

# A Comprehensive Framework for Saturation Theorem Proving

Sophie Touret

May 26, 2024

## 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 Touret, 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](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

<b>1</b>	<b>Calculi Based on a Redundancy Criterion</b>	<b>1</b>
1.1	Consequence Relations . . . . .	1
1.2	Families of Consequence Relations . . . . .	2
1.3	Inference Systems . . . . .	2
1.4	Families of Inference Systems . . . . .	3
1.5	Calculi Based on a Single Redundancy Criterion . . . . .	3
<b>2</b>	<b>Calculi Based on the Intersection of Redundancy Criteria</b>	<b>6</b>
2.1	Calculi with a Family of Redundancy Criteria . . . . .	6
<b>3</b>	<b>Variations on a Theme</b>	<b>7</b>
<b>4</b>	<b>Lifting to Non-ground Calculi</b>	<b>10</b>
4.1	Standard Lifting . . . . .	10
4.2	Strong Standard Lifting . . . . .	11
4.3	Lifting with a Family of Tiebreaker Orderings . . . . .	12
4.4	Lifting with a Family of Redundancy Criteria . . . . .	14
<b>5</b>	<b>Labeled Lifting to Non-Ground Calculi</b>	<b>17</b>
5.1	Labeled Lifting with a Family of Tiebreaker Orderings . . . . .	17
5.2	Labeled Lifting with a Family of Redundancy Criteria . . . . .	18

<b>6</b>	<b>Given Clause Prover Architectures</b>	<b>20</b>
6.1	Basis of the Given Clause Prover Architectures . . . . .	21
6.2	Given Clause Procedure . . . . .	23
6.3	Lazy Given Clause Procedure . . . . .	24

## 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  $\Rightarrow$  'f set  $\Rightarrow$  bool (**infix**  $\models$  50)

**assumes**

*bot-not-empty*:  $Bot \neq \{\}$  **and**

*bot-entails-all*:  $B \in Bot \implies \{B\} \models N1$  **and**

*subset-entailed*:  $N2 \subseteq N1 \implies N1 \models N2$  **and**

*all-formulas-entailed*:  $(\forall C \in N2. N1 \models \{C\}) \implies N1 \models N2$  **and**

*entails-trans[trans]*:  $N1 \models N2 \implies N2 \models N3 \implies N1 \models N3$

**begin**

**lemma** *entail-set-all-formulas*:  $N1 \models N2 \iff (\forall C \in N2. N1 \models \{C\})$

*<proof>*

**lemma** *entail-union*:  $N \models N1 \wedge N \models N2 \iff N \models N1 \cup N2$

*<proof>*

**lemma** *entail-unions*:  $(\forall i \in I. N \models Ni \ i) \iff N \models \bigcup (Ni \ ' I)$

*<proof>*

**lemma** *entail-all-bot*:  $(\exists B \in Bot. N \models \{B\}) \implies \forall B' \in Bot. N \models \{B'\}$

*<proof>*

**lemma** *entails-trans-strong*:  $N1 \models N2 \implies N1 \cup N2 \models N3 \implies N1 \models N3$

*<proof>*

**end**

### 1.2 Families of Consequence Relations

**locale** *consequence-relation-family* =

**fixes**

*Bot* :: 'f set **and**

*Q* :: 'q set **and**

*entails-q* :: 'q  $\Rightarrow$  'f set  $\Rightarrow$  'f set  $\Rightarrow$  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*:  $Bot \neq \{\}$   
 $\langle proof \rangle$

**definition** *entails* ::  $'f\ set \Rightarrow 'f\ set \Rightarrow bool$  (**infix**  $\models_Q 50$ ) **where**  
 $N1 \models_Q N2 \iff (\forall q \in Q. \textit{entails-q } q\ N1\ N2)$

**lemma** *intersect-cons-rel-family*: *consequence-relation Bot entails*  
 $\langle proof \rangle$

**end**

### 1.3 Inference Systems

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

**locale** *inference-system* =  
**fixes**  
 $Inf :: \langle 'f\ inference\ set \rangle$   
**begin**

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

**definition** *Inf-between* ::  $'f\ set \Rightarrow 'f\ set \Rightarrow 'f\ inference\ set$  **where**  
 $Inf\text{-between } N\ M = Inf\text{-from } (N \cup M) - Inf\text{-from } (N - M)$

**lemma** *Inf-if-Inf-from*:  $\iota \in Inf\text{-from } N \implies \iota \in Inf$   
 $\langle proof \rangle$

**lemma** *Inf-if-Inf-between*:  $\iota \in Inf\text{-between } N\ M \implies \iota \in Inf$   
 $\langle proof \rangle$

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

**lemma** *Inf-from-mono*:  $N \subseteq N' \implies Inf\text{-from } N \subseteq Inf\text{-from } N'$   
 $\langle proof \rangle$

**lemma** *Inf-between-mono*:  $N \subseteq N' \implies M \subseteq M' \implies Inf\text{-between } N\ M \subseteq Inf\text{-between } N'\ M'$   
 $\langle proof \rangle$

**end**

### 1.4 Families of Inference Systems

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

$Inf-q :: 'q \Rightarrow 'f \text{ inference set}$   
**assumes**  
 $Q\text{-nonempty}: Q \neq \{\}$   
**begin**  
**definition**  $Inf\text{-from-}q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ inference set}$  **where**  
 $Inf\text{-from-}q \ q = \text{inference-system}.Inf\text{-from} \ (Inf-q \ q)$   
**definition**  $Inf\text{-between-}q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ inference set}$  **where**  
 $Inf\text{-between-}q \ q = \text{inference-system}.Inf\text{-between} \ (Inf-q \ q)$   
**lemma**  $Inf\text{-between-}q\text{-alt}$ :  
 $Inf\text{-between-}q \ q \ N \ M = \{\iota \in Inf-q \ q. \iota \in Inf\text{-from-}q \ q \ (N \cup M) \wedge \text{set} \ (\text{prems-of } \iota) \cap M \neq \{\}\}$   
 $\langle \text{proof} \rangle$   
**end**

## 1.5 Calculi Based on a Single Redundancy Criterion

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

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

**lemma**  $red\text{-concl-to-red-inf}$ :  
**assumes**  
 $i\text{-in}: \iota \in Inf$  **and**  
 $\text{concl}: \text{concl-of } \iota \in Red-F \ N$   
**shows**  $\iota \in Red-I \ N$   
 $\langle \text{proof} \rangle$

**definition**  $\text{saturated} :: 'f \text{ set} \Rightarrow \text{bool}$  **where**  
 $\text{saturated} \ N \longleftrightarrow Inf\text{-from} \ N \subseteq Red-I \ N$

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

**lemma**  $Red-I\text{-without-red-F}$ :  
 $Red-I \ (N - Red-F \ N) = Red-I \ N$

⟨proof⟩

**lemma** *saturated-without-red-F*:  
 **assumes** *saturated*: *saturated N*  
 **shows** *saturated* ( $N - \text{Red-F } N$ )  
⟨proof⟩

**definition** *fair* :: '*f set llist*  $\Rightarrow$  *bool* **where**  
 *fair Ns*  $\longleftrightarrow$  *Inf-from* (*Liminf-llist Ns*)  $\subseteq$  *Sup-llist* (*lmap Red-I Ns*)

**inductive** *derive* :: '*f set*  $\Rightarrow$  '*f set*  $\Rightarrow$  *bool* (**infix**  $\triangleright$  50) **where**  
 *derive*:  $M - N \subseteq \text{Red-F } N \Longrightarrow M \triangleright N$

**lemma** *gt-Max-notin*:  $\langle \text{finite } A \Longrightarrow A \neq \{\} \Longrightarrow x > \text{Max } A \Longrightarrow x \notin A \rangle$  ⟨proof⟩

**lemma** *equiv-Sup-Liminf*:  
 **assumes**  
 *in-Sup*:  $C \in \text{Sup-llist } Ns$  **and**  
 *not-in-Liminf*:  $C \notin \text{Liminf-llist } Ns$   
 **shows**  
  $\exists i \in \{i. \text{enat } (\text{Suc } i) < \text{llength } Ns\}. C \in \text{lth } Ns \ i - \text{lth } Ns \ (\text{Suc } i)$   
⟨proof⟩

**lemma** *Red-in-Sup*:  
 **assumes** *deriv*:  $\langle \text{chain } (\triangleright) \ Ns \rangle$   
 **shows**  $\text{Sup-llist } Ns - \text{Liminf-llist } Ns \subseteq \text{Red-F } (\text{Sup-llist } Ns)$   
⟨proof⟩

**lemma** *Red-I-subset-Liminf*:  
 **assumes** *deriv*:  $\langle \text{chain } (\triangleright) \ Ns \rangle$  **and**  
 *i*:  $\langle \text{enat } i < \text{llength } Ns \rangle$   
 **shows**  $\langle \text{Red-I } (\text{lth } Ns \ i) \subseteq \text{Red-I } (\text{Liminf-llist } Ns) \rangle$   
⟨proof⟩

**lemma** *Red-F-subset-Liminf*:  
 **assumes** *deriv*:  $\langle \text{chain } (\triangleright) \ Ns \rangle$  **and**  
 *i*:  $\langle \text{enat } i < \text{llength } Ns \rangle$   
 **shows**  $\langle \text{Red-F } (\text{lth } Ns \ i) \subseteq \text{Red-F } (\text{Liminf-llist } Ns) \rangle$   
⟨proof⟩

**lemma** *i-in-Liminf-or-Red-F*:  
 **assumes**  
 *deriv*:  $\langle \text{chain } (\triangleright) \ Ns \rangle$  **and**  
 *i*:  $\langle \text{enat } i < \text{llength } Ns \rangle$   
 **shows**  $\langle \text{lth } Ns \ i \subseteq \text{Red-F } (\text{Liminf-llist } Ns) \cup \text{Liminf-llist } Ns \rangle$   
⟨proof⟩

**lemma** *fair-implies-Liminf-saturated*:  
 **assumes**  
 *deriv*:  $\langle \text{chain } (\triangleright) \ Ns \rangle$  **and**

```

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

end

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

lemma dynamically-complete-Liminf:
  fixes B Ns
  assumes
    bot-lem: ⟨ $B \in Bot$ ⟩ and
    deriv: ⟨chain (▷) Ns⟩ and
    fair: ⟨fair Ns⟩ and
    unsat: ⟨ $lhd Ns \models \{B\}$ ⟩
  shows ⟨ $\exists B' \in Bot. B' \in Liminf-llist Ns$ ⟩
  ⟨proof⟩

end

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

sublocale statically-complete-calculus
  ⟨proof⟩

end

sublocale statically-complete-calculus  $\subseteq$  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

```

$Red-F :: 'f\ set \Rightarrow 'f\ set$   
+ **assumes**  
 $inf-in-red-inf: Inf-between\ UNIV\ (Red-F\ N) \subseteq Red-I\ N$   
**begin**

**lemma** *sat-eq-reduc-sat: saturated N  $\longleftrightarrow$  reduc-saturated N*  
 $\langle proof \rangle$

**end**

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

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

**sublocale** *reducedly-statically-complete-calculus*  
 $\langle proof \rangle$

**sublocale** *statically-complete-calculus*  
 $\langle proof \rangle$

**end**

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

**lemma** *stat-ref-comp-imp-red-stat-ref-comp:*  
 $statically-complete-calculus\ Bot\ Inf\ entails\ Red-I\ Red-F \Longrightarrow$   
 $reducedly-statically-complete-calculus\ Bot\ Inf\ entails\ Red-I\ Red-F$   
 $\langle proof \rangle$

**end**

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

**definition** *Red-Red-I :: 'f set  $\Rightarrow$  'f inference set where*  
 $Red-Red-I\ N = Red-I\ N \cup Inf-between\ UNIV\ (Red-F\ N)$

**lemma** *reduced-calc-is-calc: calculus Bot Inf entails Red-Red-I Red-F*  
 $\langle proof \rangle$

**lemma** *inf-sub-reduced-red-inf: Inf-between UNIV (Red-F N)  $\subseteq$  Red-Red-I N*  
 $\langle proof \rangle$

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  $\implies$  reduc-calc.saturated N*

*<proof>*

**lemma** *red-sat-eg-red-calc-sat: reduc-saturated N  $\longleftrightarrow$  reduc-calc.saturated N*

*<proof>*

**lemma** *red-sat-eg-sat: reduc-saturated N  $\longleftrightarrow$  saturated (N - Red-F N)*

*<proof>*

**theorem** *stat-is-stat-red: statically-complete-calculus Bot Inf entails Red-I Red-F  $\longleftrightarrow$   
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  $\longleftrightarrow$   
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  $\longleftrightarrow$   
statically-complete-calculus Bot Inf entails Red-Red-I Red-F*

*<proof>*

**lemma** *sup-red-f-in-red-liminf:*

*chain derive Ns  $\implies$  Sup-llist (lmap Red-F Ns)  $\subseteq$  Red-F (Liminf-llist Ns)*

*<proof>*

**lemma** *sup-red-inf-in-red-liminf:*

*chain derive Ns  $\implies$  Sup-llist (lmap Red-I Ns)  $\subseteq$  Red-I (Liminf-llist Ns)*

*<proof>*

**definition** *reduc-fair :: 'f set llist  $\implies$  bool where*

*reduc-fair Ns  $\longleftrightarrow$*

*Inf-from (Liminf-llist Ns - Sup-llist (lmap Red-F Ns))  $\subseteq$  Sup-llist (lmap Red-I Ns)*

**lemma** *reduc-fair-imp-Liminf-reduc-sat:*

*chain derive Ns  $\implies$  reduc-fair Ns  $\implies$  reduc-saturated (Liminf-llist Ns)*

*<proof>*

**end**

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

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

**begin**

**sublocale** *reducedly-statically-complete-calculus*  
 $\langle proof \rangle$

**end**

**sublocale** *reducedly-statically-complete-calculus*  $\subseteq$  *reducedly-dynamically-complete-calculus*  
 $\langle proof \rangle$

**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*  
 $\langle proof \rangle$

**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*  
 $\langle proof \rangle$

**interpretation** *reduc-calc: reduced-calculus Bot Inf entails Red-Red-I Red-F*  
 $\langle proof \rangle$

**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*  
 $\langle proof \rangle$

**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*  
 $\langle proof \rangle$

**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*  
 $\langle proof \rangle$

**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* ::  $\langle 'f \text{ inference set} \rangle$  **and**

*Bot-G* ::  $\langle 'g \text{ set} \rangle$  **and**

*Inf-G* ::  $\langle 'g \text{ inference set} \rangle$  **and**

*entails-G* ::  $\langle 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool} \rangle$  (**infix**  $\models_G$  50) **and**

*Red-I-G* ::  $\langle 'g \text{ set} \Rightarrow 'g \text{ inference set} \rangle$  **and**

*Red-F-G* ::  $\langle 'g \text{ set} \Rightarrow 'g \text{ set} \rangle$

+ **fixes**

*Bot-F* ::  $\langle 'f \text{ set} \rangle$  **and**

*G-F* ::  $\langle 'f \Rightarrow 'g \text{ set} \rangle$  **and**

*G-I* ::  $\langle 'f \text{ inference} \Rightarrow 'g \text{ inference set option} \rangle$

**assumes**

*Bot-F-not-empty*:  $\text{Bot-F} \neq \{\}$  **and**

*Bot-map-not-empty*:  $\langle B \in \text{Bot-F} \Longrightarrow \text{G-F } B \neq \{\} \rangle$  **and**

*Bot-map*:  $\langle B \in \text{Bot-F} \Longrightarrow \text{G-F } B \subseteq \text{Bot-G} \rangle$  **and**

*Bot-cond*:  $\langle \text{G-F } C \cap \text{Bot-G} \neq \{\} \longrightarrow C \in \text{Bot-F} \rangle$  **and**

*inf-map*:  $\langle \iota \in \text{Inf-F} \Longrightarrow \text{G-I } \iota \neq \text{None} \Longrightarrow \text{the } (\text{G-I } \iota) \subseteq \text{Red-I-G } (\text{G-F } (\text{concl-of } \iota)) \rangle$

**begin**

**abbreviation** *G-Fset* ::  $\langle 'f \text{ set} \Rightarrow 'g \text{ set} \rangle$  **where**

$\langle \text{G-Fset } N \equiv \bigcup (\text{G-F } 'N) \rangle$

**lemma** *G-subset*:  $\langle N1 \subseteq N2 \Longrightarrow \text{G-Fset } N1 \subseteq \text{G-Fset } N2 \rangle$   $\langle \text{proof} \rangle$

**abbreviation** *entails-G* ::  $\langle 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool} \rangle$  (**infix**  $\models_G$  50) **where**

$\langle N1 \models_G N2 \equiv \text{G-Fset } N1 \models_G \text{G-Fset } N2 \rangle$

**lemma** *subs-Bot-G-entails*:

**assumes**

*not-empty*:  $\langle sB \neq \{\} \rangle$  **and**

*in-bot*:  $\langle sB \subseteq \text{Bot-G} \rangle$

**shows**  $\langle sB \models_G N \rangle$

$\langle \text{proof} \rangle$

**sublocale** *consequence-relation* *Bot-F* *entails-G*

$\langle \text{proof} \rangle$

**definition** *Red-I-G* ::  $\langle 'f \text{ set} \Rightarrow 'f \text{ inference set} \rangle$  **where**

$\langle \text{Red-I-G } N = \{ \iota \in \text{Inf-F}. (\text{G-I } \iota \neq \text{None} \wedge \text{the } (\text{G-I } \iota) \subseteq \text{Red-I-G } (\text{G-Fset } N)) \}$

$\vee (\text{G-I } \iota = \text{None} \wedge \text{G-F } (\text{concl-of } \iota) \subseteq \text{G-Fset } N \cup \text{Red-F-G } (\text{G-Fset } N)) \rangle$

**definition** *Red-F-G* ::  $\langle 'f \text{ set} \Rightarrow 'f \text{ set} \rangle$  **where**

$\langle \text{Red-F-G } N = \{ C. \forall D \in \text{G-F } C. D \in \text{Red-F-G } (\text{G-Fset } N) \} \rangle$

**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**  $\models^G$  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* ( $\models^G$ ) *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**  $\models^G$  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**  
*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-F*set *N*) ∨ (∃ *E* ∈ *N*. *Prec-F-g* *D* *E* *C* ∧ *D* ∈ *G-F* *E*)}⟩

**lemma** *Prec-trans*:

**assumes**  
⟨*Prec-F-g* *D* *A* *B*⟩ **and**

$\langle \text{Prec-F-g } D B C \rangle$

**shows**

$\langle \text{Prec-F-g } D A C \rangle$

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

**lemma** *Red-F-G-equiv-def*:

$\langle \text{Red-F-G } N = \{C. \forall Di \in \mathcal{G}\text{-F } C. Di \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \vee$   
 $(\exists E \in (N - \text{Red-F-G } N). \text{Prec-F-g } Di E C \wedge Di \in \mathcal{G}\text{-F } E)\} \rangle$

$\langle \text{proof} \rangle$

**lemma** *not-red-map-in-map-not-red*:  $\langle \mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \subseteq \mathcal{G}\text{-Fset } (N - \text{Red-F-G } N) \rangle$

$\langle \text{proof} \rangle$

**lemma** *Red-F-Bot-F*:  $\langle B \in \text{Bot-F} \implies N \models_{\mathcal{G}} \{B\} \implies N - \text{Red-F-G } N \models_{\mathcal{G}} \{B\} \rangle$

$\langle \text{proof} \rangle$

**lemma** *Red-F-of-subset-F*:  $\langle N \subseteq N' \implies \text{Red-F-G } N \subseteq \text{Red-F-G } N' \rangle$

$\langle \text{proof} \rangle$

**lemma** *Red-I-of-subset-F*:  $\langle N \subseteq N' \implies \text{Red-I-G } N \subseteq \text{Red-I-G } N' \rangle$

$\langle \text{proof} \rangle$

**lemma** *Red-F-of-Red-F-subset-F*:  $\langle N' \subseteq \text{Red-F-G } N \implies \text{Red-F-G } N \subseteq \text{Red-F-G } (N - N') \rangle$

$\langle \text{proof} \rangle$

**lemma** *Red-I-of-Red-F-subset-F*:  $\langle N' \subseteq \text{Red-F-G } N \implies \text{Red-I-G } N \subseteq \text{Red-I-G } (N - N') \rangle$

$\langle \text{proof} \rangle$

**lemma** *Red-I-of-Inf-to-N-F*:

**assumes**

*i-in*:  $\langle \iota \in \text{Inf-F} \rangle$  **and**

*concl-i-in*:  $\langle \text{concl-of } \iota \in N \rangle$

**shows**

$\langle \iota \in \text{Red-I-G } N \rangle$

$\langle \text{proof} \rangle$

**sublocale** *calculus Bot-F Inf-F entails-G Red-I-G Red-F-G*

$\langle \text{proof} \rangle$

**end**

**lemma** *wf-empty-rel: minimal-element*  $(\lambda - . \text{False}) \text{ UNIV}$

$\langle \text{proof} \rangle$

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

**lemma** *red-f-equiv*: *empt-ord.Red-F-G = Red-F-G*  
 ⟨*proof*⟩

**sublocale** *calc?*: *calculus Bot-F Inf-F entails-G Red-I-G Red-F-G*  
 ⟨*proof*⟩

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

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

**lemma** *sat-inf-imp-ground-red*:

**assumes**

*saturated*  $N$  **and**

$\iota' \in \text{Inf-from } N$  **and**

$\mathcal{G}\text{-I } \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I } \iota')$

**shows**  $\iota \in \text{Red-I-G } (\mathcal{G}\text{-Fset } N)$

⟨*proof*⟩

**lemma** *sat-imp-ground-sat*:

*saturated*  $N \implies \text{ground-Inf-overapproximated } N \implies \text{ground.saturated } (\mathcal{G}\text{-Fset } N)$

⟨*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*:  $\bigwedge N. \text{saturated } N \implies \text{ground-Inf-overapproximated } N$

**shows**

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

⟨*proof*⟩

**end**

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

**lemma** *saturated-empty-order-equiv-saturated*:  
*saturated*  $N = \text{calc.}$ *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*:  
 $\forall q \in Q. \text{tiebreaker-lifting } \text{Bot-F } \text{Inf-F } \text{Bot-G } (\text{entails-q } q) (\text{Inf-G-q } q) (\text{Red-I-q } q)$   
 $(\text{Red-F-q } q) (\text{G-F-q } q) (\text{G-I-q } q) \text{Prec-F-g}$

**begin**

**abbreviation**  $\mathcal{G}\text{-Fset-}q$  :: 'q ⇒ 'f *set* ⇒ 'g *set* **where**  
 $\mathcal{G}\text{-Fset-}q \ q \ N \equiv \bigcup (\mathcal{G}\text{-F-q } q \ 'N)$

**definition**  $\text{Red-I-}\mathcal{G}\text{-}q$  :: 'q ⇒ 'f *set* ⇒ 'f *inference set* **where**  
 $\text{Red-I-}\mathcal{G}\text{-}q \ q \ N = \{\iota \in \text{Inf-F}. (\mathcal{G}\text{-I-q } q \ \iota \neq \text{None} \wedge \text{the } (\mathcal{G}\text{-I-q } q \ \iota) \subseteq \text{Red-I-q } q \ (\mathcal{G}\text{-Fset-q } q \ N))$   
 $\vee (\mathcal{G}\text{-I-q } q \ \iota = \text{None} \wedge \mathcal{G}\text{-F-q } q \ (\text{concl-of } \iota) \subseteq (\mathcal{G}\text{-Fset-q } q \ N \cup \text{Red-F-q } q \ (\mathcal{G}\text{-Fset-q } q \ N)))\}$

**definition**  $\text{Red-F-}\mathcal{G}\text{-empty-}q$  :: 'q ⇒ 'f *set* ⇒ 'f *set* **where**  
 $\text{Red-F-}\mathcal{G}\text{-empty-}q \ q \ N = \{C. \forall D \in \mathcal{G}\text{-F-q } q \ C. D \in \text{Red-F-q } q \ (\mathcal{G}\text{-Fset-q } q \ N)\}$

**definition**  $\text{Red-F-}\mathcal{G}\text{-}q$  :: 'q ⇒ 'f *set* ⇒ 'f *set* **where**  
 $\text{Red-F-}\mathcal{G}\text{-}q \ q \ N =$   
 $\{C. \forall D \in \mathcal{G}\text{-F-q } q \ C. D \in \text{Red-F-q } q \ (\mathcal{G}\text{-Fset-q } q \ N) \vee (\exists E \in N. \text{Prec-F-g } D \ E \ C \wedge D \in \mathcal{G}\text{-F-q } q$

$E)\}$

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

**lemma** *red-crit-lifting-family*:

**assumes**  $q\text{-in}: q \in Q$

**shows**  $\text{calculus Bot-F Inf-F} \ (\text{entails-}\mathcal{G}\text{-}q \ q) \ (\text{Red-I-}\mathcal{G}\text{-}q \ q) \ (\text{Red-F-}\mathcal{G}\text{-}q \ q)$

$\langle \text{proof} \rangle$

**lemma** *red-crit-lifting-family-empty-ord*:

**assumes**  $q\text{-in}: q \in Q$

**shows**  $\text{calculus Bot-F Inf-F} \ (\text{entails-}\mathcal{G}\text{-}q \ q) \ (\text{Red-I-}\mathcal{G}\text{-}q \ q) \ (\text{Red-F-}\mathcal{G}\text{-empty-}q \ q)$

$\langle \text{proof} \rangle$

**sublocale** *consequence-relation-family Bot-F Q entails-}\mathcal{G}\text{-}q*

$\langle \text{proof} \rangle$

**sublocale** *intersection-calculus Bot-F Inf-F Q entails-}\mathcal{G}\text{-}q \ \text{Red-I-}\mathcal{G}\text{-}q \ \text{Red-F-}\mathcal{G}\text{-}q*

$\langle \text{proof} \rangle$

**abbreviation**  $\text{entails-}\mathcal{G} :: 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool}$  (**infix**  $\models \cap \mathcal{G}$  50) **where**

$(\models \cap \mathcal{G}) \equiv \text{entails}$

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

$\text{Red-I-}\mathcal{G} \equiv \text{Red-I}$

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

$\text{Red-F-}\mathcal{G} \equiv \text{Red-F}$

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

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

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

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

$\langle \text{proof} \rangle$

**abbreviation**  $\text{Red-F-}\mathcal{G}\text{-empty} :: 'f \text{ set} \Rightarrow 'f \text{ set}$  **where**

$\text{Red-F-}\mathcal{G}\text{-empty} \equiv \text{empty-ord.Red-F}$

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

**lemma** *sat-inf-imp-ground-red-fam-inter*:

**assumes**

$\text{sat-n}: \text{saturated } N$  **and**

$i'\text{-in}: \iota' \in \text{Inf-from } N$  **and**

$q\text{-in}: q \in Q$  **and**

$\text{grounding}: \mathcal{G}\text{-I-}q \ q \ \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I-}q \ q \ \iota')$

**shows**  $\iota \in \text{Red-I-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$

$\langle \text{proof} \rangle$

**abbreviation**  $\text{ground-Inf-overapproximated} :: 'q \Rightarrow 'f \text{ set} \Rightarrow \text{bool}$  **where**

$\text{ground-Inf-overapproximated } q \ N \equiv$

$\text{ground.Inf-from-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$

$\subseteq \{\iota. \exists \iota' \in \text{Inf-from } N. \mathcal{G}\text{-I-}q \ q \ \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I-}q \ q \ \iota')\} \cup \text{Red-I-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$



**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  $\Rightarrow$  'g set  $\Rightarrow$  bool (**infix**  $\models_G$  50) **and**  
*Inf-G* :: 'g inference set **and**  
*Red-I-G* :: 'g set  $\Rightarrow$  'g inference set **and**  
*Red-F-G* :: 'g set  $\Rightarrow$  'g set **and**  
*G-F* :: 'f  $\Rightarrow$  'g set **and**  
*G-I* :: 'f inference  $\Rightarrow$  'g inference set option **and**  
*Prec-F* :: 'g  $\Rightarrow$  'f  $\Rightarrow$  'f  $\Rightarrow$  bool (**infix**  $\sqsubset$  50)

**+ fixes**

*Inf-FL* ::  $\langle ('f \times 'l)$  inference set  $\rangle$

**assumes**

*Inf-F-to-Inf-FL*:  $\langle \iota_F \in \text{Inf-F} \implies \text{length } (Ll :: 'l \text{ list}) = \text{length } (\text{prems-of } \iota_F) \implies$   
 $\exists L0. \text{Infer } (\text{zip } (\text{prems-of } \iota_F) Ll) (\text{concl-of } \iota_F, L0) \in \text{Inf-FL} \rangle$  **and**  
*Inf-FL-to-Inf-F*:  $\langle \iota_{FL} \in \text{Inf-FL} \implies \text{Infer } (\text{map fst } (\text{prems-of } \iota_{FL})) (\text{fst } (\text{concl-of } \iota_{FL})) \in \text{Inf-F} \rangle$

**begin**

**definition** *to-F* ::  $\langle ('f \times 'l)$  inference  $\Rightarrow$  'f inference  $\rangle$  **where**

$\langle \text{to-F } \iota_{FL} = \text{Infer } (\text{map fst } (\text{prems-of } \iota_{FL})) (\text{fst } (\text{concl-of } \iota_{FL})) \rangle$

**abbreviation** *Bot-FL* ::  $\langle ('f \times 'l)$  set  $\rangle$  **where**

$\langle \text{Bot-FL} \equiv \text{Bot-F} \times \text{UNIV} \rangle$

**abbreviation** *G-F-L* ::  $\langle ('f \times 'l)$   $\Rightarrow$  'g set  $\rangle$  **where**

$\langle \text{G-F-L } CL \equiv \text{G-F } (\text{fst } CL) \rangle$

**abbreviation** *G-I-L* ::  $\langle ('f \times 'l)$  inference  $\Rightarrow$  'g inference set option  $\rangle$  **where**

$\langle \text{G-I-L } \iota_{FL} \equiv \text{G-I } (\text{to-F } \iota_{FL}) \rangle$

**sublocale** *standard-lifting* *Inf-FL* *Bot-G* *Inf-G* ( $\models_G$ ) *Red-I-G* *Red-F-G* *Bot-FL* *G-F-L* *G-I-L*  
 $\langle \text{proof} \rangle$

**notation** *entails-G* (**infix**  $\models_G$  50)

**lemma** *labeled-entailment-lifting*:  $NL1 \models_G NL2 \iff \text{fst } 'NL1 \models_G \text{fst } 'NL2$   
 $\langle \text{proof} \rangle$

**lemma** *red-inf-impl*:  $\iota \in \text{Red-I-G } NL \implies \text{to-F } \iota \in \text{no-labels.Red-I-G } (\text{fst } 'NL)$   
 $\langle \text{proof} \rangle$

**lemma** *labeled-saturation-lifting*:  $\text{saturated } NL \implies \text{no-labels.saturated } (\text{fst } 'NL)$   
 $\langle \text{proof} \rangle$

**lemma** *stat-ref-comp-to-labeled-sta-ref-comp*:

**assumes** *static*:

*statically-complete-calculus* *Bot-F* *Inf-F* ( $\models_G$ ) *no-labels.Red-I-G* *no-labels.Red-F-G*

**shows** *statically-complete-calculus* *Bot-FL* *Inf-FL* ( $\models_G$ ) *Red-I-G* *Red-F-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 G-F-q G-I-q λg 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**

*G-F-q* :: 'q ⇒ 'f ⇒ 'g set **and**

*G-I-q* :: 'q ⇒ 'f inference ⇒ 'g inference set option

**+ fixes**

*Inf-FL* :: ⟨('f × 'l) inference set⟩

**assumes**

*Inf-F-to-Inf-FL*:

⟨ $\iota_F \in \text{Inf-F} \implies \text{length } (Ll :: 'l \text{ list}) = \text{length } (\text{prems-of } \iota_F) \implies$

$\exists L0. \text{Infer } (\text{zip } (\text{prems-of } \iota_F) Ll) (\text{concl-of } \iota_F, L0) \in \text{Inf-FL} \rangle$  **and**

*Inf-FL-to-Inf-F*: ⟨ $\iota_{FL} \in \text{Inf-FL} \implies \text{Infer } (\text{map fst } (\text{prems-of } \iota_{FL})) (\text{fst } (\text{concl-of } \iota_{FL})) \in \text{Inf-F} \rangle$

**begin**

**definition** *to-F* :: ⟨('f × 'l) inference ⇒ 'f inference⟩ **where**

⟨*to-F*  $\iota_{FL} = \text{Infer } (\text{map fst } (\text{prems-of } \iota_{FL})) (\text{fst } (\text{concl-of } \iota_{FL})) \rangle$

**abbreviation** *Bot-FL* :: ⟨('f × 'l) set⟩ **where**

⟨*Bot-FL*  $\equiv \text{Bot-F} \times \text{UNIV} \rangle$

**abbreviation** *G-F-L-q* :: ⟨'q ⇒ ('f × 'l) ⇒ 'g set⟩ **where**

⟨*G-F-L-q*  $q \text{ CL} \equiv \text{G-F-q } q (\text{fst } \text{CL}) \rangle$

**abbreviation** *G-I-L-q* :: ⟨'q ⇒ ('f × 'l) inference ⇒ 'g inference set option⟩ **where**

⟨*G-I-L-q*  $q \iota_{FL} \equiv \text{G-I-q } q (\text{to-F } \iota_{FL}) \rangle$

**abbreviation** *G-Fset-L-q* :: 'q ⇒ ('f × 'l) set ⇒ 'g set **where**

*G-Fset-L-q*  $q \text{ N} \equiv \bigcup (\text{G-F-L-q } q 'N)$

**definition** *Red-I-G-L-q* :: 'q ⇒ ('f × 'l) set ⇒ ('f × 'l) inference set **where**

*Red-I-G-L-q*  $q \text{ N} =$

$\{\iota \in \text{Inf-FL}. (\text{G-I-L-q } q \iota \neq \text{None} \wedge \text{the } (\text{G-I-L-q } q \iota) \subseteq \text{Red-I-q } q (\text{G-Fset-L-q } q \text{ N}))$

$\vee (\text{G-I-L-q } q \iota = \text{None} \wedge \text{G-F-L-q } q (\text{concl-of } \iota) \subseteq \text{G-Fset-L-q } q \text{ N} \cup \text{Red-F-q } q (\text{G-Fset-L-q } q \text{ N}))\}$

**abbreviation** *Red-I-G-L* :: ('f × 'l) set ⇒ ('f × 'l) inference set **where**

*Red-I-G-L*  $\text{N} \equiv (\bigcap q \in Q. \text{Red-I-G-L-q } q \text{ N})$

**abbreviation** *entails-G-L-q* :: 'q ⇒ ('f × 'l) set ⇒ ('f × 'l) set ⇒ bool **where**

*entails-G-L-q*  $q \text{ N1 } \text{N2} \equiv \text{entails-q } q (\text{G-Fset-L-q } q \text{ N1}) (\text{G-Fset-L-q } q \text{ N2})$

**lemma** *lifting-q*:

**assumes**  $q \in Q$

**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*  $q$ ) ( $\lambda g$   $Cl$   $Cl'$ . *False*) *Inf-FL*  
 $\langle proof \rangle$

**lemma** *lifted-q*:

**assumes** *q-in*:  $q \in Q$

**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*  $q$ )

$\langle proof \rangle$

**lemma** *ord-fam-lifted-q*:

**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*  $q$ ) ( $\lambda g$   $Cl$   $Cl'$ . *False*)

$\langle proof \rangle$

**definition** *Red-F-G-empty-L-q* :: ' $q \Rightarrow ('f \times 'l)$  set  $\Rightarrow ('f \times 'l)$  set **where**

*Red-F-G-empty-L-q*  $q$   $N = \{C. \forall D \in \mathcal{G}\text{-F-L-q } q$   $C. D \in \text{Red-F-q } q$  ( $\mathcal{G}\text{-Fset-L-q } q$   $N$ )  $\vee$   
 ( $\exists E \in N. \text{False} \wedge D \in \mathcal{G}\text{-F-L-q } q$   $E$ )}

**abbreviation** *Red-F-G-empty-L* :: (' $f \times 'l$ ) set  $\Rightarrow ('f \times 'l)$  set **where**

*Red-F-G-empty-L*  $N \equiv (\bigcap q \in Q. \text{Red-F-G-empty-L-q } q$   $N$ )

**lemma** *all-lifted-red-crit*:

**assumes** *q-in*:  $q \in Q$

**shows** *calculus*  $Bot-FL$   $Inf-FL$  (*entails-G-L-q*  $q$ ) (*Red-I-G-L-q*  $q$ ) (*Red-F-G-empty-L-q*  $q$ )

$\langle proof \rangle$

**lemma** *all-lifted-cons-rel*:

**assumes** *q-in*:  $q \in Q$

**shows** *consequence-relation*  $Bot-FL$  (*entails-G-L-q*  $q$ )

$\langle proof \rangle$

**sublocale** *consequence-relation-family*  $Bot-FL$   $Q$  *entails-G-L-q*

$\langle proof \rangle$

**sublocale** *intersection-calculus*  $Bot-FL$   $Inf-FL$   $Q$  *entails-G-L-q* *Red-I-G-L-q* *Red-F-G-empty-L-q*

$\langle proof \rangle$

**lemma** *in-Inf-FL-imp-to-F-in-Inf-F*:  $\iota \in \text{Inf-FL} \Longrightarrow \text{to-F } \iota \in \text{Inf-F}$

$\langle proof \rangle$

**lemma** *in-Inf-from-imp-to-F-in-Inf-from*:  $\iota \in \text{Inf-from } N \Longrightarrow \text{to-F } \iota \in \text{no-labels.Inf-from } (\text{fst } 'N)$

$\langle proof \rangle$

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

**abbreviation** *entails-G-L* :: (' $f \times 'l$ ) set  $\Rightarrow ('f \times 'l)$  set  $\Rightarrow \text{bool}$  (**infix**  $\models_{\mathcal{G}L} 50$ ) **where**

$(\models_{\mathcal{G}L}) \equiv \text{entails}$

**lemmas** *entails-G-L-def* = *entails-def*

**lemma** *labeled-entailment-lifting*:  $NL1 \models_{\mathcal{G}L} NL2 \iff \text{fst } 'NL1 \models_{\mathcal{G}} \text{fst } 'NL2$

$\langle proof \rangle$

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

**lemma** *labeled-family-saturation-lifting*:  $\text{saturated NL} \implies \text{no-labels.saturated (fst ' NL)}$   
 ⟨proof⟩

**theorem** *labeled-static-ref*:

**assumes** *calc*: *statically-complete-calculus Bot-F Inf-F* ( $\models \cap \mathcal{G}$ ) *no-labels.Red-I-G*  
*no-labels.Red-F-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 G-F-q G-I-q*

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

**for**

*Bot-F* :: 'f set

**and** *Inf-F* :: 'f inference set

**and** *Bot-G* :: 'g set

**and** *Q* :: 'q set

**and** *entails-q* :: 'q  $\Rightarrow$  'g set  $\Rightarrow$  'g set  $\Rightarrow$  bool

**and** *Inf-G-q* :: 'q  $\Rightarrow$  'g inference set

**and** *Red-I-q* :: 'q  $\Rightarrow$  'g set  $\Rightarrow$  'g inference set

**and** *Red-F-q* :: 'q  $\Rightarrow$  'g set  $\Rightarrow$  'g set

**and** *G-F-q* :: 'q  $\Rightarrow$  'f  $\Rightarrow$  'g set

**and** *G-I-q* :: 'q  $\Rightarrow$  'f inference  $\Rightarrow$  'g inference set option

**+ fixes**

*Equiv-F* :: 'f  $\Rightarrow$  'f  $\Rightarrow$  bool (**infix**  $\doteq$  50) **and**

*Prec-F* :: 'f  $\Rightarrow$  'f  $\Rightarrow$  bool (**infix**  $\prec$  50) **and**

*Prec-L* :: 'l  $\Rightarrow$  'l  $\Rightarrow$  bool (**infix**  $\sqsubset L$  50) **and**

*active* :: 'l

**assumes**

*equiv-equiv-F*: *equivp* ( $\doteq$ ) **and**

*wf-prec-F*: *minimal-element* ( $\prec$ ) *UNIV* **and**

*wf-prec-L*: *minimal-element* ( $\sqsubset L$ ) *UNIV* **and**

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

**begin**

**abbreviation** *Inf-FL* :: ( $'f \times 'l$ ) *inference set where*

*Inf-FL*  $\equiv \{ \iota_{FL}. \text{Infer } (\text{map } \text{fst } (\text{prems-of } \iota_{FL})) \ (\text{fst } (\text{concl-of } \iota_{FL})) \in \text{Inf-F} \}$

**abbreviation** *Prec-eq-F* ::  $'f \Rightarrow 'f \Rightarrow \text{bool}$  (**infix**  $\preceq \cdot$  50) **where**

$C \preceq \cdot D \equiv C \doteq D \vee C \prec \cdot D$

**definition** *Prec-FL* :: ( $'f \times 'l$ )  $\Rightarrow$  ( $'f \times 'l$ )  $\Rightarrow \text{bool}$  (**infix**  $\sqsubset$  50) **where**

$CL1 \sqsubset CL2 \iff \text{fst } CL1 \prec \cdot \text{fst } CL2 \vee (\text{fst } CL1 \doteq \text{fst } CL2 \wedge \text{snd } CL1 \sqsubset L \ \text{snd } CL2)$

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

$\langle \text{proof} \rangle$

**lemma** *trans-prec-F*:  $C1 \prec \cdot C2 \implies C2 \prec \cdot C3 \implies C1 \prec \cdot C3$

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

**definition** *active-subset* :: ( $'f \times 'l$ ) *set*  $\Rightarrow$  ( $'f \times 'l$ ) *set where*

*active-subset*  $M = \{ CL \in M. \text{snd } CL = \text{active} \}$

**definition** *passive-subset* :: ( $'f \times 'l$ ) *set*  $\Rightarrow$  ( $'f \times 'l$ ) *set where*

*passive-subset*  $M = \{ CL \in M. \text{snd } CL \neq \text{active} \}$

**lemma** *active-subset-insert[simp]*:

*active-subset* (*insert*  $Cl$   $N$ ) = (*if*  $\text{snd } Cl = \text{active}$  *then*  $\{Cl\}$  *else*  $\{\}$ )  $\cup$  *active-subset*  $N$

$\langle \text{proof} \rangle$

**lemma** *active-subset-union[simp]*: *active-subset* ( $M \cup N$ ) = *active-subset*  $M \cup$  *active-subset*  $N$

$\langle \text{proof} \rangle$

**lemma** *passive-subset-insert[simp]*:

*passive-subset* (*insert*  $Cl$   $N$ ) = (*if*  $\text{snd } Cl \neq \text{active}$  *then*  $\{Cl\}$  *else*  $\{\}$ )  $\cup$  *passive-subset*  $N$

$\langle \text{proof} \rangle$

**lemma** *passive-subset-union[simp]*: *passive-subset* ( $M \cup N$ ) = *passive-subset*  $M \cup$  *passive-subset*  $N$

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

**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*  $q$ ) (*G-F-L-q*  $q$ ) (*G-I-L-q*  $q$ ) ( $\lambda g. \text{Prec-FL}$ )

$\langle \text{proof} \rangle$

**sublocale** *lifting-intersection Inf-FL Bot-G Q Inf-G-q entails-q Red-I-q Red-F-q*  
*Bot-FL G-F-L-q G-I-L-q λg. Prec-FL*  
 ⟨proof⟩

**notation** *derive (infix ▷L 50)*

**lemma** *std-Red-I-eq: std.Red-I = Red-I-G*  
 ⟨proof⟩

**lemma** *std-Red-F-eq: std.Red-F = Red-F-G-empty*  
 ⟨proof⟩

**sublocale** *statically-complete-calculus Bot-FL Inf-FL (|=∩GL) Red-I Red-F*  
 ⟨proof⟩

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

**lemma** *red-labeled-clauses:*  
**assumes**  $\langle C ∈ no-labels.Red-F-G-empty (fst ' N) \vee$   
 $(\exists C' ∈ fst ' N. C' \prec \cdot C) \vee (\exists (C', L') ∈ N. L' \sqsubset L \wedge C' \preceq \cdot C) \rangle$   
**shows**  $\langle (C, L) ∈ Red-F N \rangle$   
 ⟨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 G-F-q 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*  
*G-F-q :: 'q ⇒ 'f ⇒ 'g set and*  
*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 +*  
**assumes**  
*inf-have-prems: ιF ∈ Inf-F ⇒ prems-of ιF ≠ []*  
**begin**

**lemma** *labeled-inf-have-prems: ι ∈ Inf-FL ⇒ prems-of ι ≠ []*  
 ⟨proof⟩

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

**lemma one-step-equiv**:  $N1 \rightsquigarrow GC N2 \Rightarrow N1 \triangleright L N2$   
<proof>

**lemma gc-to-red**: *chain* ( $\rightsquigarrow GC$ )  $Ns \Rightarrow$  *chain* ( $\triangleright L$ )  $Ns$   
<proof>

**lemma (in-)** *all-ex-finite-set*:  $(\forall (j::nat) \in \{0..<m\}. \exists (n::nat). P j n) \Rightarrow$   
 $(\forall n1 n2. \forall j \in \{0..<m\}. P j n1 \rightarrow P j n2 \rightarrow n1 = n2) \Rightarrow$  *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-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-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* *G-F-q* *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  $\Rightarrow$  'g set  $\Rightarrow$  'g set  $\Rightarrow$  bool **and**  
*Inf-G-q* :: <'q  $\Rightarrow$  'g inference set> **and**  
*Red-I-q* :: 'q  $\Rightarrow$  'g set  $\Rightarrow$  'g inference set **and**  
*Red-F-q* :: 'q  $\Rightarrow$  'g set  $\Rightarrow$  'g set **and**  
*G-F-q* :: 'q  $\Rightarrow$  'f  $\Rightarrow$  'g set **and**  
*G-I-q* :: 'q  $\Rightarrow$  'f inference  $\Rightarrow$  'g inference set option **and**  
*Equiv-F* :: 'f  $\Rightarrow$  'f  $\Rightarrow$  bool (**infix**  $\doteq$  50) **and**  
*Prec-F* :: 'f  $\Rightarrow$  'f  $\Rightarrow$  bool (**infix**  $\prec$  50) **and**  
*Prec-L* :: 'l  $\Rightarrow$  'l  $\Rightarrow$  bool (**infix**  $\sqsubseteq L$  50) **and**  
*active* :: 'l

**begin**

**inductive step** :: 'f inference set  $\times$  ('f  $\times$  'l) set  $\Rightarrow$   
'f inference set  $\times$  ('f  $\times$  'l) set  $\Rightarrow$  bool (**infix**  $\rightsquigarrow LGC$  50) **where**  
*process*:  $N1 = N \cup M \Rightarrow N2 = N \cup M' \Rightarrow M \subseteq Red-F (N \cup M') \Rightarrow$   
*active-subset*  $M' = \{\} \Rightarrow (T, N1) \rightsquigarrow LGC (T, N2) \mid$   
*schedule-infer*:  $T2 = T1 \cup T' \Rightarrow N1 = N \cup \{(C, L)\} \Rightarrow N2 = N \cup \{(C, active)\} \Rightarrow$   
 $L \neq active \Rightarrow T' = no-labels.Inf-between (fst 'active-subset N) \{C\} \Rightarrow$   
 $(T1, N1) \rightsquigarrow LGC (T2, N2) \mid$   
*compute-infer*:  $T1 = T2 \cup \{\iota\} \Rightarrow N2 = N1 \cup M \Rightarrow active-subset M = \{\} \Rightarrow$   
 $\iota \in no-labels.Red-I (fst ' (N1 \cup M)) \Rightarrow (T1, N1) \rightsquigarrow LGC (T2, N2) \mid$   
*delete-orphan-infers*:  $T1 = T2 \cup T' \Rightarrow$   
 $T' \cap no-labels.Inf-from (fst 'active-subset N) = \{\} \Rightarrow (T1, N) \rightsquigarrow LGC (T2, N)$

**lemma** *premise-free-inf-always-from*:  $\iota \in Inf-F \Rightarrow prems-of \iota = [] \Rightarrow \iota \in no-labels.Inf-from N$   
<proof>

**lemma** *one-step-equiv*:  $(T1, N1) \rightsquigarrow LGC (T2, N2) \Rightarrow N1 \triangleright L N2$   
<proof>

**lemma** *lgc-to-red*:  $chain (\rightsquigarrow LGC) Ns \Rightarrow chain (\triangleright L) (lmap snd Ns)$   
<proof>

**lemma** *lgc-fair*:

**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*:  $Liminf-llist (lmap fst Ns) = \{\}$

**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*:  $\forall \iota \in \text{Inf-}F. \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst} (\text{lhd } Ns)$  **and**  
*final-schedule*: *Liminf-llist* (*lmap fst Ns*) = {} **and**  
*b-in*:  $B \in \text{Bot-}F$  **and**  
*bot-entailed*: *no-labels.entails- $\mathcal{G}$*  (*fst ' snd* (*lhd Ns*)) {*B*}  
**shows**  $\exists BL \in \text{Bot-}FL. BL \in \text{Liminf-llist} (\text{lmap snd } Ns)$   
 <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*:  $\forall \iota \in \text{Inf-}F. \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst} (\text{lhd } Ns)$  **and**  
*final-schedule*: *Liminf-llist* (*lmap fst Ns*) = {} **and**  
*b-in*:  $B \in \text{Bot-}F$  **and**  
*bot-entailed*: *no-labels.entails- $\mathcal{G}$*  (*fst ' snd* (*lhd Ns*)) {*B*}  
**shows**  $\exists i. \text{enat } i < \text{llength } Ns \wedge (\exists BL \in \text{Bot-}FL. BL \in \text{snd} (\text{lth } Ns \ i))$   
 <proof>

**end**

**end**