# A Comprehensive Framework for Saturation Theorem Proving

Sophie Tourret

May 29, 2023

**Abstract**

This Isabelle/HOL formalization is the companion of the technical report "A comprehensive framework for saturation theorem proving", itself companion of the eponym IJCAR 2020 paper, written by Uwe Waldmann, Sophie Tourret, Simon Robillard and Jasmin Blanchette. It verifies a framework for formal refutational completeness proofs of abstract provers that implement saturation calculi, such as ordered resolution or superposition, and allows to model entire prover architectures in such a way that the static refutational completeness of a calculus immediately implies the dynamic refutational completeness of a prover implementing the calculus using a variant of the given clause loop.

The technical report "A comprehensive framework for saturation theorem proving" is available at http://matryoshka.gforge.inria.fr/pubs/satur_report.pdf. The names of the Isabelle lemmas and theorems corresponding to the results in the report are indicated in the margin of the report.

## Contents

# 1 Calculi Based on a Redundancy Criterion

This section introduces the most basic notions upon which the framework is built: consequence relations and inference systems. It also defines the notion of a family of consequence relations and of redundancy criteria. This corresponds to sections 2.1 and 2.2 of the report.

**theory** *Calculus*
**imports**
*Ordered-Resolution-Prover.Lazy-List-Liminf*
*Ordered-Resolution-Prover.Lazy-List-Chain*
**begin**

## 1.1 Consequence Relations

**locale** *consequence-relation* =
**fixes**
*Bot* :: *'f set* **and**
*entails* :: *'f set* ⇒ *'f set* ⇒ *bool* (**infix** ⊨ *50*)
**assumes**
*bot-not-empty*: *Bot* ≠ {} **and**
*bot-entails-all*: *B* ∈ *Bot* ⟹ {*B*} ⊨ *N1* **and**
*subset-entailed*: *N2* ⊆ *N1* ⟹ *N1* ⊨ *N2* **and**
*all-formulas-entailed*: (∀ *C* ∈ *N2*. *N1* ⊨ {*C*}) ⟹ *N1* ⊨ *N2* **and**
*entails-trans*[*trans*]: *N1* ⊨ *N2* ⟹ *N2* ⊨ *N3* ⟹ *N1* ⊨ *N3*
**begin**

**lemma** *entail-set-all-formulas*: *N1* ⊨ *N2* ⟷ (∀ *C* ∈ *N2*. *N1* ⊨ {*C*})
⟨*proof*⟩

**lemma** *entail-union*: *N* ⊨ *N1* ∧ *N* ⊨ *N2* ⟷ *N* ⊨ *N1* ∪ *N2*
⟨*proof*⟩

**lemma** *entail-unions*: (∀ *i* ∈ *I*. *N* ⊨ *Ni i*) ⟷ *N* ⊨ ⋃ (*Ni* ‘ *I*)
⟨*proof*⟩

**lemma** *entail-all-bot*: (∃ *B* ∈ *Bot*. *N* ⊨ {*B*}) ⟹ ∀ *B'* ∈ *Bot*. *N* ⊨ {*B'*}
⟨*proof*⟩

**lemma** *entails-trans-strong*: *N1* ⊨ *N2* ⟹ *N1* ∪ *N2* ⊨ *N3* ⟹ *N1* ⊨ *N3*
⟨*proof*⟩

**end**

## 1.2 Families of Consequence Relations

**locale** *consequence-relation-family* =
**fixes**
*Bot* :: *'f set* **and**
*Q* :: *'q set* **and**
*entails-q* :: *'q* ⇒ *'f set* ⇒ *'f set* ⇒ *bool*

**assumes**
*Q-nonempty*: $Q \neq \{\}$ **and**
*q-cons-rel*: $\forall\, q \in Q.$ *consequence-relation Bot* (*entails-q q*)
**begin**

**lemma** *bot-not-empty*: $Bot \neq \{\}$
⟨*proof*⟩

**definition** *entails* :: *'f set* ⇒ *'f set* ⇒ *bool* (**infix** $\models Q$ *50*) **where**
$N1 \models Q\ N2 \longleftrightarrow (\forall\, q \in Q.$ *entails-q q N1 N2*)

**lemma** *intersect-cons-rel-family*: *consequence-relation Bot entails*
⟨*proof*⟩

**end**

## 1.3 Inference Systems

**datatype** *'f inference* =
*Infer* (*prems-of*: *'f list*) (*concl-of*: *'f*)

**locale** *inference-system* =
**fixes**
*Inf* :: ‹*'f inference set*›
**begin**

**definition** *Inf-from* :: *'f set* ⇒ *'f inference set* **where**
*Inf-from N* = $\{\iota \in Inf.$ *set* (*prems-of* $\iota$) $\subseteq N\}$

**definition** *Inf-between* :: *'f set* ⇒ *'f set* ⇒ *'f inference set* **where**
*Inf-between N M* = *Inf-from* ($N \cup M$) − *Inf-from* ($N - M$)

**lemma** *Inf-if-Inf-from*: $\iota \in$ *Inf-from N* ⟹ $\iota \in Inf$
⟨*proof*⟩

**lemma** *Inf-if-Inf-between*: $\iota \in$ *Inf-between N M* ⟹ $\iota \in Inf$
⟨*proof*⟩

**lemma** *Inf-between-alt*:
*Inf-between N M* = $\{\iota \in Inf.\ \iota \in$ *Inf-from* ($N \cup M$) ∧ *set* (*prems-of* $\iota$) ∩ $M \neq \{\}\}$
⟨*proof*⟩

**lemma** *Inf-from-mono*: $N \subseteq N'$ ⟹ *Inf-from N* ⊆ *Inf-from N'*
⟨*proof*⟩

**lemma** *Inf-between-mono*: $N \subseteq N'$ ⟹ $M \subseteq M'$ ⟹ *Inf-between N M* ⊆ *Inf-between N' M'*
⟨*proof*⟩

**end**

## 1.4 Families of Inference Systems

**locale** *inference-system-family* =
**fixes**
*Q* :: *'q set* **and**

*Inf-q* :: *'q* ⇒ *'f inference set*
**assumes**
*Q-nonempty*: $Q \neq \{\}$
**begin**

**definition** *Inf-from-q* :: *'q* ⇒ *'f set* ⇒ *'f inference set* **where**
*Inf-from-q q* = *inference-system.Inf-from* (*Inf-q q*)

**definition** *Inf-between-q* :: *'q* ⇒ *'f set* ⇒ *'f set* ⇒ *'f inference set* **where**
*Inf-between-q q* = *inference-system.Inf-between* (*Inf-q q*)

**lemma** *Inf-between-q-alt*:
*Inf-between-q q N M* = $\{\iota \in$ *Inf-q q*. $\iota \in$ *Inf-from-q q* $(N \cup M) \wedge$ *set* (*prems-of* $\iota) \cap M \neq \{\}\}$
⟨*proof*⟩

**end**

## 1.5 Calculi Based on a Single Redundancy Criterion

**locale** *calculus* = *inference-system Inf* + *consequence-relation Bot entails*
**for**
*Bot* :: *'f set* **and**
*Inf* :: ‹*'f inference set*› **and**
*entails* :: *'f set* ⇒ *'f set* ⇒ *bool* (**infix** $\models$ *50*)
+ **fixes**
*Red-I* :: *'f set* ⇒ *'f inference set* **and**
*Red-F* :: *'f set* ⇒ *'f set*
**assumes**
*Red-I-to-Inf*: *Red-I N* $\subseteq$ *Inf* **and**
*Red-F-Bot*: $B \in$ *Bot* $\Longrightarrow N \models \{B\} \Longrightarrow N -$ *Red-F N* $\models \{B\}$ **and**
*Red-F-of-subset*: $N \subseteq N' \Longrightarrow$ *Red-F N* $\subseteq$ *Red-F N'* **and**
*Red-I-of-subset*: $N \subseteq N' \Longrightarrow$ *Red-I N* $\subseteq$ *Red-I N'* **and**
*Red-F-of-Red-F-subset*: $N' \subseteq$ *Red-F N* $\Longrightarrow$ *Red-F N* $\subseteq$ *Red-F* $(N - N')$ **and**
*Red-I-of-Red-F-subset*: $N' \subseteq$ *Red-F N* $\Longrightarrow$ *Red-I N* $\subseteq$ *Red-I* $(N - N')$ **and**
*Red-I-of-Inf-to-N*: $\iota \in$ *Inf* $\Longrightarrow$ *concl-of* $\iota \in N \Longrightarrow \iota \in$ *Red-I N*
**begin**

**lemma** *Red-I-of-Inf-to-N-subset*: $\{\iota \in$ *Inf*. *concl-of* $\iota \in N\} \subseteq$ *Red-I N*
⟨*proof*⟩

**lemma** *red-concl-to-red-inf*:
**assumes**
*i-in*: $\iota \in$ *Inf* **and**
*concl*: *concl-of* $\iota \in$ *Red-F N*
**shows** $\iota \in$ *Red-I N*
⟨*proof*⟩

**definition** *saturated* :: *'f set* ⇒ *bool* **where**
*saturated N* $\longleftrightarrow$ *Inf-from N* $\subseteq$ *Red-I N*

**definition** *reduc-saturated* :: *'f set* ⇒ *bool* **where**
*reduc-saturated N* $\longleftrightarrow$ *Inf-from* $(N -$ *Red-F N*$) \subseteq$ *Red-I N*

**lemma** *Red-I-without-red-F*:
*Red-I* $(N -$ *Red-F N*$) =$ *Red-I N*

⟨*proof*⟩

**lemma** *saturated-without-red-F*:
**assumes** *saturated*: *saturated N*
**shows** *saturated* (*N* − *Red-F N*)
⟨*proof*⟩

**definition** *fair* :: *'f set llist* ⇒ *bool* **where**
*fair Ns* ⟷ *Inf-from* (*Liminf-llist Ns*) ⊆ *Sup-llist* (*lmap Red-I Ns*)

**inductive** *derive* :: *'f set* ⇒ *'f set* ⇒ *bool* (**infix** ▷ *50*) **where**
*derive*: *M* − *N* ⊆ *Red-F N* ⟹ *M* ▷ *N*

**lemma** *gt-Max-notin*: ‹*finite A* ⟹ *A* ≠ {} ⟹ *x* > *Max A* ⟹ *x* ∉ *A*› ⟨*proof*⟩

**lemma** *equiv-Sup-Liminf*:
**assumes**
*in-Sup*: *C* ∈ *Sup-llist Ns* **and**
*not-in-Liminf*: *C* ∉ *Liminf-llist Ns*
**shows**
∃ *i* ∈ {*i*. *enat* (*Suc i*) < *llength Ns*}. *C* ∈ *lnth Ns i* − *lnth Ns* (*Suc i*)
⟨*proof*⟩

**lemma** *Red-in-Sup*:
**assumes** *deriv*: *chain* (▷) *Ns*
**shows** *Sup-llist Ns* − *Liminf-llist Ns* ⊆ *Red-F* (*Sup-llist Ns*)
⟨*proof*⟩

**lemma** *Red-I-subset-Liminf*:
**assumes** *deriv*: ‹*chain* (▷) *Ns*› **and**
*i*: ‹*enat i* < *llength Ns*›
**shows** ‹*Red-I* (*lnth Ns i*) ⊆ *Red-I* (*Liminf-llist Ns*)›
⟨*proof*⟩

**lemma** *Red-F-subset-Liminf*:
**assumes** *deriv*: ‹*chain* (▷) *Ns*› **and**
*i*: ‹*enat i* < *llength Ns*›
**shows** ‹*Red-F* (*lnth Ns i*) ⊆ *Red-F* (*Liminf-llist Ns*)›
⟨*proof*⟩

**lemma** *i-in-Liminf-or-Red-F*:
**assumes**
*deriv*: ‹*chain* (▷) *Ns*› **and**
*i*: ‹*enat i* < *llength Ns*›
**shows** ‹*lnth Ns i* ⊆ *Red-F* (*Liminf-llist Ns*) ∪ *Liminf-llist Ns*›
⟨*proof*⟩

**lemma** *fair-implies-Liminf-saturated*:
**assumes**
*deriv*: ‹*chain* (▷) *Ns*› **and**

*fair*: ‹*fair Ns*›
**shows** ‹*saturated* (*Liminf-llist Ns*)›
⟨*proof*⟩

**end**

**locale** *statically-complete-calculus* = *calculus* +
**assumes** *statically-complete*: $B \in Bot \Longrightarrow saturated\ N \Longrightarrow N \models \{B\} \Longrightarrow \exists\, B' \in Bot.\ B' \in N$
**begin**

**lemma** *dynamically-complete-Liminf*:
**fixes** *B Ns*
**assumes**
*bot-elem*: ‹$B \in Bot$› **and**
*deriv*: ‹*chain* (⊳) *Ns*› **and**
*fair*: ‹*fair Ns*› **and**
*unsat*: ‹*lhd Ns* ⊨ {*B*}›
**shows** ‹$\exists\, B' \in Bot.\ B' \in$ *Liminf-llist Ns*›
⟨*proof*⟩

**end**

**locale** *dynamically-complete-calculus* = *calculus* +
**assumes**
*dynamically-complete*: $B \in Bot \Longrightarrow chain\ (\rhd)\ Ns \Longrightarrow fair\ Ns \Longrightarrow lhd\ Ns \models \{B\} \Longrightarrow$
$\exists\, i \in \{i.\ enat\ i < llength\ Ns\}.\ \exists\, B' \in Bot.\ B' \in lnth\ Ns\ i$
**begin**

**sublocale** *statically-complete-calculus*
⟨*proof*⟩

**end**

**sublocale** *statically-complete-calculus* ⊆ *dynamically-complete-calculus*
⟨*proof*⟩

**end**

# 2 Calculi Based on the Intersection of Redundancy Criteria

In this section, section 2.3 of the report is covered, on calculi equipped with a family of redundancy criteria.

**theory** *Intersection-Calculus*
**imports**
*Calculus*
*Ordered-Resolution-Prover.Lazy-List-Liminf*
*Ordered-Resolution-Prover.Lazy-List-Chain*
**begin**

## 2.1 Calculi with a Family of Redundancy Criteria

**locale** *intersection-calculus* =

*inference-system Inf* + *consequence-relation-family Bot Q entails-q*
**for**
*Bot* :: *'f set* **and**
*Inf* :: ‹*'f inference set*› **and**
*Q* :: *'q set* **and**
*entails-q* :: *'q* ⇒ *'f set* ⇒ *'f set* ⇒ *bool*
+ **fixes**
*Red-I-q* :: *'q* ⇒ *'f set* ⇒ *'f inference set* **and**
*Red-F-q* :: *'q* ⇒ *'f set* ⇒ *'f set*
**assumes**
*Q-nonempty*: *Q* ≠ {} **and**
*all-red-crit*: ∀ *q* ∈ *Q*. *calculus Bot Inf* (*entails-q q*) (*Red-I-q q*) (*Red-F-q q*)
**begin**

**definition** *Red-I* :: *'f set* ⇒ *'f inference set* **where**
*Red-I N* = (⋂ *q* ∈ *Q*. *Red-I-q q N*)

**definition** *Red-F* :: *'f set* ⇒ *'f set* **where**
*Red-F N* = (⋂ *q* ∈ *Q*. *Red-F-q q N*)

**sublocale** *calculus Bot Inf entails Red-I Red-F*
⟨*proof*⟩

**lemma** *sat-int-to-sat-q*: *calculus.saturated Inf Red-I N* ⟷
(∀ *qi* ∈ *Q*. *calculus.saturated Inf* (*Red-I-q qi*) *N*) **for** *N*
⟨*proof*⟩

**lemma** *stat-ref-comp-from-bot-in-sat*:
(∀ *N*. *calculus.saturated Inf Red-I N* ∧ (∀ *B* ∈ *Bot*. *B* ∉ *N*) ⟶
(∃ *B* ∈ *Bot*. ∃ *qi* ∈ *Q*. ¬ *entails-q qi N* {*B*})) ⟹
*statically-complete-calculus Bot Inf entails Red-I Red-F*
⟨*proof*⟩

**end**

**end**

# 3 Variations on a Theme

In this section, section 2.4 of the report is covered, demonstrating that various notions of redundancy are equivalent.

**theory** *Calculus-Variations*
**imports** *Calculus*
**begin**

**locale** *reduced-calculus* = *calculus Bot Inf entails Red-I Red-F*
**for**
*Bot* :: *'f set* **and**
*Inf* :: ‹*'f inference set*› **and**
*entails* :: *'f set* ⇒ *'f set* ⇒ *bool* (**infix** ⊨ *50*) **and**
*Red-I* :: *'f set* ⇒ *'f inference set* **and**

*Red-F* :: *'f set* ⇒ *'f set*
+ **assumes**
*inf-in-red-inf*: *Inf-between UNIV* (*Red-F N*) ⊆ *Red-I N*
**begin**

**lemma** *sat-eq-reduc-sat*: *saturated N* ⟷ *reduc-saturated N*
⟨*proof*⟩

**end**

**locale** *reducedly-statically-complete-calculus* = *calculus* +
**assumes** *reducedly-statically-complete*:
*B* ∈ *Bot* ⟹ *reduc-saturated N* ⟹ *N* ⊨ {*B*} ⟹ ∃ *B'*∈*Bot*. *B'* ∈ *N*

**locale** *reducedly-statically-complete-reduced-calculus* = *reduced-calculus* +
**assumes** *reducedly-statically-complete*:
*B* ∈ *Bot* ⟹ *reduc-saturated N* ⟹ *N* ⊨ {*B*} ⟹ ∃ *B'*∈*Bot*. *B'* ∈ *N*
**begin**

**sublocale** *reducedly-statically-complete-calculus*
⟨*proof*⟩

**sublocale** *statically-complete-calculus*
⟨*proof*⟩

**end**

**context** *reduced-calculus*
**begin**

**lemma** *stat-ref-comp-imp-red-stat-ref-comp*:
*statically-complete-calculus Bot Inf entails Red-I Red-F* ⟹
*reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F*
⟨*proof*⟩

**end**

**context** *calculus*
**begin**

**definition** *Red-Red-I* :: *'f set* ⇒ *'f inference set* **where**
*Red-Red-I N* = *Red-I N* ∪ *Inf-between UNIV* (*Red-F N*)

**lemma** *reduced-calc-is-calc*: *calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**lemma** *inf-subs-reduced-red-inf*: *Inf-between UNIV* (*Red-F N*) ⊆ *Red-Red-I N*
⟨*proof*⟩

The following is a lemma and not a sublocale as was previously used in similar cases. Here, a sublocale cannot be used because it would create an infinitely descending chain of sublocales.

**lemma** *reduc-calc*: *reduced-calculus Bot Inf entails Red-Red-I Red-F*

⟨*proof*⟩

**interpretation** *reduc-calc*: *reduced-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**lemma** *sat-imp-red-calc-sat*: *saturated N* ⟹ *reduc-calc.saturated N*
⟨*proof*⟩

**lemma** *red-sat-eq-red-calc-sat*: *reduc-saturated N* ⟷ *reduc-calc.saturated N*
⟨*proof*⟩

**lemma** *red-sat-eq-sat*: *reduc-saturated N* ⟷ *saturated* (*N* − *Red-F N*)
⟨*proof*⟩

**theorem** *stat-is-stat-red*: *statically-complete-calculus Bot Inf entails Red-I Red-F* ⟷
*statically-complete-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**theorem** *red-stat-red-is-stat-red*:
*reducedly-statically-complete-calculus Bot Inf entails Red-Red-I Red-F* ⟷
*statically-complete-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**theorem** *red-stat-is-stat-red*:
*reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F* ⟷
*statically-complete-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**lemma** *sup-red-f-in-red-liminf*:
*chain derive Ns* ⟹ *Sup-llist* (*lmap Red-F Ns*) ⊆ *Red-F* (*Liminf-llist Ns*)
⟨*proof*⟩

**lemma** *sup-red-inf-in-red-liminf*:
*chain derive Ns* ⟹ *Sup-llist* (*lmap Red-I Ns*) ⊆ *Red-I* (*Liminf-llist Ns*)
⟨*proof*⟩

**definition** *reduc-fair* :: *'f set llist* ⇒ *bool* **where**
*reduc-fair Ns* ⟷
*Inf-from* (*Liminf-llist Ns* − *Sup-llist* (*lmap Red-F Ns*)) ⊆ *Sup-llist* (*lmap Red-I Ns*)

**lemma** *reduc-fair-imp-Liminf-reduc-sat*:
*chain derive Ns* ⟹ *reduc-fair Ns* ⟹ *reduc-saturated* (*Liminf-llist Ns*)
⟨*proof*⟩

**end**

**locale** *reducedly-dynamically-complete-calculus* = *calculus* +
**assumes**

*reducedly-dynamically-complete*: $B \in Bot \Longrightarrow$ *chain derive Ns* $\Longrightarrow$ *reduc-fair Ns* $\Longrightarrow$
*lhd Ns* $\models \{B\} \Longrightarrow \exists\, i \in \{i.\ enat\ i < llength\ Ns\}.\ \exists\ B' \in Bot.\ B' \in lnth\ Ns\ i$
**begin**

**sublocale** *reducedly-statically-complete-calculus*
⟨*proof*⟩

**end**

**sublocale** *reducedly-statically-complete-calculus* $\subseteq$ *reducedly-dynamically-complete-calculus*
⟨*proof*⟩

**context** *calculus*
**begin**

**lemma** *dyn-equiv-stat*: *dynamically-complete-calculus Bot Inf entails Red-I Red-F* =
*statically-complete-calculus Bot Inf entails Red-I Red-F*
⟨*proof*⟩

**lemma** *red-dyn-equiv-red-stat*:
*reducedly-dynamically-complete-calculus Bot Inf entails Red-I Red-F* =
*reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F*
⟨*proof*⟩

**interpretation** *reduc-calc*: *reduced-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**theorem** *dyn-ref-eq-dyn-ref-red*:
*dynamically-complete-calculus Bot Inf entails Red-I Red-F* $\longleftrightarrow$
*dynamically-complete-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**theorem** *red-dyn-ref-red-eq-dyn-ref-red*:
*reducedly-dynamically-complete-calculus Bot Inf entails Red-Red-I Red-F* $\longleftrightarrow$
*dynamically-complete-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**theorem** *red-dyn-ref-eq-dyn-ref-red*:
*reducedly-dynamically-complete-calculus Bot Inf entails Red-I Red-F* $\longleftrightarrow$
*dynamically-complete-calculus Bot Inf entails Red-Red-I Red-F*
⟨*proof*⟩

**end**

**end**

# 4 Lifting to Non-ground Calculi

The section 3.1 to 3.3 of the report are covered by the current section. Various forms of lifting are proven correct. These allow to obtain the dynamic refutational completeness of a non-ground calculus from the static refutational completeness of its ground counterpart.

**theory** *Lifting-to-Non-Ground-Calculi*
**imports**
*Intersection-Calculus*
*Calculus-Variations*
*Well-Quasi-Orders.Minimal-Elements*
**begin**

## 4.1 Standard Lifting

**locale** *standard-lifting* = *inference-system Inf-F* +
*ground*: *calculus Bot-G Inf-G entails-G Red-I-G Red-F-G*
**for**
*Inf-F* :: ‹*'f inference set*› **and**
*Bot-G* :: ‹*'g set*› **and**
*Inf-G* :: ‹*'g inference set*› **and**
*entails-G* :: ‹*'g set* ⇒ *'g set* ⇒ *bool*› (**infix** ⊨*G* *50*) **and**
*Red-I-G* :: ‹*'g set* ⇒ *'g inference set*› **and**
*Red-F-G* :: ‹*'g set* ⇒ *'g set*›
+ **fixes**
*Bot-F* :: ‹*'f set*› **and**
*𝒢-F* :: ‹*'f* ⇒ *'g set*› **and**
*𝒢-I* :: ‹*'f inference* ⇒ *'g inference set option*›
**assumes**
*Bot-F-not-empty*: *Bot-F* ≠ {} **and**
*Bot-map-not-empty*: ‹*B* ∈ *Bot-F* ⟹ *𝒢-F B* ≠ {}› **and**
*Bot-map*: ‹*B* ∈ *Bot-F* ⟹ *𝒢-F B* ⊆ *Bot-G*› **and**
*Bot-cond*: ‹*𝒢-F C* ∩ *Bot-G* ≠ {} ⟶ *C* ∈ *Bot-F*› **and**
*inf-map*: ‹*ι* ∈ *Inf-F* ⟹ *𝒢-I ι* ≠ *None* ⟹ *the* (*𝒢-I ι*) ⊆ *Red-I-G* (*𝒢-F* (*concl-of ι*))›
**begin**

**abbreviation** *𝒢-Fset* :: ‹*'f set* ⇒ *'g set*› **where**
‹*𝒢-Fset N* ≡ ⋃ (*𝒢-F* ‘ *N*)›

**lemma** *𝒢-subset*: ‹*N1* ⊆ *N2* ⟹ *𝒢-Fset N1* ⊆ *𝒢-Fset N2*› ⟨*proof*⟩

**abbreviation** *entails-𝒢* :: ‹*'f set* ⇒ *'f set* ⇒ *bool*› (**infix** ⊨*𝒢* *50*) **where**
‹*N1* ⊨*𝒢* *N2* ≡ *𝒢-Fset N1* ⊨*G* *𝒢-Fset N2*›

**lemma** *subs-Bot-G-entails*:
**assumes**
*not-empty*: ‹*sB* ≠ {}› **and**
*in-bot*: ‹*sB* ⊆ *Bot-G*›
**shows** ‹*sB* ⊨*G* *N*›
⟨*proof*⟩

**sublocale** *consequence-relation Bot-F entails-𝒢*
⟨*proof*⟩

**definition** *Red-I-𝒢* :: *'f set* ⇒ *'f inference set* **where**
‹*Red-I-𝒢 N* = {*ι* ∈ *Inf-F*. (*𝒢-I ι* ≠ *None* ∧ *the* (*𝒢-I ι*) ⊆ *Red-I-G* (*𝒢-Fset N*))
∨ (*𝒢-I ι* = *None* ∧ *𝒢-F* (*concl-of ι*) ⊆ *𝒢-Fset N* ∪ *Red-F-G* (*𝒢-Fset N*))}›

**definition** *Red-F-𝒢* :: *'f set* ⇒ *'f set* **where**
‹*Red-F-𝒢 N* = {*C*. ∀ *D* ∈ *𝒢-F C*. *D* ∈ *Red-F-G* (*𝒢-Fset N*)}›
**end**

## 4.2 Strong Standard Lifting

**locale** *strong-standard-lifting* = *inference-system Inf-F* +
  *ground*: *calculus Bot-G Inf-G entails-G Red-I-G Red-F-G*
  **for**
    *Inf-F* :: ‹*'f inference set*› **and**
    *Bot-G* :: ‹*'g set*› **and**
    *Inf-G* :: ‹*'g inference set*› **and**
    *entails-G* :: ‹*'g set* ⇒ *'g set* ⇒ *bool*› (**infix** ⊨*G* *50*) **and**
    *Red-I-G* :: ‹*'g set* ⇒ *'g inference set*› **and**
    *Red-F-G* :: ‹*'g set* ⇒ *'g set*›
  + **fixes**
    *Bot-F* :: ‹*'f set*› **and**
    *𝒢-F* :: ‹*'f* ⇒ *'g set*› **and**
    *𝒢-I* :: ‹*'f inference* ⇒ *'g inference set option*›
  **assumes**
    *Bot-F-not-empty*: *Bot-F* ≠ {} **and**
    *Bot-map-not-empty*: ‹*B* ∈ *Bot-F* ⟹ *𝒢-F B* ≠ {}› **and**
    *Bot-map*: ‹*B* ∈ *Bot-F* ⟹ *𝒢-F B* ⊆ *Bot-G*› **and**
    *Bot-cond*: ‹*𝒢-F C* ∩ *Bot-G* ≠ {} ⟶ *C* ∈ *Bot-F*› **and**
    *strong-inf-map*: ‹*ι* ∈ *Inf-F* ⟹ *𝒢-I ι* ≠ *None* ⟹ *concl-of* ‘ (*the* (*𝒢-I ι*)) ⊆ (*𝒢-F* (*concl-of ι*))› **and**
    *inf-map-in-Inf*: ‹*ι* ∈ *Inf-F* ⟹ *𝒢-I ι* ≠ *None* ⟹ *the* (*𝒢-I ι*) ⊆ *Inf-G*›
**begin**

**sublocale** *standard-lifting Inf-F Bot-G Inf-G* (⊨*G*) *Red-I-G Red-F-G Bot-F 𝒢-F 𝒢-I*
⟨*proof*⟩

**end**

## 4.3 Lifting with a Family of Tiebreaker Orderings

**locale** *tiebreaker-lifting* =
  *empty-ord?*: *standard-lifting Inf-F Bot-G Inf-G entails-G Red-I-G Red-F-G Bot-F 𝒢-F 𝒢-I*
  **for**
    *Bot-F* :: ‹*'f set*› **and**
    *Inf-F* :: ‹*'f inference set*› **and**
    *Bot-G* :: ‹*'g set*› **and**
    *entails-G* :: ‹*'g set* ⇒ *'g set* ⇒ *bool*› (**infix** ⊨*G* *50*) **and**
    *Inf-G* :: ‹*'g inference set*› **and**
    *Red-I-G* :: ‹*'g set* ⇒ *'g inference set*› **and**
    *Red-F-G* :: ‹*'g set* ⇒ *'g set*› **and**
    *𝒢-F* :: *'f* ⇒ *'g set* **and**
    *𝒢-I* :: *'f inference* ⇒ *'g inference set option*
  + **fixes**
    *Prec-F-g* :: ‹*'g* ⇒ *'f* ⇒ *'f* ⇒ *bool*›
  **assumes**
    *all-wf*: *minimal-element* (*Prec-F-g g*) *UNIV*
**begin**

**definition** *Red-F-𝒢* :: *'f set* ⇒ *'f set* **where**
  ‹*Red-F-𝒢 N* = {*C*. ∀ *D* ∈ *𝒢-F C*. *D* ∈ *Red-F-G* (*𝒢-Fset N*) ∨ (∃ *E* ∈ *N*. *Prec-F-g D E C* ∧ *D* ∈ *𝒢-F E*)}›

**lemma** *Prec-trans*:
  **assumes**
    ‹*Prec-F-g D A B*› **and**

‹*Prec-F-g D B C*›
**shows**
‹*Prec-F-g D A C*›
⟨*proof*⟩

**lemma** *prop-nested-in-set*: $D \in P\ C \Longrightarrow C \in \{C.\ \forall D \in P\ C.\ A\ D \vee B\ C\ D\} \Longrightarrow A\ D \vee B\ C\ D$
⟨*proof*⟩

**lemma** *Red-F-$\mathcal{G}$-equiv-def*:
‹*Red-F-$\mathcal{G}$ N* = {*C*. ∀ *Di* ∈ $\mathcal{G}$-*F C*. *Di* ∈ *Red-F-G* ($\mathcal{G}$-*Fset N*) ∨
(∃ *E* ∈ (*N* − *Red-F-$\mathcal{G}$ N*). *Prec-F-g Di E C* ∧ *Di* ∈ $\mathcal{G}$-*F E*)}›
⟨*proof*⟩

**lemma** *not-red-map-in-map-not-red*: ‹$\mathcal{G}$-*Fset N* − *Red-F-G* ($\mathcal{G}$-*Fset N*) ⊆ $\mathcal{G}$-*Fset* (*N* − *Red-F-$\mathcal{G}$ N*)›
⟨*proof*⟩

**lemma** *Red-F-Bot-F*: ‹*B* ∈ *Bot-F* ⟹ *N* ⊨$\mathcal{G}$ {*B*} ⟹ *N* − *Red-F-$\mathcal{G}$ N* ⊨$\mathcal{G}$ {*B*}›
⟨*proof*⟩

**lemma** *Red-F-of-subset-F*: ‹*N* ⊆ *N'* ⟹ *Red-F-$\mathcal{G}$ N* ⊆ *Red-F-$\mathcal{G}$ N'*›
⟨*proof*⟩

**lemma** *Red-I-of-subset-F*: ‹*N* ⊆ *N'* ⟹ *Red-I-$\mathcal{G}$ N* ⊆ *Red-I-$\mathcal{G}$ N'*›
⟨*proof*⟩

**lemma** *Red-F-of-Red-F-subset-F*: ‹*N'* ⊆ *Red-F-$\mathcal{G}$ N* ⟹ *Red-F-$\mathcal{G}$ N* ⊆ *Red-F-$\mathcal{G}$* (*N* − *N'*)›
⟨*proof*⟩

**lemma** *Red-I-of-Red-F-subset-F*: ‹*N'* ⊆ *Red-F-$\mathcal{G}$ N* ⟹ *Red-I-$\mathcal{G}$ N* ⊆ *Red-I-$\mathcal{G}$* (*N* − *N'*) ›
⟨*proof*⟩

**lemma** *Red-I-of-Inf-to-N-F*:
**assumes**
*i-in*: ‹*ι* ∈ *Inf-F*› **and**
*concl-i-in*: ‹*concl-of ι* ∈ *N*›
**shows**
‹*ι* ∈ *Red-I-$\mathcal{G}$ N* ›
⟨*proof*⟩

**sublocale** *calculus Bot-F Inf-F entails-$\mathcal{G}$ Red-I-$\mathcal{G}$ Red-F-$\mathcal{G}$*
⟨*proof*⟩

**end**

**lemma** *wf-empty-rel*: *minimal-element* (λ- -. *False*) *UNIV*
⟨*proof*⟩

**lemma** *standard-empty-tiebreaker-equiv*: *standard-lifting Inf-F Bot-G Inf-G entails-G Red-I-G Red-F-G Bot-F* $\mathcal{G}$*-F* $\mathcal{G}$*-I* = *tiebreaker-lifting Bot-F Inf-F Bot-G entails-G Inf-G Red-I-G Red-F-G* $\mathcal{G}$*-F* $\mathcal{G}$*-I* ($\lambda g\ C\ C'$. *False*)
⟨*proof*⟩

**context** *standard-lifting*
**begin**

**interpretation** *empt-ord*: *tiebreaker-lifting Bot-F Inf-F Bot-G entails-G Inf-G Red-I-G Red-F-G* $\mathcal{G}$*-F* $\mathcal{G}$*-I* $\lambda g\ C\ C'$. *False*
⟨*proof*⟩

**lemma** *red-f-equiv*: *empt-ord.Red-F-*$\mathcal{G}$ = *Red-F-*$\mathcal{G}$
⟨*proof*⟩

**sublocale** *calc?*: *calculus Bot-F Inf-F entails-*$\mathcal{G}$ *Red-I-*$\mathcal{G}$ *Red-F-*$\mathcal{G}$
⟨*proof*⟩

**lemma** *grounded-inf-in-ground-inf*: $\iota \in$ *Inf-F* $\Longrightarrow$ $\mathcal{G}$*-I* $\iota \neq$ *None* $\Longrightarrow$ *the* ($\mathcal{G}$*-I* $\iota$) $\subseteq$ *Inf-G*
⟨*proof*⟩

**abbreviation** *ground-Inf-overapproximated* :: *'f set* ⇒ *bool* **where**
*ground-Inf-overapproximated N* ≡ *ground.Inf-from* ($\mathcal{G}$*-Fset N*)
$\subseteq \{\iota.\ \exists \iota' \in$ *Inf-from N*. $\mathcal{G}$*-I* $\iota' \neq$ *None* $\wedge\ \iota \in$ *the* ($\mathcal{G}$*-I* $\iota'$)$\}$ ∪ *Red-I-G* ($\mathcal{G}$*-Fset N*)

**lemma** *sat-inf-imp-ground-red*:
**assumes**
*saturated N* **and**
$\iota' \in$ *Inf-from N* **and**
$\mathcal{G}$*-I* $\iota' \neq$ *None* $\wedge\ \iota \in$ *the* ($\mathcal{G}$*-I* $\iota'$)
**shows** $\iota \in$ *Red-I-G* ($\mathcal{G}$*-Fset N*)
⟨*proof*⟩

**lemma** *sat-imp-ground-sat*:
*saturated N* $\Longrightarrow$ *ground-Inf-overapproximated N* $\Longrightarrow$ *ground.saturated* ($\mathcal{G}$*-Fset N*)
⟨*proof*⟩

**theorem** *stat-ref-comp-to-non-ground*:
**assumes**
*stat-ref-G*: *statically-complete-calculus Bot-G Inf-G entails-G Red-I-G Red-F-G* **and**
*sat-n-imp*: $\bigwedge N$. *saturated N* $\Longrightarrow$ *ground-Inf-overapproximated N*
**shows**
*statically-complete-calculus Bot-F Inf-F entails-*$\mathcal{G}$ *Red-I-*$\mathcal{G}$ *Red-F-*$\mathcal{G}$
⟨*proof*⟩

**end**

**context** *tiebreaker-lifting*
**begin**

**lemma** *saturated-empty-order-equiv-saturated*:
*saturated N = calc.saturated N*
⟨*proof*⟩

**lemma** *static-empty-order-equiv-static*:
*statically-complete-calculus Bot-F Inf-F entails-𝒢 Red-I-𝒢 Red-F-𝒢 =*
*statically-complete-calculus Bot-F Inf-F entails-𝒢 Red-I-𝒢 empty-ord.Red-F-𝒢*
⟨*proof*⟩

**theorem** *static-to-dynamic*:
*statically-complete-calculus Bot-F Inf-F entails-𝒢 Red-I-𝒢 empty-ord.Red-F-𝒢 =*
*dynamically-complete-calculus Bot-F Inf-F entails-𝒢 Red-I-𝒢 Red-F-𝒢*
⟨*proof*⟩

**end**

## 4.4 Lifting with a Family of Redundancy Criteria

**locale** *lifting-intersection = inference-system Inf-F +*
*ground*: *inference-system-family Q Inf-G-q +*
*ground*: *consequence-relation-family Bot-G Q entails-q*
**for**
*Inf-F* :: *'f inference set* **and**
*Bot-G* :: *'g set* **and**
*Q* :: *'q set* **and**
*Inf-G-q* :: *‹'q ⇒ 'g inference set›* **and**
*entails-q* :: *'q ⇒ 'g set ⇒ 'g set ⇒ bool* **and**
*Red-I-q* :: *'q ⇒ 'g set ⇒ 'g inference set* **and**
*Red-F-q* :: *'q ⇒ 'g set ⇒ 'g set*
+ **fixes**
*Bot-F* :: *'f set* **and**
*𝒢-F-q* :: *'q ⇒ 'f ⇒ 'g set* **and**
*𝒢-I-q* :: *'q ⇒ 'f inference ⇒ 'g inference set option* **and**
*Prec-F-g* :: *'g ⇒ 'f ⇒ 'f ⇒ bool*
**assumes**
*standard-lifting-family*:
*∀ q ∈ Q. tiebreaker-lifting Bot-F Inf-F Bot-G (entails-q q) (Inf-G-q q) (Red-I-q q)*
*(Red-F-q q) (𝒢-F-q q) (𝒢-I-q q) Prec-F-g*
**begin**

**abbreviation** *𝒢-Fset-q* :: *'q ⇒ 'f set ⇒ 'g set* **where**
*𝒢-Fset-q q N ≡ ⋃ (𝒢-F-q q ‘ N)*

**definition** *Red-I-𝒢-q* :: *'q ⇒ 'f set ⇒ 'f inference set* **where**
*Red-I-𝒢-q q N = {ι ∈ Inf-F. (𝒢-I-q q ι ≠ None ∧ the (𝒢-I-q q ι) ⊆ Red-I-q q (𝒢-Fset-q q N))*
*∨ (𝒢-I-q q ι = None ∧ 𝒢-F-q q (concl-of ι) ⊆ (𝒢-Fset-q q N ∪ Red-F-q q (𝒢-Fset-q q N)))}*

**definition** *Red-F-𝒢-empty-q* :: *'q ⇒ 'f set ⇒ 'f set* **where**
*Red-F-𝒢-empty-q q N = {C. ∀ D ∈ 𝒢-F-q q C. D ∈ Red-F-q q (𝒢-Fset-q q N)}*

**definition** *Red-F-𝒢-q* :: *'q ⇒ 'f set ⇒ 'f set* **where**
*Red-F-𝒢-q q N =*
*{C. ∀ D ∈ 𝒢-F-q q C. D ∈ Red-F-q q (𝒢-Fset-q q N) ∨ (∃ E ∈ N. Prec-F-g D E C ∧ D ∈ 𝒢-F-q q*

$E)\}$

**abbreviation** *entails-$\mathcal{G}$-q* :: *'q* ⇒ *'f set* ⇒ *'f set* ⇒ *bool* **where**
*entails-$\mathcal{G}$-q q N1 N2* ≡ *entails-q q* (*$\mathcal{G}$-Fset-q q N1*) (*$\mathcal{G}$-Fset-q q N2*)

**lemma** *red-crit-lifting-family*:
**assumes** *q-in*: *q* ∈ *Q*
**shows** *calculus Bot-F Inf-F* (*entails-$\mathcal{G}$-q q*) (*Red-I-$\mathcal{G}$-q q*) (*Red-F-$\mathcal{G}$-q q*)
⟨*proof*⟩

**lemma** *red-crit-lifting-family-empty-ord*:
**assumes** *q-in*: *q* ∈ *Q*
**shows** *calculus Bot-F Inf-F* (*entails-$\mathcal{G}$-q q*) (*Red-I-$\mathcal{G}$-q q*) (*Red-F-$\mathcal{G}$-empty-q q*)
⟨*proof*⟩

**sublocale** *consequence-relation-family Bot-F Q entails-$\mathcal{G}$-q*
⟨*proof*⟩

**sublocale** *intersection-calculus Bot-F Inf-F Q entails-$\mathcal{G}$-q Red-I-$\mathcal{G}$-q Red-F-$\mathcal{G}$-q*
⟨*proof*⟩

**abbreviation** *entails-$\mathcal{G}$* :: *'f set* ⇒ *'f set* ⇒ *bool* (**infix** ⊨∩$\mathcal{G}$ *50*) **where**
(⊨∩$\mathcal{G}$) ≡ *entails*

**abbreviation** *Red-I-$\mathcal{G}$* :: *'f set* ⇒ *'f inference set* **where**
*Red-I-$\mathcal{G}$* ≡ *Red-I*

**abbreviation** *Red-F-$\mathcal{G}$* :: *'f set* ⇒ *'f set* **where**
*Red-F-$\mathcal{G}$* ≡ *Red-F*

**lemmas** *entails-$\mathcal{G}$-def* = *entails-def*
**lemmas** *Red-I-$\mathcal{G}$-def* = *Red-I-def*
**lemmas** *Red-F-$\mathcal{G}$-def* = *Red-F-def*

**sublocale** *empty-ord*: *intersection-calculus Bot-F Inf-F Q entails-$\mathcal{G}$-q Red-I-$\mathcal{G}$-q Red-F-$\mathcal{G}$-empty-q*
⟨*proof*⟩

**abbreviation** *Red-F-$\mathcal{G}$-empty* :: *'f set* ⇒ *'f set* **where**
*Red-F-$\mathcal{G}$-empty* ≡ *empty-ord.Red-F*

**lemmas** *Red-F-$\mathcal{G}$-empty-def* = *empty-ord.Red-F-def*

**lemma** *sat-inf-imp-ground-red-fam-inter*:
**assumes**
*sat-n*: *saturated N* **and**
*i′-in*: *ι′* ∈ *Inf-from N* **and**
*q-in*: *q* ∈ *Q* **and**
*grounding*: *$\mathcal{G}$-I-q q ι′* ≠ *None* ∧ *ι* ∈ *the* (*$\mathcal{G}$-I-q q ι′*)
**shows** *ι* ∈ *Red-I-q q* (*$\mathcal{G}$-Fset-q q N*)
⟨*proof*⟩

**abbreviation** *ground-Inf-overapproximated* :: *'q* ⇒ *'f set* ⇒ *bool* **where**
*ground-Inf-overapproximated q N* ≡
*ground.Inf-from-q q* (*$\mathcal{G}$-Fset-q q N*)
⊆ {*ι*. ∃*ι′*∈ *Inf-from N*. *$\mathcal{G}$-I-q q ι′* ≠ *None* ∧ *ι* ∈ *the* (*$\mathcal{G}$-I-q q ι′*)} ∪ *Red-I-q q* (*$\mathcal{G}$-Fset-q q N*)

**abbreviation** *ground-saturated* :: *'q* ⇒ *'f set* ⇒ *bool* **where**
*ground-saturated q N* ≡ *ground.Inf-from-q q* (*𝒢-Fset-q q N*) ⊆ *Red-I-q q* (*𝒢-Fset-q q N*)

**lemma** *sat-imp-ground-sat-fam-inter*:
*saturated N* ⟹ *q* ∈ *Q* ⟹ *ground-Inf-overapproximated q N* ⟹ *ground-saturated q N*
⟨*proof*⟩

**theorem** *stat-ref-comp-to-non-ground-fam-inter*:
**assumes**
*stat-ref-G*:
∀ *q* ∈ *Q*. *statically-complete-calculus Bot-G* (*Inf-G-q q*) (*entails-q q*) (*Red-I-q q*)
(*Red-F-q q*) **and**
*sat-n-imp*: ⋀*N*. *saturated N* ⟹ ∃ *q* ∈ *Q*. *ground-Inf-overapproximated q N*
**shows**
*statically-complete-calculus Bot-F Inf-F entails-𝒢 Red-I-𝒢 Red-F-𝒢-empty*
⟨*proof*⟩

**lemma** *sat-eq-sat-empty-order*: *saturated N* = *empty-ord.saturated N*
⟨*proof*⟩

**lemma** *static-empty-ord-inter-equiv-static-inter*:
*statically-complete-calculus Bot-F Inf-F entails Red-I Red-F* =
*statically-complete-calculus Bot-F Inf-F entails Red-I Red-F-𝒢-empty*
⟨*proof*⟩

**theorem** *stat-eq-dyn-ref-comp-fam-inter*: *statically-complete-calculus Bot-F Inf-F*
*entails Red-I Red-F-𝒢-empty* =
*dynamically-complete-calculus Bot-F Inf-F entails Red-I Red-F*
⟨*proof*⟩

**end**

**end**

# 5 Labeled Lifting to Non-Ground Calculi

This section formalizes the extension of the lifting results to labeled calculi. This corresponds to section 3.4 of the report.

**theory** *Labeled-Lifting-to-Non-Ground-Calculi*
**imports** *Lifting-to-Non-Ground-Calculi*
**begin**

**lemma** *po-on-empty-rel*[*simp*]: *po-on* (λ- -. *False*) *UNIV*
⟨*proof*⟩

## 5.1 Labeled Lifting with a Family of Tiebreaker Orderings

**locale** *labeled-tiebreaker-lifting* = *no-labels*: *tiebreaker-lifting Bot-F Inf-F*
*Bot-G entails-G Inf-G Red-I-G Red-F-G 𝒢-F 𝒢-I Prec-F*

**for**
*Bot-F* :: *'f set* **and**
*Inf-F* :: *'f inference set* **and**
*Bot-G* :: *'g set* **and**
*entails-G* :: *'g set* ⇒ *'g set* ⇒ *bool* (**infix** $\models G$ *50*) **and**
*Inf-G* :: *'g inference set* **and**
*Red-I-G* :: *'g set* ⇒ *'g inference set* **and**
*Red-F-G* :: *'g set* ⇒ *'g set* **and**
$\mathcal{G}$*-F* :: *'f* ⇒ *'g set* **and**
$\mathcal{G}$*-I* :: *'f inference* ⇒ *'g inference set option* **and**
*Prec-F* :: *'g* ⇒ *'f* ⇒ *'f* ⇒ *bool* (**infix** ⊏ *50*)
+ **fixes**
*Inf-FL* :: ‹(*'f* × *'l*) *inference set*›
**assumes**
*Inf-F-to-Inf-FL*: ‹$\iota_F$ ∈ *Inf-F* ⟹ *length* (*Ll* :: *'l list*) = *length* (*prems-of* $\iota_F$) ⟹
∃ *L0*. *Infer* (*zip* (*prems-of* $\iota_F$) *Ll*) (*concl-of* $\iota_F$, *L0*) ∈ *Inf-FL*› **and**
*Inf-FL-to-Inf-F*: ‹$\iota_{FL}$ ∈ *Inf-FL* ⟹ *Infer* (*map fst* (*prems-of* $\iota_{FL}$)) (*fst* (*concl-of* $\iota_{FL}$)) ∈ *Inf-F*›
**begin**

**definition** *to-F* :: ‹(*'f* × *'l*) *inference* ⇒ *'f inference*› **where**
‹*to-F* $\iota_{FL}$ = *Infer* (*map fst* (*prems-of* $\iota_{FL}$)) (*fst* (*concl-of* $\iota_{FL}$))›

**abbreviation** *Bot-FL* :: ‹(*'f* × *'l*) *set*› **where**
‹*Bot-FL* ≡ *Bot-F* × *UNIV*›

**abbreviation** $\mathcal{G}$*-F-L* :: ‹(*'f* × *'l*) ⇒ *'g set*› **where**
‹$\mathcal{G}$*-F-L CL* ≡ $\mathcal{G}$*-F* (*fst CL*)›

**abbreviation** $\mathcal{G}$*-I-L* :: ‹(*'f* × *'l*) *inference* ⇒ *'g inference set option*› **where**
‹$\mathcal{G}$*-I-L* $\iota_{FL}$ ≡ $\mathcal{G}$*-I* (*to-F* $\iota_{FL}$)›

**sublocale** *standard-lifting Inf-FL Bot-G Inf-G* ($\models G$) *Red-I-G Red-F-G Bot-FL* $\mathcal{G}$*-F-L* $\mathcal{G}$*-I-L*
⟨*proof*⟩

**notation** *entails-*$\mathcal{G}$ (**infix** $\models\mathcal{G}L$ *50*)

**lemma** *labeled-entailment-lifting*: *NL1* $\models\mathcal{G}L$ *NL2* ⟷ *fst* ‘ *NL1* $\models\mathcal{G}$ *fst* ‘ *NL2*
⟨*proof*⟩

**lemma** *red-inf-impl*: $\iota$ ∈ *Red-I-*$\mathcal{G}$ *NL* ⟹ *to-F* $\iota$ ∈ *no-labels.Red-I-*$\mathcal{G}$ (*fst* ‘ *NL*)
⟨*proof*⟩

**lemma** *labeled-saturation-lifting*: *saturated NL* ⟹ *no-labels.saturated* (*fst* ‘ *NL*)
⟨*proof*⟩

**lemma** *stat-ref-comp-to-labeled-sta-ref-comp*:
**assumes** *static*:
*statically-complete-calculus Bot-F Inf-F* ($\models\mathcal{G}$) *no-labels.Red-I-*$\mathcal{G}$ *no-labels.Red-F-*$\mathcal{G}$
**shows** *statically-complete-calculus Bot-FL Inf-FL* ($\models\mathcal{G}L$) *Red-I-*$\mathcal{G}$ *Red-F-*$\mathcal{G}$

⟨*proof*⟩

**end**

## 5.2 Labeled Lifting with a Family of Redundancy Criteria

**locale** *labeled-lifting-intersection* = *no-labels*: *lifting-intersection Inf-F*
*Bot-G Q Inf-G-q entails-q Red-I-q Red-F-q Bot-F* $\mathcal{G}$*-F-q* $\mathcal{G}$*-I-q* $\lambda g$ *Cl Cl'. False*
**for**
*Bot-F* :: *'f set* **and**
*Inf-F* :: *'f inference set* **and**
*Bot-G* :: *'g set* **and**
*Q* :: *'q set* **and**
*entails-q* :: *'q* ⇒ *'g set* ⇒ *'g set* ⇒ *bool* **and**
*Inf-G-q* :: *'q* ⇒ *'g inference set* **and**
*Red-I-q* :: *'q* ⇒ *'g set* ⇒ *'g inference set* **and**
*Red-F-q* :: *'q* ⇒ *'g set* ⇒ *'g set* **and**
$\mathcal{G}$*-F-q* :: *'q* ⇒ *'f* ⇒ *'g set* **and**
$\mathcal{G}$*-I-q* :: *'q* ⇒ *'f inference* ⇒ *'g inference set option*
+ **fixes**
*Inf-FL* :: ‹(*'f* × *'l*) *inference set*›
**assumes**
*Inf-F-to-Inf-FL*:
‹$\iota_F$ ∈ *Inf-F* ⟹ *length* (*Ll* :: *'l list*) = *length* (*prems-of* $\iota_F$) ⟹
∃ *L0*. *Infer* (*zip* (*prems-of* $\iota_F$) *Ll*) (*concl-of* $\iota_F$, *L0*) ∈ *Inf-FL*› **and**
*Inf-FL-to-Inf-F*: ‹$\iota_{FL}$ ∈ *Inf-FL* ⟹ *Infer* (*map fst* (*prems-of* $\iota_{FL}$)) (*fst* (*concl-of* $\iota_{FL}$)) ∈ *Inf-F*›
**begin**

**definition** *to-F* :: ‹(*'f* × *'l*) *inference* ⇒ *'f inference*› **where**
‹*to-F* $\iota_{FL}$ = *Infer* (*map fst* (*prems-of* $\iota_{FL}$)) (*fst* (*concl-of* $\iota_{FL}$))›

**abbreviation** *Bot-FL* :: ‹(*'f* × *'l*) *set*› **where**
‹*Bot-FL* ≡ *Bot-F* × *UNIV*›

**abbreviation** $\mathcal{G}$*-F-L-q* :: ‹*'q* ⇒ (*'f* × *'l*) ⇒ *'g set*› **where**
‹$\mathcal{G}$*-F-L-q q CL* ≡ $\mathcal{G}$*-F-q q* (*fst CL*)›

**abbreviation** $\mathcal{G}$*-I-L-q* :: ‹*'q* ⇒ (*'f* × *'l*) *inference* ⇒ *'g inference set option*› **where**
‹$\mathcal{G}$*-I-L-q q* $\iota_{FL}$ ≡ $\mathcal{G}$*-I-q q* (*to-F* $\iota_{FL}$)›

**abbreviation** $\mathcal{G}$*-Fset-L-q* :: *'q* ⇒ (*'f* × *'l*) *set* ⇒ *'g set* **where**
$\mathcal{G}$*-Fset-L-q q N* ≡ ⋃ ($\mathcal{G}$*-F-L-q q* ‘ *N*)

**definition** *Red-I-*$\mathcal{G}$*-L-q* :: *'q* ⇒ (*'f* × *'l*) *set* ⇒ (*'f* × *'l*) *inference set* **where**
*Red-I-*$\mathcal{G}$*-L-q q N* =
{$\iota$ ∈ *Inf-FL*. ($\mathcal{G}$*-I-L-q q* $\iota$ ≠ *None* ∧ *the* ($\mathcal{G}$*-I-L-q q* $\iota$) ⊆ *Red-I-q q* ($\mathcal{G}$*-Fset-L-q q N*))
∨ ($\mathcal{G}$*-I-L-q q* $\iota$ = *None* ∧ $\mathcal{G}$*-F-L-q q* (*concl-of* $\iota$) ⊆ $\mathcal{G}$*-Fset-L-q q N* ∪ *Red-F-q q* ($\mathcal{G}$*-Fset-L-q q N*))}

**abbreviation** *Red-I-*$\mathcal{G}$*-L* :: (*'f* × *'l*) *set* ⇒ (*'f* × *'l*) *inference set* **where**
*Red-I-*$\mathcal{G}$*-L N* ≡ (⋂ *q* ∈ *Q*. *Red-I-*$\mathcal{G}$*-L-q q N*)

**abbreviation** *entails-*$\mathcal{G}$*-L-q* :: *'q* ⇒ (*'f* × *'l*) *set* ⇒ (*'f* × *'l*) *set* ⇒ *bool* **where**
*entails-*$\mathcal{G}$*-L-q q N1 N2* ≡ *entails-q q* ($\mathcal{G}$*-Fset-L-q q N1*) ($\mathcal{G}$*-Fset-L-q q N2*)

**lemma** *lifting-q*:
**assumes** *q* ∈ *Q*

**shows** *labeled-tiebreaker-lifting Bot-F Inf-F Bot-G* (*entails-q q*) (*Inf-G-q q*) (*Red-I-q q*)
(*Red-F-q q*) (*$\mathcal{G}$-F-q q*) (*$\mathcal{G}$-I-q q*) (*λg Cl Cl′. False*) *Inf-FL*
⟨*proof*⟩

**lemma** *lifted-q*:
**assumes** *q-in*: *q ∈ Q*
**shows** *standard-lifting Inf-FL Bot-G* (*Inf-G-q q*) (*entails-q q*) (*Red-I-q q*) (*Red-F-q q*)
*Bot-FL* (*$\mathcal{G}$-F-L-q q*) (*$\mathcal{G}$-I-L-q q*)
⟨*proof*⟩

**lemma** *ord-fam-lifted-q*:
**assumes** *q-in*: *q ∈ Q*
**shows** *tiebreaker-lifting Bot-FL Inf-FL Bot-G* (*entails-q q*) (*Inf-G-q q*) (*Red-I-q q*)
(*Red-F-q q*) (*$\mathcal{G}$-F-L-q q*) (*$\mathcal{G}$-I-L-q q*) (*λg Cl Cl′. False*)
⟨*proof*⟩

**definition** *Red-F-$\mathcal{G}$-empty-L-q* :: *′q ⇒* (*′f × ′l*) *set ⇒* (*′f × ′l*) *set* **where**
*Red-F-$\mathcal{G}$-empty-L-q q N* = {*C*. *∀ D ∈ $\mathcal{G}$-F-L-q q C*. *D ∈ Red-F-q q* (*$\mathcal{G}$-Fset-L-q q N*) *∨*
(*∃ E ∈ N*. *False ∧ D ∈ $\mathcal{G}$-F-L-q q E*)}

**abbreviation** *Red-F-$\mathcal{G}$-empty-L* :: (*′f × ′l*) *set ⇒* (*′f × ′l*) *set* **where**
*Red-F-$\mathcal{G}$-empty-L N ≡* (*⋂ q ∈ Q*. *Red-F-$\mathcal{G}$-empty-L-q q N*)

**lemma** *all-lifted-red-crit*:
**assumes** *q-in*: *q ∈ Q*
**shows** *calculus Bot-FL Inf-FL* (*entails-$\mathcal{G}$-L-q q*) (*Red-I-$\mathcal{G}$-L-q q*) (*Red-F-$\mathcal{G}$-empty-L-q q*)
⟨*proof*⟩

**lemma** *all-lifted-cons-rel*:
**assumes** *q-in*: *q ∈ Q*
**shows** *consequence-relation Bot-FL* (*entails-$\mathcal{G}$-L-q q*)
⟨*proof*⟩

**sublocale** *consequence-relation-family Bot-FL Q entails-$\mathcal{G}$-L-q*
⟨*proof*⟩

**sublocale** *intersection-calculus Bot-FL Inf-FL Q entails-$\mathcal{G}$-L-q Red-I-$\mathcal{G}$-L-q Red-F-$\mathcal{G}$-empty-L-q*
⟨*proof*⟩

**lemma** *in-Inf-FL-imp-to-F-in-Inf-F*: *ι ∈ Inf-FL ⟹ to-F ι ∈ Inf-F*
⟨*proof*⟩

**lemma** *in-Inf-from-imp-to-F-in-Inf-from*: *ι ∈ Inf-from N ⟹ to-F ι ∈ no-labels.Inf-from* (*fst ‘ N*)
⟨*proof*⟩

**notation** *no-labels.entails-$\mathcal{G}$* (**infix** *⊨∩$\mathcal{G}$ 50*)

**abbreviation** *entails-$\mathcal{G}$-L* :: (*′f × ′l*) *set ⇒* (*′f × ′l*) *set ⇒ bool* (**infix** *⊨∩$\mathcal{G}$L 50*) **where**
(*⊨∩$\mathcal{G}$L*) *≡ entails*

**lemmas** *entails-$\mathcal{G}$-L-def = entails-def*

**lemma** *labeled-entailment-lifting*: *NL1 ⊨∩$\mathcal{G}$L NL2 ⟷ fst ‘ NL1 ⊨∩$\mathcal{G}$ fst ‘ NL2*
⟨*proof*⟩

**lemma** *red-inf-impl*: $\iota \in$ *Red-I NL* $\Longrightarrow$ *to-F* $\iota \in$ *no-labels.Red-I-*$\mathcal{G}$ (*fst* ‘ *NL*)
⟨*proof*⟩

**lemma** *labeled-family-saturation-lifting*: *saturated NL* $\Longrightarrow$ *no-labels.saturated* (*fst* ‘ *NL*)
⟨*proof*⟩

**theorem** *labeled-static-ref*:
**assumes** *calc*: *statically-complete-calculus Bot-F Inf-F* ($\models\cap\mathcal{G}$) *no-labels.Red-I-*$\mathcal{G}$
*no-labels.Red-F-*$\mathcal{G}$*-empty*
**shows** *statically-complete-calculus Bot-FL Inf-FL* ($\models\cap\mathcal{G}L$) *Red-I Red-F*
⟨*proof*⟩

**end**

**end**

# 6 Given Clause Prover Architectures

This section covers all the results presented in the section 4 of the report. This is where abstract architectures of provers are defined and proven dynamically refutationally complete.

**theory** *Given-Clause-Architectures*
**imports**
*Lambda-Free-RPOs.Lambda-Free-Util*
*Labeled-Lifting-to-Non-Ground-Calculi*
**begin**

## 6.1 Basis of the Given Clause Prover Architectures

**locale** *given-clause-basis* = *std?*: *labeled-lifting-intersection Bot-F Inf-F Bot-G Q*
*entails-q Inf-G-q Red-I-q Red-F-q* $\mathcal{G}$*-F-q* $\mathcal{G}$*-I-q*
{$\iota_{FL}$ :: ('*f* × '*l*) *inference*. *Infer* (*map fst* (*prems-of* $\iota_{FL}$)) (*fst* (*concl-of* $\iota_{FL}$)) ∈ *Inf-F*}
**for**
*Bot-F* :: '*f set*
**and** *Inf-F* :: '*f inference set*
**and** *Bot-G* :: '*g set*
**and** *Q* :: '*q set*
**and** *entails-q* :: '*q* ⇒ '*g set* ⇒ '*g set* ⇒ *bool*
**and** *Inf-G-q* :: ‹'*q* ⇒ '*g inference set*›
**and** *Red-I-q* :: '*q* ⇒ '*g set* ⇒ '*g inference set*
**and** *Red-F-q* :: '*q* ⇒ '*g set* ⇒ '*g set*
**and** $\mathcal{G}$*-F-q* :: '*q* ⇒ '*f* ⇒ '*g set*
**and** $\mathcal{G}$*-I-q* :: '*q* ⇒ '*f inference* ⇒ '*g inference set option*
+ **fixes**
*Equiv-F* :: '*f* ⇒ '*f* ⇒ *bool* (**infix** $\doteq$ *50*) **and**
*Prec-F* :: '*f* ⇒ '*f* ⇒ *bool* (**infix** $\prec\cdot$ *50*) **and**
*Prec-L* :: '*l* ⇒ '*l* ⇒ *bool* (**infix** $\sqsubset L$ *50*) **and**
*active* :: '*l*
**assumes**
*equiv-equiv-F*: *equivp* ($\doteq$) **and**
*wf-prec-F*: *minimal-element* ($\prec\cdot$) *UNIV* **and**
*wf-prec-L*: *minimal-element* ($\sqsubset L$) *UNIV* **and**

*compat-equiv-prec*: $C1 \doteq D1 \Longrightarrow C2 \doteq D2 \Longrightarrow C1 \prec\cdot C2 \Longrightarrow D1 \prec\cdot D2$ **and**
*equiv-F-grounding*: $q \in Q \Longrightarrow C1 \doteq C2 \Longrightarrow \mathcal{G}\text{-}F\text{-}q\ q\ C1 \subseteq \mathcal{G}\text{-}F\text{-}q\ q\ C2$ **and**
*prec-F-grounding*: $q \in Q \Longrightarrow C2 \prec\cdot C1 \Longrightarrow \mathcal{G}\text{-}F\text{-}q\ q\ C1 \subseteq \mathcal{G}\text{-}F\text{-}q\ q\ C2$ **and**
*active-minimal*: $l2 \neq active \Longrightarrow active \sqsubset L\ l2$ **and**
*at-least-two-labels*: $\exists\, l2.\ active \sqsubset L\ l2$ **and**
*static-ref-comp*: *statically-complete-calculus Bot-F Inf-F* $(\models \cap \mathcal{G})$
*no-labels.Red-I-*$\mathcal{G}$ *no-labels.Red-F-*$\mathcal{G}$*-empty*
**begin**

**abbreviation** *Inf-FL* :: $('f \times 'l)$ *inference set* **where**
*Inf-FL* $\equiv \{\iota_{FL}.\ Infer\ (map\ fst\ (prems\text{-}of\ \iota_{FL}))\ (fst\ (concl\text{-}of\ \iota_{FL})) \in Inf\text{-}F\}$

**abbreviation** *Prec-eq-F* :: $'f \Rightarrow 'f \Rightarrow bool$ (**infix** $\preceq\cdot$ *50*) **where**
$C \preceq\cdot D \equiv C \doteq D \vee C \prec\cdot D$

**definition** *Prec-FL* :: $('f \times 'l) \Rightarrow ('f \times 'l) \Rightarrow bool$ (**infix** $\sqsubset$ *50*) **where**
$Cl1 \sqsubset Cl2 \longleftrightarrow fst\ Cl1 \prec\cdot fst\ Cl2 \vee (fst\ Cl1 \doteq fst\ Cl2 \wedge snd\ Cl1 \sqsubset L\ snd\ Cl2)$

**lemma** *irrefl-prec-F*: $\neg\ C \prec\cdot C$
⟨*proof*⟩

**lemma** *trans-prec-F*: $C1 \prec\cdot C2 \Longrightarrow C2 \prec\cdot C3 \Longrightarrow C1 \prec\cdot C3$
⟨*proof*⟩

**lemma** *wf-prec-FL*: *minimal-element* $(\sqsubset)$ *UNIV*
⟨*proof*⟩

**definition** *active-subset* :: $('f \times 'l)\ set \Rightarrow ('f \times 'l)\ set$ **where**
*active-subset* $M = \{CL \in M.\ snd\ CL = active\}$

**definition** *passive-subset* :: $('f \times 'l)\ set \Rightarrow ('f \times 'l)\ set$ **where**
*passive-subset* $M = \{CL \in M.\ snd\ CL \neq active\}$

**lemma** *active-subset-insert*[*simp*]:
*active-subset* $(insert\ Cl\ N) = (if\ snd\ Cl = active\ then\ \{Cl\}\ else\ \{\}) \cup active\text{-}subset\ N$
⟨*proof*⟩

**lemma** *active-subset-union*[*simp*]: *active-subset* $(M \cup N) = active\text{-}subset\ M \cup active\text{-}subset\ N$
⟨*proof*⟩

**lemma** *passive-subset-insert*[*simp*]:
*passive-subset* $(insert\ Cl\ N) = (if\ snd\ Cl \neq active\ then\ \{Cl\}\ else\ \{\}) \cup passive\text{-}subset\ N$
⟨*proof*⟩

**lemma** *passive-subset-union*[*simp*]: *passive-subset* $(M \cup N) = passive\text{-}subset\ M \cup passive\text{-}subset\ N$
⟨*proof*⟩

**sublocale** *std?*: *statically-complete-calculus Bot-FL Inf-FL* $(\models \cap \mathcal{G}L)$ *Red-I Red-F*
⟨*proof*⟩

**lemma** *labeled-tiebreaker-lifting*:
**assumes** *q-in*: $q \in Q$
**shows** *tiebreaker-lifting Bot-FL Inf-FL Bot-G* (*entails-q q*) (*Inf-G-q q*)
(*Red-I-q q*) (*Red-F-q q*) ($\mathcal{G}$*-F-L-q q*) ($\mathcal{G}$*-I-L-q q*) ($\lambda g.$ *Prec-FL*)
⟨*proof*⟩

**sublocale** *lifting-intersection Inf-FL Bot-G Q Inf-G-q entails-q Red-I-q Red-F-q Bot-FL* $\mathcal{G}$*-F-L-q* $\mathcal{G}$*-I-L-q* $\lambda g$*. Prec-FL*
⟨*proof*⟩

**notation** *derive* (**infix** $\triangleright L$ *50*)

**lemma** *std-Red-I-eq*: *std.Red-I* = *Red-I-*$\mathcal{G}$
⟨*proof*⟩

**lemma** *std-Red-F-eq*: *std.Red-F* = *Red-F-*$\mathcal{G}$*-empty*
⟨*proof*⟩

**sublocale** *statically-complete-calculus Bot-FL Inf-FL* ($\models \cap \mathcal{G} L$) *Red-I Red-F*
⟨*proof*⟩

**lemma** *labeled-red-inf-eq-red-inf*:
**assumes** *i-in*: $\iota \in$ *Inf-FL*
**shows** $\iota \in$ *Red-I N* $\longleftrightarrow$ *to-F* $\iota \in$ *no-labels.Red-I-*$\mathcal{G}$ (*fst* ‘ *N*)
⟨*proof*⟩

**lemma** *red-labeled-clauses*:
**assumes** ‹$C \in$ *no-labels.Red-F-*$\mathcal{G}$*-empty* (*fst* ‘ *N*) ∨
($\exists\, C' \in$ *fst* ‘ *N*. $C' \prec\cdot C$) ∨ ($\exists (C', L') \in N.\ L' \sqsubset L\ L \wedge C' \preceq\cdot C$)›
**shows** ‹($C$, $L$) ∈ *Red-F N*›
⟨*proof*⟩

**end**

## 6.2 Given Clause Procedure

**locale** *given-clause* = *given-clause-basis Bot-F Inf-F Bot-G Q entails-q Inf-G-q Red-I-q Red-F-q* $\mathcal{G}$*-F-q* $\mathcal{G}$*-I-q Equiv-F Prec-F Prec-L active*
**for**
*Bot-F* :: *'f set* **and**
*Inf-F* :: *'f inference set* **and**
*Bot-G* :: *'g set* **and**
*Q* :: *'q set* **and**
*entails-q* :: *'q* ⇒ *'g set* ⇒ *'g set* ⇒ *bool* **and**
*Inf-G-q* :: ‹*'q* ⇒ *'g inference set*› **and**
*Red-I-q* :: *'q* ⇒ *'g set* ⇒ *'g inference set* **and**
*Red-F-q* :: *'q* ⇒ *'g set* ⇒ *'g set* **and**
$\mathcal{G}$*-F-q* :: *'q* ⇒ *'f* ⇒ *'g set* **and**
$\mathcal{G}$*-I-q* :: *'q* ⇒ *'f inference* ⇒ *'g inference set option* **and**
*Equiv-F* :: *'f* ⇒ *'f* ⇒ *bool* (**infix** $\doteq$ *50*) **and**
*Prec-F* :: *'f* ⇒ *'f* ⇒ *bool* (**infix** $\prec\cdot$ *50*) **and**
*Prec-L* :: *'l* ⇒ *'l* ⇒ *bool* (**infix** $\sqsubset L$ *50*) **and**
*active* :: *'l* +
**assumes**
*inf-have-prems*: $\iota F \in$ *Inf-F* ⟹ *prems-of* $\iota F \neq []$
**begin**

**lemma** *labeled-inf-have-prems*: $\iota \in$ *Inf-FL* ⟹ *prems-of* $\iota \neq []$
⟨*proof*⟩

**inductive** *step* :: $('f \times 'l)$ *set* $\Rightarrow$ $('f \times 'l)$ *set* $\Rightarrow$ *bool* (**infix** $\rightsquigarrow GC$ *50*) **where**
  *process*: $N1 = N \cup M \Longrightarrow N2 = N \cup M' \Longrightarrow M \subseteq$ *Red-F* $(N \cup M') \Longrightarrow$
    *active-subset* $M' = \{\} \Longrightarrow N1 \rightsquigarrow GC\ N2$
| *infer*: $N1 = N \cup \{(C, L)\} \Longrightarrow N2 = N \cup \{(C, active)\} \cup M \Longrightarrow L \neq active \Longrightarrow$
    *active-subset* $M = \{\} \Longrightarrow$
    *no-labels.Inf-between* (*fst* ‘ *active-subset* $N$) $\{C\}$
    $\subseteq$ *no-labels.Red-I* (*fst* ‘ $(N \cup \{(C, active)\} \cup M)) \Longrightarrow$
    $N1 \rightsquigarrow GC\ N2$

**lemma** *one-step-equiv*: $N1 \rightsquigarrow GC\ N2 \Longrightarrow N1 \rhd L\ N2$
⟨*proof*⟩

**lemma** *gc-to-red*: *chain* $(\rightsquigarrow GC)$ *Ns* $\Longrightarrow$ *chain* $(\rhd L)$ *Ns*
  ⟨*proof*⟩

**lemma** (**in**−) *all-ex-finite-set*: $(\forall\, (j{::}nat){\in}\{0..{<}m\}.\ \exists\, (n{::}nat).\ P\ j\ n) \Longrightarrow$
  $(\forall\, n1\ n2.\ \forall\, j{\in}\{0..{<}m\}.\ P\ j\ n1 \longrightarrow P\ j\ n2 \longrightarrow n1 = n2) \Longrightarrow$ *finite* $\{n.\ \exists\, j \in \{0..{<}m\}.\ P\ j\ n\}$ **for** $m$ $P$
⟨*proof*⟩

**lemma** *gc-fair*:
  **assumes**
    *deriv*: *chain* $(\rightsquigarrow GC)$ *Ns* **and**
    *init-state*: *active-subset* (*lhd Ns*) = {} **and**
    *final-state*: *passive-subset* (*Liminf-llist Ns*) = {}
  **shows** *fair Ns*
  ⟨*proof*⟩

**theorem** *gc-complete-Liminf*:
  **assumes**
    *deriv*: *chain* $(\rightsquigarrow GC)$ *Ns* **and**
    *init-state*: *active-subset* (*lhd Ns*) = {} **and**
    *final-state*: *passive-subset* (*Liminf-llist Ns*) = {} **and**
    *b-in*: $B \in$ *Bot-F* **and**
    *bot-entailed*: *no-labels.entails-*$\mathcal{G}$ (*fst* ‘ *lhd Ns*) $\{B\}$
  **shows** $\exists\, BL \in$ *Bot-FL*. $BL \in$ *Liminf-llist Ns*
⟨*proof*⟩

**theorem** *gc-complete*:
  **assumes**
    *deriv*: *chain* $(\rightsquigarrow GC)$ *Ns* **and**
    *init-state*: *active-subset* (*lhd Ns*) = {} **and**
    *final-state*: *passive-subset* (*Liminf-llist Ns*) = {} **and**
    *b-in*: $B \in$ *Bot-F* **and**
    *bot-entailed*: *no-labels.entails-*$\mathcal{G}$ (*fst* ‘ *lhd Ns*) $\{B\}$
  **shows** $\exists\, i.$ *enat* $i <$ *llength Ns* $\wedge$ $(\exists\, BL \in$ *Bot-FL*. $BL \in$ *lnth Ns* $i)$
⟨*proof*⟩

**end**

## 6.3 Lazy Given Clause Procedure

**locale** *lazy-given-clause* = *given-clause-basis Bot-F Inf-F Bot-G Q entails-q Inf-G-q Red-I-q Red-F-q* $\mathcal{G}$*-F-q* $\mathcal{G}$*-I-q Equiv-F Prec-F Prec-L active*
**for**
*Bot-F* :: *'f set* **and**
*Inf-F* :: *'f inference set* **and**
*Bot-G* :: *'g set* **and**
*Q* :: *'q set* **and**
*entails-q* :: *'q* ⇒ *'g set* ⇒ *'g set* ⇒ *bool* **and**
*Inf-G-q* :: ‹*'q* ⇒ *'g inference set*› **and**
*Red-I-q* :: *'q* ⇒ *'g set* ⇒ *'g inference set* **and**
*Red-F-q* :: *'q* ⇒ *'g set* ⇒ *'g set* **and**
$\mathcal{G}$*-F-q* :: *'q* ⇒ *'f* ⇒ *'g set* **and**
$\mathcal{G}$*-I-q* :: *'q* ⇒ *'f inference* ⇒ *'g inference set option* **and**
*Equiv-F* :: *'f* ⇒ *'f* ⇒ *bool* (**infix** ≐ *50*) **and**
*Prec-F* :: *'f* ⇒ *'f* ⇒ *bool* (**infix** ≺· *50*) **and**
*Prec-L* :: *'l* ⇒ *'l* ⇒ *bool* (**infix** ⊏*L* *50*) **and**
*active* :: *'l*
**begin**

**inductive** *step* :: *'f inference set* × (*'f* × *'l*) *set* ⇒
*'f inference set* × (*'f* × *'l*) *set* ⇒ *bool* (**infix** ⇝*LGC* *50*) **where**
*process*: *N1* = *N* ∪ *M* ⟹ *N2* = *N* ∪ *M'* ⟹ *M* ⊆ *Red-F* (*N* ∪ *M'*) ⟹
*active-subset M'* = {} ⟹ (*T*, *N1*) ⇝*LGC* (*T*, *N2*) |
*schedule-infer*: *T2* = *T1* ∪ *T'* ⟹ *N1* = *N* ∪ {(*C*, *L*)} ⟹ *N2* = *N* ∪ {(*C*, *active*)} ⟹
*L* ≠ *active* ⟹ *T'* = *no-labels.Inf-between* (*fst* ‘ *active-subset N*) {*C*} ⟹
(*T1*, *N1*) ⇝*LGC* (*T2*, *N2*) |
*compute-infer*: *T1* = *T2* ∪ {*ι*} ⟹ *N2* = *N1* ∪ *M* ⟹ *active-subset M* = {} ⟹
*ι* ∈ *no-labels.Red-I* (*fst* ‘ (*N1* ∪ *M*)) ⟹ (*T1*, *N1*) ⇝*LGC* (*T2*, *N2*) |
*delete-orphan-infers*: *T1* = *T2* ∪ *T'* ⟹
*T'* ∩ *no-labels.Inf-from* (*fst* ‘ *active-subset N*) = {} ⟹ (*T1*, *N*) ⇝*LGC* (*T2*, *N*)

**lemma** *premise-free-inf-always-from*: *ι* ∈ *Inf-F* ⟹ *prems-of ι* = [] ⟹ *ι* ∈ *no-labels.Inf-from N*
⟨*proof*⟩

**lemma** *one-step-equiv*: (*T1*, *N1*) ⇝*LGC* (*T2*, *N2*) ⟹ *N1* ▷*L* *N2*
⟨*proof*⟩

**lemma** *lgc-to-red*: *chain* (⇝*LGC*) *Ns* ⟹ *chain* (▷*L*) (*lmap snd Ns*)
⟨*proof*⟩

**lemma** *lgc-fair*:
**assumes**
*deriv*: *chain* (⇝*LGC*) *Ns* **and**
*init-state*: *active-subset* (*snd* (*lhd Ns*)) = {} **and**
*final-state*: *passive-subset* (*Liminf-llist* (*lmap snd Ns*)) = {} **and**
*no-prems-init*: ∀ *ι* ∈ *Inf-F*. *prems-of ι* = [] ⟶ *ι* ∈ *fst* (*lhd Ns*) **and**
*final-schedule*: *Liminf-llist* (*lmap fst Ns*) = {}
**shows** *fair* (*lmap snd Ns*)
⟨*proof*⟩

**theorem** *lgc-complete-Liminf*:
**assumes**

*deriv*: *chain* ($\rightsquigarrow$*LGC*) *Ns* **and**
*init-state*: *active-subset* (*snd* (*lhd Ns*)) = {} **and**
*final-state*: *passive-subset* (*Liminf-llist* (*lmap snd Ns*)) = {} **and**
*no-prems-init*: $\forall\, \iota \in$ *Inf-F*. *prems-of* $\iota = [] \longrightarrow \iota \in$ *fst* (*lhd Ns*) **and**
*final-schedule*: *Liminf-llist* (*lmap fst Ns*) = {} **and**
*b-in*: *B* $\in$ *Bot-F* **and**
*bot-entailed*: *no-labels.entails-*$\mathcal{G}$ (*fst* ‘ *snd* (*lhd Ns*)) {*B*}
**shows** $\exists$ *BL* $\in$ *Bot-FL*. *BL* $\in$ *Liminf-llist* (*lmap snd Ns*)
⟨*proof*⟩

**theorem** *lgc-complete*:
**assumes**
*deriv*: *chain* ($\rightsquigarrow$*LGC*) *Ns* **and**
*init-state*: *active-subset* (*snd* (*lhd Ns*)) = {} **and**
*final-state*: *passive-subset* (*Liminf-llist* (*lmap snd Ns*)) = {} **and**
*no-prems-init*: $\forall\, \iota \in$ *Inf-F*. *prems-of* $\iota = [] \longrightarrow \iota \in$ *fst* (*lhd Ns*) **and**
*final-schedule*: *Liminf-llist* (*lmap fst Ns*) = {} **and**
*b-in*: *B* $\in$ *Bot-F* **and**
*bot-entailed*: *no-labels.entails-*$\mathcal{G}$ (*fst* ‘ *snd* (*lhd Ns*)) {*B*}
**shows** $\exists$ *i*. *enat i* < *llength Ns* $\wedge$ ($\exists$ *BL* $\in$ *Bot-FL*. *BL* $\in$ *snd* (*lnth Ns i*))
⟨*proof*⟩

**end**

**end**