ **proof** –
 have $M = \{\#\} + M$ **and** $N = \{\#\} + N$ **using** *assms* **by** *auto*
 thus *?thesis* **using** *assms(2,3)* **unfolding** *mul-def* **by** *fast*
qed

lemma lemma2-6-5-a: *assumes* $t: \text{trans } r$ **and** $\text{ds } r \ S \subseteq S$ **and** $(M, N) \in \text{mul-eq } r$
shows $(M -s S, N -s S) \in \text{mul-eq } r$
proof –
 from *assms(1,3)*
 obtain $I \ J \ K$ **where** $a: N=I+J$ **and** $b:M=I+K$ **and** $c:\text{set-mset } K \subseteq \text{dm } r \ J$
 unfolding *mul-eq-def* **by** *best*
 from $a \ b$ **have** $M -s S = I -s S + K -s S$
 $N -s S = I -s S + J -s S$ **by** (*auto simp add: lemmaA-3-8*)
 moreover **have** $\text{set-mset } (K-sS) \subseteq \text{dm } r \ (J-sS)$ **proof** –
 have $\text{set-mset } (K-sS) \subseteq \text{set-mset } (K-s(\text{ds } r \ S))$ **using** *assms(2)* **unfolding**
 diff-def **by** *auto*
 moreover **have** $\text{set-mset}(K-s(\text{ds } r \ S)) \subseteq (\text{dm } r \ J) - (\text{ds } r \ S)$ **using** c **un-**
 folding *diff-def* **by** *auto*
 moreover **have** $(\text{dm } r \ J) - (\text{ds } r \ S) \subseteq \text{dm } r \ (J -s S)$ **using** *lemma2-6-1-diff*
 by *fast*
 ultimately **show** *?thesis* **by** *auto*
 qed
 ultimately **show** *?thesis* **by** (*rule in-mul-eqI*)
qed

lemma lemma2-6-5-a': *assumes* $t:\text{trans } r$ **and** $(M, N) \in \text{mul-eq } r$ **shows** $(M -s \text{ds } r \ S, N -s \text{ds } r \ S) \in \text{mul-eq } r$
 using *assms lemma2-6-5-a[OF t] ds-ds-subseteq-ds[OF t]* **by** *auto*

Lemma 2.6.6

lemma lemma2-6-6-a: *assumes* $t: \text{trans } r$ **and** $(M, N) \in \text{mul-eq } r$ **shows** $(Q + M, Q + N) \in \text{mul-eq } r$ **proof** –
 obtain $I \ J \ K$ **where** $A: Q+N=(Q+I)+J$ **and** $B: Q+M=(Q+I)+K$ **and** $C:\text{set-mset } K \subseteq \text{dm } r \ J$
 using *assms(2)* **unfolding** *mul-eq-def* **by** *auto*
 thus *?thesis* **using** C **unfolding** *mul-eq-def* **by** *blast*
qed

lemma add-left-one:

assumes $\exists I \ J \ K. \text{add-mset } q \ N = I + J \wedge \text{add-mset } q \ M = I + K \wedge (J \cap \#K = \{\#\}) \wedge \text{set-mset } K \subseteq \text{dm } r \ J$
 shows $\exists I2 \ J \ K. N = I2 + J \wedge M = I2 + K \wedge \text{set-mset } K \subseteq \text{dm } r \ J$ **proof** –
 from *assms* **obtain** $I \ J \ K$ **where** $A: \{\#q\} + N = I + J$ **and** $B: \{\#q\} + M = I + K$

and $C:(J \cap \# K = \{\#\})$ **and** $D:\text{set-mset } K \subseteq \text{dm } r J$ **by** *auto*
have $q \in \# I$ **proof** (*cases* $q \in \# I$)
case *True* **thus** *?thesis* **by** *auto*
next
case *False*
have $q \in \# J$ **using** *False A* **by** (*metis UnE multi-member-this set-mset-union*)
moreover **have** $q \in \# K$ **using** *False B* **by** (*metis UnE multi-member-this set-mset-union*)
moreover **have** $\neg q \in \# (J \cap \# K)$ **using** *C* **by** *auto*
ultimately **show** *?thesis* **by** *auto*
qed
hence $\exists I2. I = \text{add-mset } q I2$ **by** (*metis multi-member-split*)
hence $\exists I2. \text{add-mset } q N = (\text{add-mset } q I2) + J \wedge \text{add-mset } q M = (\text{add-mset } q I2) + K$ **using** *A B* **by** *auto*
thus *?thesis* **using** *D* **by** *auto*
qed

lemma *lemma2-6-6-b-one* :
assumes *trans r* **and** *irrefl r* **and** $(\text{add-mset } q M, \text{add-mset } q N) \in \text{mul-eq } r$
shows $(M, N) \in \text{mul-eq } r$
using *add-left-one[OF lemma2-6-3[OF assms]]* **unfolding** *mul-eq-def* **by** *auto*

lemma *lemma2-6-6-b'* : **assumes** *trans r* **and** *i: irrefl r* **and** $(Q + M, Q + N) \in \text{mul-eq } r$
shows $(M, N) \in \text{mul-eq } r$ **using** *assms(3)* **proof** (*induct Q arbitrary: M N*)
case *empty* **thus** *?case* **by** *auto*
next
case $(\text{add } q Q)$ **thus** *?case* **unfolding** *union-assoc* **using** *lemma2-6-6-b-one[OF assms(1,2)]*
by (*metis union-mset-add-mset-left*)
qed

lemma *lemma2-6-9*: **assumes** *t: trans r* **and** $(M, N) \in \text{mul } r$ **shows** $(Q+M, Q+N) \in \text{mul } r$ **proof** –
obtain $I J K$ **where** $F1:N = I + J$ **and** $F2:M = I + K$ **and** $F3:\text{set-mset } K \subseteq \text{dm } r J$ **and** $F4:J \neq \{\#\}$
using *assms* **unfolding** *mul-def* **by** *auto*
have $Q+N = Q+I+J$ **and** $Q+M = Q+I+K$ **by** (*auto simp: F1 F2 union-commute union-lcomm*)
thus *?thesis* **using** *assms(1) F3 F4* **unfolding** *mul-def* **by** *blast*
qed

Lemma 2.6.7

lemma *lemma2-6-7-a*: **assumes** *t: trans r* **and** $\text{set-mset } Q \subseteq \text{dm } r N - \text{dm } r M$
and $(M, N) \in \text{mul-eq } r$
shows $(Q+M, N) \in \text{mul-eq } r$ **proof** –
obtain $I J K$ **where** $A:N=I+J$ **and** $B:M=I+K$ **and** $C:\text{set-mset } K \subseteq \text{dm } r J$
using *assms* **unfolding** *mul-eq-def* **by** *auto*
hence $\text{set-mset}(Q+K) \subseteq \text{dm } r J$ **using** *assms lemma2-6-1-diff* **unfolding** *dm-def*

ds-def **by** *auto*
hence $(I+(Q+K),I+J) \in \text{mul-eq } r$ **unfolding** *mul-eq-def* **by** *fast*
thus *?thesis* **using** *A B C union-assoc union-commute* **by** *metis*
qed

missing?; similar to lemma_2.6.2?

lemma *lemma2-6-8*: **assumes** *t: trans r* **and** $S \subseteq T$ **shows** $(M -s T, M -s S) \in \text{mul-eq } r$ **proof** –
from *assms(2)* **have** $(M -s T) \subseteq\# (M -s S)$
by (*metis Un-absorb2 Un-commute lemmaA-3-10 lemmaA-3-9 mset-subset-eq-add-right*)
thus *?thesis* **using** *lemma2-6-2-a[OF t]* **by** *auto*
qed

1.1.3 Lexicographic maximum measure

Def 3.1: lexicographic maximum measure

fun *lexmax* :: '*a* *rel* \Rightarrow '*a* *list* \Rightarrow '*a* *multiset* **where**
lexmax *r* [] = {#}
| *lexmax* *r* ($\alpha\#\sigma$) = {# $\alpha\#$ } + (*lexmax* *r* $\sigma -s ds$ *r* { α })

notation

lexmax (-|-| [1000] 1000)

lemma *lexmax-singleton*: $r[\alpha] = \{\#\alpha\#$ **unfolding** *lexmax.simps* *diff-def* **by** *simp*

Lemma 3.2

Lemma 3.2(1)

lemma *lemma3-2-1*: **assumes** *t: trans r* **shows** $r \downarrow m r|\sigma| = r \downarrow l \sigma$ **proof** (*induct* σ)

case *Nil* **show** *?case* **unfolding** *dm-def dl-def* **by** *auto*

next

case (*Cons* α σ)

have dm *r* {# $\alpha\#$ } \cup dm *r* ($r|\sigma| -s ds$ *r* { α }) = dm *r* {# $\alpha\#$ } \cup dl *r* σ (**is** $?L = ?R$) **proof** –

have $?L \subseteq ?R$ **unfolding** *Cons[symmetric]* *diff-def dm-def ds-def* **by** *auto*

moreover **have** $?R \subseteq ?L$ **proof**

fix *x* **assume** *A*: $x \in ?R$ **show** $x \in ?L$ **proof** (*cases* $x \in dm$ *r* {# $\alpha\#$ })

case *True* **thus** *?thesis* **by** *auto*

next

case *False*

hence *mem*: $x \in dm$ *r* (*lexmax* *r* σ) **using** *A Cons* **by** *auto*

have $x \notin (ds$ *r* (ds *r* { α })) **using** *False ds-ds-subseteq-ds[OF t]* **unfolding**

dm-def **by** *auto*

thus *?thesis* **using** *mem lemma2-6-1-diff* **by** *fast*

qed

qed

ultimately **show** *?thesis* **by** *auto*

qed
thus *?case* **unfolding** *lemma2-6-1-multiset lexmax.simps dl-def dm-def ds-def* **by**
auto
qed

Lemma 3.2(2)

lemma *lemma3-2-2*: $r|\sigma@tau| = r|\sigma| + (r|\tau| -s r \downarrow l \sigma)$ **proof** (*induct* σ)
case *Nil* **thus** *?case* **unfolding** *dl-def ds-def lexmax.simps* **apply** *auto* **unfolding**
diff-empty **by** *auto*
next
case (*Cons* $\alpha \sigma$)
have *lexmax* $r (\alpha\#\sigma@tau) = \{\#\alpha\#\} + ((lexmax\ r (\sigma@tau)) -s (ds\ r \{\alpha\}))$ **by** *auto*
moreover **have** $\dots = \{\#\alpha\#\} + ((lexmax\ r \sigma + ((lexmax\ r \tau) -s (dl\ r \sigma))) -s (ds\ r \{\alpha\}))$
using *Cons* **by** *auto*
moreover **have** $\dots = \{\#\alpha\#\} + ((lexmax\ r \sigma) -s (ds\ r \{\alpha\})) + (((lexmax\ r \tau) -s (dl\ r \sigma)) -s (ds\ r \{\alpha\}))$
unfolding *lemmaA-3-8* **unfolding** *union-assoc* **by** *auto*
moreover **have** $\dots = lexmax\ r (\alpha\#\sigma) + (((lexmax\ r \tau) -s (dl\ r \sigma)) -s (ds\ r \{\alpha\}))$ **by** *simp*
moreover **have** $\dots = lexmax\ r (\alpha\#\sigma) + ((lexmax\ r \tau) -s (dl\ r (\alpha\#\sigma)))$ **un-**
folding *lemmaA-3-9 dl-def dm-def lemma2-6-1-set[symmetric]* **by** *auto*
ultimately **show** *?case* **unfolding** *diff-def* **by** *simp*
qed

Definition 3.3

definition $D :: 'a\ rel \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow bool$ **where**
 $D\ r\ \tau\ \sigma\ \sigma'\ \tau' = ((r|\sigma@tau'|, r|\tau| + r|\sigma|) \in mul\text{-}eq\ r$
 $\wedge (r|\tau@sigma'|, r|\tau| + r|\sigma|) \in mul\text{-}eq\ r)$

lemma *D-eq*: **assumes** *trans* r **and** *irrefl* r **and** $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows $(r|\tau'| -s dl\ r\ \sigma, r|\tau|) \in mul\text{-}eq\ r$ **and** $(r|\sigma'| -s dl\ r\ \tau, r|\sigma|) \in mul\text{-}eq\ r$
using *assms* **unfolding** *D-def lemma3-2-2* **using** *union-commute lemma2-6-6-b'*
apply *metis*
using *assms* **unfolding** *D-def lemma3-2-2* **using** *union-commute lemma2-6-6-b'*
by *metis*

lemma *D-inv*: **assumes** *trans* r **and** *irrefl* r **and** $(r|\tau'| -s dl\ r\ \sigma, r|\tau|) \in mul\text{-}eq\ r$
and $(r|\sigma'| -s dl\ r\ \tau, r|\sigma|) \in mul\text{-}eq\ r$
shows $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
using *assms* **unfolding** *D-def lemma3-2-2* **using** *lemma2-6-6-a[OF assms(1)]*
union-commute **by** *metis*

lemma *D*: **assumes** *trans* r **and** *irrefl* r
shows $D\ r\ \tau\ \sigma\ \sigma'\ \tau' = ((r|\tau'| -s dl\ r\ \sigma, r|\tau|) \in mul\text{-}eq\ r$
 $\wedge (r|\sigma'| -s dl\ r\ \tau, r|\sigma|) \in mul\text{-}eq\ r)$
using *assms* *D-eq* *D-inv* **by** *auto*

lemma *mirror-D*: **assumes** *trans r* **and** *irrefl r* **and** $D r \tau \sigma \sigma' \tau'$ **shows** $D r \sigma \tau \tau' \sigma'$

using *assms D* **by** *metis*

Proposition 3.4

definition $LD-1'$:: $'a \text{ rel} \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow \text{bool}$

where $LD-1' r \beta \alpha \sigma 1 \sigma 2 \sigma 3 =$

$(\text{set } \sigma 1 \subseteq \text{ds } r \{\beta\} \wedge \text{length } \sigma 2 \leq 1 \wedge \text{set } \sigma 2 \subseteq \{\alpha\} \wedge \text{set } \sigma 3 \subseteq \text{ds } r \{\alpha, \beta\})$

definition LD' :: $'a \text{ rel} \Rightarrow 'a \Rightarrow 'a$

$\Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow \text{bool}$

where $LD' r \beta \alpha \sigma 1 \sigma 2 \sigma 3 \tau 1 \tau 2 \tau 3 = (LD-1' r \beta \alpha \sigma 1 \sigma 2 \sigma 3 \wedge LD-1' r \alpha \beta \tau 1 \tau 2 \tau 3)$

auxiliary properties on multisets

lemma *lexmax-le-multiset*: **assumes** *t:trans r* **shows** $r|\sigma| \subseteq\# \text{mset } \sigma$ **proof** (*induct* σ)

case *Nil* **thus** *?case unfolding lexmax.simps* **by** *auto*

next

case (*Cons s* τ) **hence** *lexmax r* $\tau -s \text{ds } r \{s\} \subseteq\# \text{mset } \tau$ **using** *lemmaA-3-10 mset-subset-eq-add-right subset-mset.order-trans* **by** *metis*

thus *?case* **by** *simp*

qed

lemma *split*: **assumes** $LD-1' r \beta \alpha \sigma 1 \sigma 2 \sigma 3$ **shows** $\sigma 2 = [] \vee \sigma 2 = [\alpha]$

using *assms unfolding LD-1'-def* **by** (*cases* $\sigma 2$) *auto*

lemma *proposition3-4-step*: **assumes** *trans r* **and** *irrefl r* **and** $LD-1' r \beta \alpha \sigma 1 \sigma 2 \sigma 3$

shows $(r|\sigma 1 @ \sigma 2 @ \sigma 3| -s (\text{dm } r \{\#\beta\#}), r|[\alpha]|) \in \text{mul-eq } r$ **proof** –

from *assms* **have** $\text{set } \sigma 1 \subseteq \text{dm } r \{\#\beta\#}$ **unfolding** *LD'-def LD-1'-def dm-def* **by** *auto*

hence $\text{set-mset } (\text{lexmax } r \sigma 1) \subseteq \text{dm } r \{\#\beta\#}$ **using** *submultiset-implies-subset[OF lexmax-le-multiset[OF assms(1)]]* **by** *auto*

hence *one*: $\text{lexmax } r \sigma 1 -s \text{dm } r \{\#\beta\#} = \{\#\}$ **using** *subset-implies-remove-empty* **by** *auto*

from *assms* **have** $\text{set } \sigma 3 \subseteq \text{dl } r \sigma 2 \cup \text{dl } r \sigma 1 \cup \text{dm } r \{\#\beta\#} \cup \text{dm } r \{\#\alpha\#}$ (*is ?l* \subseteq *?r*) **unfolding** *LD'-def LD-1'-def dm-def ds-def* **by** *auto*

hence $\text{set-mset } (\text{lexmax } r \sigma 3) \subseteq ?r$ **using** *submultiset-implies-subset[OF lexmax-le-multiset[OF assms(1)]]* **by** *auto*

hence *pre3*: $\text{lexmax } r \sigma 3 -s ?r = \{\#\}$ **using** *subset-implies-remove-empty* **by** *auto*

show *?thesis* **proof** (*cases* $\sigma 2 = []$)

case *True*

hence *two*: $(\text{lexmax } r \sigma 2 -s \text{dl } r \sigma 1) -s \text{dm } r \{\#\beta\#} = \{\#\}$ **unfolding** *diff-def* **by** *auto*

from *pre3* **have** $((\text{lexmax } r \sigma 3 -s \text{dl } r \sigma 2) -s \text{dl } r \sigma 1) -s \text{dm } r \{\#\beta\#} -s \text{dm } r \{\#\alpha\#} = \{\#\}$ **unfolding** *True dl-def lemmaA-3-9* **by** *auto*

hence *three*: $((\text{lexmax } r \sigma 3 -s \text{dl } r \sigma 2) -s \text{dl } r \sigma 1) -s \text{dm } r \{\#\beta\#}, \{\#\alpha\#}$

$\in \text{mul } r$ **using** *remove-is-empty-imp-mul* **by** *metis*
show *?thesis* **using** *three unfolding lemma3-2-2 lexmax-singleton lemmaA-3-8*
one two mul-def mul-eq-def **by** *auto*
next
case *False* **hence** $\text{eq}: \sigma 2 = [\alpha]$ **using** *split[OF assms(3)]* **by** *fast*
have *two*: $((\text{lexmax } r \ \sigma 2 \ -s \ \text{dl } r \ \sigma 1) \ -s \ \text{dm } r \ \{\#\beta\#\}, \{\#\alpha\#\}) \in \text{mul-eq } r$ **using**
lemma2-6-8[OF assms(1) empty-subsetI] **unfolding** *eq lexmax.simps diff-from-empty*
lemmaA-3-9 diff-empty **by** *auto*
from *pre3* **have** $\text{lexmax } r \ \sigma 3 \ -s \ ((\text{dl } r \ \sigma 2 \cup \ \text{dm } r \ \{\#\alpha\#\}) \cup \ \text{dl } r \ \sigma 1 \cup \ \text{dm } r$
 $\{\#\beta\#\}) = \{\#\}$ **unfolding** *eq lemmaA-3-9* **using** *Un-assoc Un-commute* **by** *metis*
hence *three*: $((\text{lexmax } r \ \sigma 3 \ -s \ \text{dl } r \ \sigma 2) \ -s \ \text{dl } r \ \sigma 1) \ -s \ \text{dm } r \ \{\#\beta\#\} = \{\#\}$
using *Un-absorb* **unfolding** *lemmaA-3-9 eq dm-def dl-def* **by** *auto*
show *?thesis* **unfolding** *lemma3-2-2 lexmax-singleton lemmaA-3-8 one three*
using two **by** *auto*
qed
qed

lemma *proposition3-4*:

assumes *t*: *trans* r **and** *i*: *irrefl* r **and** *ld*: $LD' \ r \ \beta \ \alpha \ \sigma 1 \ \sigma 2 \ \sigma 3 \ \tau 1 \ \tau 2 \ \tau 3$
shows $D \ r \ [\beta] \ [\alpha] \ (\sigma 1 @ \sigma 2 @ \sigma 3) \ (\tau 1 @ \tau 2 @ \tau 3)$
using *proposition3-4-step[OF t i] ld* **unfolding** *LD'-def D[OF assms(1,2)] dl-def*
dm-def **by** *auto*

lemma *lexmax-decompose*: **assumes** $\alpha \in \# \ r | \sigma$ **shows** $\exists \sigma 1 \ \sigma 3. (\sigma = \sigma 1 @ [\alpha] @ \sigma 3$
 $\wedge \alpha \notin \text{dl } r \ \sigma 1)$

using *assms* **proof** (*induct* σ)
case *Nil* **thus** *?case* **by** *auto*
next
case (*Cons* $\beta \ as$) **thus** *?case* **proof** (*cases* $\alpha = \beta$)
case *True*
from *this* **obtain** $\sigma 1 \ \sigma 3$ **where** *dec*: $\beta \# as = \sigma 1 @ [\alpha] @ \sigma 3$ **and** *empty*: $\sigma 1 = []$
by *auto*
hence $\alpha \notin \text{dl } r \ \sigma 1$ **unfolding** *dl-def ds-def* **by** *auto*
thus *?thesis* **using** *dec* **by** *auto*
next
case *False* **hence** $\alpha \in \# \ r | as | -s \ ds \ r \ \{\beta\}$ **using** *Cons(2)* **by** *auto*
hence x : $\alpha \in \# \ r | as | \wedge \alpha \notin ds \ r \ \{\beta\}$
by *simp*
from *this* **obtain** $\sigma 1 \ \sigma 3$ **where** $as = \sigma 1 @ [\alpha] @ \sigma 3$ **and** w : $\alpha \notin \text{dl } r \ \sigma 1$ **using**
Cons(1) **by** *auto*
hence $\beta \# as = (\beta \# \sigma 1) @ [\alpha] @ \sigma 3$ **and** $\alpha \notin \text{dl } r \ (\beta \# \sigma 1)$ **using** $x \ w$ **unfolding**
dm-def dl-def ds-def **by** *auto*
thus *?thesis* **by** *fast*
qed
qed

lemma *lexmax-elt*: **assumes** *trans* r **and** $x \in (\text{set } \sigma)$ **and** $x \notin \text{set-mset } r | \sigma |$

shows $\exists y. (x,y) \in r \wedge y \in \text{set-mset } r|\sigma|$ **using** *assms(2,3)* **proof** (*induct* σ)
case *Nil* **thus** *?case* **by** *auto*
next
case (*Cons a as*) **thus** *?case* **proof** (*cases* $x \notin \text{set-mset } r|as|$)
case *True*
from *Cons True* **obtain** y **where** $s: (x, y) \in r \wedge y \in \text{set-mset } r|as|$ **by** *auto*
thus *?thesis* **proof** (*cases* $y \in ds\ r\ \{a\}$)
case *True* **thus** *?thesis* **using** *transD[OF assms(1)] s* **unfolding** *dm-def ds-def*
by *auto*
next
case *False* **thus** *?thesis* **using** s **unfolding** *lexmax.simps diff-def* **by** *auto*
qed
next
case *False* **thus** *?thesis* **using** *Cons* **unfolding** *diff-def dm-def ds-def lex-*
max.simps **by** *auto*
qed
qed

lemma *lexmax-set*: **assumes** *trans r* **and** $\text{set-mset } r|\sigma| \subseteq r \downarrow s\ S$ **shows** $\text{set } \sigma \subseteq r \downarrow s\ S$ **proof**
fix x **assume** $A: x \in \text{set } \sigma$ **show** $x \in ds\ r\ S$ **proof** (*cases* $x \in \text{set-mset } r|\sigma|$)
case *True* **thus** *?thesis* **using** *assms* **by** *auto*
next
case *False* **from** *lexmax-elt[OF assms(1) A False]* **obtain** y
where $rel: (x,y) \in r \wedge y \in \text{set-mset } r|\sigma|$ **by** *auto*
hence $y \in ds\ r\ S$ **using** *assms* **by** *auto*
thus *?thesis* **using** rel *assms transD* **unfolding** *dm-def ds-def* **by** *fast*
qed
qed

lemma *drop-left-mult-eq*:
assumes *trans r* **and** *irrefl r* **and** $(N+M, M) \in \text{mul-eq } r$ **shows** $N = \{\#\}$ **proof**
 $-$
have $(M+N, M+\{\#\}) \in \text{mul-eq } r$ **using** *assms(3)* **apply** *auto* **using** *union-commute*
by *metis*
hence $k: (N, \{\#\}) \in \text{mul-eq } r$ **using** *lemma2-6-6-b'[OF assms(1,2)]* **by** *fast*
from *this* **obtain** $I\ J\ K$ **where** $\{\#\} = I + J \wedge N = I + K \wedge \text{set-mset } K \subseteq dm$
 $r\ J$ **unfolding** *mul-eq-def* **by** *fast*
thus *?thesis* **unfolding** *dm-def ds-def* **by** *auto*
qed

generalized to lists

lemma *proposition3-4-inv-lists*:
assumes $t: \text{trans } r$ **and** $i: \text{irrefl } r$ **and** $k: (r|\sigma| -s\ r \downarrow l\ \beta, \{\#\alpha\#\}) \in \text{mul-eq } r$ (**is**
 $(?M, -) \in -$)
shows $\exists \sigma 1\ \sigma 2\ \sigma 3. ((\sigma = \sigma 1 @ \sigma 2 @ \sigma 3) \wedge \text{set } \sigma 1 \subseteq dl\ r\ \beta \wedge \text{length } \sigma 2 \leq 1 \wedge$
 $\text{set } \sigma 2 \subseteq \{\alpha\}) \wedge \text{set } \sigma 3 \subseteq dl\ r\ (\alpha\#\beta)$ **proof** (*cases* $\alpha \in \#\ ?M$)
case *True* **hence** $\alpha \in \#\ r|\sigma|$ **by** *simp*
from *this* **obtain** $\sigma 1\ \sigma 3$ **where** $\text{sigma}: \sigma = \sigma 1 @ [\alpha] @ \sigma 3$ **and** $\text{alpha}: \alpha \notin dl\ r\ \sigma 1$

using *lexmax-decompose* **by** *metis*
hence *dec*: $((r|\sigma 1|-sdl\ r\ \beta) + (r|[\alpha]|-s\ (dl\ r\ \sigma 1 \cup dl\ r\ \beta)) + (r|\sigma 3|-s\ (dl\ r\ [\alpha] \cup dl\ r\ \sigma 1 \cup dl\ r\ \beta)), \{\#\alpha\#\}) \in mul\text{-}eq\ r$ (**is** $(?M1 + ?M2 + ?M3, -) \in -$)
using *k unfolding sigma lemma3-2-2 lemmaA-3-8 lemmaA-3-9 LD-1'-def union-assoc*
by *auto*

from *True* **have** *key*: $\alpha \notin dl\ r\ \beta$ **by** *simp*
hence $?M2 = \{\#\alpha\#\}$ **unfolding** *lexmax-singleton diff-def* **using** *alpha* **by** *auto*
hence $(?M1 + ?M3 + \{\#\alpha\#\}, \{\#\alpha\#\}) \in mul\text{-}eq\ r$ **using** *dec union-assoc union-commute*
by *metis*
hence *w*: $?M1 + ?M3 = \{\#\}$ **using** *drop-left-mult-eq assms(1,2)* **by** *blast*
from *w* **have** $(r|\sigma 1|-sdl\ r\ \beta) = \{\#\}$ **by** *auto*
hence *set-mset* $r|\sigma 1| \subseteq ds\ r\ (set\ \beta)$ **using** *remove-empty-implies-subset* **unfolding** *dl-def dm-def* **by** *auto*
hence *sigma1*: $set\ \sigma 1 \subseteq ds\ r\ (set\ \beta)$ **using** *lexmax-set[OF assms(1)]* **by** *auto*

have *sigma2*: $length\ [\alpha] \leq 1 \wedge set\ [\alpha] \subseteq \{\alpha\}$ **by** *auto*

have *sub*: $dl\ r\ \sigma 1 \subseteq dl\ r\ \beta$ **using** *subset-imp-ds-subset[OF assms(1) sigma1]*
unfolding *dm-def dl-def* **by** *auto*
hence *sub2*: $dl\ r\ \sigma 1 \cup dl\ r\ \beta = dl\ r\ \beta$ **by** *auto*
from *w* **have** $?M3 = \{\#\}$ **by** *auto*
hence $r|\sigma 3|-s\ (ds\ r\ \{\alpha\} \cup ds\ r\ (set\ \beta)) = \{\#\}$ **unfolding** *Un-assoc sub2*
unfolding *dl-def* **by** *auto*
hence $r|\sigma 3|-s\ (ds\ r\ (\{\alpha\} \cup (set\ \beta))) = \{\#\}$ **unfolding** *lemma2-6-1-set[symmetric]*
by *metis*
hence *set-mset* $r|\sigma 3| \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **using** *remove-empty-implies-subset*
by *auto*
hence *sigma3*: $set\ \sigma 3 \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **using** *lexmax-set[OF assms(1)]*
by *auto*
show *?thesis* **using** *sigma sigma1 sigma2 sigma3* **unfolding** *dl-def* **apply** *auto*
by (*metis One-nat-def append-Cons append-Nil sigma2*)
next
case *False* **hence** *set-mset* $?M \subseteq dm\ r\ \{\#\alpha\#\}$ **using** *mul-eq-singleton[OF k]*
by (*auto dest: diff-eq-singleton-imp*)
hence *set-mset* $r|\sigma| \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **unfolding** *diff-def dm-def dl-def*
ds-def **by** *auto*
hence $set\ \sigma \subseteq ds\ r\ (\{\alpha\} \cup (set\ \beta))$ **using** *lexmax-set[OF assms(1)]* **by** *auto*
thus *?thesis* **unfolding** *dl-def* **apply** *auto* **by** (*metis append-Nil bot-least empty-set*
le0 length-0-conv)
qed

lemma *proposition3-4-inv-step*:
assumes *t*: *trans* *r* **and** *i*: *irrefl* *r* **and** *k*: $(r|\sigma| -s\ r\ \downarrow l\ [\beta], \{\#\alpha\#\}) \in mul\text{-}eq\ r$ (**is** $(?M, -) \in -$)
shows $\exists\ \sigma 1\ \sigma 2\ \sigma 3. ((\sigma = \sigma 1 @ \sigma 2 @ \sigma 3) \wedge LD-1'\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3)$
using *proposition3-4-inv-lists[OF assms]* **unfolding** *LD-1'-def dl-def* **by** *auto*

lemma *proposition3-4-inv*:

assumes t : $\text{trans } r$ **and** i : $\text{irrefl } r$ **and** $D r$ $[\beta]$ $[\alpha]$ $\sigma \tau$
shows $\exists \sigma 1 \sigma 2 \sigma 3 \tau 1 \tau 2 \tau 3. (\sigma = \sigma 1 @ \sigma 2 @ \sigma 3 \wedge \tau = \tau 1 @ \tau 2 @ \tau 3 \wedge LD' r \beta \alpha$
 $\sigma 1 \sigma 2 \sigma 3 \tau 1 \tau 2 \tau 3)$
using $\text{proposition3-4-inv-step}[OF \text{ assms}(1,2)]$ $D\text{-eq}[OF \text{ assms}]$ **unfolding** lexmax-singleton
 $LD'\text{-def}$ **by** metis

Lemma 3.5

lemma lemma3-5-1 :

assumes t : $\text{trans } r$ **and** $\text{irrefl } r$ **and** $D r \tau \sigma \sigma' \tau'$ **and** $D r v \sigma' \sigma'' v'$
shows $(\text{lexmax } r (\tau @ v @ \sigma''), \text{lexmax } r (\tau @ v) + \text{lexmax } r \sigma) \in \text{mul-eq } r$ **proof**
 –
have $\text{lexmax } r (\tau @ v @ \sigma'') = (\text{lexmax } r (\tau @ v) + ((\text{lexmax } r \sigma'') -s (\text{dl } r$
 $(\tau @ v))))$
unfolding $\text{append-assoc[symmetric]}$ **using** lemma3-2-2 **by** fast
moreover have $x \dots = \text{lexmax } r (\tau @ v) + (((\text{lexmax } r \sigma'') -s \text{dl } r v) -s \text{dl } r \tau)$
by $(\text{auto simp: lemma2-6-1-list lemmaA-3-9 Un-commute})$
moreover have $(\dots, \text{lexmax } r (\tau @ v) + (\text{lexmax } r \sigma' -s \text{dl } r \tau)) \in \text{mul-eq } r$ **(is**
 $(-, ?R) \in -)$
using $\text{lemma2-6-6-a}[OF t \text{ lemma2-6-5-a}'[OF t D\text{-eq}(2)[OF \text{ assms}(1,2,4)]]]$ **un-**
folding dl-def **by** auto
moreover have $(?R, \text{lexmax } r (\tau @ v) + \text{lexmax } r \sigma) \in \text{mul-eq } r$
using $\text{lemma2-6-6-a}[OF t D\text{-eq}(2)[OF \text{ assms}(1-3)]]$ **by** auto
ultimately show $?thesis$ **using** $\text{mul-eq-trans}[OF t]$ **by** metis
qed

lemma claim1 : **assumes** t : $\text{trans } r$ **and** $D r \tau \sigma \sigma' \tau'$

shows $(r|\sigma @ \tau' + ((r|v' -s r \downarrow l (\sigma @ \tau')) \cap s r \downarrow l \tau), r|\sigma + r|\tau) \in \text{mul-eq } r$ **(is**
 $(?F + ?H, ?G) \in -)$

proof –

have 1 : $(?F, ?G) \in \text{mul-eq } r$ **using** $\text{assms}(2)$ **unfolding** $D\text{-def}$ **by** $(\text{auto simp:}$
 $\text{union-commute})$

have 2 : $\text{set-mset } ?H \subseteq (\text{dm } r ?G) - (\text{dm } r ?F)$ **(is** $(?L \subseteq -)$) **proof** –

have $\text{set-mset } ?H = \text{set-mset } ((\text{lexmax } r v' \cap s \text{dl } r \tau) -s \text{dl } r (\sigma @ \tau'))$ **unfolding**
 lemmaA-3-11 **by** auto

moreover have $\dots \subseteq (\text{dl } r \tau - \text{dl } r (\sigma @ \tau'))$ **unfolding** $\text{diff-def intersect-def}$
by auto

moreover have $\dots \subseteq ((\text{dl } r \sigma \cup \text{dl } r \tau) - \text{dl } r (\sigma @ \tau'))$ **by** auto

ultimately show $?thesis$ **unfolding** $\text{lemma2-6-1-multiset lemma3-2-1}[OF t]$ **by**
 auto

qed

show $?thesis$ **using** $\text{lemma2-6-7-a}[OF t 2 1]$ **by** $(\text{auto simp: union-commute})$

qed

lemma step3 : **assumes** t : $\text{trans } r$ **and** $D r \tau \sigma \sigma' \tau'$

shows $r \downarrow l (\sigma @ \tau) \supseteq (r \downarrow m (r|\sigma' + r|\tau))$ **proof** –

have a : $\text{dl } r (\sigma @ \tau) = \text{dm } r (\text{lexmax } r \tau + \text{lexmax } r \sigma)$ **and** b : $\text{dl } r (\tau @ \sigma') = \text{dm}$
 $r (\text{lexmax } r \sigma' + \text{lexmax } r \tau)$

unfolding $\text{lemma2-6-1-list lemma3-2-1}[OF t, \text{symmetric}]$ $\text{lemma2-6-1-multiset}$
by auto

show *?thesis* **using** *assms(2)* *lemma2-6-2-b[OF t]* *lemma3-2-1[OF t,symmetric]*
unfolding *D-def a b[symmetric]* **by** *auto*
qed

lemma *claim2*: **assumes** *t*: *trans r* **and** *D r τ σ σ' τ'*
shows $((r|v'| -s \ r \ \downarrow l \ (\sigma @ \tau')) -s \ r \ \downarrow l \ \tau, (r|v'| -s \ r \ \downarrow l \ \sigma') -s \ r \ \downarrow l \ \tau) \in \text{mul-eq } r$
(is *?L,?R*) **∈ -)**
proof –
have *?L = lexmax r v' -s (dl r (σ@τ'@τ))* **unfolding** *lemmaA-3-9 append-assoc[symmetric]*
lemma2-6-1-list **by** *auto*
moreover **have** $(\dots, \text{lexmax } r \ v' -s \ \text{dl } r \ (\sigma @ \tau)) \in \text{mul-eq } r$ **(is** *(-,?R)*) **∈ -)** **using**
lemma2-6-8[OF t dl-monotone] **by** *auto*
moreover **have** $(?R, \text{lexmax } r \ v' -s \ \text{dm } r \ (\text{lexmax } r \ \sigma' + \text{lexmax } r \ \tau)) \in \text{mul-eq}$
 r **(is** *(-,?R)*) **∈ -)** **using** *lemma2-6-8[OF t step3[OF assms]]* **by** *auto*
moreover **have** *?R = (lexmax r v' -s dl r σ') -s dl r τ* **unfolding** *lemma3-2-1[OF*
t,symmetric] *lemma2-6-1-multiset* *lemmaA-3-9[symmetric]* **by** *auto*
ultimately show *?thesis* **using** *mul-eq-trans[OF t]* **by** *metis*
qed

lemma *lemma3-5-2*: **assumes** *trans r* **and** *irrefl r* **and** *D r τ σ σ' τ'* **and** *D r v*
 $\sigma' \ \sigma'' \ v'$
shows $(r|(\sigma @ \tau' @ v')|, r|\sigma| + r|(\tau @ v)|) \in \text{mul-eq } r$
proof –
have *0: lexmax r (σ@τ'@v') = lexmax r (σ@τ') + (lexmax r v' -s dl r (σ@τ'))*
(is *?L = -)*
unfolding *append-assoc[symmetric]* *lemma3-2-2* **by** *auto*
moreover **have** $\dots = \text{lexmax } r \ (\sigma @ \tau') + ((\text{lexmax } r \ v' -s \ \text{dl } r \ (\sigma @ \tau')) \cap s \ \text{dl } r$
 $\tau) + ((\text{lexmax } r \ v' -s \ \text{dl } r \ (\sigma @ \tau')) -s \ \text{dl } r \ \tau)$
using *lemmaA-3-10* **unfolding** *union-assoc* **by** *auto*
moreover **have** $(\dots, \text{lexmax } r \ \sigma + \text{lexmax } r \ \tau + ((\text{lexmax } r \ v' -s \ \text{dl } r \ (\sigma @ \tau'))$
 $-s \ \text{dl } r \ \tau)) \in \text{mul-eq } r$ **(is** *(-,?R)*) **∈ -)**
using *assms claim1 lemma2-6-6-a union-commute* **by** *metis*
moreover **have** $(?R, \text{lexmax } r \ \sigma + \text{lexmax } r \ \tau + (((\text{lexmax } r \ v' -s \ \text{dl } r \ \sigma') -s$
 $\text{dl } r \ \tau))) \in \text{mul-eq } r$ **(is** *(-,?R)*) **∈ -)**
using *lemma2-6-6-a[OF assms(1) claim2[OF assms(1,3)]]* **by** *auto*
moreover **have** $(?R, \text{lexmax } r \ \sigma + \text{lexmax } r \ \tau + \text{lexmax } r \ v -s \ \text{dl } r \ \tau) \in \text{mul-eq}$
 r **(is** *(-,?R)*) **∈ -)**
using *lemma2-6-6-a[OF assms(1) lemma2-6-5-a'[OF assms(1) D-eq(1)[OF assms(1,2,4)]]*
unfolding *dl-def* **by** *auto*
moreover **have** *?R = lexmax r σ + lexmax r (τ@v)* **unfolding** *union-assoc*
lemma3-2-2 **by** *auto*
ultimately show *?thesis* **using** *mul-eq-trans[OF assms(1)]* **by** *metis*
qed

lemma *lemma3-5*: **assumes** *trans r* **and** *irrefl r* **and** *D r τ σ σ' τ'* **and** *D r v*
 $\sigma' \ \sigma'' \ v'$
shows *D r (τ@v) σ σ'' (τ'@v')*
unfolding *D-def append-assoc* **using** *assms lemma3-5-1 lemma3-5-2 union-commute*
by *metis*

lemma step2: assumes $trans\ r$ and $\tau \neq []$ shows $(M \cap s\ dl\ r\ \tau, lexmax\ r\ \tau) \in mul\ r$ **proof** –
from $assms$ **obtain** $x\ xs$ **where** $\tau = x \# xs$ **using** $list.exhaust$ **by** $auto$
hence $x: lexmax\ r\ \tau \neq \{\#\}$ **by** $auto$
from $assms(2)$ **have** $y: set-mset\ (M \cap s\ dl\ r\ \tau) \subseteq dm\ r\ (lexmax\ r\ \tau)$ **unfolding**
 $lemma3-2-1[OF\ assms(1)]\ intersect-def$ **by** $auto$
show $?thesis$ **using** $lemma2-6-4[OF\ assms(1)\ x\ y]$ **by** $auto$
qed

Lemma 3.6

lemma lemma3-6: assumes $t: trans\ r$ and $ne: \tau \neq []$ and $D: D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows $(r|\sigma'| + r|v|, r|\sigma| + r|\tau@v|) \in mul\ r$ (**is** $(?L, ?R) \in -$) **proof** –
have $?L = ((lexmax\ r\ \sigma' + lexmax\ r\ v) \cap s\ dl\ r\ \tau) + ((lexmax\ r\ \sigma' + lexmax\ r\ v) - s\ dl\ r\ \tau)$
unfolding $lemmaA-3-10[symmetric]$ **by** $auto$
moreover **have** $(\dots, lexmax\ r\ \tau + ((lexmax\ r\ \sigma' + lexmax\ r\ v) - s\ dl\ r\ \tau)) \in mul\ r$ (**is** $(-, ?R2) \in -$)
using $lemma2-6-9[OF\ t\ step2[OF\ t\ ne]]\ union-commute$ **by** $metis$
moreover **have** $?R2 = lexmax\ r\ \tau + (lexmax\ r\ \sigma' - s\ dl\ r\ \tau) + (lexmax\ r\ v - s\ dl\ r\ \tau)$
unfolding $lemmaA-3-8\ union-assoc[symmetric]$ **by** $auto$
moreover **have** $\dots = lexmax\ r\ (\tau@v) + (lexmax\ r\ v - s\ dl\ r\ \tau)$ **unfolding**
 $lemma3-2-2$ **by** $auto$
moreover **have** $(\dots, lexmax\ r\ \sigma + lexmax\ r\ \tau + (lexmax\ r\ v - s\ dl\ r\ \tau)) \in mul-eq\ r$ (**is** $(-, ?R5) \in -$)
using D **unfolding** $D-def$ **using** $lemma2-6-6-a[OF\ t]\ union-commute$ **by** $metis$
moreover **have** $?R5 = ?R$ **unfolding** $lemma3-2-2\ union-assoc$ **by** $auto$
ultimately **show** $?thesis$ **using** $mul-and-mul-eq-imp-mul\ t$ **by** $metis$
qed

lemma lemma3-6-v: assumes $trans\ r$ and $irrefl\ r$ and $\sigma \neq []$ and $D\ r\ \tau\ \sigma\ \sigma'\ \tau'$
shows $(r|\tau'| + r|v|, r|\tau| + r|\sigma@v|) \in mul\ r$
using $assms\ lemma3-6\ mirror-D$ **by** $fast$

1.1.4 Labeled Rewriting

Theorem 3.7

type-synonym $('a, 'b)\ lars = ('a \times 'b \times 'a)\ set$
type-synonym $('a, 'b)\ seq = ('a \times ('b \times 'a))\ list$

inductive-set $seq :: ('a, 'b)\ lars \Rightarrow ('a, 'b)\ seq\ set$ **for** ars
where $(a, []) \in seq\ ars$
 $| (a, \alpha, b) \in ars \Longrightarrow (b, ss) \in seq\ ars \Longrightarrow (a, (\alpha, b) \# ss) \in seq\ ars$

definition $lst :: ('a, 'b)\ seq \Rightarrow 'a$
where $lst\ ss = (if\ snd\ ss = []\ then\ fst\ ss\ else\ snd\ (last\ (snd\ ss)))$

results on seqs

lemma seq-tail1: **assumes** $seq: (s, x\#xs) \in seq\ lars$
shows $(snd\ x, xs) \in seq\ lars$ **and** $(s, fst\ x, snd\ x) \in lars$ **and** $lst\ (s, x\#xs) = lst\ (snd\ x, xs)$
proof –
 show $(snd\ x, xs) \in seq\ lars$ **using** *assms* **by** (*cases*) *auto*
next
 show $(s, fst\ x, snd\ x) \in lars$ **using** *assms* **by** (*cases*) *auto*
next
 show $lst\ (s, x\#xs) = lst\ (snd\ x, xs)$ **using** *assms* **unfolding** *lst-def* **by** (*cases*) *auto*
qed

lemma seq-chop: **assumes** $(s, ss@ts) \in seq\ ars$ **shows** $(s, ss) \in seq\ ars$ $(lst(s, ss), ts) \in seq\ ars$ **proof** –
 show $(s, ss) \in seq\ ars$ **using** *assms* **proof** (*induct ss arbitrary: s*)
 case *Nil* **show** *?case* **using** *seq.intros(1)* **by** *fast*
 next
 case (*Cons x xs*) **hence** $k:(s, x\#(xs@ts)) \in seq\ ars$ **by** *auto*
 from *Cons* **have** $(snd\ x, xs) \in seq\ ars$ **using** *seq-tail1(1)* **unfolding** *append.simps*
by *fast*
 thus *?case* **using** *seq.intros(2)*[*OF seq-tail1(2)*][*OF k*] **by** *auto*
 qed
next
 show $(lst(s, ss), ts) \in seq\ ars$ **using** *assms* **proof** (*induct ss arbitrary: s*)
 case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
 next
 case (*Cons x xs*)
 hence $(lst\ (snd\ x, xs), ts) \in seq\ ars$ **using** *seq-tail1(1)* **unfolding** *append.simps*
by *fast*
 thus *?case* **unfolding** *lst-def* **by** *auto*
 qed
qed

lemma seq-concat-helper:
assumes $(s, ls) \in seq\ ars$ **and** $ss2 \in seq\ ars$ **and** $lst\ (s, ls) = fst\ ss2$
shows $(s, ls@snd\ ss2) \in seq\ ars \wedge (lst\ (s, ls@snd\ ss2) = lst\ ss2)$
using *assms* **proof** (*induct ls arbitrary: s ss2 rule:list.induct*)
 case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
 next
 case (*Cons x xs*)
 hence $(snd\ x, xs) \in seq\ ars$ **and** $mem:(s, fst\ x, snd\ x) \in ars$ **and** $lst\ (snd\ x, xs) = fst\ ss2$
 using *seq-tail1*[*OF Cons(2)*] *Cons(4)* **by** *auto*
 thus *?case* **using** *Cons seq.intros(2)*[*OF mem*] **unfolding** *lst-def* **by** *auto*
 qed

lemma seq-concat:
assumes $ss1 \in seq\ ars$ **and** $ss2 \in seq\ ars$ **and** $lst\ ss1 = fst\ ss2$
shows $(fst\ ss1, snd\ ss1@snd\ ss2) \in seq\ ars$ **and** $(lst\ (fst\ ss1, snd\ ss1@snd\ ss2) =$

$lst\ ss2$)
proof –
show $(fst\ ss1, snd\ ss1 @ snd\ ss2) \in seq\ ars$ **using** *seq-concat-helper* **assms** **by** *force*
next
show $(lst\ (fst\ ss1, snd\ ss1 @ snd\ ss2) = lst\ ss2)$
using *assms surjective-pairing seq-concat-helper* **by** *metis*
qed

diagrams

definition $diagram :: ('a, 'b)\ lars \Rightarrow ('a, 'b)\ seq \times ('a, 'b)\ seq \times ('a, 'b)\ seq \times ('a, 'b)\ seq \Rightarrow bool$
where $diagram\ ars\ d = (let\ (\tau, \sigma, \sigma', \tau') = d\ in\ \{\sigma, \tau, \sigma', \tau'\} \subseteq seq\ ars \wedge$
 $fst\ \sigma = fst\ \tau \wedge lst\ \sigma = fst\ \tau' \wedge lst\ \tau = fst\ \sigma' \wedge lst\ \sigma' = lst\ \tau')$

definition $labels :: ('a, 'b)\ seq \Rightarrow 'b\ list$
where $labels\ ss = map\ fst\ (snd\ ss)$

definition $D2 :: 'b\ rel \Rightarrow ('a, 'b)\ seq \times ('a, 'b)\ seq \times ('a, 'b)\ seq \times ('a, 'b)\ seq \Rightarrow bool$
where $D2\ r\ d = (let\ (\tau, \sigma, \sigma', \tau') = d\ in\ D\ r\ (labels\ \tau)\ (labels\ \sigma)\ (labels\ \sigma')\ (labels\ \tau'))$

lemma *lemma3-5-d*: **assumes** $diagram\ ars\ (\tau, \sigma, \sigma', \tau')$ **and** $diagram\ ars\ (v, \sigma', \sigma'', v')$
shows $diagram\ ars\ ((fst\ \tau, snd\ \tau @ snd\ v), \sigma, \sigma'', (fst\ \tau'), snd\ \tau' @ snd\ v')$ **proof** –
from *assms* **have** $\tau \in seq\ ars$ **and** $upsilon: v \in seq\ ars$ **and** $o: lst\ \tau = fst\ v$
and $\tau': \tau' \in seq\ ars$ **and** $upsilon': v' \in seq\ ars$ **and** $l: lst\ \tau' = fst\ v'$
unfolding *diagram-def* **by** *auto*
show *?thesis* **using** *assms seq-concat[OF tau' upsilon' l] seq-concat[OF tau upsilon o]* **unfolding** *diagram-def* **by** *auto*
qed

lemma *lemma3-5-d-v*: **assumes** $diagram\ ars\ (\tau, \sigma, \sigma', \tau')$ **and** $diagram\ ars\ (\tau', v, v', \tau'')$
shows $diagram\ ars\ (\tau, (fst\ \sigma, snd\ \sigma @ snd\ v), (fst\ \sigma', snd\ \sigma' @ snd\ v'), \tau'')$ **proof** –
from *assms* **have** $d1: diagram\ ars\ (\sigma, \tau, \tau', \sigma')$ **and** $d2: diagram\ ars\ (v, \tau', \tau'', v')$
unfolding *diagram-def* **by** *auto*
show *?thesis* **using** *lemma3-5-d[OF d1 d2]* **unfolding** *diagram-def* **by** *auto*
qed

lemma *lemma3-5'*: **assumes** $trans\ r$ **and** $irrefl\ r$ **and** $D2\ r\ (\tau, \sigma, \sigma', \tau')$ **and** $D2\ r\ (v, \sigma', \sigma'', v')$
shows $D2\ r\ ((fst\ \tau, snd\ \tau @ snd\ v), \sigma, \sigma'', (fst\ \tau'), snd\ \tau' @ snd\ v')$
using *assms lemma3-5'[OF assms(1,2)]* **unfolding** *labels-def D2-def* **by** *auto*

lemma *lemma3-5'-v*: **assumes** $trans\ r$ **and** $irrefl\ r$ **and** $D2\ r\ (\tau, \sigma, \sigma', \tau')$ **and** $D2\ r\ (\tau', v, v', \tau'')$
shows $D2\ r\ (\tau, (fst\ \sigma, snd\ \sigma @ snd\ v), (fst\ \sigma', snd\ \sigma' @ snd\ v'), \tau'')$ **proof** –
from *assms(3,4)* **have** $D1: D2\ r\ (\sigma, \tau, \tau', \sigma')$ **and** $D2: D2\ r\ (v, \tau', \tau'', v')$
unfolding *D2-def* **using** *mirror-D[OF assms(1,2)]* **by** *auto*
show *?thesis* **using** *lemma3-5'[OF assms(1,2) D1 D2]* **mirror-D**[*OF assms(1,2)*]

unfolding *D2-def* **by** *auto*
qed

lemma *trivial-diagram*: **assumes** $\sigma \in \text{seq ars}$ **shows** *diagram ars* $(\sigma, (\text{fst } \sigma, []), (\text{lst } \sigma, []), \sigma)$
using *assms seq.intros(1)* **unfolding** *diagram-def Let-def lst-def* **by** *auto*

lemma *trivial-D2*: **assumes** $\sigma \in \text{seq ars}$ **shows** *D2 r* $(\sigma, (\text{fst } \sigma, []), (\text{lst } \sigma, []), \sigma)$
using *assms* **unfolding** *D2-def D-def labels-def* **using** *mul-eq-reflexive* **by** *auto*

definition *DD* :: $('a, 'b) \text{ lars} \Rightarrow 'b \text{ rel} \Rightarrow ('a, 'b) \text{ seq} \times ('a, 'b) \text{ seq} \times ('a, 'b) \text{ seq} \times ('a, 'b) \text{ seq} \Rightarrow \text{bool}$
where *DD ars r d* = $(\text{diagram ars } d \wedge \text{D2 } r \text{ } d)$

lemma *lemma3-5-DD*: **assumes** *trans r* **and** *irrefl r* **and** *DD ars r* $(\tau, \sigma, \sigma', \tau')$
and *DD ars r* $(v, \sigma', \sigma'', v')$
shows *DD ars r* $((\text{fst } \tau, \text{snd } \tau @ \text{snd } v), \sigma, \sigma'', (\text{fst } \tau'), \text{snd } \tau' @ \text{snd } v')$
using *assms lemma3-5-d lemma3-5'[OF assms(1,2)]* **unfolding** *DD-def* **by** *fast*

lemma *lemma3-5-DD-v*: **assumes** *trans r* **and** *irrefl r* **and** *DD ars r* $(\tau, \sigma, \sigma', \tau')$
and *DD ars r* (τ', v, v', τ'')
shows *DD ars r* $(\tau, (\text{fst } \sigma, \text{snd } \sigma @ \text{snd } v), (\text{fst } \sigma', \text{snd } \sigma' @ \text{snd } v'), \tau'')$
using *assms lemma3-5-d-v lemma3-5'-v* **unfolding** *DD-def* **by** *fast*

lemma *trivial-DD*: **assumes** $\sigma \in \text{seq ars}$ **shows** *DD ars r* $(\sigma, (\text{fst } \sigma, []), (\text{lst } \sigma, []), \sigma)$
using *assms trivial-diagram trivial-D2* **unfolding** *DD-def* **by** *fast*

lemma *mirror-DD*: **assumes** *trans r* **and** *irrefl r* **and** *DD ars r* $(\tau, \sigma, \sigma', \tau')$ **shows**
DD ars r $(\sigma, \tau, \tau', \sigma')$
using *assms mirror-D* **unfolding** *DD-def D2-def diagram-def* **by** *auto*

well-foundedness of *rel r*

definition *measure* :: $'b \text{ rel} \Rightarrow ('a, 'b) \text{ seq} \times ('a, 'b) \text{ seq} \Rightarrow 'b \text{ multiset}$
where *measure r P* = $r | \text{labels } (\text{fst } P) | + r | \text{labels } (\text{snd } P) |$

definition *pex* :: $'b \text{ rel} \Rightarrow (('a, 'b) \text{ seq} \times ('a, 'b) \text{ seq}) \text{ rel}$
where *pex r* = $\{(P1, P2). (\text{measure } r \text{ } P1, \text{measure } r \text{ } P2) \in \text{mul } r\}$

lemma *wfi*: **assumes** *retr = pex r* **and** $\neg \text{wf } (\text{retr})$ **shows** $\neg \text{wf } (\text{mul } r)$ **proof** –
have $\neg \text{SN } ((\text{retr})^{-1})$ **using** *assms* **unfolding** *SN-iff-wf converse-converse* **by**
auto

from this obtain *s* **where** $\forall i. (s \text{ } i, s \text{ } (\text{Suc } i)) \in \text{retr}^{-1}$ **unfolding** *SN-def*
SN-on-def **by** *auto*

hence fact: $\forall i. (\text{measure } r \text{ } (s \text{ } i), \text{measure } r \text{ } (s \text{ } (\text{Suc } i))) \in (\text{mul } r)^{-1}$ **unfolding**
assms(1) pex-def **by** *auto*

have $\neg \text{SN } ((\text{mul } r)^{-1})$ **using** *chain-imp-not-SN-on[OF fact]* **unfolding** *SN-on-def*
by *auto*

thus *?thesis* **unfolding** *SN-iff-wf converse-converse* **by** *auto*

qed

lemma *wf*: **assumes** *trans r* **and** *wf r* **shows** *wf (pex r)* **using** *wf-mul[OF assms]*
wfi **by** *auto*

main result

definition *peak* :: ('a,'b) *lars* \Rightarrow ('a,'b) *seq* \times ('a,'b) *seq* \Rightarrow *bool*
where *peak ars p* = (let (τ, σ) = *p* in $\{\tau, \sigma\} \subseteq \text{seq ars} \wedge \text{fst } \tau = \text{fst } \sigma$)

definition *local-peak* :: ('a,'b) *lars* \Rightarrow ('a,'b) *seq* \times ('a,'b) *seq* \Rightarrow *bool*
where *local-peak ars p* = (let (τ, σ) = *p* in *peak ars p* \wedge *length (snd } \tau) = 1* \wedge
length (snd } \sigma) = 1)

proof of Theorem 3.7

lemma *LD-imp-D*: **assumes** *trans r* **and** *wf r* **and** $\forall P. (\text{local-peak ars } P \longrightarrow (\exists$
 $\sigma' \tau'. \text{DD ars } r (\text{fst } P, \text{snd } P, \sigma', \tau')))$
and *peak ars P* **shows** $(\exists \sigma' \tau'. \text{DD ars } r (\text{fst } P, \text{snd } P, \sigma', \tau'))$ **proof** –
have *i*: *irrefl r* **using** *assms(1,2)* *acyclic-irrefl trancl-id wf-acyclic* **by** *metis*
have *wf*: *wf (pex r)* **using** *wf[OF assms(1,2)]* .
show *?thesis* **using** *assms(4)* **proof** (*induct rule:wf-induct-rule[OF wf]*)
case (*1 P*)
obtain *s } \tau } \sigma* **where** *decompose:P = (} \tau, } \sigma)* **and** *tau:} \tau \in \text{seq ars}* **and** *sigma:} \sigma \in*
seq ars
and *tau-s: fst } \tau = s* **and** *sigma-s: fst } \sigma = s* **using** *1* **unfolding** *peak-def* **by**
auto
show *?case* **proof** (*cases snd } \tau*)
case *Nil* **from** *mirror-DD[OF assms(1) i trivial-DD[OF sigma]]*
show *?thesis* **using** *tau-s sigma-s Nil surjective-pairing unfolding decompose*
fst-conv snd-conv DD-def **by** *metis*
next
case (*Cons } \beta*-*step v-step*)
hence *tau-dec: } \tau = (s, [} \beta*-*step]@v-step)* **apply** *auto* **using** *tau-s surjective-pairing*
by *metis*
hence *tau2: (s, [} \beta*-*step]@v-step) \in \text{seq ars}* **using** *tau* **by** *auto*
show *?thesis* **proof** (*cases snd } \sigma*)
case *Nil* **from** *trivial-DD[OF tau]*
show *?thesis* **using** *tau-s sigma-s Nil surjective-pairing unfolding decompose*
fst-conv snd-conv DD-def **by** *metis*
next
case (*Cons } \alpha*-*step } \rho*-*step*)
hence *sigma-dec: } \sigma = (s, [} \alpha*-*step]@} \rho*-*step)* **apply** *auto* **using** *sigma-s surjec-*
tive-pairing **by** *metis*
hence *sigma2:(s, [} \alpha*-*step]@} \rho*-*step) \in \text{seq ars}* **using** *sigma* **by** *auto*

have *alpha:(s, [} \alpha*-*step]) \in \text{seq ars}* (**is** *?} \alpha \in -*)
and *rho: (lst (s, [} \alpha*-*step]), } \rho*-*step) \in \text{seq ars}* (**is** *?} \rho \in -*) **using** *seq-chop[OF*
sigma2] **by** *auto*
have *beta:(s, [} \beta*-*step]) \in \text{seq ars}* (**is** *?} \beta \in -*)
and *upsilon: (lst (s, [} \beta*-*step]), v-step) \in \text{seq ars}* (**is** *?v \in -*) **using** *seq-chop[OF*

$\tau_2]$ **by auto**
have *local-peak ars* (β, α) **using** *alpha beta unfolding local-peak-def peak-def*
by auto
from this obtain $\kappa \mu$ **where** $D:DD$ *ars* r $(\beta, \alpha, \kappa, \mu)$ **using** *assms(3)* **apply**
auto by metis
hence κ : $\kappa \in seq$ *ars* **and** μ : $\mu \in seq$ *ars* **unfolding** *DD-def diagram-def* **by**
auto
have $P-IH1$: *peak ars* (v, κ) **using** *upsilon kappa D unfolding DD-def dia-*
gram-def peak-def **by auto**
have β -ne: *labels* $\beta \neq []$ **unfolding** *labels-def* **by auto**
have dec : D r (*labels* β) (*labels* α) (*labels* κ) (*labels* μ) **using** D **unfolding**
DD-def D2-def **by auto**
have $x1$: $((v, \kappa), (\tau, \alpha)) \in pex$ r **using** *lemma3-6[OF assms(1) beta-ne dec]*
unfolding *pex-def measure-def decompose labels-def tau-dec* **by** (*simp add:*
add.commute)
have (*lexmax* r (*labels* τ) + *lexmax* r (*labels* α)), *lexmax* r (*labels* τ) + *lexmax*
 r (*labels* σ) \in *mul-eq* r (**is** $(l, r) \in -$)
unfolding *sigma-dec labels-def snd-conv list.map lexmax.simps diff-from-empty*
using *assms(1)* **by** (*simp add: lemma2-6-2-a*)
hence $((v, \kappa), P) \in pex$ r **using** $x1$ **unfolding** *sigma-s pex-def measure-def*
decompose **using** *mul-and-mul-eq-imp-mul[OF assms(1)]* **by auto**
from this obtain $\kappa' v'$ **where** $IH1$: DD *ars* r (v, κ, κ', v') **using** $1(1)[OF -$
 $P-IH1]$ **unfolding** *decompose* **by auto**
hence κ' : $\kappa' \in seq$ *ars* **and** v' : $v' \in seq$ *ars* **using** D **unfolding** *DD-def*
diagram-def **by auto**
have τ' : $(fst \mu, snd \mu @ (snd v')) \in seq$ *ars* (**is** $\tau' \in -$) **using** *seq-concat(1)[OF*
mu upsilon] D $IH1$ **unfolding** *DD-def diagram-def* **by auto**
have $DIH1$: DD *ars* r $(\tau, \alpha, \kappa', \tau')$ **using** *lemma3-5-DD[OF assms(1) i D IH1]*
tau-dec **by auto**
hence dec - $dih1$: D r (*labels* τ) (*labels* α) (*labels* κ') (*labels* τ') **using** $DIH1$
unfolding *DD-def D2-def* **by simp**

have $P-IH2$: *peak ars* (τ', ρ) **using** τ' *rho D unfolding DD-def diagram-def*
peak-def **by auto**
have α -ne: *labels* $\alpha \neq []$ **unfolding** *labels-def* **by simp**
have $((\tau', \rho), P) \in pex$ r **using** *lemma3-6-v[OF assms(1) i alpha-ne dec-dih1]*
unfolding *pex-def measure-def decompose labels-def sigma-dec* **by auto**
from this obtain $\rho' \tau''$ **where** $IH2$: DD *ars* r $(\tau', \rho, \rho', \tau'')$ **using** $1(1)[OF -$
 $P-IH2]$ **by auto**
show $?thesis$ **using** *lemma3-5-DD-v[OF assms(1) i DIH1 IH2]* **unfolding**
decompose fst-conv snd-conv sigma-dec **by fast**
qed
qed
qed
qed

CR with unlabeled

definition *unlabel* :: (a, b) *lars* \Rightarrow a *rel*
where *unlabel ars* = $\{(a, c). \exists b. (a, b, c) \in ars\}$

lemma *step-imp-seq*: **assumes** $(a,b) \in (\text{unlabel ars})$
shows $\exists ss \in \text{seq ars}. \text{fst } ss = a \wedge \text{lst } ss = b$ **proof** –
obtain α **where** $\text{step}:(a,\alpha,b) \in \text{ars}$ **using** *assms* **unfolding** *unlabel-def* **by** *auto*
hence $ss: (a,[(\alpha,b)]) \in \text{seq ars}$ (**is** $?ss \in -$) **using** *seq.intros* **by** *fast*
have $\text{fst } ?ss = a$ **and** $\text{lst } ?ss = b$ **unfolding** *lst-def* **by** *auto*
thus *?thesis* **using** *ss* **unfolding** *lst-def* **by** *fast*
qed

lemma *steps-imp-seq*: **assumes** $(a,b) \in (\text{unlabel ars})^{\wedge*}$
shows $\exists ss \in \text{seq ars}. \text{fst } ss = a \wedge \text{lst } ss = b$ **using** *assms(1)* **proof**
fix n **assume** $A: (a,b) \in (\text{unlabel ars})^{\wedge n}$ **thus** *?thesis* **proof** (*induct n arbitrary: a b ars*)
case 0 **hence** $eq: a = b$ **by** *auto*
have $(a,[]) \in \text{seq ars}$ **using** *seq.intros(1)* **by** *fast*
thus *?case* **using** *fst-eqD* *snd-conv* *lst-def* *eq* **by** *metis*
next
case $(\text{Suc } m)$
obtain c **where** $\text{steps}: (a,c) \in (\text{unlabel ars})^{\wedge m}$ **and** $\text{step}: (c,b) \in (\text{unlabel ars})$
using *Suc* **by** *auto*
obtain $ss \ ts$ **where** $ss1: ss \in \text{seq ars}$ **and** $ss2: \text{fst } ss = a$
and $ts1: ts \in \text{seq ars}$ **and** $ts3: \text{lst } ts = b$ **and** $eq: \text{lst } ss = \text{fst } ts$
using *Suc steps step-imp-seq[OF step]* **by** *metis*
show *?case* **using** *seq-concat[OF ss1 ts1 eq]* **unfolding** $ss2 \ ts3$ **by** *force*
qed
qed

lemma *step-imp-unlabeled-step*: **assumes** $(a,b,c) \in \text{ars}$ **shows** $(a,c) \in (\text{unlabel ars})$
using *assms* **unfolding** *unlabel-def* **by** *auto*

lemma *seq-imp-steps*:
assumes $ss \in \text{seq ars}$ **and** $\text{fst } ss = a$ **and** $\text{lst } ss = b$ **shows** $(a,b) \in (\text{unlabel ars})^{\wedge*}$
proof –
from *assms surjective-pairing* **obtain** ls **where** $(a,ls) \in \text{seq (ars)}$ **and** $\text{lst } (a,ls) = b$ **by** *metis*
thus *?thesis* **proof** (*induct ls arbitrary: a b rule:list.induct*)
case *Nil* **thus** *?case* **unfolding** *lst-def* **by** *auto*
next
case $(\text{Cons } x \ xs)$
have $\text{fst}:(a,\text{fst } x,\text{snd } x) \in \text{ars}$ **using** *Cons seq-tail1(2)* *surjective-pairing* **by** *metis*
have $(\text{snd } x,b) \in (\text{unlabel ars})^{\wedge*}$ **using** *Cons seq-tail1(1,3)* **by** *metis*
thus *?case* **using** *step-imp-unlabeled-step[OF fst]* **by** *auto*
qed
qed

lemma *seq-vs-steps*: **shows** $(a,b) \in (\text{unlabel ars})^{\wedge*} = (\exists ss. \text{fst } ss = a \wedge \text{lst } ss = b \wedge ss \in \text{seq ars})$
using *seq-imp-steps steps-imp-seq* **by** *metis*

lemma *D-imp-CR*: **assumes** $\forall P. (\text{peak ars } P \longrightarrow (\exists \sigma' \tau'. \text{DD ars } r (\text{fst } P, \text{snd } P, \sigma', \tau')))$ **shows** *CR* (*unlabel ars*) **proof**
fix *a b c* **assume** $A: (a, b) \in (\text{unlabel ars})^{\wedge*}$ **and** $B: (a, c) \in (\text{unlabel ars})^{\wedge*}$ **show**
 $(b, c) \in (\text{unlabel ars})^{\downarrow}$ **proof** –
obtain *ss1 ss2* **where** *peak ars* (*ss1, ss2*) **and** $b: \text{lst } ss1 = b$ **and** $c: \text{lst } ss2 = c$
unfolding *peak-def* **using** *A B* **unfolding** *seq-vs-steps* **by** *auto*
from *this* **obtain** *ss3 ss4* **where** *dia: diagram ars* (*ss1, ss2, ss3, ss4*) **using**
assms(1) **unfolding** *DD-def* **apply** *auto* **using** *surjective-pairing* **by** *metis*
from *dia* **obtain** *d* **where** $ss3: ss3 \in \text{seq ars}$ **and** $ss4: ss4 \in \text{seq ars}$
and $ss3-1: \text{fst } ss3 = b$ **and** $ss3-2: \text{lst } ss3 = d$ **and** $ss4-1: \text{fst } ss4 = c$ **and**
 $ss4-2: \text{lst } ss4 = d$
using *b c* **unfolding** *diagram-def* **by** *auto*
show *?thesis* **using** *seq-imp-steps[OF ss3 ss3-1 ss3-2]* *seq-imp-steps[OF ss4 ss4-1 ss4-2]* **by** *auto*
qed
qed

definition *LD* :: $'b \text{ set} \Rightarrow 'a \text{ rel} \Rightarrow \text{bool}$
where $LD \ L \ \text{ars} = (\exists (r::('b \ \text{rel})) (\text{lrs}::('a, 'b) \ \text{lars}). (\text{ars} = \text{unlabel lrs}) \wedge \text{trans } r \wedge \text{wf } r \wedge (\forall P. (\text{local-peak lrs } P \longrightarrow (\exists \sigma' \tau'. (\text{DD lrs } r (\text{fst } P, \text{snd } P, \sigma', \tau'))))))$

lemma *sound*: **assumes** *LD L ars* **shows** *CR ars*
using *assms LD-imp-D D-imp-CR* **unfolding** *LD-def* **by** *metis*

1.1.5 Application: Newman's Lemma

lemma *measure*:
assumes *lab-eq: lrs = {(a, c, b). c = a \wedge (a, b) \in ars}* **and** $(s, (\alpha, t) \# ss) \in \text{seq lrs}$
shows $\text{set} (\text{labels } (t, ss)) \subseteq \text{ds } ((\text{ars}^{\wedge+})^{-1}) \{\alpha\}$ **using** *assms(2)* **proof** (*induct ss*
arbitrary: s α t)
case *Nil* **thus** *?case* **unfolding** *labels-def* **by** *auto*
next
case (*Cons x xs*)
from *this* **obtain** $\beta \ u$ **where** $x: x = (\beta, u)$ **using** *surjective-pairing* **by** *metis*
have $t: \text{trans } ((\text{ars}^{\wedge+})^{-1})$ **by** (*metis trans-converse trans-trancl*)
from *Cons(1) x* **have** $s0: (s, \alpha, t) \in \text{lrs}$ **and** $cs: (t, (\beta, u) \# xs) \in \text{seq lrs}$ **using**
Cons.prem1 seq-tail1(1) snd-conv fst-conv seq-tail1(2) **by** *auto*
have $ih: \text{set} (\text{labels } (u, xs)) \subseteq \text{ds } ((\text{ars}^{\wedge+})^{-1}) \{\beta\}$ **using** *Cons(1)[OF cs]* **by** *auto*
have $key: \{\beta\} \subseteq \text{ds } ((\text{ars}^{\wedge+})^{-1}) \{\alpha\}$ **using** $s0 \ cs \ \text{seq-tail1}(2)[\text{OF } cs]$ **unfolding**
ds-def lab-eq **by** *auto*
show *?case* **using** *ih subset-imp-ds-subset[OF t key]* *key* **unfolding** *x labels-def*
by *auto*
qed

lemma *newman*: **assumes** *WCR ars* **and** *SN ars* **shows** *CR ars* **proof** –
from *assms* **obtain** *L* **where** $L = \{a . \exists b. (a, b) \in \text{ars}\}$ **by** *auto*
from *assms* **obtain** *lrs* **where** *lab-eq: (lrs = {(a, c, b). c = a \wedge (a, b) \in ars})* **by**
auto

have $lab: ars = \text{unlabel lrs unfolding unlabel-def lab-eq by auto}$
have $t: \text{trans } ((ars \hat{+})^{-1}) \text{ using trans-converse trans-trancl by auto}$
have $w: \text{wf } ((ars \hat{+})^{-1}) \text{ using assms(2) wf-trancl trancl-converse unfolding}$
SN-iff-wf by metis
have $ps: \forall P. (\text{local-peak lrs } P \dashrightarrow (\exists \sigma' \tau'. DD \text{ lrs } ((ars \hat{+})^{-1}) (\text{fst } P, \text{snd } P, \sigma', \tau')))$ **proof**
fix P **show** $\text{local-peak lrs } P \dashrightarrow (\exists \sigma' \tau'. DD \text{ lrs } ((ars \hat{+})^{-1}) (\text{fst } P, \text{snd } P, \sigma', \tau'))$
proof
assume $A: \text{local-peak lrs } P$ **show** $(\exists \sigma' \tau'. DD \text{ lrs } ((ars \hat{+})^{-1}) (\text{fst } P, \text{snd } P, \sigma', \tau'))$ **(is ?DD) proof** $-$
from $lab\text{-eq}$ **have** $lab: ars = \text{unlabel lrs unfolding unlabel-def by auto}$
from A **obtain** $\tau \sigma$ **where** $ts: \{\tau, \sigma\} \subseteq \text{seq lrs}$ **and** $l1: \text{length } (\text{snd } \tau) = 1$ **and**
 $l2: \text{length } (\text{snd } \sigma) = 1$ **and** $P: P = (\tau, \sigma)$
and $p: \text{fst } \tau = \text{fst } \sigma$ **unfolding local-peak-def peak-def by auto**

from $l1$ **obtain** βb **where** $1: \text{snd } \tau = [(\beta, b)]$ **by** $(\text{auto simp add: length-Suc-conv})$
from $this$ **obtain** a **where** $\text{tau}: \tau = (a, [(\beta, b)])$ **by** $(\text{metis surjective-pairing})$
hence $alb: (a, \beta, b) \in \text{lrs}$ **using** ts **by** $(\text{metis fst-conv insert-subset seq-tail1(2) snd-conv})$
have $ab: (a, b) \in ars$ **and** $a\text{-eq}: a = \beta$ **using** alb **unfolding lab-eq by auto**

from $l2$ **obtain** αc **where** $2: \text{snd } \sigma = [(\alpha, c)]$ **by** $(\text{auto simp add: length-Suc-conv})$
hence $\text{sigma}: \sigma = (a, [(\alpha, c)])$ **using** ts **by** $(\text{metis fst-conv p prod.collapse tau})$
hence $alc: (a, \alpha, c) \in \text{lrs}$ **using** ts **by** $(\text{metis fst-conv insert-subset seq-tail1(2) snd-conv})$
hence $ac: (a, c) \in ars$ **and** $a\text{-eq}: a = \alpha$ **using** alb **unfolding lab-eq by auto**

from tau sigma **have** $fl: \text{fst } \tau = a \wedge \text{fst } \sigma = a \wedge \text{lst } \tau = b \wedge \text{lst } \sigma = c$ **unfolding**
 lst-def by auto
from $ab ac$ **obtain** d **where** $(b, d) \in ars \hat{*}$ **and** $(c, d) \in ars \hat{*}$ **using** assms(1)
by auto
from $this$ **obtain** $\sigma' \tau'$ **where** $\text{sigma}': \sigma' \in \text{seq lrs}$ **and** $\text{sigma}'1: \text{fst } \sigma' = b$
and $\text{lst } \sigma' = d$
and $\text{tau}': \tau' \in \text{seq lrs}$ **and** $\text{fst } \tau' = c$ **and** $\text{lst } \tau' = d$ **using**
 $\text{steps-imp-seq unfolding lab by metis}$
hence $d: \text{diagram lrs } (\text{fst } P, \text{snd } P, \sigma', \tau')$ **using** $P A ts fl$ **unfolding**
 $\text{local-peak-def peak-def diagram-def by auto}$

have $s1: (a, (\beta, b) \# \text{snd } \sigma') \in \text{seq lrs}$ **using** $\langle \text{fst } \sigma' = b \rangle \text{seq.intros(2)[OF alb]}$
 sigma' **by auto**
have $vv: \text{set } (\text{labels } \sigma') \subseteq ds ((ars \hat{+})^{-1}) \{\beta\}$ **using** $\text{measure[OF lab-eq s1]}$
by $(\text{metis } \langle \text{fst } \sigma' = b \rangle \text{surjective-pairing})$
have $s2: (a, (\alpha, c) \# \text{snd } \tau') \in \text{seq lrs}$ **using** $\langle \text{fst } \tau' = c \rangle \text{seq.intros(2)[OF alc]}$
 tau' **by auto**
hence $ww: \text{set } (\text{labels } \tau') \subseteq ds ((ars \hat{+})^{-1}) \{\alpha\}$ **using** $\text{measure[OF lab-eq] s2}$
by $(\text{metis } \langle \text{fst } \tau' = c \rangle \text{surjective-pairing})$
from w **have** $i: \text{irrefl } ((ars \hat{+})^{-1})$ **by** $(\text{metis SN-imp-acyclic acyclic-converse acyclic-irrefl assms(2) trancl-converse})$

from $vv\ ww$ **have** $ld: LD' ((ars^{\wedge+})^{-1}) \beta \alpha (labels\ \sigma') [] [] (labels\ \tau') [] []$
unfolding $LD'-def\ LD-1'-def$ **by** $auto$
have $D: D ((ars^{\wedge+})^{-1}) (labels\ (fst\ P)) (labels\ (snd\ P)) (labels\ \sigma') (labels\ \tau')$
using $proposition3-4[OF\ t\ i\ ld]$ **unfolding** $P\ sigma\ tau\ lst-def\ labels-def$ **by** $auto$

from $d\ D$ **have** $DD\ lrs\ ((ars^{\wedge+})^{-1}) (fst\ P, snd\ P, \sigma', \tau')$ **unfolding** $DD-def\ D2-def$ **by** $auto$
thus $?thesis$ **by** $fast$
qed
qed
qed
have $LD\ L\ ars$ **using** $lab\ t\ w\ ps$ **unfolding** $LD-def$ **by** $fast$
thus $?thesis$ **using** $sound$ **by** $auto$
qed

1.2 Conversion Version

This section follows [2].

auxiliary results on multisets

lemma $mul-eq-add-right: (M, M+P) \in mul-eq\ r$ **proof** –
have $M = M + \{\#\}$ $set-mset\ \{\#\} \subseteq dm\ r\ P$ **by** $auto$
thus $?thesis$ **unfolding** $mul-eq-def$ **by** $fast$
qed

lemma $mul-add-right: assumes\ (M, N) \in mul\ r$ **shows** $(M, N+P) \in mul\ r$ **proof** –
from $assms$ **obtain** $I\ J\ K$ **where** $M = I + K\ N = I + J$ $set-mset\ K \subseteq dm\ r\ J$
 $J \neq \{\#\}$ **unfolding** $mul-def$ **by** $auto$
hence $b: M = I + K\ N + P = I + (J + P)$ $set-mset\ K \subseteq ds\ r\ (set-mset\ J \cup set-mset\ P)$
 $J+P \neq \{\#\}$ **unfolding** $dm-def\ lemma2-6-1-set$ **using** $union-assoc$ **by** $auto$
hence $set-mset\ K \subseteq ds\ r\ (set-mset\ (J+P))$ **by** $auto$
thus $?thesis$ **using** b **unfolding** $mul-def$ **unfolding** $dm-def$ **by** $fast$
qed

lemma $mul-eq-and-ds-imp-ds:$
assumes $t: trans\ r$ **and** $(M, N) \in mul-eq\ r$ **and** $set-mset\ N \subseteq ds\ r\ S$
shows $set-mset\ M \subseteq ds\ r\ S$ **proof** –
from $assms$ **obtain** $I\ J\ K$ **where** $a: M = I + K$ **and** $N = I + J$ **and** $c: set-mset\ K \subseteq dm\ r\ J$
unfolding $mul-eq-def$ **by** $auto$
hence $k1: set-mset\ I \subseteq ds\ r\ S$ $set-mset\ J \subseteq ds\ r\ S$ **using** $assms$ **by** $auto$
hence $ds\ r\ (set-mset\ J) \subseteq ds\ r\ S$ **using** $subset-imp-ds-subset[OF\ t]$ **by** $auto$
thus $?thesis$ **using** $k1\ a\ c$ **unfolding** $dm-def$ **by** $auto$
qed

lemma $lemma2-6-2-set: assumes\ S \subseteq T$ **shows** $ds\ r\ S \subseteq ds\ r\ T$ **using** $assms$
unfolding $ds-def$ **by** $auto$

lemma *leq-imp-subseteq*: **assumes** $M \subseteq\# N$ **shows** *set-mset* $M \subseteq \text{set-mset } N$
using *assms mset-subset-eqD* **by** *auto*

lemma *mul-add-mul-eq-imp-mul*: **assumes** $(M,N) \in \text{mul } r$ **and** $(P,Q) \in \text{mul-eq } r$ **shows** $(M+P,N+Q) \in \text{mul } r$ **proof** –
from *assms(1)* **obtain** $I J K$ **where** $a:M = I + K$ $N = I + J$ *set-mset* $K \subseteq \text{dm } r$
 $J J \neq \{\#\}$ **unfolding** *mul-def* **by** *auto*
from *assms(2)* **obtain** $I2 J2 K2$ **where** $b:P = I2 + K2$ $Q = I2 + J2$ *set-mset* $K2 \subseteq \text{dm } r$
 $J2$ **unfolding** *mul-eq-def* **by** *auto*
have $M + P = (I + I2) + (K + K2)$ **using** *a b union-commute union-assoc* **by** *metis*
moreover **have** $N + Q = (I + I2) + (J + J2)$ **using** *a b union-commute union-assoc* **by** *metis*
moreover **have** *set-mset* $(K + K2) \subseteq \text{dm } r$ $(J + J2)$ **using** *a b unfolding lemma2-6-1-multiset* **by** *auto*
ultimately show *?thesis* **using** *a b unfolding mul-def* **by** *fast*
qed

labeled conversion

type-synonym $('a,'b)$ *conv* = $('a \times ((\text{bool} \times 'b \times 'a) \text{ list}))$

inductive-set *conv* :: $('a,'b)$ *lars* $\Rightarrow ('a,'b)$ *conv set for ars*

where $(a,[]) \in \text{conv ars}$

| $(a,\alpha,b) \in \text{ars} \Rightarrow (b,ss) \in \text{conv ars} \Rightarrow (a,(\text{True},\alpha,b) \# ss) \in \text{conv ars}$

| $(b,\alpha,a) \in \text{ars} \Rightarrow (b,ss) \in \text{conv ars} \Rightarrow (a,(\text{False},\alpha,b) \# ss) \in \text{conv ars}$

definition *labels-conv* :: $('a,'b)$ *conv* $\Rightarrow 'b$ *list*

where *labels-conv* $c = \text{map } (\lambda q. (\text{fst } (\text{snd } q))) (\text{snd } c)$

definition *measure-conv* :: $'b$ *rel* $\Rightarrow ('a,'b)$ *conv* $\Rightarrow 'b$ *multiset*

where *measure-conv* $r c = \text{lexmax } r (\text{labels-conv } c)$

fun *lst-conv* :: $('a,'b)$ *conv* $\Rightarrow 'a$

where *lst-conv* $(s,[]) = s$

| *lst-conv* $(s,(d,\alpha,t) \# ss) = \text{lst-conv } (t,ss)$

definition *local-diagram1* :: $('a,'b)$ *lars* $\Rightarrow ('a,'b)$ *seq* $\Rightarrow ('a,'b)$ *seq* $\Rightarrow ('a,'b)$ *seq*
 $\Rightarrow ('a,'b)$ *seq* $\Rightarrow ('a,'b)$ *seq* $\Rightarrow \text{bool}$

where *local-diagram1* $\text{ars } \beta \alpha \sigma 1 \sigma 2 \sigma 3 =$

$(\text{local-peak } \text{ars } (\beta,\alpha) \wedge \{\sigma 1,\sigma 2,\sigma 3\} \subseteq \text{seq ars} \wedge \text{lst } \beta = \text{fst } \sigma 1 \wedge \text{lst } \sigma 1 = \text{fst } \sigma 2$
 $\wedge \text{lst } \sigma 2 = \text{fst } \sigma 3)$

definition *LDD1* :: $('a,'b)$ *lars* $\Rightarrow 'b$ *rel* $\Rightarrow ('a,'b)$ *seq* $\Rightarrow ('a,'b)$ *seq* $\Rightarrow ('a,'b)$ *seq*
 $\Rightarrow ('a,'b)$ *seq* $\Rightarrow ('a,'b)$ *seq* $\Rightarrow \text{bool}$

where *LDD1* $\text{ars } r \beta \alpha \sigma 1 \sigma 2 \sigma 3 = (\text{local-diagram1 } \text{ars } \beta \alpha \sigma 1 \sigma 2 \sigma 3 \wedge$

$\text{LD-1}' r (\text{hd } (\text{labels } \beta)) (\text{hd } (\text{labels } \alpha)) (\text{labels } \sigma 1) (\text{labels } \sigma 2) (\text{labels } \sigma 3))$

definition *LDD* :: $('a,'b)$ *lars* $\Rightarrow 'b$ *rel* $\Rightarrow ('a,'b)$ *seq* $\times ('a,'b)$ *seq* $\times ('a,'b)$ *seq* \times
 $(('a,'b)$ *seq* $\times ('a,'b)$ *seq* $\times ('a,'b)$ *seq* $\times ('a,'b)$ *seq* $\times ('a,'b)$ *seq* $\Rightarrow \text{bool}$

where $LDD\ ars\ r\ d = (let\ (\beta, \alpha, \sigma 1, \sigma 2, \sigma 3, \tau 1, \tau 2, \tau 3) = d\ in\ LDD1\ ars\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3 \wedge LDD1\ ars\ r\ \alpha\ \beta\ \tau 1\ \tau 2\ \tau 3 \wedge lst\ \sigma 3 = lst\ \tau 3)$

definition $local\text{-}triangle1 :: ('a, 'b)\ lars \Rightarrow ('a, 'b)\ seq \Rightarrow ('a, 'b)\ seq \Rightarrow ('a, 'b)\ conv \Rightarrow ('a, 'b)\ seq \Rightarrow ('a, 'b)\ conv \Rightarrow bool$

where $local\text{-}triangle1\ ars\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3 = (local\text{-}peak\ ars\ (\beta, \alpha) \wedge \sigma 2 \in seq\ ars \wedge \{\sigma 1, \sigma 3\} \subseteq conv\ ars \wedge lst\ \beta = fst\ \sigma 1 \wedge lst\text{-}conv\ \sigma 1 = fst\ \sigma 2 \wedge lst\ \sigma 2 = fst\ \sigma 3)$

definition $LT1 :: ('a, 'b)\ lars \Rightarrow 'b\ rel \Rightarrow ('a, 'b)\ seq \Rightarrow ('a, 'b)\ seq \Rightarrow ('a, 'b)\ conv \Rightarrow ('a, 'b)\ seq \Rightarrow ('a, 'b)\ conv \Rightarrow bool$

where $LT1\ ars\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3 = (local\text{-}triangle1\ ars\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3 \wedge LD\text{-}1'\ r\ (hd\ (labels\ \beta))\ (hd\ (labels\ \alpha))\ (labels\text{-}conv\ \sigma 1)\ (labels\ \sigma 2)\ (labels\text{-}conv\ \sigma 3))$

definition $LT :: ('a, 'b)\ lars \Rightarrow 'b\ rel \Rightarrow ('a, 'b)\ seq \times ('a, 'b)\ seq \times ('a, 'b)\ conv \times ('a, 'b)\ seq \times ('a, 'b)\ conv \times ('a, 'b)\ conv \times ('a, 'b)\ seq \times ('a, 'b)\ conv \Rightarrow bool$

where $LT\ ars\ r\ t = (let\ (\beta, \alpha, \sigma 1, \sigma 2, \sigma 3, \tau 1, \tau 2, \tau 3) = t\ in\ LT1\ ars\ r\ \beta\ \alpha\ \sigma 1\ \sigma 2\ \sigma 3 \wedge LT1\ ars\ r\ \alpha\ \beta\ \tau 1\ \tau 2\ \tau 3 \wedge lst\text{-}conv\ \sigma 3 = lst\text{-}conv\ \tau 3)$

lemma $conv\text{-}tail1$: **assumes** $conv: (s, (d, \alpha, t) \# xs) \in conv\ ars$

shows $(t, xs) \in conv\ ars$ **and** $d \Longrightarrow (s, \alpha, t) \in ars$ **and** $\neg d \Longrightarrow (t, \alpha, s) \in ars$ **and** $lst\text{-}conv\ (s, (d, \alpha, t) \# xs) = lst\text{-}conv\ (t, xs)$ **proof** –

show $(t, xs) \in conv\ ars$ **using** $assms$ **by** $(cases)\ auto$

show $d \Longrightarrow (s, \alpha, t) \in ars$ **using** $assms$ **by** $(cases)\ auto$

show $\neg d \Longrightarrow (t, \alpha, s) \in ars$ **using** $assms$ **by** $(cases)\ auto$

show $lst\text{-}conv\ (s, (d, \alpha, t) \# xs) = lst\text{-}conv\ (t, xs)$ **unfolding** $lst\text{-}conv.simps$ **by** $auto$ **qed**

lemma $conv\text{-}chop$: **assumes** $(s, ss1 @ ss2) \in conv\ ars$ **shows** $(s, ss1) \in conv\ ars$ $(lst\text{-}conv\ (s, ss1), ss2) \in conv\ ars$ **proof** –

show $(s, ss1) \in conv\ ars$ **using** $assms$ **proof** $(induct\ ss1\ arbitrary: s)$

case Nil **thus** $?case$ **using** $conv.intros$ **by** $fast$

next

case $(Cons\ t'\ ts)$ **from** $this$ **obtain** $d\ \alpha\ t$ **where** $dec: t' = (d, \alpha, t)$ **using** $prod\text{-}cases3$ **by** $metis$

from $Cons$ **have** $(s, t' \# ts @ ss2) \in conv\ ars$ **by** $auto$

hence $(t, ts @ ss2) \in conv\ ars$ **and** $d1: d \Longrightarrow (s, \alpha, t) \in ars$ **and** $d2: \neg d \Longrightarrow (t, \alpha, s) \in ars$ **using** $conv\text{-}tail1(1-3)$ **unfolding** dec **by** $auto$

hence $(t, ts) \in conv\ ars$ **using** $Cons$ **by** $auto$

thus $?case$ **unfolding** dec **using** $Cons\ conv.intros\ d1\ d2$ **by** $(cases\ d)\ auto$

qed

show $(lst\text{-}conv\ (s, ss1), ss2) \in conv\ ars$ **using** $assms$ **proof** $(induct\ ss1\ arbitrary: s)$

case Nil **thus** $?case$ **using** $conv.intros$ **unfolding** $last.simps$ **by** $auto$

next

case $(Cons\ t'\ ts)$ **from** $this$ **obtain** $d\ \alpha\ t$ **where** $dec: t' = (d, \alpha, t)$ **using** $prod\text{-}cases3$ **by** $metis$

from $Cons$ **have** $(s, t' \# ts @ ss2) \in conv\ ars$ **by** $auto$

hence $(snd\ (snd\ t'), ts @ ss2) \in conv\ ars$ **using** $conv\text{-}tail1(1)$ **unfolding** dec

by *auto*
 thus ?case using *Cons(1) unfolding dec last.simps by auto*
 qed
 qed

lemma *conv-concat-helper*:
 assumes $(s,ls) \in \text{conv ars}$ and $ss2 \in \text{conv ars}$ and $\text{lst-conv } (s,ls) = \text{fst } ss2$
 shows $(s,ls@snd\ ss2) \in \text{conv ars} \wedge (\text{lst-conv } (s,ls@snd\ ss2) = \text{lst-conv } ss2)$
 using *assms proof (induct ls arbitrary: s ss2 rule:list.induct)*
 case *Nil* thus ?case **unfolding** *lst-def* **by auto**
 next
 case $(\text{Cons } x\ xs)$ **from this obtain** $d\ \alpha\ t$ **where** *dec: x = (d,α,t)* **using** *prod-cases3*
by metis
 hence *tl: (t,xs) ∈ conv ars* and $d1:d \implies (s,\alpha,t) \in \text{ars}$ and $d2:\neg d \implies (t,\alpha,s) \in \text{ars}$ and $\text{lst:lst-conv } (t,xs) = \text{fst } ss2$
 using *conv-tail1 Cons(2) Cons(4) by auto*
 have $(t,xs@snd\ ss2) \in \text{conv ars}$ and $\text{lst:lst-conv } (t,xs@snd\ ss2) = \text{lst-conv } ss2$
 using *Cons(1)[OF tl Cons(3) lst] by auto*
 thus ?case using *conv.intros d1 d2 unfolding dec lst-conv.simps by (cases d)*
auto
 qed

lemma *conv-concat*:
 assumes $ss1 \in \text{conv ars}$ and $ss2 \in \text{conv ars}$ and $\text{lst-conv } ss1 = \text{fst } ss2$
 shows $(\text{fst } ss1, \text{snd } ss1@snd\ ss2) \in \text{conv ars}$ and $(\text{lst-conv } (\text{fst } ss1, \text{snd } ss1@snd\ ss2)) = \text{lst-conv } ss2)$
proof –
 show $(\text{fst } ss1, \text{snd } ss1@snd\ ss2) \in \text{conv ars}$ **using** *conv-concat-helper assms* **by**
force
 next
 show $(\text{lst-conv } (\text{fst } ss1, \text{snd } ss1@snd\ ss2)) = \text{lst-conv } ss2)$
 using *assms surjective-pairing conv-concat-helper* **by metis**
 qed

lemma *conv-concat-labels*:
 assumes $ss1 \in \text{conv ars}$ and $ss2 \in \text{conv ars}$ and $\text{set } (\text{labels-conv } ss1) \subseteq S$ and
 $\text{set } (\text{labels-conv } ss2) \subseteq T$
 shows $\text{set } (\text{labels-conv } (\text{fst } ss1, \text{snd } ss1@snd\ ss2)) \subseteq S \cup T$ **using** *assms unfolding*
labels-conv-def **by auto**

lemma *seq-decompose*:
 assumes $\sigma \in \text{seq ars}$ and $\text{labels } \sigma = \sigma 1'@ \sigma 2'$
 shows $\exists \sigma 1\ \sigma 2. (\{\sigma 1, \sigma 2\} \subseteq \text{seq ars} \wedge \sigma = (\text{fst } \sigma 1, \text{snd } \sigma 1@snd\ \sigma 2) \wedge \text{lst } \sigma 1 = \text{fst } \sigma 2 \wedge \text{lst } \sigma 2 = \text{lst } \sigma \wedge \text{labels } \sigma 1 = \sigma 1' \wedge \text{labels } \sigma 2 = \sigma 2')$ **proof** –
 obtain $s\ ss$ **where** $\sigma\text{-dec: } \sigma = (s,ss)$ **using** *assms(1) surjective-pairing* **by metis**
 show ?thesis **using** *assms unfolding σ-dec* **proof** *(induct ss arbitrary: s σ 1' σ 2' rule:list.induct)*
 case *Nil* thus ?case **unfolding** *labels-def lst-def* **by auto**
 next

case (*Cons* x xs)
have $step: (s, x) \in ars$ **and** $x:(snd\ x, xs) \in seq\ ars$ **using** *seq-tail1* [*OF Cons*(2)]
surjective-pairing **by** *auto*
hence $steps: (s, [x]) \in seq\ ars$ **by** (*metis Cons*(2) *append-Cons append-Nil seq-chop*(1))
from *Cons*(3) **have** $a:fst\ x\#\#labels\ (snd\ x, xs) = \sigma 1'\@ \sigma 2'$ **unfolding** *labels-def*
snd-conv **by** *auto*
show $?case\ proof\ (cases\ \sigma 1' = [])$
case *True*
from *a True* **obtain** $l\ ls$ **where** $\sigma 2'\text{-dec}: \sigma 2' = l\#\#ls$ **and** $y1:fst\ x = l$ **and**
 $y2:labels\ (snd\ x, xs) = []\@ls$ **by** *auto*
obtain $\sigma 1\ \sigma 2$ **where** $ih: \sigma 1 \in seq\ ars\ \sigma 2 \in seq\ ars\ (snd\ x, xs) = (fst\ \sigma 1, snd\ \sigma 1\ @\ snd\ \sigma 2)\ lst\ \sigma 1 = fst\ \sigma 2$
 $labels\ \sigma 1 = []\ labels\ \sigma 2 = ls$ **using** *Cons*(1) [*OF x y2*] **by** *blast*
hence $c:fst\ (snd\ x, xs) = fst\ \sigma 1$ **by** *auto*

have $1: \sigma 1 = (snd\ x, [])$ **using** *ih* **unfolding** *labels-def* **apply** *auto* **by** (*metis*
surjective-pairing)
hence $2: snd\ x = fst\ \sigma 1\ xs = snd\ \sigma 2$ **using** *ih* **by** *auto*
have $3: snd\ x = fst\ \sigma 2$ **using** *ih 1* **unfolding** *lst-def* **by** *auto*
have $\sigma 2 = (snd\ x, xs)$ **using** *x 1 2 3* *surjective-pairing* **by** *metis*
hence $l:lst\ (s, [x]) = fst\ \sigma 2$ **unfolding** *lst-def* **by** *auto*
have $m:\{(s, []), (s, x\#\#snd\ \sigma 2)\} \subseteq seq\ ars$ (**is** $\{\sigma 1, \sigma 2\} \subseteq -$) **using** *seq.intros*(1)
seq-concat(1) [*OF steps - l*] *ih* **by** *auto*
moreover $have\ (s, x\#\#xs) = (fst\ \sigma 1, snd\ \sigma 1\ @\ snd\ \sigma 2)$ **using** $m\ 2$ **by**
auto
moreover $have\ lst\ \sigma 1 = fst\ \sigma 2$ **using** m **unfolding** *lst-def* **by** *auto*
moreover $have\ lst\ \sigma 2 = lst\ (s, x\#\#xs)$ **unfolding** *lst-def* **using** $2\ 3$ **by** *auto*
moreover $have\ labels\ (s, []) = \sigma 1'$ **unfolding** *labels-def* **using** *True* **by** *auto*
moreover $have\ labels\ \sigma 2 = \sigma 2'$ **using** *ih y1* **unfolding** $\sigma 2'\text{-dec}$ *labels-def* **by**
auto
ultimately **show** $?thesis$ **by** *metis*
next
case *False* **from** *False* **obtain** $l\ ls$ **where** $\sigma 1'\text{-dec}: \sigma 1' = l\#\#ls$ **using** *list.exhaust*
by *auto*
hence $y1:fst\ x = l$ **and** $y2:labels\ (snd\ x, xs) = ls\ @\ \sigma 2'$ **using** *a* **by** *auto*
obtain $\sigma 1\ \sigma 2$ **where** $ih: \sigma 1 \in seq\ ars\ \sigma 2 \in seq\ ars\ (snd\ x, xs) = (fst\ \sigma 1, snd\ \sigma 1\ @\ snd\ \sigma 2)$
 $lst\ \sigma 1 = fst\ \sigma 2\ lst\ \sigma 2 = lst\ (snd\ x, xs)\ labels\ \sigma 1 = ls\ labels\ \sigma 2 = \sigma 2'$ **using**
Cons(1) [*OF x y2*] **by** *blast*
hence $\{(s, x\#\#snd\ \sigma 1), \sigma 2\} \subseteq seq\ ars$ (**is** $\{\sigma 1, -\} \subseteq seq\ ars$) **using** *seq-concat*(1) [*OF*
steps] *ih* **unfolding** *lst-def* **by** *auto*
moreover $have\ (s, x\#\#xs) = (fst\ \sigma 1, snd\ \sigma 1\ @\ snd\ \sigma 2)$ **using** *ih* **by** *auto*
moreover $have\ lst\ \sigma 1 = fst\ \sigma 2$ **using** *ih* **unfolding** *lst-def* **by** *auto*
moreover $have\ lst\ \sigma 2 = lst\ (s, x\#\#xs)$ **using** *ih* **unfolding** *lst-def* **by** *auto*
moreover $have\ labels\ \sigma 1 = \sigma 1'$ **using** *ih* $\sigma 1'\text{-dec}$ *y1* **unfolding** *labels-def* **by**
auto
moreover $have\ labels\ \sigma 2 = \sigma 2'$ **using** *ih* **by** *auto*
ultimately **show** $?thesis$ **by** *blast*
qed

qed
qed

lemma *seq-imp-conv*:

assumes $(s,ss) \in \text{seq ars}$

shows $(s, \text{map } (\lambda \text{step. } (\text{True}, \text{step})) \text{ ss}) \in \text{conv ars} \wedge$
 $\text{lst-conv } (s, \text{map } (\lambda \text{step. } (\text{True}, \text{step})) \text{ ss}) = \text{lst } (s, \text{ss}) \wedge$
 $\text{labels } (s, \text{ss}) = \text{labels-conv } (s, \text{map } (\lambda \text{step. } (\text{True}, \text{step})) \text{ ss})$

using *assms* **proof** (*induct ss arbitrary: s rule:list.induct*)

case Nil show ?*case* **unfolding** *lst-def labels-def labels-conv-def* **apply** *auto*
using *conv.intros* **by** *fast*

next

case (*Cons t' ts*) **have** *t'-dec*: $t' = (\text{fst } t', \text{snd } t')$ **using** *surjective-pairing* **by** *auto*
have *step*: $(s, \text{fst } t', \text{snd } t') \in \text{ars}$ **and** $x: (\text{snd } t', \text{ts}) \in \text{seq ars}$ **using** *seq-tail1[OF Cons(2)]* **by** *auto*

have *y1*: $(\text{snd } t', \text{map } (\text{Pair True}) \text{ ts}) \in \text{conv ars}$ **and**

y2: $\text{lst } (\text{snd } t', \text{ts}) = \text{lst-conv } (\text{snd } t', \text{map } (\text{Pair True}) \text{ ts})$ **and**

y3: $\text{labels } (\text{snd } t', \text{ts}) = \text{labels-conv } (\text{snd } t', \text{map } (\text{Pair True}) \text{ ts})$ **using**
Cons(1)[OF x] **by** *auto*

have *k*: $(s, (\text{True}, \text{fst } t', \text{snd } t') \# \text{map } (\text{Pair True}) \text{ ts}) \in \text{conv ars}$ **using** *step y1*
conv.intros **by** *fast*

moreover **have** $\text{lst } (s, (\text{fst } t', \text{snd } t') \# \text{ts}) = \text{lst-conv } (s, \text{map } (\text{Pair True}) ((\text{fst } t', \text{snd } t') \# \text{ts}))$ **using** *y2* **unfolding** *list.map lst-def lst-conv.simps* **by** *auto*

moreover **have** $\text{labels } (s, (\text{fst } t', \text{snd } t') \# \text{ts}) = \text{labels-conv } (s, \text{map } (\text{Pair True}) ((\text{fst } t', \text{snd } t') \# \text{ts}))$ **using** *y3* **unfolding** *list.map labels-def labels-conv-def* **by** *auto*

ultimately show ?*case* **by** *auto*

qed

fun *conv-mirror* :: $('a, 'b) \text{ conv} \Rightarrow ('a, 'b) \text{ conv}$

where *conv-mirror* $\sigma = (\text{let } (s, \text{ss}) = \sigma \text{ in case ss of}$

$\square \Rightarrow (s, \text{ss})$

$| x \# xs \Rightarrow \text{let } (d, \alpha, t) = x \text{ in}$

$(\text{fst } (\text{conv-mirror } (t, \text{xs})), \text{snd } (\text{conv-mirror } (t, \text{xs}))) @ [(\neg d, \alpha, s)]))$

lemma *conv-mirror*: **assumes** $\sigma \in \text{conv ars}$

shows *conv-mirror* $\sigma \in \text{conv ars} \wedge$

$\text{set } (\text{labels-conv } (\text{conv-mirror } \sigma)) = \text{set } (\text{labels-conv } \sigma) \wedge$

$\text{fst } \sigma = \text{lst-conv } (\text{conv-mirror } \sigma) \wedge$

$\text{lst-conv } \sigma = \text{fst } (\text{conv-mirror } \sigma)$ **proof** –

from *assms* **obtain** *s ss* **where** $\sigma\text{-dec}$: $\sigma = (s, \text{ss})$ **using** *surjective-pairing* **by**
metis

show ?*thesis* **using** *assms* **unfolding** $\sigma\text{-dec}$ **proof** (*induct ss arbitrary:s rule:list.induct*)

case Nil thus ?*case* **using** *conv.intros conv-mirror.simps* **by** *auto*

next

case (*Cons t' ts*) **from this** **obtain** *d α t* **where** $t'\text{-dec}$: $t' = (d, \alpha, t)$ **by** (*metis*
prod-cases3)

have *1*: $(t, \text{ts}) \in \text{conv ars}$ **and** *2*: $d \Longrightarrow (s, \alpha, t) \in \text{ars}$ **and** *3*: $\neg d \Longrightarrow (t, \alpha, s) \in$
 ars **and** *4*: $\text{lst-conv } (s, t' \# \text{ts}) = \text{lst-conv } (t, \text{ts})$

using *Cons(2) conv-tail1* **unfolding** $t'\text{-dec}$ **by** *auto*

have $r: (t, [(-d, \alpha, s)]) \in \text{conv ars}$ **using** $2\ 3\ \text{conv.intros}(3)[OF - \text{conv.intros}(1)]$
 $\text{conv.intros}(2)[OF - \text{conv.intros}(1)]$ **by** $(\text{cases } d)$ **auto**
have $\text{conv-mirror } (s, t' \# ts) \in \text{conv ars}$ **using** $\text{conv-concat}[OF - r]$ $\text{Cons}(1)[OF$
 $1]$ t' -dec **by** **auto**
moreover **have** $\text{set } (\text{labels-conv } (\text{conv-mirror } (s, t' \# ts))) = \text{set } (\text{labels-conv}$
 $(s, t' \# ts))$ **using** $\text{Cons}(1)[OF 1]$ **unfolding** $\text{labels-conv-def } t'$ -dec **by** **auto**
moreover **have** $\text{fst } (s, t' \# ts) = \text{lst-conv } (\text{conv-mirror } (s, t' \# ts))$ **using** t' -dec
 $\text{Cons}(1)[OF 1]$ $\text{conv-concat}(2)[OF - r]$ **by** **auto**
moreover **have** $\text{lst-conv } (s, t' \# ts) = \text{fst } (\text{conv-mirror } (s, t' \# ts))$ **using** t' -dec
 $4\ \text{Cons}(1)[OF 1]$ **by** **auto**
ultimately show $?case$ **by** **auto**
qed
qed

lemma *DD-subset-helper*:

assumes $t:\text{trans } r$ **and** $(r|\tau @ \sigma', r|\tau + r|\sigma) \in \text{mul-eq } r$ **and** $\text{set-mset } (r|\tau +$
 $r|\sigma) \subseteq \text{ds } r\ S$
shows $\text{set-mset } r|\sigma' \subseteq \text{ds } r\ S$ **proof** –
from *assms* **have** $d: (r|\tau + r|\sigma' - s\ \text{dl } r\ \tau), r|\tau + r|\sigma \in \text{mul-eq } r$ **unfolding**
 lemma3-2-2 **by** **auto**
from *assms* **have** $\text{assm}:\text{set-mset } (r|\tau + r|\sigma) \subseteq \text{ds } r\ S$ **unfolding** measure-def
by **auto**
hence $\text{tau}:\text{ds } r\ (\text{set-mset } r|\tau) \subseteq \text{ds } r\ S$ **using** $\text{subset-imp-ds-subset}[OF t]$ **by**
auto
have $\text{set-mset } (r|\tau + (r|\sigma' - s\ \text{dl } r\ \tau)) \subseteq \text{ds } r\ S$ **using** $\text{mul-eq-and-ds-imp-ds}[OF$
 $t\ d\ \text{assm}]$ **by** **auto**
hence $\text{set-mset } (r|\sigma' - s\ \text{ds } r\ (\text{set } \tau)) \subseteq \text{ds } r\ S$ **unfolding** dl-def **by** **auto**
hence $\text{set-mset } (r|\sigma' - s\ \text{ds } r\ (\text{set } \tau)) \cup \text{ds } r\ (\text{set } \tau) \subseteq \text{ds } r\ S$ **using** tau **by**
 $(\text{metis } t\ \text{dl-def } \text{dm-def } \text{le-sup-iff } \text{lemma3-2-1})$
thus $?thesis$ **unfolding** diff-def **by** **auto**
qed

lemma *DD-subset-ds*:

assumes $t:\text{trans } r$ **and** $DD: DD\ \text{ars } r\ (\tau, \sigma, \sigma', \tau')$ **and** $\text{set-mset } (\text{measure } r\ (\tau, \sigma))$
 $\subseteq \text{ds } r\ S$ **shows** $\text{set-mset } (\text{measure } r\ (\sigma', \tau')) \subseteq \text{ds } r\ S$ **proof** –
have $d1: (r|\text{labels } \tau @ \text{labels } \sigma', r|\text{labels } \tau + r|\text{labels } \sigma) \in \text{mul-eq } r$ **using** DD
unfolding $DD\text{-def } D2\text{-def } D\text{-def}$ **by** **auto**
have $d2: (r|\text{labels } \sigma @ \text{labels } \tau', r|\text{labels } \sigma + r|\text{labels } \tau) \in \text{mul-eq } r$ **using** DD
unfolding $DD\text{-def } D2\text{-def } D\text{-def}$
by $(\text{auto simp: union-commute})$
show $?thesis$ **using** $DD\text{-subset-helper}[OF t\ d1]$ $DD\text{-subset-helper}[OF t\ d2]$ $\text{assms}(3)$
unfolding measure-def **by** **auto**
qed

lemma *conv-imp-valley*:

assumes $t:\text{trans } r$
and $IH: !!y. ((y, ((s, [\alpha\text{-step}] @ \varrho\text{-step}), (s, [\beta\text{-step}] @ v\text{-step}))) \in \text{pex } r \implies \text{peak ars } y$
 $\implies \exists \sigma' \tau'. DD\ \text{ars } r\ (\text{fst } y, \text{snd } y, \sigma', \tau')$ **(is** $!!y. ((y, ?P) \in - \implies - \implies -)$
and $\delta 1 \in \text{conv ars}$

and $set\text{-}mset (measure\text{-}conv\ r\ \delta 1) \subseteq dm\ r\ M$
and $(M, \{\#fst\ \alpha\text{-}step, fst\ \beta\text{-}step\#\}) \in mul\text{-}eq\ r$
shows $\exists\ \sigma\ \tau. (\{\sigma, \tau\} \subseteq seq\ ars \wedge fst\ \sigma = fst\ \delta 1 \wedge fst\ \tau = lst\text{-}conv\ \delta 1 \wedge lst\ \sigma = lst\ \tau \wedge set\text{-}mset (measure\ r\ (\sigma, \tau)) \subseteq dm\ r\ M)$ **proof** –
from *assms* **obtain** $s\ ss$ **where** $sigma1: \delta 1 = (s, ss)$ **using** *surjective-pairing* **by** *metis*
show *?thesis* **using** *assms(3,4)* **unfolding** *sigma1* **proof** (*induct ss arbitrary: s rule:list.induct*)
case *Nil*
hence $(s, []) \in seq\ ars$ **using** *seq.intros(1)* **by** *fast*
moreover **have** $set\text{-}mset (measure\ r\ ((s, []), (s, []))) \subseteq dm\ r\ M$ **unfolding** *measure-def labels-def* **by** *auto*
ultimately **show** *?case* **by** *auto*
next
case $(Cons\ t'\ ts)$ **obtain** $d\ \beta\ t$ **where** $dec: t' = (d, \beta, t)$ **using** *surjective-pairing* **by** *metis*
hence $dic: \{\beta\} \subseteq ds\ r\ (set\text{-}mset\ M)$ **using** *Cons(3)* **unfolding** *dec measure-conv-def labels-conv-def dm-def* **by** *auto*
have $one: ds\ r\ \{\beta\} \subseteq dm\ r\ M$ **unfolding** *dm-def* **using** *subset-imp-ds-subset[OF t dic]* **by** *auto*
have $set\text{-}mset (measure\text{-}conv\ r\ (t, ts) -s\ ds\ r\ \{\beta\}) \subseteq dm\ r\ M$ **using** *Cons(3)* **unfolding** *measure-conv-def labels-conv-def dec* **by** *auto*
hence $set\text{-}mset (measure\text{-}conv\ r\ (t, ts)) \subseteq dm\ r\ M \cup ds\ r\ \{\beta\}$ **unfolding** *set-mset-def diff-def* **by** *auto*
hence $ts2: set\text{-}mset (measure\text{-}conv\ r\ (t, ts)) \subseteq dm\ r\ M$ **using** *dic one* **by** *auto*
from *Cons(2)* **have** $ts: (t, ts) \in conv\ ars$ **unfolding** *dec* **using** *conv-tail1(1)* **by** *fast*
from *Cons(1)[OF ts ts2]* **obtain** $\sigma'\ \tau$ **where**
ih: $\{\sigma', \tau\} \subseteq seq\ ars \wedge fst\ \sigma' = fst\ (t, ts) \wedge fst\ \tau = lst\text{-}conv\ (t, ts) \wedge lst\ \sigma' = lst\ \tau \wedge set\text{-}mset (measure\ r\ (\sigma', \tau)) \subseteq dm\ r\ M$ **by** *metis*
have $diff: !!x. x \in \# r | map\ fst\ (snd\ \sigma') | -sds\ r\ \{\beta\} \implies x \in \# r | map\ fst\ (snd\ \sigma')$ **by** *simp*
show *?case* **proof** (*cases d*)
case *True* **hence** $step: (s, \beta, t) \in ars$ **using** *conv-tail1(2)* *Cons(2)* **unfolding** *dec* **by** *auto*
have $(s, (\beta, t) \# snd\ \sigma') \in seq\ ars$ (**is** $?\sigma \in -$) **using** *seq.intros(2)[OF step]* **using** *ih(1)* **by** *auto*
hence $\{?\sigma, \tau\} \subseteq seq\ ars$ **using** *ih* **by** *auto*
moreover **have** $(fst\ ?\sigma) = fst\ (s, t' \# ts)$ **by** *auto*
moreover **have** $fst\ \tau = lst\text{-}conv\ (s, t' \# ts)$ **using** *ih* **unfolding** *dec lst-conv.simps* **by** *auto*
moreover **have** $lst\ ?\sigma = lst\ \tau$ **by** (*metis $\langle (s, (\beta, t) \# snd\ \sigma') \in seq\ ars \rangle$* *fst-conv ih seq-tail1(3) snd-conv surjective-pairing*)
moreover **have** $set\text{-}mset (measure\ r\ (?\sigma, \tau)) \subseteq dm\ r\ M$ **using** *diff ih dic* **unfolding** *measure-def labels-def dm-def* **by** *auto*
ultimately **show** *?thesis* **by** *blast*
next
case *False* **hence** $step: (t, \beta, s) \in ars$ **using** *conv-tail1(3)* *Cons(2)* **unfolding** *dec* **by** *auto*

hence $(t,[(\beta,s)]) \in \text{seq ars}$ **using** *seq.intros* **by** *metis*
hence $p:\text{peak ars } ((t,[(\beta,s))),\sigma')$ **(is peak ars ?y)** **using** *seq.intros* **unfolding**
peak-def **using** *ih* **by** *auto*
hence $mp:\text{set-mset } (\text{measure } r ?y) \subseteq ds \ r \ (\text{set-mset } M)$ **using** *ih* *dic* **unfolding**
measure-def *labels-def* *dm-def*
by *simp*
hence $ne: M \neq \{\#\}$ **using** *dec* *dic* **unfolding** *dm-def* *ds-def* **by** *auto*
hence $x:(\text{measure } r ?y, M) \in \text{mul } r$ **using** *mp* **unfolding** *dm-def* *mul-def* **apply**
auto **by** (*metis* *add-0*)
have $(\{\#\text{fst } \alpha\text{-step}\#\} + \{\#\text{fst } \beta\text{-step}\#\}, \text{measure } r ?P) \in \text{mul-eq } r$ **unfolding**
assms(2) *measure-def* *labels-def* **apply** *auto*
unfolding *union-lcomm* *union-assoc[symmetric]* **using** *mul-eq-add-right* **[where**
 $M = \{\#\text{fst } \alpha\text{-step}\#\} + \{\#\text{fst } \beta\text{-step}\#\}$ **unfolding** *union-assoc* **by** *auto*
hence $(M, \text{measure } r ?P) \in \text{mul-eq } r$ **using** *assms(5)* *mul-eq-trans* *t* **by** (*auto*
simp: add-mset-commute)
hence $w:(?y, ?P) \in \text{pex } r$ **unfolding** *assms(1)* *pex-def* **using** *mul-and-mul-eq-imp-mul[OF*
t x] **by** *auto*
obtain $\sigma'' \ \tau''$ **where** $DD:DD \text{ ars } r \ ((t,[(\beta,s))),\sigma',\sigma'',\tau'')$ **using** *IH[OF w p]* **by**
auto
have $\text{sigma}: \sigma'' \in \text{seq ars}$ $\text{fst } \sigma'' = \text{fst } (s, t' \# ts)$ $\text{lst } \sigma'' = \text{lst } \tau''$ **using** *DD*
unfolding *DD-def* *diagram-def* *lst-def* **by** *auto*
have $\text{tau}'': \tau'' \in \text{seq ars}$ **and** $\text{eq}: \text{lst } \tau = \text{fst } \tau''$ **using** *DD* **unfolding** *DD-def*
diagram-def **using** *ih* **by** *auto*
have $\text{tau}: (\text{fst } \tau, \text{snd } \tau @ \text{snd } \tau'') \in \text{seq ars}$ **(is ?\tau''' \in -)** **and** $\text{lst } \tau'' = \text{lst } (\text{fst}$
 $\tau, \text{snd } \tau @ \text{snd } \tau'')$ **using** *seq-concat[OF - tau'' eq]* **ih** **by** *auto*
hence $\text{tau2}: \text{fst } ?\tau''' = \text{lst-conv } (s, t' \# ts)$ $\text{lst } \sigma'' = \text{lst } ?\tau'''$ **using** *DD* **ih** **un-**
folding *DD-def* *diagram-def* *dec* *lst-conv.simps* **by** *auto*
have $\text{set-mset } (\text{measure } r (\sigma'', \tau'')) \subseteq ds \ r \ (\text{set-mset } M)$ **using** *DD-subset-ds[OF*
t DD mp] **unfolding** *measure-def* **by** *auto*
hence $\text{set-mset } (r|\text{labels } \sigma''| + r|\text{labels } \tau| + (r|\text{labels } \tau''| - s \ (dl \ r \ (\text{labels } \tau))))$
 $\subseteq dm \ r \ M$ **using** *ih* **unfolding** *measure-def* *dm-def* *diff-def* **by** *auto*
hence $\text{fin}: \text{set-mset } (\text{measure } r (\sigma'', ?\tau''')) \subseteq dm \ r \ M$ **unfolding** *measure-def*
labels-def **apply** *auto* **unfolding** *lemma3-2-2* **by** *auto*
show *?thesis* **using** *sigma* *tau* *tau2* *fin* **by** *blast*
qed
qed
qed

lemma *labels-multiset*: **assumes** $\text{length } (\text{labels } \sigma) \leq 1$ **and** $\text{set } (\text{labels } \sigma) \subseteq \{\alpha\}$
shows $(r|\text{labels } \sigma|, \{\#\alpha\#\}) \in \text{mul-eq } r$ **proof** (*cases* *snd* σ)
case *Nil* **hence** $r|\text{labels } \sigma| = \{\#\}$ **unfolding** *labels-def* **by** *auto*
thus *?thesis* **unfolding** *mul-eq-def* **by** *auto*
next
case (*Cons* x xs) **hence** $l:\text{length } (\text{labels } \sigma) = 1$ **using** *assms(1)* **unfolding** *la-*
labels-def **by** *auto*
from *this* **have** $\text{labels } \sigma \neq []$ **by** *auto*
from *this* **obtain** a as **where** $\text{labels } \sigma = a \# as$ **using** *neq-Nil-conv* **by** *metis*
hence $\text{leq}: \text{labels } \sigma = [a]$ **using** l **by** *auto*
hence $\text{set } (\text{labels } \sigma) = \{\alpha\}$ **using** *assms(2)* **by** *auto*

hence $(r|labels\ \sigma|) = \{\#\alpha\#\}$ **unfolding** *leq lexmax.simps diff-def* **by auto**
thus *?thesis using mul-eq-reflexive* **by auto**
qed

lemma *decreasing-imp-local-decreasing*:

assumes *t:trans r* **and** *i:irrefl r* **and** *DD: DD ars r* $(\tau, \sigma, \sigma', \tau')$ **and** *set (labels τ)*
 $\subseteq ds\ r\ \{\beta\}$

and *length (labels σ)* ≤ 1 **and** *set (labels σ)* $\subseteq \{\alpha\}$

shows $\exists \sigma 1\ \sigma 2\ \sigma 3. (\sigma' = (fst\ \sigma 1, snd\ \sigma 1 @ snd\ \sigma 2 @ snd\ \sigma 3) \wedge lst\ \sigma 1 = fst\ \sigma 2 \wedge lst$
 $\sigma 2 = fst\ \sigma 3 \wedge lst\ \sigma 3 = lst\ \sigma')$

$\wedge LD-1'\ r\ \beta\ \alpha\ (labels\ \sigma 1)\ (labels\ \sigma 2)\ (labels\ \sigma 3))$

$set\ (labels\ \tau') \subseteq ds\ r\ (\{\alpha, \beta\})$

proof –

show $\exists \sigma 1\ \sigma 2\ \sigma 3. (\sigma' = (fst\ \sigma 1, snd\ \sigma 1 @ snd\ \sigma 2 @ snd\ \sigma 3) \wedge$
 $lst\ \sigma 1 = fst\ \sigma 2 \wedge lst\ \sigma 2 = fst\ \sigma 3 \wedge lst\ \sigma 3 = lst\ \sigma' \wedge LD-1'\ r\ \beta\ \alpha\ (labels$
 $\sigma 1)\ (labels\ \sigma 2)\ (labels\ \sigma 3))$ **proof** –

from *DD* **have** $\sigma':\sigma' \in seq\ ars$ **using** *assms unfolding DD-def diagram-def* **by auto**

from *DD* **have** $x:(r|labels\ \sigma'| -s\ dl\ r\ (labels\ \tau), r|labels\ \sigma|) \in mul-eq\ r$ **unfolding**
DD-def D2-def **using** *D-eq(2)[OF t i]* **by auto**

have $dl\ r\ (labels\ \tau) \subseteq ds\ r\ (ds\ r\ \{\beta\})$ **using** *assms(4) unfolding dl-def ds-def*
by auto

hence $dl\ r\ (labels\ \tau) \subseteq ds\ r\ \{\beta\}$ **using** *ds-ds-subseteq-ds[OF t]* **by auto**

hence $x:(r|labels\ \sigma'| -s\ ds\ r\ \{\beta\}, r|labels\ \sigma|) \in mul-eq\ r$ **using** *x unfolding*
diff-def **by** *(metis diff-def lemma2-6-8 mul-eq-trans t)*

hence $x:(r|labels\ \sigma'| -s\ dl\ r\ [\beta], \{\#\alpha\#\}) \in mul-eq\ r$ **using** *labels-multiset[OF*
assms(5,6)] unfolding dl-def **using** *mul-eq-trans[OF t x]* **by auto**

obtain $\sigma 1'\ \sigma 2'\ \sigma 3'$ **where** $l:labels\ \sigma' = \sigma 1' @ (\sigma 2' @ \sigma 3')$ **and** $\sigma 1'l: set\ \sigma 1' \subseteq$
 $ds\ r\ \{\beta\}$ **and**

$\sigma 2'l: length\ \sigma 2' \leq 1 \wedge set\ \sigma 2' \subseteq \{\alpha\}$ **and** $\sigma 3'l: set\ \sigma 3' \subseteq ds\ r\ \{\alpha, \beta\}$ **using**
proposition3-4-inv-lists[OF t i x] **unfolding** *dl-def* **by auto**

obtain $\sigma 1\ \sigma 23$ **where** $\sigma 1:\sigma 1 \in seq\ ars$ **and** $\sigma 23:\sigma 23 \in seq\ ars$ **and** *lf1: lst $\sigma 1$*
 $= fst\ \sigma 23$ **and** *lf1b: lst $\sigma' = lst\ \sigma 23$* **and**

$\sigma'-eq:\sigma' = (fst\ \sigma 1, snd\ \sigma 1 @ snd\ \sigma 23)$ **and** $\sigma 1l:labels\ \sigma 1 = \sigma 1'$ **and** $l2:labels\ \sigma 23$
 $= \sigma 2' @ \sigma 3'$

using *seq-decompose[OF $\sigma' l$]* **by auto**

obtain $\sigma 2\ \sigma 3$ **where** $\sigma 2:\sigma 2 \in seq\ ars$ **and** $\sigma 3:\sigma 3 \in seq\ ars$ **and** *lf2:lst $\sigma 2 =$*
 $fst\ \sigma 3$ **and** *lf2b:lst $\sigma 23 = lst\ \sigma 3$* **and**

$\sigma 23-eq:\sigma 23 = (fst\ \sigma 2, snd\ \sigma 2 @ snd\ \sigma 3)$ **and** $\sigma 2l: labels\ \sigma 2 = \sigma 2'$ **and** $\sigma 3l: labels$
 $\sigma 3 = \sigma 3'$

using *seq-decompose[OF $\sigma 23\ l2$]* **by auto**

have $\sigma' = (fst\ \sigma 1, snd\ \sigma 1 @ snd\ \sigma 2 @ snd\ \sigma 3)$ **using** $\sigma 1\ \sigma 2\ \sigma 3\ \sigma'-eq\ \sigma 23-eq$
by auto

moreover **have** $lst\ \sigma 1 = fst\ \sigma 2$ **using** *lf1 $\sigma 23-eq$* **by auto**

moreover **have** $lst\ \sigma 2 = fst\ \sigma 3$ **using** *lf2* **by auto**

moreover **have** $lst\ \sigma 3 = lst\ \sigma'$ **using** *lf1b lf2b* **by auto**

moreover **have** $set\ (labels\ \sigma 1) \subseteq ds\ r\ \{\beta\}$ **using** $\sigma 1l\ \sigma 1'l$ **by auto**

moreover **have** $length\ (labels\ \sigma 2) \leq 1 \wedge set\ (labels\ \sigma 2) \subseteq \{\alpha\}$ **using** $\sigma 2l\ \sigma 2'l$
by auto

moreover have $set (labels \sigma 3) \subseteq ds r \{\alpha, \beta\}$ **using** $\sigma 3 l \sigma 3' l$ **by auto**
ultimately show *?thesis unfolding LD-1'-def by fast*
qed
show $set (labels \tau') \subseteq ds r (\{\alpha, \beta\})$ **proof** –
have $x: (r|labels \tau'|-s dl r (labels \sigma), r|labels \tau|) \in mul-eq r$ **using** $DD D-eq[OF t i]$ **unfolding** $DD-def D2-def$ **by auto**
have $y: set-mset r|labels \tau| \subseteq ds r \{\beta\}$ **using** $leq-imp-subseteq[OF lexmax-le-multiset[OF t]]$ $assms(4)$ **by auto**
hence $set-mset (r|labels \tau'|-s ds r (set (labels \sigma))) \subseteq ds r \{\beta\}$ **using** $mul-eq-and-ds-imp-ds[OF t x y]$ **unfolding** $dl-def$ **by auto**
hence $set-mset (r|labels \tau'|) \subseteq ds r \{\beta\} \cup ds r (set (labels \sigma))$ **unfolding** $diff-def$ **by auto**
hence $set-mset (r|labels \tau'|) \subseteq ds r \{\alpha, \beta\}$ **using** $assms(6)$ **unfolding** $ds-def$ **by auto**
thus *?thesis using lexmax-set[OF t] by auto*
qed
qed

lemma *local-decreasing-extended-imp-decreasing:*

assumes $LT1$ $ars r (s, [\beta\text{-step}]) (s, [\alpha\text{-step}]) \gamma 1 \gamma 2 \gamma 3$

and $t: trans r$ **and** $i: irrefl r$

and $IH: !!y. ((y, ((s, [\beta\text{-step}]@v\text{-step}), (s, [\alpha\text{-step}]@p\text{-step}))) \in pex r \implies peak ars y \implies \exists \sigma' \tau'. DD ars r (fst y, snd y, \sigma', \tau') (is !!y. ((y, ?P) \in - \implies - \implies -))$

shows $\exists \sigma 1 \sigma 2 \sigma 3' \gamma 1'''. (\{\sigma 1, \sigma 2, \sigma 3', \gamma 1'''\} \subseteq seq ars \wedge$

$set (labels \sigma 1) \subseteq ds r \{fst \beta\text{-step}\} \wedge length (labels \sigma 2) \leq 1 \wedge set (labels \sigma 2) \subseteq \{fst \alpha\text{-step}\} \wedge set (labels \sigma 3') \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\} \wedge$

$set (labels \gamma 1''') \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\} \wedge$

$snd \beta\text{-step} = fst \sigma 1 \wedge lst \sigma 1 = fst \sigma 2 \wedge lst \sigma 2 = fst \sigma 3' \wedge lst \sigma 3' = lst \gamma 1'''$
 $\wedge fst \gamma 1'''' = fst \gamma 3$

proof –

from $assms labels-multiset$ **have** $s2: (r|labels \gamma 2|, \{\#fst \alpha\text{-step}\}) \in mul-eq r$ **unfolding** $LT1-def local-triangle1-def LD-1'-def labels-def$ **by auto**

from $assms$ **have** $\gamma 1: \gamma 1 \in conv ars$ **and** $\gamma 3: \gamma 3 \in conv ars$ **and** $\gamma 2-l: length (labels \gamma 2) \leq 1$

and $\gamma 2-s: set (labels \gamma 2) \subseteq \{fst \alpha\text{-step}\}$ **and** $\gamma 3-s: set (labels conv \gamma 3) \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\}$

and $set (labels conv \gamma 1) \subseteq ds r \{fst \beta\text{-step}\}$ **unfolding** $LT-def LD'-def LT1-def LD-1'-def labels-def local-triangle1-def$ **by auto**

hence $set-mset (measure conv r \gamma 1) \subseteq ds r \{fst \beta\text{-step}\}$ **unfolding** $measure conv-def$ **using** $lexmax-le-multiset[OF t]$ **by** $(metis set-mset-mset submultiset-implies-subset subset-trans)$

hence $\gamma 1-s: set-mset (measure conv r \gamma 1) \subseteq dm r \{\#fst \beta\text{-step}\}$ **unfolding** $dm-def$ **by auto**

have $x: (\{\#fst \beta\text{-step}\}, \{\#fst \beta\text{-step}, fst \alpha\text{-step}\}) \in mul-eq r$ **using** $mul-eq-add-right[of \{\#\text{-}\#\}]$ **by auto**

obtain $\gamma 1' \gamma 1''$ **where** $\gamma 1': \gamma 1' \in seq ars$ **and** $\gamma 1'': \gamma 1'' \in seq ars$ **and** $eqx: fst \gamma 1' = fst \gamma 1$

and $fst \gamma 1'' = lst conv \gamma 1$ **and** $\gamma 1'\text{-eq}: lst \gamma 1' = lst \gamma 1''$ **and** $m2: set-mset (measure r (\gamma 1', \gamma 1'')) \subseteq dm r \{\#fst \beta\text{-step}\}$

using *conv-imp-valley*[*OF t IH $\gamma_1 \gamma_1\text{-s } x$*] **apply auto by fast**
hence $Q:\text{peak ars } (\gamma_1'', \gamma_2)$ (**is peak ars** $?Q$) **unfolding peak-def using** *assms(1)*
unfolding *LT-def LD'-def LT1-def local-triangle1-def* **by auto**
from m_2 **have** γ_1' 's: *set (labels γ_1') \subseteq ds r {fst β -step}* **and** γ_1'' 's: *set (labels γ_1'') \subseteq ds r {fst β -step}* **unfolding** *measure-def dm-def* **using** *lexmax-set[OF t]*
by auto
from m_2 **have** $\tau_1'\text{-m}:(r|\text{labels } \gamma_1'', \{\#\text{fst } \beta\text{-step}\}) \in \text{mul } r$ **unfolding** *measure-def mul-def* **apply auto by** (*metis dm-def empty-neutral(1) set-mset-single add-mset-not-empty*)
hence $y:(r|\text{labels } \gamma_1'' + r|\text{labels } \gamma_2, \{\#\text{fst } \alpha\text{-step}\} + \{\#\text{fst } \beta\text{-step}\}) \in \text{mul } r$
using *mul-add-mul-eq-imp-mul[OF $\tau_1'\text{-m } s_2$] union-commute* **by metis**
have $(r|\text{labels } \gamma_1'' + r|\text{labels } \gamma_2, \{\#\text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\}) + (r|\text{map fst } \varrho\text{-step} | \text{sds } r \{\text{fst } \alpha\text{-step}\} + r|\text{map fst } \nu\text{-step} | \text{sds } r \{\text{fst } \beta\text{-step}\})) \in \text{mul } r$ **using** *mul-add-right[OF y]* **by** (*auto simp: add-mset-commute*)
hence $q:(?Q, ?P) \in \text{pex } r$ **unfolding** *pex-def measure-def labels-def* **apply auto by** (*metis union-assoc union-commute union-lcomm*)
obtain $\gamma_1''' \sigma'$ **where** $DD:DD \text{ ars } r (\gamma_1'', \gamma_2, \sigma', \gamma_1''')$ **using** *IH[OF q Q]* **by auto**
from DD **have** $\sigma': \sigma' \in \text{seq ars}$ **and** $\gamma_1''': \gamma_1''' \in \text{seq ars}$ **unfolding** *DD-def diagram-def* **by auto**
from *decreasing-imp-local-decreasing[OF t i DD γ_1'' 's $\gamma_2\text{-l } \gamma_2\text{-s}$]* **obtain** $\sigma_1' \sigma_2' \sigma_3'$ **where** $\sigma'\text{-dec}: \sigma' = (\text{fst } \sigma_1', \text{snd } \sigma_1' @ \text{snd } \sigma_2' @ \text{snd } \sigma_3')$
and $\text{eq}_1: \text{lst } \sigma_1' = \text{fst } \sigma_2' \text{ lst } \sigma_2' = \text{fst } \sigma_3' \text{ lst } \sigma_3' = \text{lst } \sigma'$
and $\sigma_1\text{s}: \text{set (labels } \sigma_1') \subseteq \text{ds } r \{\text{fst } \beta\text{-step}\}$ **and** $\sigma_2\text{l}: \text{length (labels } \sigma_2') \leq 1$
and $\sigma_2'\text{s}: \text{set (labels } \sigma_2') \subseteq \{\text{fst } \alpha\text{-step}\}$ **and** $\text{set (labels } \sigma_3') \subseteq \text{ds } r \{\text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\}$
and $\sigma_3'\text{s}: \text{set (labels } \sigma_3') \subseteq \text{ds } r \{\text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\}$ **and** $\gamma_1'''\text{s}: \text{set (labels } \gamma_1''') \subseteq \text{ds } r \{\text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\}$ **unfolding** *LD-1'-def* **by blast**
have $\sigma'\text{-ds}: (\text{fst } \sigma_1', \text{snd } \sigma_1' @ \text{snd } \sigma_2' @ \text{snd } \sigma_3') \in \text{seq ars}$ **using** $\sigma' \sigma'\text{-dec}$ **by auto**
have $\sigma_1'\text{:}\sigma_1' \in \text{seq ars}$ **and** $\text{tmp}: (\text{fst } \sigma_2', \text{snd } \sigma_2' @ \text{snd } \sigma_3') \in \text{seq ars}$ **using** *seq-chop[OF $\sigma'\text{-ds}$] surjective-pairing eq1* **by auto**
have $\sigma_2'\text{:}\sigma_2' \in \text{seq ars}$ **and** $\sigma_3'\text{:}\sigma_3' \in \text{seq ars}$ **using** *seq-chop[OF tmp]* **using** *surjective-pairing eq1* **by auto**
have $\text{eq}: \text{lst } \gamma_1' = \text{fst } \sigma_1'$ **using** *DD $\gamma_1'\text{-eq}$* **unfolding** *DD-def diagram-def* **apply auto unfolding $\sigma'\text{-dec}$ **by auto**
have $\sigma_1: (\text{fst } \gamma_1', \text{snd } \gamma_1' @ \text{snd } \sigma_1') \in \text{seq ars}$ (**is** $? \sigma_1 \in -$) **and** $\text{eq}_0: \text{lst } (\text{fst } \gamma_1', \text{snd } \gamma_1' @ \text{snd } \sigma_1') = \text{lst } \sigma_1'$ **using** *seq-concat[OF $\gamma_1' \sigma_1' \text{ eq}$]* **by auto**
moreover **have** $\text{set (labels } ? \sigma_1) \subseteq \text{ds } r \{\text{fst } \beta\text{-step}\}$ **using** $\sigma_1\text{s } \gamma_1'\text{s}$ **unfolding** *labels-def dm-def* **by auto**
moreover **have** $\text{snd } \beta\text{-step} = \text{fst } ? \sigma_1$ **using** *assms(1)* **unfolding** *LT1-def local-triangle1-def lst-def $\sigma'\text{-dec}$ eqx* **by auto**
moreover **have** $\text{lst } ? \sigma_1 = \text{fst } \sigma_2'$ **unfolding** *eq0 eq1* **by auto**
moreover **have** $\text{lst } \sigma_3' = \text{lst } \gamma_1'''$ **unfolding** *eq1* **using** *DD* **unfolding** *DD-def diagram-def* **by auto**
moreover **have** $\text{fst } \gamma_1''' = \text{fst } \gamma_3$ **using** *DD assms(1)* **unfolding** *DD-def diagram-def LT1-def local-triangle1-def* **by auto**
ultimately show $? \text{thesis}$ **using** $\sigma_2' \sigma_2'\text{s } \sigma_3' \sigma_3'\text{s } \gamma_1''' \gamma_1'''\text{s } \text{eq}_1$ *surjective-pairing $\sigma_2\text{l}$* **by blast****

qed

lemma *LDD-imp-DD*:

assumes $t:trans\ r$ **and** $i:irrefl\ r$ **and** $LDD\ ars\ r\ (\tau,\sigma,\sigma 1,\sigma 2,\sigma 3,\tau 1,\tau 2,\tau 3)$

shows $\exists\ \sigma'\ \tau'.\ DD\ ars\ r\ (\tau,\sigma,\sigma',\tau')$ **proof** –

from *assms* **have** $length\ (labels\ \sigma) = 1$ **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def* **unfolding** *labels-def* **by** *auto*

from *this* **obtain** α **where** $l: labels\ \sigma = [\alpha]$ **by** (*metis append-Nil append-butlast-last-id butlast-conv-take diff-self-eq-0 length-0-conv take-0 zero-neq-one*)

hence $\sigma l: labels\ \sigma = [hd\ (labels\ \sigma)]$ **by** *auto*

from *assms* **have** $length\ (labels\ \tau) = 1$ **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def* **unfolding** *labels-def* **by** *auto*

from *this* **obtain** β **where** $l: labels\ \tau = [\beta]$ **by** (*metis append-Nil append-butlast-last-id butlast-conv-take diff-self-eq-0 length-0-conv take-0 zero-neq-one*)

hence $\tau l: labels\ \tau = [hd\ (labels\ \tau)]$ **by** *auto*

from *assms* **have** $\sigma':(fst\ \sigma 1, snd\ \sigma 1 @ snd\ \sigma 2 @ snd\ \sigma 3) \in seq\ ars$ (**is** $?\sigma' \in -$) **and** $\tau':(fst\ \tau 1, snd\ \tau 1 @ snd\ \tau 2 @ snd\ \tau 3) \in seq\ ars$ (**is** $?\tau' \in -$)

unfolding *LDD-def LDD1-def local-diagram1-def local-peak-def peak-def* **apply** *auto* **apply** (*metis fst-eqD seq-concat(1) snd-eqD*) **by** (*metis fst-eqD seq-concat(1) snd-eqD*)

from *assms* **have** $sigmas: fst\ ?\sigma' = fst\ \sigma 1\ lst\ ?\sigma' = lst\ \sigma 3$ **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def peak-def* **apply** *auto* **by** (*metis (opaque-lifting, no-types) \sigma' append-assoc seq-chop(1) seq-concat(2) seq-concat-helper*)

from *assms* **have** $taus: fst\ ?\tau' = fst\ \tau 1\ lst\ ?\tau' = lst\ \tau 3$ **unfolding** *LDD-def LDD1-def local-diagram1-def local-peak-def peak-def* **apply** *auto* **by** (*metis (opaque-lifting, no-types) \tau' append-assoc seq-chop(1) seq-concat(2) seq-concat-helper*)

have *diagram ars* $(\tau,\sigma,?\sigma',?\tau')$ **using** *assms* **unfolding** *LDD-def LDD1-def local-diagram1-def diagram-def local-peak-def* **apply** *auto*

unfolding *peak-def* **apply** *auto* **using** *sigmas taus \sigma' \tau'* **by** *auto*

moreover **have** $D2\ r\ (\tau,\sigma,?\sigma',?\tau')$ **using** *assms proposition3-4[OF t i] \sigma l \tau l* **unfolding** *LDD-def LDD1-def D2-def LD'-def labels-def* **by** *auto*

ultimately show *?thesis* **unfolding** *DD-def* **by** *auto*

qed

lemma *LT-imp-DD*:

assumes $t:trans\ r$

and $i:irrefl\ r$

and $IH:!!y. ((y,((s,[\beta\text{-step}]@v\text{-step}), (s,[\alpha\text{-step}]@p\text{-step}))) \in pex\ r \implies peak\ ars\ y \implies \exists\ \sigma'\ \tau'.\ DD\ ars\ r\ (fst\ y, snd\ y, \sigma', \tau'))$ (**is** $!!y. ((y,?P) \in - \implies - \implies -)$)

and $LT: LT\ ars\ r\ ((s,[\beta\text{-step}]), (s,[\alpha\text{-step}]), \gamma 1, \gamma 2, \gamma 3, \delta 1, \delta 2, \delta 3)$

shows $\exists\ \kappa\ \mu. DD\ ars\ r\ ((s,[\beta\text{-step}]), (s,[\alpha\text{-step}]), \kappa, \mu)$

proof –

from *LT* **have** $LTa: LT1\ ars\ r\ (s,[\beta\text{-step}])\ (s,[\alpha\text{-step}])\ \gamma 1\ \gamma 2\ \gamma 3$ **and** $LTb: LT1\ ars\ r\ (s,[\alpha\text{-step}])\ (s,[\beta\text{-step}])\ \delta 1\ \delta 2\ \delta 3$ **unfolding** *LT-def* **by** *auto*

from *local-decreasing-extended-imp-decreasing[OF LTa t i IH]* **obtain** $\sigma 1\ \sigma 2\ \sigma 3'\ \gamma 1''''$ **where** $sigmas: \{\sigma 1, \sigma 2, \sigma 3', \gamma 1''''\} \subseteq seq\ ars$ **and**

$onetwo1: set\ (labels\ \sigma 1) \subseteq ds\ r\ \{fst\ \beta\text{-step}\} \wedge length\ (labels\ \sigma 2) \leq 1 \wedge set\ (labels\ \sigma 2) \subseteq \{fst\ \alpha\text{-step}\} \wedge$

$set (labels \sigma 3') \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\} \wedge set (labels \gamma 1''') \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\} \wedge$
 $snd \beta\text{-step} = fst \sigma 1 \wedge lst \sigma 1 = fst \sigma 2 \wedge lst \sigma 2 = fst \sigma 3' \wedge lst \sigma 3' = lst \gamma 1'''$
 $\wedge fst \gamma 1''' = fst \gamma 3$ **by metis**

have $ih2: !!y. (y, (s, [\alpha\text{-step}] @ \rho\text{-step}), (s, [\beta\text{-step}] @ \nu\text{-step})) \in pex r \implies peak$
 $ars y \implies \exists \sigma' \tau'. DD ars r (fst y, snd y, \sigma', \tau')$
using *IH unfolding pex-def measure-def by (auto simp: union-commute)*

from *local-decreasing-extended-imp-decreasing[OF LTb t i ih2]* **obtain** $\tau 1 \tau 2$
 $\tau 3' \delta 1'''$ **where** $taus: \{\tau 1, \tau 2, \tau 3', \delta 1'''\} \subseteq seq ars$ **and**
 $onetwo2: set (labels \tau 1) \subseteq ds r \{fst \alpha\text{-step}\} \wedge length (labels \tau 2) \leq 1 \wedge set$
 $(labels \tau 2) \subseteq \{fst \beta\text{-step}\} \wedge$
 $set (labels \tau 3') \subseteq ds r \{fst \beta\text{-step}, fst \alpha\text{-step}\} \wedge set (labels \delta 1''') \subseteq ds r \{fst$
 $\beta\text{-step}, fst \alpha\text{-step}\} \wedge$
 $snd \alpha\text{-step} = fst \tau 1 \wedge lst \tau 1 = fst \tau 2 \wedge lst \tau 2 = fst \tau 3' \wedge lst \tau 3' = lst \delta 1'''$
 $\wedge fst \delta 1''' = fst \delta 3$ **by metis**

have $\gamma 3: \gamma 3 \in conv ars$ **and** $\gamma 3m: set (labels\text{-conv } \gamma 3) \subseteq ds r \{fst \alpha\text{-step}, fst$
 $\beta\text{-step}\}$ **and**
 $\delta 3: \delta 3 \in conv ars$ **(is ?c2 ∈ -)** **and** $\delta 3m: set (labels\text{-conv } \delta 3) \subseteq ds r \{fst$
 $\beta\text{-step}, fst \alpha\text{-step}\}$ **and**
 $eq: lst\text{-conv } \gamma 3 = lst\text{-conv } \delta 3$ **using** *LT unfolding LT-def LT1-def lo-*
cal-triangle1-def LD-1'-def labels-def by auto
hence $\delta 3m: set (labels\text{-conv } \delta 3) \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\}$ **using** *in-*
sert-commute by metis

have $\delta 1''': \delta 1''' \in seq ars$ **using** *taus by auto*
have $\gamma 1''': \gamma 1''' \in seq ars$ **using** *sigmas by auto*
have $c1: (fst \delta 1''', map (Pair True) (snd \delta 1''')) \in conv ars$ **(is ?c0 ∈ -)**
using *seq-imp-conv \delta 1''' surjective-pairing by metis*
have $c11: lst \delta 1''' = lst\text{-conv } ?c0$ **using** *seq-imp-conv \delta 1''' surjective-pairing*
by metis
have $c1l: set (labels\text{-conv } ?c0) \subseteq ds r \{fst \beta\text{-step}, fst \alpha\text{-step}\}$ **using** *onetwo2*
seq-imp-conv \delta 1''' surjective-pairing by metis
hence $c1l: set (labels\text{-conv } ?c0) \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\}$ **using** *insert-commute*
by metis

have $c1: conv\text{-mirror } ?c0 \in conv ars$ **(is ?c1 ∈ -)**
 $set (labels\text{-conv } (conv\text{-mirror } ?c0)) \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\}$
 $fst (conv\text{-mirror } ?c0) = lst \tau 3'$
 $lst\text{-conv } (conv\text{-mirror } ?c0) = fst \delta 3$
using *conv-mirror[OF c1] c11 c1l c1 onetwo2 by auto*

have $c2: ?c2 \in conv ars$
 $set (labels\text{-conv } ?c2) \subseteq ds r \{fst \alpha\text{-step}, fst \beta\text{-step}\}$
 $fst ?c2 = fst \delta 3$

$lst\text{-conv } ?c2 = lst\text{-conv } \gamma3$ **using** $\delta3$ eq $\delta3m$ **by auto**

have $conv\text{-mirror } \gamma3 \in conv\text{ ars}$ (**is** $?c3 \in -$) set ($labels\text{-conv } (conv\text{-mirror } \gamma3)$)
 $= set$ ($labels\text{-conv } \gamma3$) **using** $conv\text{-mirror}[OF \ \gamma3]$ **by auto**

hence $c3: ?c3 \in conv\text{ ars}$
 set ($labels\text{-conv } ?c3$) $\subseteq ds\ r$ $\{fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\}$
 $fst\ ?c3 = lst\text{-conv } \delta3$
 $lst\text{-conv } ?c3 = fst\ \gamma1'''$ **using** $conv\text{-mirror}[OF \ \gamma3]$ eq $onetwo1\ \gamma3m$

by auto

have $c4: (fst\ \gamma1''', map\ (Pair\ True)\ (snd\ \gamma1''')) \in conv\text{ ars}$ (**is** $?c4 \in -$) set
 $(labels\text{-conv } (fst\ \gamma1''', map\ (Pair\ True)\ (snd\ \gamma1'''))) = set$ ($labels\ \gamma1'''$)
using $seq\text{-imp-conv } \gamma1'''$ $surjective\text{-pairing}$ **apply** $metis$ **using** $seq\text{-imp-conv}$
 $\gamma1'''$ $surjective\text{-pairing}$ **by** $metis$

have $c4: ?c4 \in conv\text{ ars}$
 $lst\text{-conv } ?c4 = lst\ \gamma1'''$
 set ($labels\text{-conv } ?c4$) $\subseteq ds\ r$ $\{fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\}$
using $seq\text{-imp-conv } \gamma1'''$ $surjective\text{-pairing}$ **apply** $metis$ **using** $seq\text{-imp-conv}$
 $\gamma1'''$ $surjective\text{-pairing}$ **apply** $metis$ **using** $c4(2)$ $onetwo1$ **by auto**

have $eq: lst\text{-conv } ?c1 = fst\ ?c2$ **using** $c1\ c2$ **by auto**

have $c12: (fst\ ?c1, snd\ ?c1@snd\ ?c2) \in conv\text{ ars}$ (**is** $?c12 \in -$)
 $fst\ (fst\ ?c1, snd\ ?c1@snd\ ?c2) = fst\ ?c1\ lst\text{-conv } (fst\ ?c1, snd\ ?c1@snd\ ?c2) =$
 $lst\text{-conv } ?c2$ **using** $conv\text{-concat}[OF\ c1(1)\ c2(1)\ eq]$ **by auto**

have $eq: lst\text{-conv } ?c12 = fst\ ?c3$ **using** $c12\ c3$ **by auto**

have $c123: (fst\ ?c12, snd\ ?c12@snd\ ?c3) \in conv\text{ ars}$ (**is** $?c123 \in -$)
 $fst\ (fst\ ?c12, snd\ ?c12@snd\ ?c3) = fst\ ?c12\ lst\text{-conv } (fst\ ?c12, snd\ ?c12@snd\ ?c3) =$
 $lst\text{-conv } ?c3$ **using** $conv\text{-concat}[OF\ c12(1)\ c3(1)\ eq]$ **by auto**

have $eq: lst\text{-conv } ?c123 = fst\ ?c4$ **using** $c123\ c2\ c4\ onetwo1\ onetwo2$ **apply**
 $auto$ **unfolding** $Let\text{-def}$ **using** $c3(4)$ **apply** $auto$ **by** $metis$

have $c1234: (fst\ ?c123, snd\ ?c123@snd\ ?c4) \in conv\text{ ars}$ (**is** $?c1234 \in -$)
 $fst\ (fst\ ?c123, snd\ ?c123@snd\ ?c4) = fst\ ?c123\ lst\text{-conv } (fst\ ?c123, snd\ ?c123@snd\ ?c4) =$
 $lst\text{-conv } ?c4$ **using** $conv\text{-concat}[OF\ c123(1)\ c4(1)\ eq]$ **by auto**

hence $c1234: ?c1234 \in conv\text{ ars}$ (**is** $?c1234 \in -$)
 $fst\ (?c1234) = lst\ \tau3'\ lst\text{-conv } ?c1234 = lst\ \sigma3'$ **apply** $auto$ **unfolding** $Let\text{-def}$
using $c1(3)$ **apply** $auto$ **apply** $metis$ **using** $c4(2)$ $onetwo1$ **by** $metis$

have $c12l: set$ ($labels\text{-conv } ?c12$) $\subseteq ds\ r$ $\{fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\}$ **using** $conv\text{-concat-labels}[OF\ c1(1)\ c2(1)]\ c1\ c2$ **by auto**

have $c123l: set$ ($labels\text{-conv } ?c123$) $\subseteq ds\ r$ $\{fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\}$ **using**
 $conv\text{-concat-labels}[OF\ c12(1)\ c3(1)]\ c12l\ c3$ **by auto**

have $c1234l: set$ ($labels\text{-conv } ?c1234$) $\subseteq ds\ r$ $\{fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\}$ **using**
 $conv\text{-concat-labels}[OF\ c123(1)\ c4(1)]\ c123l\ c4$ **by auto**

hence $set\text{-mset } (r|labels\text{-conv } ?c1234|) \subseteq ds\ r$ $\{fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\}$ **using**
 $submultiset\text{-implies-subset}[OF\ lexmax\text{-le-multiset}[OF\ t]]$ **by auto**

hence $set\text{-mset } (measure\text{-conv } r\ ?c1234) \subseteq dm\ r$ $\{\#fst\ \beta\text{-step},\ fst\ \alpha\text{-step}\#$
unfolding $measure\text{-conv-def}\ dm\text{-def}$ **by** ($auto\ simp: add\text{-mset-commute}$)

hence $m: set\text{-mset } (measure\text{-conv } r\ ?c1234) \subseteq dm\ r$ $\{\#fst\ \alpha\text{-step},\ fst\ \beta\text{-step}\#$

by (auto simp: add-mset-commute)

from $c1234$ m obtain ϱ where $\varrho:\varrho \in \text{conv ars}$ and $\varrho m:\text{set-mset}$ (measure-conv $r \varrho \subseteq \text{dm } r \{\# \text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\# \}$)
and $\text{eq1}:\text{fst } \varrho = \text{lst } \tau 3'$ and $\text{eq2}:\text{lst-conv } \varrho = \text{lst } \sigma 3'$ by auto

have $M:(\{\# \text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\# \}, \{\# \text{fst } \beta\text{-step}, \text{fst } \alpha\text{-step}\# \}) \in \text{mul-eq } r$ using $\text{mul-eq-reflexive add-mset-commute}$ by metis

from $\text{conv-imp-valley}[OF t IH \varrho \varrho m M]$ obtain $\tau 3'' \sigma 3''$ where
 $\tau 3'':\tau 3'' \in \text{seq ars}$ and $\sigma 3'':\sigma 3'' \in \text{seq ars}$ and $\text{eq}:\text{fst } \tau 3'' = \text{fst } \varrho \wedge \text{fst } \sigma 3'' = \text{lst-conv } \varrho \wedge \text{lst } \tau 3'' = \text{lst } \sigma 3'' \wedge$
 set-mset (measure r ($\tau 3'', \sigma 3''$)) $\subseteq \text{dm } r \{\# \text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\# \}$ apply auto by fast

have $s1:\text{set}(\text{labels } \sigma 3'') \subseteq \text{ds } r \{\text{fst } \alpha\text{-step}, \text{fst } \beta\text{-step}\}$ using eq unfolding $\text{dm-def measure-def}$ apply auto by (metis (opaque-lifting, no-types) insert-commute lexmax-set subsetD t)

have $s2:\text{set}(\text{labels } \tau 3'') \subseteq \text{ds } r \{\text{fst } \beta\text{-step}, \text{fst } \alpha\text{-step}\}$ using eq unfolding $\text{dm-def measure-def}$ apply auto by (metis (opaque-lifting, no-types) insert-commute lexmax-set subsetD t)

have $\sigma 1\text{-eq}:\text{lst}(s, [\beta\text{-step}]) = \text{fst } \sigma 1$ and $\tau 1\text{-eq}:\text{lst}(s, [\alpha\text{-step}]) = \text{fst } \tau 1$ using $\text{onetwo1 onetwo2 surjective-pairing}$ unfolding lst-def by auto

have $\text{eqn}:\text{lst } \tau 3'' = \text{lst } \sigma 3''$ and $\sigma\text{-eq}:\text{lst } \sigma 3' = \text{fst } \sigma 3''$ and $\tau\text{-eq}:\text{lst } \tau 3' = \text{fst } \tau 3''$ using eq eq1 eq2 by auto

have $\sigma 3':(\text{fst } \sigma 3', \text{snd } \sigma 3' @ \text{snd } \sigma 3'') \in \text{seq ars}$ (is $?\sigma 3 \in -$) using $\text{seq-concat}[OF - \sigma 3'' \sigma\text{-eq}]$ sigmas by blast

have $\tau 3':(\text{fst } \tau 3', \text{snd } \tau 3' @ \text{snd } \tau 3'') \in \text{seq ars}$ (is $?\tau 3 \in -$) using $\text{seq-concat}[OF - \tau 3'' \tau\text{-eq}]$ taus by blast

have $\text{fst } ?\sigma 3 = \text{lst } \sigma 2$ and $\text{fst } ?\tau 3 = \text{lst } \tau 2$ using onetwo1 onetwo2 by auto

have $\text{lst}:\text{lst } ?\sigma 3 = \text{lst } ?\tau 3$ using eqn $\sigma 3'' \sigma 3' \sigma\text{-eq } \tau 3'' \tau\text{-eq } \tau 3' \text{seq-chop}(1)$ $\text{seq-concat}(2)$ $\text{surjective-pairing}$ by metis

have $\text{local-diagram1 ars}(s, [\beta\text{-step}]) (s, [\alpha\text{-step}]) \sigma 1 \sigma 2 (\text{fst } \sigma 3', \text{snd } \sigma 3' @ \text{snd } \sigma 3'')$

using sigmas $\sigma 3'$ $\text{onetwo1 } \sigma 1\text{-eq}$ LT unfolding $\text{local-diagram1-def}$ LT-def LT1-def $\text{local-triangle1-def}$ by auto

moreover have $\text{local-diagram1 ars}(s, [\alpha\text{-step}]) (s, [\beta\text{-step}]) \tau 1 \tau 2 (\text{fst } \tau 3', \text{snd } \tau 3' @ \text{snd } \tau 3'')$

using taus $\tau 3'$ $\text{onetwo2 } \tau 1\text{-eq}$ LT unfolding $\text{local-diagram1-def}$ LT-def LT1-def $\text{local-triangle1-def}$ by auto

moreover have $\text{LD-1}' r (\text{hd}(\text{labels}(s, [\beta\text{-step}]))) (\text{hd}(\text{labels}(s, [\alpha\text{-step}]))) (\text{labels } \sigma 1) (\text{labels } \sigma 2) (\text{labels}(\text{fst } \sigma 3', \text{snd } \sigma 3' @ \text{snd } \sigma 3''))$

using $\text{onetwo1 } s1$ unfolding $\text{LD-1}'\text{-def}$ labels-def by auto

moreover have $\text{LD-1}' r (\text{hd}(\text{labels}(s, [\alpha\text{-step}]))) (\text{hd}(\text{labels}(s, [\beta\text{-step}]))) (\text{labels } \tau 1) (\text{labels } \tau 2) (\text{labels}(\text{fst } \tau 3', \text{snd } \tau 3' @ \text{snd } \tau 3''))$

using $\text{onetwo2 } s2$ unfolding $\text{LD-1}'\text{-def}$ labels-def by auto

ultimately have LDD: $\text{LDD ars } r ((s, [\beta\text{-step}]), (s, [\alpha\text{-step}]), \sigma 1, \sigma 2, ?\sigma 3, \tau 1, \tau 2, ?\tau 3)$ using lst unfolding LDD-def LDD1-def $\text{local-diagram1-def}$ by auto

from LDD-imp-DD[OF t i LDD] show $?thesis$ by auto

qed

lemma *LT-imp-D*: **assumes** $t:trans\ r$ **and** $wf\ r$ **and** $\forall p. (local\ peak\ ars\ p \longrightarrow (\exists \gamma 1\ \gamma 2\ \gamma 3\ \delta 1\ \delta 2\ \delta 3. LT\ ars\ r\ (fst\ p, snd\ p, \gamma 1, \gamma 2, \gamma 3, \delta 1, \delta 2, \delta 3)))$
and *peak ars P* **shows** $(\exists\ \sigma'\ \tau'. DD\ ars\ r\ (fst\ P, snd\ P, \sigma', \tau'))$ **proof** –
have $i: irrefl\ r$ **using** $assms(1,2)$ **acyclic-irrefl** **trancl-id** **wf-acyclic** **by** *metis*
have $wf: wf\ (pex\ r)$ **using** $wf[OF\ assms(1,2)]$.
show *?thesis* **using** $assms(4)$ **proof** (*induct rule:wf-induct-rule[OF wf]*)
case $(1\ P)$
obtain $s\ \tau\ \sigma$ **where** $decompose:P = (\tau, \sigma)$ **and** $tau:\tau \in seq\ ars$ **and** $sigma:\sigma \in seq\ ars$
and $tau-s: fst\ \tau = s$ **and** $sigma-s: fst\ \sigma = s$ **using** 1 **unfolding** *peak-def* **by** *auto*
show *?case* **proof** (*cases* $snd\ \tau$)
case *Nil* **from** *mirror-DD[OF assms(1) i trivial-DD[OF sigma]]*
show *?thesis* **using** $tau-s\ sigma-s\ Nil$ **surjective-pairing** **unfolding** *decompose* **fst-conv** **snd-conv** **DD-def** **by** *metis*
next
case (*Cons* β -*step* v -*step*)
hence $tau-dec: \tau = (s, [\beta\text{-step}]@v\text{-step})$ **apply** *auto* **using** $tau-s$ **surjective-pairing** **by** *metis*
hence $tau2: (s, [\beta\text{-step}]@v\text{-step}) \in seq\ ars$ **using** tau **by** *auto*
show *?thesis* **proof** (*cases* $snd\ \sigma$)
case *Nil* **from** *trivial-DD[OF tau]*
show *?thesis* **using** $tau-s\ sigma-s\ Nil$ **surjective-pairing** **unfolding** *decompose* **fst-conv** **snd-conv** **DD-def** **by** *metis*
next
case (*Cons* α -*step* ϱ -*step*)
hence $sigma-dec: \sigma = (s, [\alpha\text{-step}]@{\varrho}\text{-step})$ **apply** *auto* **using** $sigma-s$ **surjective-pairing** **by** *metis*
hence $sigma2: (s, [\alpha\text{-step}]@{\varrho}\text{-step}) \in seq\ ars$ **using** $sigma$ **by** *auto*

have $alpha: (s, [\alpha\text{-step}]) \in seq\ ars$ **(is** $?{\alpha} \in -$ **)**
and $rho: (lst\ (s, [\alpha\text{-step}]), {\varrho}\text{-step}) \in seq\ ars$ **(is** $?{\varrho} \in -$ **)** **using** *seq-chop[OF sigma2]* **by** *auto*
hence $alpha': (s, fst\ \alpha\text{-step}, snd\ \alpha\text{-step}) \in ars$ **by** (*metis* *seq-tail1* (2))
have $beta: (s, [\beta\text{-step}]) \in seq\ ars$ **(is** $?{\beta} \in -$ **)**
and $upsilon: (lst\ (s, [\beta\text{-step}]), v\text{-step}) \in seq\ ars$ **(is** $?{v} \in -$ **)** **using** *seq-chop[OF tau2]* **by** *auto*

have $lp: local\ peak\ ars\ (?{\beta}, ?{\alpha})$ **using** $alpha\ beta$ **unfolding** *local-peak-def* *peak-def* **by** *auto*

from this **obtain** $\gamma 1\ \gamma 2\ \gamma 3\ \delta 1\ \delta 2\ \delta 3$ **where** $LT: LT\ ars\ r\ (?{\beta}, ?{\alpha}, \gamma 1, \gamma 2, \gamma 3, \delta 1, \delta 2, \delta 3)$ **using** $assms(3)$ **apply** *auto* **by** *metis*
have $P: P = ((s, [\beta\text{-step}]@v\text{-step}), (s, [\alpha\text{-step}]@{\varrho}\text{-step}))$ **(is** $P = ?P$ **)** **using** *decompose* **unfolding** $tau-dec\ sigma-dec$ **by** *auto*
obtain $\kappa\ \mu$ **where** $D: DD\ ars\ r\ (?{\beta}, ?{\alpha}, \kappa, \mu)$ **using** *LT-imp-DD[OF t i 1(1) LT]* **unfolding** P **by** *fast*

hence *kappa*: $\kappa \in \text{seq ars}$ **and** *mu*: $\mu \in \text{seq ars}$ **unfolding** *DD-def diagram-def* **by**
auto
have *P-IH1*: *peak ars* ($?v, \kappa$) **using** *upsilon kappa D* **unfolding** *DD-def dia-*
gram-def peak-def **by** *auto*
have *beta-ne*: *labels* $? \beta \neq []$ **unfolding** *labels-def* **by** *auto*
have *dec*: *D r* (*labels* $? \beta$) (*labels* $? \alpha$) (*labels* κ) (*labels* μ) **using** *D* **unfolding**
DD-def D2-def **by** *auto*
have *x1*: ($(?v, \kappa), (\tau, ? \alpha)$) $\in \text{pex } r$ **using** *lemma3-6*[*OF assms(1) beta-ne dec*]
unfolding *pex-def measure-def decompose labels-def tau-dec* **apply** (*auto simp:*
add-mset-commute) **using** *union-commute* **by** *metis*
have (*lexmax* *r* (*labels* τ) + *lexmax* *r* (*labels* ($? \alpha$)), *lexmax* *r* (*labels* τ) + *lexmax*
r (*labels* σ)) $\in \text{mul-eq } r$ (**is** ($?l, ?r$) $\in -$)
unfolding *sigma-dec labels-def snd-conv list.map lexmax.simps diff-from-empty*
by (*simp add: lemma2-6-2-a t*)
hence ($(?v, \kappa), P$) $\in \text{pex } r$ **using** *x1* **unfolding** *sigma-s pex-def measure-def*
decompose **using** *mul-and-mul-eq-imp-mul*[*OF assms(1)*] **by** *auto*
from *this* **obtain** $\kappa' v'$ **where** *IH1*: *DD ars* *r* ($?v, \kappa, \kappa', v'$) **using** *1(1)*[*OF -*
P-IH1] **unfolding** *decompose* **by** *auto*
hence *kappa'*: $\kappa' \in \text{seq ars}$ **and** *upsilon'*: $v' \in \text{seq ars}$ **using** *D* **unfolding** *DD-def*
diagram-def **by** *auto*
have *tau'*: (*fst* $\mu, \text{snd } \mu @ (\text{snd } v')$) $\in \text{seq ars}$ (**is** $? \tau' \in -$) **using** *seq-concat(1)*[*OF*
mu upsilon'] *D IH1* **unfolding** *DD-def diagram-def* **by** *auto*
have *DIH1*: *DD ars* *r* ($\tau, ? \alpha, \kappa', ? \tau'$) **using** *lemma3-5-DD*[*OF assms(1) i D IH1*]
tau-dec **by** *auto*
hence *dec-dih1*: *D r* (*labels* τ) (*labels* $? \alpha$) (*labels* κ') (*labels* $? \tau'$) **using** *DIH1*
unfolding *DD-def D2-def* **by** *simp*

have *P-IH2*: *peak ars* ($? \tau', ? \rho$) **using** *tau' rho D* **unfolding** *DD-def diagram-def*
peak-def **by** *auto*
have *alpha-ne*: *labels* $? \alpha \neq []$ **unfolding** *labels-def* **by** *simp*
have ($(? \tau', ? \rho), P$) $\in \text{pex } r$ **using** *lemma3-6-v*[*OF assms(1) i alpha-ne dec-dih1*]
unfolding *pex-def measure-def decompose labels-def sigma-dec* **by** *auto*
from *this* **obtain** $\rho' \tau''$ **where** *IH2*: *DD ars* *r* ($? \tau', ? \rho, \rho', \tau''$) **using** *1(1)*[*OF -*
P-IH2] **by** *auto*
show *?thesis* **using** *lemma3-5-DD-v*[*OF assms(1) i DIH1 IH2*] **unfolding**
decompose fst-conv snd-conv sigma-dec **by** *fast*
qed
qed
qed
qed

definition *LD-conv* :: $'b \text{ set} \Rightarrow 'a \text{ rel} \Rightarrow \text{bool}$

where *LD-conv* *L ars* = ($\exists (r :: ('b \text{ rel})) (lrs :: ('a, 'b) \text{ lars}). (ars = \text{unlabel } lrs) \wedge$
trans *r* $\wedge \text{wf } r \wedge (\forall p. (\text{local-peak } lrs \text{ } p \longrightarrow (\exists \gamma 1 \gamma 2 \gamma 3 \delta 1 \delta 2 \delta 3. \text{LT } lrs \text{ } r (\text{fst}$
p, snd } p, \gamma 1, \gamma 2, \gamma 3, \delta 1, \delta 2, \delta 3)))))

lemma *sound-conv*: **assumes** *LD-conv* *L ars* **shows** *CR ars*

using *assms* *LT-imp-D D-imp-CR* **unfolding** *LD-conv-def* **by** *metis*

```
hide-const (open) D
hide-const (open) seq
hide-const (open) measure

hide-fact (open) split

end
```

References

- [1] V. van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.
- [2] V. van Oostrom. Confluence by decreasing diagrams – converted. In *Proc. 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320, 2008.
- [3] H. Zankl. Confluence by decreasing diagrams – formalized. In *Proc. 24th International Conference on Rewriting Techniques and Applications*, number 21 in *Leibniz International Proceedings in Informatics*, pages 352–367, 2013.