

# 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>8</b>
2.1	Calculi with a Family of Redundancy Criteria . . . . .	9
<b>3</b>	<b>Variations on a Theme</b>	<b>13</b>
<b>4</b>	<b>Lifting to Non-ground Calculi</b>	<b>21</b>
4.1	Standard Lifting . . . . .	21
4.2	Strong Standard Lifting . . . . .	22
4.3	Lifting with a Family of Tiebreaker Orderings . . . . .	23
4.4	Lifting with a Family of Redundancy Criteria . . . . .	30
<b>5</b>	<b>Labeled Lifting to Non-Ground Calculi</b>	<b>34</b>
5.1	Labeled Lifting with a Family of Tiebreaker Orderings . . . . .	34
5.2	Labeled Lifting with a Family of Redundancy Criteria . . . . .	37

<b>6</b>	<b>Given Clause Prover Architectures</b>	<b>41</b>
6.1	Basis of the Given Clause Prover Architectures . . . . .	41
6.2	Given Clause Procedure . . . . .	48
6.3	Lazy Given Clause Procedure . . . . .	54

## 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*:  $N1 \models N2 \iff (\forall C \in N2. N1 \models \{C\})$

**by** (*meson all-formulas-entailed empty-subsetI insert-subset subset-entailed entails-trans*)

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

**using** *entail-set-all-formulas*[*of N N1*] *entail-set-all-formulas*[*of N N2*]

*entail-set-all-formulas*[*of N N1*  $\cup$  *N2*] **by** *blast*

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

**using** *entail-set-all-formulas*[*of N*  $\bigcup$  (*Ni* ' *I*)] *entail-set-all-formulas*[*of N*]

*Complete-Lattices.UN-ball-beq-simps(2)*[*of Ni I*  $\lambda C. N \models \{C\}$ , *symmetric*]

**by** *meson*

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

**using** *bot-entails-all entails-trans* **by** *blast*

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

**by** (*meson entail-union entails-trans order-refl subset-entailed*)

**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 ≠ {} and
  q-cons-rel: ∀ q ∈ Q. consequence-relation Bot (entails-q q)
begin

lemma bot-not-empty: Bot ≠ {}
  using Q-nonempty consequence-relation.bot-not-empty consequence-relation-family.q-cons-rel
  consequence-relation-family-axioms by blast

definition entails :: 'f set ⇒ 'f set ⇒ bool (infix |=Q 50) where
  N1 |=Q N2 ⟷ (∀ q ∈ Q. entails-q q N1 N2)

lemma intersect-cons-rel-family: consequence-relation Bot entails
  unfolding consequence-relation-def entails-def
  by (intro conjI bot-not-empty) (metis consequence-relation-def q-cons-rel)+

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 = {ι ∈ Inf. set (prems-of ι) ⊆ N}

definition Inf-between :: 'f set ⇒ 'f set ⇒ 'f inference set where
  Inf-between N M = Inf-from (N ∪ M) − Inf-from (N − M)

lemma Inf-if-Inf-from: ι ∈ Inf-from N ⟹ ι ∈ Inf
  unfolding Inf-from-def by simp

lemma Inf-if-Inf-between: ι ∈ Inf-between N M ⟹ ι ∈ Inf
  unfolding Inf-between-def Inf-from-def by simp

lemma Inf-between-alt:
  Inf-between N M = {ι ∈ Inf. ι ∈ Inf-from (N ∪ M) ∧ set (prems-of ι) ∩ M ≠ {}}
  unfolding Inf-from-def Inf-between-def by auto

lemma Inf-from-mono: N ⊆ N' ⟹ Inf-from N ⊆ Inf-from N'
  unfolding Inf-from-def by auto

lemma Inf-between-mono: N ⊆ N' ⟹ M ⊆ M' ⟹ Inf-between N M ⊆ Inf-between N' M'
  unfolding Inf-between-alt using Inf-from-mono[of N ∪ M N' ∪ M'] by auto

end

```

## 1.4 Families of Inference Systems

**locale** *inference-system-family* =

**fixes**

$Q :: 'q \text{ set}$  **and**

$Inf\text{-}q :: 'q \Rightarrow 'f \text{ inference set}$

**assumes**

$Q\text{-nonempty}: Q \neq \{\}$

**begin**

**definition**  $Inf\text{-from-}q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ inference set}$  **where**

$Inf\text{-from-}q \ q = \text{inference-system.}Inf\text{-from} \ (Inf\text{-}q \ q)$

**definition**  $Inf\text{-between-}q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ inference set}$  **where**

$Inf\text{-between-}q \ q = \text{inference-system.}Inf\text{-between} \ (Inf\text{-}q \ q)$

**lemma**  $Inf\text{-between-}q\text{-alt}$ :

$Inf\text{-between-}q \ q \ N \ M = \{\iota \in Inf\text{-}q \ q. \iota \in Inf\text{-from-}q \ q \ (N \cup M) \wedge \text{set} \ (\text{prems-of } \iota) \cap M \neq \{\}\}$

**unfolding**  $Inf\text{-from-}q\text{-def}$   $Inf\text{-between-}q\text{-def}$   $\text{inference-system.}Inf\text{-between-}alt$  **by** *auto*

**end**

## 1.5 Calculi Based on a Single Redundancy Criterion

**locale** *calculus* = *inference-system*  $Inf$  + *consequence-relation*  $Bot$  *entails*

**for**

$Bot :: 'f \text{ set}$  **and**

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

$\text{entails} :: 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool}$  (**infix**  $\models$  50)

+ **fixes**

$Red\text{-}I :: 'f \text{ set} \Rightarrow 'f \text{ inference set}$  **and**

$Red\text{-}F :: 'f \text{ set} \Rightarrow 'f \text{ set}$

**assumes**

$Red\text{-}I\text{-to-}Inf: Red\text{-}I \ N \subseteq Inf$  **and**

$Red\text{-}F\text{-}Bot: B \in Bot \Longrightarrow N \models \{B\} \Longrightarrow N - Red\text{-}F \ N \models \{B\}$  **and**

$Red\text{-}F\text{-of-}subset: N \subseteq N' \Longrightarrow Red\text{-}F \ N \subseteq Red\text{-}F \ N'$  **and**

$Red\text{-}I\text{-of-}subset: N \subseteq N' \Longrightarrow Red\text{-}I \ N \subseteq Red\text{-}I \ N'$  **and**

$Red\text{-}F\text{-of-}Red\text{-}F\text{-subset}: N' \subseteq Red\text{-}F \ N \Longrightarrow Red\text{-}F \ N \subseteq Red\text{-}F \ (N - N')$  **and**

$Red\text{-}I\text{-of-}Red\text{-}F\text{-subset}: N' \subseteq Red\text{-}F \ N \Longrightarrow Red\text{-}I \ N \subseteq Red\text{-}I \ (N - N')$  **and**

$Red\text{-}I\text{-of-}Inf\text{-to-}N: \iota \in Inf \Longrightarrow \text{concl-of } \iota \in N \Longrightarrow \iota \in Red\text{-}I \ N$

**begin**

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

**using**  $Red\text{-}I\text{-of-}Inf\text{-to-}N$  **by** *blast*

**lemma**  $red\text{-concl-to-red-inf}$ :

**assumes**

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

$\text{concl}: \text{concl-of } \iota \in Red\text{-}F \ N$

**shows**  $\iota \in Red\text{-}I \ N$

**proof** –

**have**  $\iota \in Red\text{-}I \ (Red\text{-}F \ N)$  **by** (*simp add: Red-I-of-Inf-to-N i-in concl*)

**then have**  $i\text{-in-}Red: \iota \in Red\text{-}I \ (N \cup Red\text{-}F \ N)$  **by** (*simp add: Red-I-of-Inf-to-N concl i-in*)

**have**  $red\text{-}n\text{-subs}: Red\text{-}F \ N \subseteq Red\text{-}F \ (N \cup Red\text{-}F \ N)$  **by** (*simp add: Red-F-of-subset*)

**then have**  $\iota \in Red\text{-}I \ ((N \cup Red\text{-}F \ N) - (Red\text{-}F \ N - N))$  **using**  $Red\text{-}I\text{-of-}Red\text{-}F\text{-subset}$   $i\text{-in-}Red$

by (*meson Diff-subset subsetCE subset-trans*)  
**then show** *?thesis* **by** (*metis Diff-cancel Diff-subset Un-Diff Un-Diff-cancel contra-subsetD calculus.Red-I-of-subset calculus-axioms sup-bot.right-neutral*)  
**qed**

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

**definition** *reduc-saturated* :: 'f set  $\Rightarrow$  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  
**using** *Red-I-of-subset [of N - Red-F N N]*  
**and** *Red-I-of-Red-F-subset [of Red-F N N]* **by** *blast*

**lemma** *saturated-without-red-F*:  
**assumes** *saturated: saturated N*  
**shows** *saturated (N - Red-F N)*

**proof** –  
**have** *Inf-from (N - Red-F N)  $\subseteq$  Inf-from N* **unfolding** *Inf-from-def* **by** *auto*  
**also have** *Inf-from N  $\subseteq$  Red-I N* **using** *saturated* **unfolding** *saturated-def* **by** *auto*  
**also have** *Red-I N  $\subseteq$  Red-I (N - Red-F N)* **using** *Red-I-of-Red-F-subset* **by** *auto*  
**finally have** *Inf-from (N - Red-F N)  $\subseteq$  Red-I (N - Red-F N)* **by** *auto*  
**then show** *?thesis* **unfolding** *saturated-def* **by** *auto*  
**qed**

**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  $\subseteq$  Red-F N  $\Longrightarrow$  M  $\triangleright$  N*

**lemma** *gt-Max-notin*:  $\langle$ finite A  $\Longrightarrow$  A  $\neq$   $\{\}$   $\Longrightarrow$  x > Max A  $\Longrightarrow$  x  $\notin$  A  $\rangle$  **by** *auto*

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

**proof** –  
**obtain** *i* **where** *C-D-i: C  $\in$  Sup-upto-llist Ns (enat i)* **and** *enat i < llength Ns*  
**using** *elem-Sup-llist-imp-Sup-upto-llist in-Sup* **by** *fast*  
**then obtain** *j* **where** *j: j  $\geq$  i  $\wedge$  enat j < llength Ns  $\wedge$  C  $\notin$  lth Ns j*  
**using** *not-in-Liminf* **unfolding** *Sup-upto-llist-def chain-def Liminf-llist-def* **by** *auto*  
**obtain** *k* **where** *k: C  $\in$  lth Ns k enat k < llength Ns k  $\leq$  i* **using** *C-D-i*  
**unfolding** *Sup-upto-llist-def* **by** *auto*  
**let** *?S =  $\{i. i < j \wedge i \geq k \wedge C \in \text{lth } Ns\ i\}$*   
**define** *l* **where** *l = Max ?S*  
**have**  $\langle k \in \{i. i < j \wedge k \leq i \wedge C \in \text{lth } Ns\ i\} \rangle$  **using** *k j* **by** (*auto simp: order.order-iff-strict*)  
**then have** *nempty:  $\{i. i < j \wedge k \leq i \wedge C \in \text{lth } Ns\ i\} \neq \{\}$*  **by** *auto*  
**then have** *l-prop: l < j  $\wedge$  l  $\geq$  k  $\wedge$  C  $\in$  lth Ns l* **using** *Max-in[of ?S, OF - nempty]*  
**unfolding** *l-def* **by** *auto*  
**then have** *C  $\in$  lth Ns l - lth Ns (Suc l)* **using** *j gt-Max-notin[OF - nempty, of Suc l]*

```

  unfolding l-def[symmetric] by (auto intro: Suc-lessI)
then show ?thesis
proof (rule bexI[of - l])
  show  $l \in \{i. \text{enat } (Suc\ i) < \text{llength } Ns\}$ 
  using l-prop j
  by (clarify, metis Suc-leI dual-order.order-iff-strict enat-ord-simps(2) less-trans)
qed
qed

```

**lemma** *Red-in-Sup*:

```

  assumes deriv: chain ( $\triangleright$ ) Ns
  shows Sup-llist Ns – Liminf-llist Ns  $\subseteq$  Red-F (Sup-llist Ns)
proof
  fix C
  assume C-in-subset: C  $\in$  Sup-llist Ns – Liminf-llist Ns
  {
    fix C i
    assume
      in-ith-elem: C  $\in$  lnth Ns i – lnth Ns (Suc i) and
      i: enat (Suc i) < llength Ns
    have lnth Ns i  $\triangleright$  lnth Ns (Suc i) using i deriv in-ith-elem chain-lnth-rel by auto
    then have C  $\in$  Red-F (lnth Ns (Suc i)) using in-ith-elem derive.cases by blast
    then have C  $\in$  Red-F (Sup-llist Ns) using Red-F-of-subset
    by (meson contra-subsetD i lnth-subset-Sup-llist)
  }
  then show C  $\in$  Red-F (Sup-llist Ns) using equiv-Sup-Liminf[of C] C-in-subset by fast
qed

```

**lemma** *Red-I-subset-Liminf*:

```

  assumes deriv:  $\langle$ chain ( $\triangleright$ ) Ns $\rangle$  and
  i:  $\langle$ enat i < llength Ns $\rangle$ 
  shows  $\langle$ Red-I (lnth Ns i)  $\subseteq$  Red-I (Liminf-llist Ns) $\rangle$ 
proof –
  have Sup-in-diff:  $\langle$ Red-I (Sup-llist Ns)  $\subseteq$  Red-I (Sup-llist Ns – (Sup-llist Ns – Liminf-llist Ns)) $\rangle$ 
  using Red-I-of-Red-F-subset[OF Red-in-Sup] deriv by auto
  also have  $\langle$ Sup-llist Ns – (Sup-llist Ns – Liminf-llist Ns) = Liminf-llist Ns $\rangle$ 
  by (simp add: Liminf-llist-subset-Sup-llist double-diff)
  then have Red-I-Sup-in-Liminf:  $\langle$ Red-I (Sup-llist Ns)  $\subseteq$  Red-I (Liminf-llist Ns) $\rangle$ 
  using Sup-in-diff by auto
  have  $\langle$ lnth Ns i  $\subseteq$  Sup-llist Ns $\rangle$  unfolding Sup-llist-def using i by blast
  then have Red-I (lnth Ns i)  $\subseteq$  Red-I (Sup-llist Ns) using Red-I-of-subset
  unfolding Sup-llist-def by auto
  then show ?thesis using Red-I-Sup-in-Liminf by auto
qed

```

**lemma** *Red-F-subset-Liminf*:

```

  assumes deriv:  $\langle$ chain ( $\triangleright$ ) Ns $\rangle$  and
  i:  $\langle$ enat i < llength Ns $\rangle$ 
  shows  $\langle$ Red-F (lnth Ns i)  $\subseteq$  Red-F (Liminf-llist Ns) $\rangle$ 
proof –
  have Sup-in-diff:  $\langle$ Red-F (Sup-llist Ns)  $\subseteq$  Red-F (Sup-llist Ns – (Sup-llist Ns – Liminf-llist Ns)) $\rangle$ 
  using Red-F-of-Red-F-subset[OF Red-in-Sup] deriv by auto

```

**also have**  $\langle \text{Sup-llist } Ns - (\text{Sup-llist } Ns - \text{Liminf-llist } Ns) = \text{Liminf-llist } Ns \rangle$   
**by** (*simp add: Liminf-llist-subset-Sup-llist double-diff*)  
**then have**  $\text{Red-F-Sup-in-Liminf}: \langle \text{Red-F } (\text{Sup-llist } Ns) \subseteq \text{Red-F } (\text{Liminf-llist } Ns) \rangle$   
**using** *Sup-in-diff* **by** *auto*  
**have**  $\langle \text{lnth } Ns \ i \subseteq \text{Sup-llist } Ns \rangle$  **unfolding** *Sup-llist-def* **using** *i* **by** *blast*  
**then have**  $\text{Red-F } (\text{lnth } Ns \ i) \subseteq \text{Red-F } (\text{Sup-llist } Ns)$  **using** *Red-F-of-subset*  
**unfolding** *Sup-llist-def* **by** *auto*  
**then show** *?thesis* **using** *Red-F-Sup-in-Liminf* **by** *auto*  
**qed**

**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**  $\langle \text{lnth } Ns \ i \subseteq \text{Red-F } (\text{Liminf-llist } Ns) \cup \text{Liminf-llist } Ns \rangle$   
**proof** (*rule,rule*)  
**fix** *C*  
**assume** *C*:  $\langle C \in \text{lnth } Ns \ i \rangle$   
**and** *C-not-Liminf*:  $\langle C \notin \text{Liminf-llist } Ns \rangle$   
**have**  $\langle C \in \text{Sup-llist } Ns \rangle$  **unfolding** *Sup-llist-def* **using** *C i* **by** *auto*  
**then obtain** *j* **where** *j*:  $\langle C \in \text{lnth } Ns \ j - \text{lnth } Ns \ (\text{Suc } j) \rangle \langle \text{enat } (\text{Suc } j) < \text{llength } Ns \rangle$   
**using** *equiv-Sup-Liminf[of C Ns]* *C-not-Liminf* **by** *auto*  
**then have**  $\langle C \in \text{Red-F } (\text{lnth } Ns \ (\text{Suc } j)) \rangle$   
**using** *deriv* **by** (*meson chain-lnth-rel contra-subsetD derive.cases*)  
**then show**  $\langle C \in \text{Red-F } (\text{Liminf-llist } Ns) \rangle$  **using** *Red-F-subset-Liminf[of Ns Suc j]* *deriv j(2)* **by** *blast*  
**qed**

**lemma** *fair-implies-Liminf-saturated:*

**assumes**  
*deriv*:  $\langle \text{chain } (\triangleright) \ Ns \rangle$  **and**  
*fair*:  $\langle \text{fair } Ns \rangle$   
**shows**  $\langle \text{saturated } (\text{Liminf-llist } Ns) \rangle$   
**unfolding** *saturated-def*  
**proof**  
**fix** *ι*  
**assume** *ι*:  $\langle \iota \in \text{Inf-from } (\text{Liminf-llist } Ns) \rangle$   
**have**  $\langle \iota \in \text{Sup-llist } (\text{lmap } \text{Red-I } Ns) \rangle$  **using** *fair ι* **unfolding** *fair-def* **by** *auto*  
**then obtain** *i* **where** *i*:  $\langle \text{enat } i < \text{llength } Ns \rangle \langle \iota \in \text{Red-I } (\text{lnth } Ns \ i) \rangle$   
**unfolding** *Sup-llist-def* **by** *auto*  
**then show**  $\langle \iota \in \text{Red-I } (\text{Liminf-llist } Ns) \rangle$   
**using** *deriv i-in-Liminf-or-Red-F[of Ns i]* *Red-I-subset-Liminf* **by** *blast*  
**qed**

**end**

**locale** *statically-complete-calculus = calculus +*

**assumes** *statically-complete*:  $B \in \text{Bot} \implies \text{saturated } N \implies N \models \{B\} \implies \exists B' \in \text{Bot}. B' \in N$   
**begin**

**lemma** *dynamically-complete-Liminf:*

**fixes** *B Ns*  
**assumes**  
*bot-elem*:  $\langle B \in \text{Bot} \rangle$  **and**

*deriv*:  $\langle \text{chain } (\triangleright) \text{ } Ns \rangle$  **and**  
*fair*:  $\langle \text{fair } Ns \rangle$  **and**  
*unsat*:  $\langle \text{lhd } Ns \models \{B\} \rangle$   
**shows**  $\langle \exists B' \in \text{Bot}. B' \in \text{Liminf-llist } Ns \rangle$   
**proof** –  
**note** *lhd-is* = *lhd-conv-lnth*[*OF chain-not-lnull*[*OF deriv*]]  
**have** *non-empty*:  $\langle \neg \text{lnull } Ns \rangle$  **using** *chain-not-lnull*[*OF deriv*] .  
**have** *subs*:  $\langle \text{lhd } Ns \subseteq \text{Sup-llist } Ns \rangle$   
**using** *lhd-subset-Sup-llist*[*of Ns*] *non-empty* **by** (*simp add: lhd-conv-lnth*)  
**have**  $\langle \text{Sup-llist } Ns \models \{B\} \rangle$   
**using** *unsat subset-entailed*[*OF subs*] *entails-trans*[*of Sup-llist Ns lhd Ns*] **by** *auto*  
**then have** *Sup-no-Red*:  $\langle \text{Sup-llist } Ns - \text{Red-F } (\text{Sup-llist } Ns) \models \{B\} \rangle$   
**using** *bot-elem Red-F-Bot* **by** *auto*  
**have** *Sup-no-Red-in-Liminf*:  $\langle \text{Sup-llist } Ns - \text{Red-F } (\text{Sup-llist } Ns) \subseteq \text{Liminf-llist } Ns \rangle$   
**using** *deriv Red-in-Sup* **by** *auto*  
**have** *Liminf-entails-Bot*:  $\langle \text{Liminf-llist } Ns \models \{B\} \rangle$   
**using** *Sup-no-Red subset-entailed*[*OF Sup-no-Red-in-Liminf*] *entails-trans* **by** *blast*  
**have**  $\langle \text{saturated } (\text{Liminf-llist } Ns) \rangle$   
**using** *deriv fair fair-implies-Liminf-saturated* **unfolding** *saturated-def* **by** *auto*  
**then show** *?thesis*  
**using** *bot-elem statically-complete Liminf-entails-Bot* **by** *auto*  
**qed**

**end**

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

**sublocale** *statically-complete-calculus*

**proof**

**fix** *B N*  
**assume**  
*bot-elem*:  $\langle B \in \text{Bot} \rangle$  **and**  
*saturated-N*: *saturated N* **and**  
*refut-N*:  $N \models \{B\}$   
**define** *Ns* **where**  $Ns = \text{LCons } N \text{ LNil}$   
**have**[*simp*]:  $\langle \neg \text{lnull } Ns \rangle$  **by** (*auto simp: Ns-def*)  
**have** *deriv-Ns*:  $\langle \text{chain } (\triangleright) \text{ } Ns \rangle$  **by** (*simp add: chain.chain-singleton Ns-def*)  
**have** *liminf-is-N*:  $\text{Liminf-llist } Ns = N$  **by** (*simp add: Ns-def Liminf-llist-LCons*)  
**have** *head-Ns*:  $N = \text{lhd } Ns$  **by** (*simp add: Ns-def*)  
**have** *Sup-llist* (*lmap Red-I Ns*) = *Red-I N* **by** (*simp add: Ns-def*)  
**then have** *fair-Ns*: *fair Ns* **using** *saturated-N* **by** (*simp add: fair-def saturated-def liminf-is-N*)  
**obtain** *i B'* **where** *B'-is-bot*:  $\langle B' \in \text{Bot} \rangle$  **and** *B'-in*:  $B' \in \text{lnth } Ns \ i$  **and**  $\langle i < \text{llength } Ns \rangle$   
**using** *dynamically-complete*[*of B Ns*] *bot-elem fair-Ns head-Ns saturated-N deriv-Ns refut-N*  
**by** *auto*  
**then have**  $i = 0$   
**by** (*auto simp: Ns-def enat-0-iff*)  
**show**  $\langle \exists B' \in \text{Bot}. B' \in N \rangle$   
**using** *B'-is-bot B'-in* **unfolding**  $\langle i = 0 \rangle$  *head-Ns*[*symmetric*] *Ns-def* **by** *auto*  
**qed**



end

**sublocale** *statically-complete-calculus*  $\subseteq$  *dynamically-complete-calculus*

**proof**

**fix**  $B\ Ns$

**assume**

$\langle B \in Bot \rangle$  **and**

$\langle chain (\triangleright) Ns \rangle$  **and**

$\langle fair Ns \rangle$  **and**

$\langle lhd Ns \models \{B\} \rangle$

**then have**  $\langle \exists B' \in Bot. B' \in Liminf\text{-}l\text{-}list\ Ns \rangle$

**by** (*rule dynamically-complete-Liminf*)

**then show**  $\langle \exists i \in \{i. enat\ i < llength\ Ns\}. \exists B' \in Bot. B' \in lnth\ Ns\ i \rangle$

**unfolding** *Liminf-l-list-def* **by** *auto*

qed

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**

$Inf :: \langle 'f\ inference\ set \rangle$  **and**

$Q :: 'q\ set$  **and**

$entails-q :: 'q \Rightarrow 'f\ set \Rightarrow 'f\ set \Rightarrow bool$

**+ fixes**

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

$Red-F-q :: 'q \Rightarrow 'f\ set \Rightarrow 'f\ set$

**assumes**

$Q\text{-nonempty}: Q \neq \{\}$  **and**

$all\text{-red-crit}: \forall q \in Q. calculus\ Bot\ Inf\ (entails-q\ q)\ (Red-I-q\ q)\ (Red-F-q\ q)$

**begin**

**definition** *Red-I* ::  $'f\ set \Rightarrow 'f\ inference\ set$  **where**

$Red-I\ N = (\bigcap q \in Q. Red-I-q\ q\ N)$

**definition** *Red-F* ::  $'f\ set \Rightarrow 'f\ set$  **where**

$Red-F\ N = (\bigcap q \in Q. Red-F-q\ q\ N)$

```

sublocale calculus Bot Inf entails Red-I Red-F
  unfolding calculus-def calculus-axioms-def
proof (intro conjI)
  show consequence-relation Bot entails
  using intersect-cons-rel-family .
next
show  $\forall N. \text{Red-I } N \subseteq \text{Inf}$ 
  unfolding Red-I-def
proof
  fix N
  show  $(\bigcap q \in Q. \text{Red-I-}q \ q \ N) \subseteq \text{Inf}$ 
  proof (intro Inter-subset)
  fix Red-Is
  assume one-red-inf: Red-Is  $\in (\lambda q. \text{Red-I-}q \ q \ N) \text{ ' } Q$ 
  show Red-Is  $\subseteq \text{Inf}$ 
  using one-red-inf all-red-crit calculus.Red-I-to-Inf by blast
next
  show  $(\lambda q. \text{Red-I-}q \ q \ N) \text{ ' } Q \neq \{\}$ 
  using Q-nonempty by blast
  qed
qed
next
show  $\forall B \ N. B \in \text{Bot} \longrightarrow N \models_Q \{B\} \longrightarrow N - \text{Red-F } N \models_Q \{B\}$ 
proof (intro allI impI)
  fix B N
  assume
  B-in:  $B \in \text{Bot}$  and
  N-unsat:  $N \models_Q \{B\}$ 
show  $N - \text{Red-F } N \models_Q \{B\}$  unfolding entails-def Red-F-def
proof
  fix qi
  assume qi-in:  $qi \in Q$ 
  define entails-qi (infix  $\models_{qi}$  50) where entails-qi = entails-q qi
  have cons-rel-qi: consequence-relation Bot entails-qi
  unfolding entails-qi-def using qi-in all-red-crit calculus.axioms(1) by blast
  define Red-F-qi where Red-F-qi = Red-F-q qi
  have red-qi-in:  $\text{Red-F } N \subseteq \text{Red-F-}qi \ N$ 
  unfolding Red-F-def Red-F-qi-def using qi-in image-iff by blast
  then have  $N - \text{Red-F-}qi \ N \subseteq N - \text{Red-F } N$  by blast
  then have entails-1:  $N - \text{Red-F } N \models_{qi} N - \text{Red-F-}qi \ N$ 
  using qi-in all-red-crit
  unfolding calculus-def consequence-relation-def entails-qi-def by metis
  have N-unsat-qi:  $N \models_{qi} \{B\}$  using qi-in N-unsat unfolding entails-qi-def entails-def
  by simp
  then have N-unsat-qi:  $N - \text{Red-F-}qi \ N \models_{qi} \{B\}$ 
  using qi-in all-red-crit Red-F-qi-def calculus.Red-F-Bot[OF B-in] entails-qi-def
  by fastforce
  show  $N - (\bigcap q \in Q. \text{Red-F-}q \ q \ N) \models_{qi} \{B\}$ 
  using consequence-relation.entails-trans[OF cons-rel-qi entails-1 N-unsat-qi]
  unfolding Red-F-def .
  qed
qed
next
show  $\forall N1 \ N2. N1 \subseteq N2 \longrightarrow \text{Red-F } N1 \subseteq \text{Red-F } N2$ 
proof (intro allI impI)

```

```

fix N1 :: 'f set
and N2 :: 'f set
assume
  N1-in-N2: N1 ⊆ N2
show Red-F N1 ⊆ Red-F N2
proof
  fix C
  assume C ∈ Red-F N1
  then have ∀ qi ∈ Q. C ∈ Red-F-q qi N1 unfolding Red-F-def by blast
  then have ∀ qi ∈ Q. C ∈ Red-F-q qi N2
    using N1-in-N2 all-red-crit calculus.axioms(2) calculus.Red-F-of-subset by blast
  then show C ∈ Red-F N2 unfolding Red-F-def by blast
qed
qed
next
show ∀ N1 N2. N1 ⊆ N2 ⟶ Red-I N1 ⊆ Red-I N2
proof (intro allI impI)
  fix N1 :: 'f set
  and N2 :: 'f set
  assume
    N1-in-N2: N1 ⊆ N2
  show Red-I N1 ⊆ Red-I N2
  proof
    fix ι
    assume ι ∈ Red-I N1
    then have ∀ qi ∈ Q. ι ∈ Red-I-q qi N1 unfolding Red-I-def by blast
    then have ∀ qi ∈ Q. ι ∈ Red-I-q qi N2
      using N1-in-N2 all-red-crit calculus.axioms(2) calculus.Red-I-of-subset by blast
    then show ι ∈ Red-I N2 unfolding Red-I-def by blast
  qed
qed
next
show ∀ N2 N1. N2 ⊆ Red-F N1 ⟶ Red-F N1 ⊆ Red-F (N1 - N2)
proof (intro allI impI)
  fix N2 N1
  assume N2-in-Red-N1: N2 ⊆ Red-F N1
  show Red-F N1 ⊆ Red-F (N1 - N2)
  proof
    fix C
    assume C ∈ Red-F N1
    then have ∀ qi ∈ Q. C ∈ Red-F-q qi N1 unfolding Red-F-def by blast
    moreover have ∀ qi ∈ Q. N2 ⊆ Red-F-q qi N1 using N2-in-Red-N1 unfolding Red-F-def by
blast
ultimately have ∀ qi ∈ Q. C ∈ Red-F-q qi (N1 - N2)
  using all-red-crit calculus.axioms(2) calculus.Red-F-of-Red-F-subset
  by blast
  then show C ∈ Red-F (N1 - N2) unfolding Red-F-def by blast
  qed
qed
next
show ∀ N2 N1. N2 ⊆ Red-F N1 ⟶ Red-I N1 ⊆ Red-I (N1 - N2)
proof (intro allI impI)
  fix N2 N1
  assume N2-in-Red-N1: N2 ⊆ Red-F N1
  show Red-I N1 ⊆ Red-I (N1 - N2)

```

```

proof
  fix  $\iota$ 
  assume  $\iota \in \text{Red-I } N1$ 
  then have  $\forall qi \in Q. \iota \in \text{Red-I-q } qi \ N1$  unfolding Red-I-def by blast
  moreover have  $\forall qi \in Q. N2 \subseteq \text{Red-F-q } qi \ N1$  using N2-in-Red-N1 unfolding Red-F-def by
blast
  ultimately have  $\forall qi \in Q. \iota \in \text{Red-I-q } qi \ (N1 - N2)$ 
  using all-red-crit calculus.axioms(2) calculus.Red-I-of-Red-F-subset by blast
  then show  $\iota \in \text{Red-I } (N1 - N2)$  unfolding Red-I-def by blast
  qed
qed
next
show  $\forall \iota N. \iota \in \text{Inf} \longrightarrow \text{concl-of } \iota \in N \longrightarrow \iota \in \text{Red-I } N$ 
proof (intro allI impI)
  fix  $\iota N$ 
  assume
    i-in:  $\iota \in \text{Inf}$  and
    concl-in:  $\text{concl-of } \iota \in N$ 
  then have  $\forall qi \in Q. \iota \in \text{Red-I-q } qi \ N$ 
  using all-red-crit calculus.axioms(2) calculus.Red-I-of-Inf-to-N by blast
  then show  $\iota \in \text{Red-I } N$  unfolding Red-I-def by blast
  qed
qed

```

```

lemma sat-int-to-sat-q: calculus.saturated Inf Red-I N  $\longleftrightarrow$ 
  ( $\forall qi \in Q. \text{calculus.saturated Inf (Red-I-q } qi) \ N$ ) for  $N$ 
proof
  fix  $N$ 
  assume inter-sat: calculus.saturated Inf Red-I N
  show  $\forall qi \in Q. \text{calculus.saturated Inf (Red-I-q } qi) \ N$ 
  proof
    fix  $qi$ 
    assume qi-in:  $qi \in Q$ 
    then interpret one: calculus Bot Inf entails-q qi Red-I-q qi Red-F-q qi
    by (metis all-red-crit)
    show one.saturated N
    using qi-in inter-sat
    unfolding one.saturated-def saturated-def Red-I-def by blast
  qed
next
  fix  $N$ 
  assume all-sat:  $\forall qi \in Q. \text{calculus.saturated Inf (Red-I-q } qi) \ N$ 
  show saturated N
  unfolding saturated-def Red-I-def
  proof
    fix  $\iota$ 
    assume  $\iota$ -in:  $\iota \in \text{Inf-from } N$ 
    have  $\forall \text{Red-I-qi} \in \text{Red-I-q } \iota \ Q. \iota \in \text{Red-I-qi } N$ 
    proof
      fix Red-I-qi
      assume red-inf-in:  $\text{Red-I-qi} \in \text{Red-I-q } \iota \ Q$ 
      then obtain qi where
        qi-in:  $qi \in Q$  and
        red-inf-qi-def:  $\text{Red-I-qi} = \text{Red-I-q } qi$  by blast

```

```

then interpret one: calculus Bot Inf entails-q qi Red-I-q qi Red-F-q qi
  by (metis all-red-crit)
have one.saturated N using qi-in all-sat red-inf-qi-def by blast
then show  $\iota \in \text{Red-I-qi } N$  unfolding one.saturated-def using \iota-in red-inf-qi-def by blast
qed
then show  $\iota \in (\bigcap q \in Q. \text{Red-I-q } q \ N)$  by blast
qed
qed

```

**lemma** *stat-ref-comp-from-bot-in-sat:*

```

 $(\forall N. \text{calculus.saturated Inf Red-I } N \wedge (\forall B \in \text{Bot}. B \notin N) \longrightarrow$ 
 $(\exists B \in \text{Bot}. \exists qi \in Q. \neg \text{entails-q } qi \ N \ \{B\})) \implies$ 
statically-complete-calculus Bot Inf entails Red-I Red-F

```

**proof** (*rule ccontr*)

**assume**

```

N-saturated: \forall N. (calculus.saturated Inf Red-I } N \wedge (\forall B \in \text{Bot}. B \notin N)) \longrightarrow
 $(\exists B \in \text{Bot}. \exists qi \in Q. \neg \text{entails-q } qi \ N \ \{B\})$  and

```

```

no-stat-ref-comp: \neg statically-complete-calculus Bot Inf (\models Q) Red-I Red-F

```

**obtain** *N1 B1 where B1-in:*

```

B1 \in Bot and N1-saturated: calculus.saturated Inf Red-I } N1 and

```

```

N1-unsat: N1 \models Q \{B1\} and no-B-in-N1: \forall B \in Bot. B \notin N1

```

```

using no-stat-ref-comp by (metis calculus-axioms statically-complete-calculus.intro
statically-complete-calculus-axioms.intro)

```

**obtain** *B2 qi where*

```

qi-in: qi \in Q and

```

```

no-qi: \neg entails-q qi } N1 \{B2\}

```

```

using N-saturated N1-saturated no-B-in-N1 by auto

```

**have**  $N1 \models Q \{B2\}$  **using** *N1-unsat B1-in intersect-cons-rel-family*

```

unfolding consequence-relation-def by metis

```

**then have**  $\text{entails-q } qi \ N1 \ \{B2\}$  **unfolding** *entails-def* **using** *qi-in* **by** *blast*

**then show** *False* **using** *no-qi* **by** *simp*

**qed**

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

```

```

Red-I :: 'f set \Rightarrow 'f inference set and

```

```

Red-F :: 'f set \Rightarrow '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
  fix N
  assume saturated N
  then show reduc-saturated N
    using Red-I-without-red-F saturated-without-red-F
    unfolding saturated-def reduc-saturated-def
    by blast
next
  fix N
  assume red-sat-n: reduc-saturated N
  show saturated N unfolding saturated-def
    using red-sat-n inf-in-red-inf unfolding reduc-saturated-def Inf-from-def Inf-between-def
    by blast
qed

end

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

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

sublocale reducedly-statically-complete-calculus
  by (simp add: calculus-axioms reducedly-statically-complete
    reducedly-statically-complete-calculus-axioms.intro
    reducedly-statically-complete-calculus-def)

sublocale statically-complete-calculus
proof
  fix B N
  assume
    bot-lem: ⟨B ∈ Bot⟩ and
    saturated-N: saturated N and
    refut-N: N ⊨ {B}
  have reduc-saturated-N: reduc-saturated N using saturated-N sat-eq-reduc-sat by blast
  show  $\exists B' \in Bot. B' \in N$  using reducedly-statically-complete[OF bot-lem reduc-saturated-N refut-N] .
qed

end

context reduced-calculus
begin

lemma stat-ref-comp-imp-red-stat-ref-comp:

```

*statically-complete-calculus Bot Inf entails Red-I Red-F*  $\implies$   
*reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F*

**proof**  
**fix**  $B N$   
**assume**  
*stat-ref-comp: statically-complete-calculus Bot Inf ( $\models$ ) Red-I Red-F* **and**  
*bot-elim:  $\langle B \in Bot \rangle$*  **and**  
*saturated-N: reduc-saturated N* **and**  
*refut-N:  $N \models \{B\}$*   
**have** *reduc-saturated-N: saturated N* **using** *saturated-N sat-eq-reduc-sat* **by** *blast*  
**show**  $\exists B' \in Bot. B' \in N$   
**using** *statically-complete-calculus.statically-complete[OF stat-ref-comp*  
*bot-elim reduc-saturated-N refut-N]* .  
**qed**

**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*

**proof**  
**fix**  $N$   
**show** *Red-Red-I N  $\subseteq$  Inf*  
**unfolding** *Red-Red-I-def Inf-between-def Inf-from-def* **using** *Red-I-to-Inf* **by** *auto*  
**next**  
**fix**  $B N$   
**assume**  
*b-in:  $B \in Bot$*  **and**  
*n-entails:  $N \models \{B\}$*   
**show**  $N - Red-F N \models \{B\}$   
**by** (*simp add: Red-F-Bot b-in n-entails*)  
**next**  
**fix**  $N N' :: 'f set$   
**assume**  $N \subseteq N'$   
**then show**  $Red-F N \subseteq Red-F N'$  **by** (*simp add: Red-F-of-subset*)  
**next**  
**fix**  $N N' :: 'f set$   
**assume**  $n-in: N \subseteq N'$   
**then have**  $Inf-from (UNIV - (Red-F N')) \subseteq Inf-from (UNIV - (Red-F N))$   
**using** *Red-F-of-subset[OF n-in]* **unfolding** *Inf-from-def* **by** *auto*  
**then have**  $Inf-between UNIV (Red-F N) \subseteq Inf-between UNIV (Red-F N')$   
**unfolding** *Inf-between-def* **by** *auto*  
**then show**  $Red-Red-I N \subseteq Red-Red-I N'$   
**unfolding** *Red-Red-I-def* **using** *Red-I-of-subset[OF n-in]* **by** *blast*  
**next**  
**fix**  $N N' :: 'f set$   
**assume**  $N' \subseteq Red-F N$   
**then show**  $Red-F N \subseteq Red-F (N - N')$  **by** (*simp add: Red-F-of-Red-F-subset*)  
**next**  
**fix**  $N N' :: 'f set$   
**assume**  $np-subst: N' \subseteq Red-F N$

```

have Red-F  $N \subseteq \text{Red-F } (N - N')$  by (simp add: Red-F-of-Red-F-subset np-subst)
then have Inf-from ( $UNIV - (\text{Red-F } (N - N'))$ )  $\subseteq$  Inf-from ( $UNIV - (\text{Red-F } N)$ )
  by (metis Diff-subset Red-F-of-subset eq-iff)
then have Inf-between  $UNIV (\text{Red-F } N) \subseteq$  Inf-between  $UNIV (\text{Red-F } (N - N'))$ 
  unfolding Inf-between-def by auto
then show Red-Red-I  $N \subseteq \text{Red-Red-I } (N - N')$ 
  unfolding Red-Red-I-def using Red-I-of-Red-F-subset[OF np-subst] by blast
next
fix  $\iota \in N$ 
assume  $\iota \in \text{Inf}$ 
  concl-of  $\iota \in N$ 
then show  $\iota \in \text{Red-Red-I } N$ 
  by (simp add: Red-I-of-Inf-to-N Red-Red-I-def)
qed

```

```

lemma inf-subst-reduced-red-inf: Inf-between  $UNIV (\text{Red-F } N) \subseteq \text{Red-Red-I } N$ 
  unfolding Red-Red-I-def by simp

```

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
  using inf-subst-reduced-red-inf reduced-calc-is-calc
  by (simp add: reduced-calculus.intro reduced-calculus-axioms-def)

```

```

interpretation reduc-calc: reduced-calculus Bot Inf entails Red-Red-I Red-F
  by (fact reduc-calc)

```

```

lemma sat-imp-red-calc-sat: saturated  $N \implies \text{reduc-calc.saturated } N$ 
  unfolding saturated-def reduc-calc.saturated-def Red-Red-I-def by blast

```

```

lemma red-sat-eq-red-calc-sat: reduc-saturated  $N \longleftrightarrow \text{reduc-calc.saturated } N$ 

```

```

proof
assume red-sat-n: reduc-saturated  $N$ 
show reduc-calc.saturated  $N$ 
  unfolding reduc-calc.saturated-def
proof
fix  $\iota$ 
assume i-in:  $\iota \in \text{Inf-from } N$ 
show  $\iota \in \text{Red-Red-I } N$ 
  using i-in red-sat-n
  unfolding reduc-saturated-def Inf-between-def Inf-from-def Red-Red-I-def by blast
qed

```

```

next
assume red-sat-n: reduc-calc.saturated  $N$ 
show reduc-saturated  $N$ 
  unfolding reduc-saturated-def
proof
fix  $\iota$ 
assume i-in:  $\iota \in \text{Inf-from } (N - \text{Red-F } N)$ 
show  $\iota \in \text{Red-I } N$ 
  using i-in red-sat-n
  unfolding Inf-from-def reduc-calc.saturated-def Red-Red-I-def Inf-between-def by blast
qed

```



qed

**lemma** *red-sat-eq-sat*:  $\text{reduc-saturated } N \longleftrightarrow \text{saturated } (N - \text{Red-F } N)$   
**unfolding** *reduc-saturated-def saturated-def* **by** (*simp add: Red-I-without-red-F*)

**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**

**assume**

*stat-ref1*: *statically-complete-calculus Bot Inf entails Red-I Red-F*

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

**using** *reduc-calc.calculus-axioms*

**unfolding** *statically-complete-calculus-def statically-complete-calculus-axioms-def*

**proof**

**show**  $\forall B N. B \in \text{Bot} \longrightarrow \text{reduc-calc.saturated } N \longrightarrow N \models \{B\} \longrightarrow (\exists B' \in \text{Bot}. B' \in N)$

**proof** (*clarify*)

**fix**  $B N$

**assume**

*b-in*:  $B \in \text{Bot}$  **and**

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

*n-imp-b*:  $N \models \{B\}$

**have**  $\text{saturated } (N - \text{Red-F } N)$  **using** *red-sat-eq-red-calc-sat[of N] red-sat-eq-sat[of N] n-sat* **by**

*blast*

**moreover have**  $(N - \text{Red-F } N) \models \{B\}$  **using** *n-imp-b b-in* **by** (*simp add: reduc-calc.Red-F-Bot*)

**ultimately show**  $\exists B' \in \text{Bot}. B' \in N$

**using** *stat-ref1* **by** (*meson DiffD1 b-in statically-complete-calculus.statically-complete*)

qed

qed

**next**

**assume**

*stat-ref3*: *statically-complete-calculus Bot Inf entails Red-Red-I Red-F*

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

**unfolding** *statically-complete-calculus-def statically-complete-calculus-axioms-def*

**using** *calculus-axioms*

**proof**

**show**  $\forall B N. B \in \text{Bot} \longrightarrow \text{saturated } N \longrightarrow N \models \{B\} \longrightarrow (\exists B' \in \text{Bot}. B' \in N)$

**proof** *clarify*

**fix**  $B N$

**assume**

*b-in*:  $B \in \text{Bot}$  **and**

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

*n-imp-b*:  $N \models \{B\}$

**then show**  $\exists B' \in \text{Bot}. B' \in N$

**using** *stat-ref3 sat-imp-red-calc-sat[OF n-sat]*

**by** (*meson statically-complete-calculus.statically-complete*)

qed

qed

qed

**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*

**using** *reduc-calc.stat-ref-comp-imp-red-stat-ref-comp*  
**by** (*metis reduc-calc.sat-eq-reduc-sat reducedly-statically-complete-calculus.axioms(2)*  
*reducedly-statically-complete-calculus-axioms-def reduced-calc-is-calc*  
*statically-complete-calculus.intro statically-complete-calculus-axioms.intro*)

**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*  
**using** *reduc-calc.calculus-axioms calculus-axioms red-sat-eq-red-calc-sat*  
**unfolding** *statically-complete-calculus-def statically-complete-calculus-axioms-def*  
*reducedly-statically-complete-calculus-def reducedly-statically-complete-calculus-axioms-def*  
**by** *blast*

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

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

**proof**

**fix** *N*

**assume**

*deriv: chain derive Ns and*

*n-in-sup: N  $\in$  Sup-llist (lmap Red-F Ns)*

**obtain** *i0* **where** *i-smaller: enat i0 < llength Ns and n-in: N  $\in$  Red-F (lnth Ns i0)*

**using** *n-in-sup* **by** (*metis Sup-llist-imp-exists-index llength-lmap lnth-lmap*)

**have** *Red-F (lnth Ns i0)  $\subseteq$  Red-F (Liminf-llist Ns)*

**using** *i-smaller* **by** (*simp add: deriv Red-F-subset-Liminf*)

**then show** *N  $\in$  Red-F (Liminf-llist Ns)*

**using** *n-in* **by** *fast*

**qed**

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

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

**proof**

**fix**  *$\iota$*

**assume**

*deriv: chain derive Ns and*

*i-in-sup:  $\iota \in$  Sup-llist (lmap Red-I Ns)*

**obtain** *i0* **where** *i-smaller: enat i0 < llength Ns and n-in:  $\iota \in$  Red-I (lnth Ns i0)*

**using** *i-in-sup* **unfolding** *Sup-llist-def* **by** *auto*

**have** *Red-I (lnth Ns i0)  $\subseteq$  Red-I (Liminf-llist Ns)*

**using** *i-smaller* **by** (*simp add: deriv Red-I-subset-Liminf*)

**then show**  *$\iota \in$  Red-I (Liminf-llist Ns)*

**using** *n-in* **by** *fast*

**qed**

**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)*

**unfolding** *reduc-saturated-def*

**proof** -

**fix** *Ns*

**assume**

*deriv*: chain derive  $Ns$  **and**  
*red-fair*: reduc-fair  $Ns$   
**have**  $Inf\text{-from} (Liminf\text{-l}list\ Ns - Red\text{-}F (Liminf\text{-l}list\ Ns))$   
 $\subseteq Inf\text{-from} (Liminf\text{-l}list\ Ns - Sup\text{-}l}list (lmap\ Red\text{-}F\ Ns))$   
**using** *sup-red-f-in-red-liminf*[*OF deriv*] **unfolding** *Inf-from-def* **by** *blast*  
**then have**  $Inf\text{-from} (Liminf\text{-l}list\ Ns - Red\text{-}F (Liminf\text{-l}list\ Ns)) \subseteq Sup\text{-}l}list (lmap\ Red\text{-}I\ Ns)$   
**using** *red-fair* **unfolding** *reduc-fair-def* **by** *simp*  
**then show**  $Inf\text{-from} (Liminf\text{-l}list\ Ns - Red\text{-}F (Liminf\text{-l}list\ Ns)) \subseteq Red\text{-}I (Liminf\text{-l}list\ Ns)$   
**using** *sup-red-inf-in-red-liminf*[*OF deriv*] **by** *fast*  
**qed**

**end**

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

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

**begin**

**sublocale** *reducedly-statically-complete-calculus*

**proof**

**fix**  $B\ N$

**assume**

*bot-elem*:  $\langle B \in Bot \rangle$  **and**

*saturated-N*: *reduc-saturated*  $N$  **and**

*refut-N*:  $N \models \{B\}$

**define**  $Ns$  **where**  $Ns = LCons\ N\ LNil$

**have**[*simp*]:  $\langle \neg\ lnull\ Ns \rangle$  **by** (*auto simp: Ns-def*)

**have** *deriv-D*:  $\langle chain\ (\triangleright)\ Ns \rangle$  **by** (*simp add: chain.chain-singleton Ns-def*)

**have** *liminf-is-N*:  $Liminf\text{-l}list\ Ns = N$  **by** (*simp add: Ns-def Liminf-l}list-LCons*)

**have** *head-D*:  $N = lhd\ Ns$  **by** (*simp add: Ns-def*)

**have**  $Sup\text{-}l}list (lmap\ Red\text{-}F\ Ns) = Red\text{-}F\ N$  **by** (*simp add: Ns-def*)

**moreover have**  $Sup\text{-}l}list (lmap\ Red\text{-}I\ Ns) = Red\text{-}I\ N$  **by** (*simp add: Ns-def*)

**ultimately have** *fair-D*: *reduc-fair*  $Ns$

**using** *saturated-N liminf-is-N* **unfolding** *reduc-fair-def* *reduc-saturated-def*

**by** (*simp add: reduc-fair-def* *reduc-saturated-def* *liminf-is-N*)

**obtain**  $i\ B'$  **where** *B'-is-bot*:  $\langle B' \in Bot \rangle$  **and** *B'-in*:  $B' \in lnth\ Ns\ i$  **and**  $\langle i < llength\ Ns \rangle$

**using** *reducedly-dynamically-complete*[*of B Ns*] *bot-elem* *fair-D* *head-D* *saturated-N* *deriv-D* *refut-N*

**by** *auto*

**then have**  $i = 0$

**by** (*auto simp: Ns-def enat-0-iff*)

**show**  $\langle \exists B' \in Bot. B' \in N \rangle$

**using** *B'-is-bot* *B'-in* **unfolding**  $\langle i = 0 \rangle$  *head-D*[*symmetric*] *Ns-def* **by** *auto*

**qed**

**end**

**sublocale** *reducedly-statically-complete-calculus*  $\subseteq$  *reducedly-dynamically-complete-calculus*

**proof**

**fix**  $B\ Ns$

**assume**

*bot-elem*:  $\langle B \in Bot \rangle$  **and**

*deriv*:  $\langle chain\ (\triangleright)\ Ns \rangle$  **and**

*fair*:  $\langle reduc\text{-}fair\ Ns \rangle$  **and**

*unsat*:  $\langle lhd\ Ns \models \{B\} \rangle$

**have** *non-empty*:  $\langle \neg \text{Inull } Ns \rangle$  **using** *chain-not-Inull*[*OF deriv*] .  
**have** *subs*:  $\langle \text{lhd } Ns \subseteq \text{Sup-llist } Ns \rangle$   
**using** *lhd-subset-Sup-llist*[*of Ns*] *non-empty* **by** (*simp add: lhd-conv-lnth*)  
**have**  $\langle \text{Sup-llist } Ns \models \{B\} \rangle$   
**using** *unsat subset-entailed*[*OF subs*] *entails-trans*[*of Sup-llist Ns lhd Ns*] **by** *auto*  
**then have** *Sup-no-Red*:  $\langle \text{Sup-llist } Ns - \text{Red-F } (\text{Sup-llist } Ns) \models \{B\} \rangle$   
**using** *bot-elem Red-F-Bot* **by** *auto*  
**have** *Sup-no-Red-in-Liminf*:  $\langle \text{Sup-llist } Ns - \text{Red-F } (\text{Sup-llist } Ns) \subseteq \text{Liminf-llist } Ns \rangle$   
**using** *deriv Red-in-Sup* **by** *auto*  
**have** *Liminf-entails-Bot*:  $\langle \text{Liminf-llist } Ns \models \{B\} \rangle$   
**using** *Sup-no-Red subset-entailed*[*OF Sup-no-Red-in-Liminf*] *entails-trans* **by** *blast*  
**have**  $\langle \text{reduc-saturated } (\text{Liminf-llist } Ns) \rangle$   
**using** *deriv fair reduc-fair-imp-Liminf-reduc-sat* **unfolding** *reduc-saturated-def*  
**by** *auto*  
**then have**  $\langle \exists B' \in \text{Bot}. B' \in \text{Liminf-llist } Ns \rangle$   
**using** *bot-elem reducedly-statically-complete Liminf-entails-Bot*  
**by** *auto*  
**then show**  $\langle \exists i \in \{i. \text{enat } i < \text{llength } Ns\}. \exists B' \in \text{Bot}. B' \in \text{lnth } Ns \ i \rangle$   
**unfolding** *Liminf-llist-def* **by** *auto*  
**qed**

**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**

**assume** *dynamically-complete-calculus Bot Inf entails Red-I Red-F*  
**then interpret** *dynamically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** *simp*  
**show** *statically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** (*simp add: statically-complete-calculus-axioms*)

**next**

**assume** *statically-complete-calculus Bot Inf entails Red-I Red-F*  
**then interpret** *statically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** *simp*  
**show** *dynamically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** (*simp add: dynamically-complete-calculus-axioms*)

**qed**

**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**

**assume** *reducedly-dynamically-complete-calculus Bot Inf entails Red-I Red-F*  
**then interpret** *reducedly-dynamically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** *simp*  
**show** *reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** (*simp add: reducedly-statically-complete-calculus-axioms*)

**next**

**assume** *reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F*  
**then interpret** *reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** *simp*  
**show** *reducedly-dynamically-complete-calculus Bot Inf entails Red-I Red-F*  
**by** (*simp add: reducedly-dynamically-complete-calculus-axioms*)

qed

**interpretation** *reduc-calc*: *reduced-calculus Bot Inf entails Red-Red-I Red-F*  
by (*fact reduc-calc*)

**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*  
using *dyn-equiv-stat stat-is-stat-red reduc-calc.dyn-equiv-stat* by *meson*

**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*  
using *red-dyn-equiv-red-stat dyn-equiv-stat red-stat-red-is-stat-red*  
by (*simp add: reduc-calc.dyn-equiv-stat reduc-calc.red-dyn-equiv-red-stat*)

**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*  
using *red-dyn-equiv-red-stat dyn-equiv-stat red-stat-is-stat-red*  
*reduc-calc.dyn-equiv-stat reduc-calc.red-dyn-equiv-red-stat*  
by *blast*

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**  
 $\mathcal{G}-F :: \langle 'f \Rightarrow 'g \text{ set} \rangle$  **and**  
 $\mathcal{G}-I :: \langle 'f \text{ inference} \Rightarrow 'g \text{ inference set option} \rangle$   
**assumes**  
 $Bot-F\text{-not-empty}: Bot-F \neq \{\}$  **and**  
 $Bot\text{-map-not-empty}: \langle B \in Bot-F \Longrightarrow \mathcal{G}-F B \neq \{\} \rangle$  **and**  
 $Bot\text{-map}: \langle B \in Bot-F \Longrightarrow \mathcal{G}-F B \subseteq Bot-G \rangle$  **and**  
 $Bot\text{-cond}: \langle \mathcal{G}-F C \cap Bot-G \neq \{\} \longrightarrow C \in Bot-F \rangle$  **and**  
 $inf\text{-map}: \langle \iota \in Inf-F \Longrightarrow \mathcal{G}-I \iota \neq None \Longrightarrow the (\mathcal{G}-I \iota) \subseteq Red-I-G (\mathcal{G}-F (concl-of \iota)) \rangle$   
**begin**

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

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

**abbreviation**  $entails\text{-}\mathcal{G} :: \langle 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow bool \rangle$  (**infix**  $\models_{\mathcal{G}}$  50) **where**  
 $\langle N1 \models_{\mathcal{G}} N2 \equiv \mathcal{G}\text{-Fset } N1 \models_{\mathcal{G}} \mathcal{G}\text{-Fset } N2 \rangle$

**lemma**  $subs\text{-}Bot\text{-}G\text{-}entails$ :

**assumes**  
 $not\text{-empty}: \langle sB \neq \{\} \rangle$  **and**  
 $in\text{-bot}: \langle sB \subseteq Bot-G \rangle$

**shows**  $\langle sB \models_{\mathcal{G}} N \rangle$

**proof** –

**have**  $\langle \exists B. B \in sB \rangle$  **using**  $not\text{-empty}$  **by auto**  
**then obtain**  $B$  **where**  $B\text{-in}: \langle B \in sB \rangle$  **by auto**  
**then have**  $r\text{-trans}: \langle \{B\} \models_{\mathcal{G}} N \rangle$  **using**  $ground.bot\text{-}entails\text{-}all$   $in\text{-bot}$  **by auto**  
**have**  $l\text{-trans}: \langle sB \models_{\mathcal{G}} \{B\} \rangle$  **using**  $B\text{-in}$   $ground.subset\text{-}entailed$  **by auto**  
**then show**  $?thesis$  **using**  $r\text{-trans}$   $ground.entails\text{-}trans[of sB \{B\}]$  **by auto**

**qed**

**sublocale**  $consequence\text{-}relation Bot-F entails\text{-}\mathcal{G}$

**proof**

**show**  $Bot-F \neq \{\}$  **using**  $Bot-F\text{-not-empty}$  .

**next**

**show**  $\langle B \in Bot-F \Longrightarrow \{B\} \models_{\mathcal{G}} N \rangle$  **for**  $B N$

**proof** –

**assume**  $\langle B \in Bot-F \rangle$

**then show**  $\langle \{B\} \models_{\mathcal{G}} N \rangle$

**using**  $Bot\text{-map}$   $ground.bot\text{-}entails\text{-}all[of - \mathcal{G}\text{-Fset } N]$   $subs\text{-}Bot\text{-}G\text{-}entails$   $Bot\text{-map}\text{-not}\text{-empty}$   
**by auto**

**qed**

**next**

**fix**  $N1 N2 :: \langle 'f \text{ set} \rangle$

**assume**

$\langle N2 \subseteq N1 \rangle$

**then show**  $\langle N1 \models_{\mathcal{G}} N2 \rangle$  **using**  $\mathcal{G}\text{-subset}$   $ground.subset\text{-}entailed$  **by auto**

**next**

**fix**  $N1 N2$

**assume**

$N1\text{-entails}\text{-}C: \langle \forall C \in N2. N1 \models_{\mathcal{G}} \{C\} \rangle$

**show**  $\langle N1 \models_{\mathcal{G}} N2 \rangle$  **using**  $ground.all\text{-}formulas\text{-}entailed$   $N1\text{-entails}\text{-}C$

**by** ( $smt UN\text{-}E UN\text{-}I$   $ground.entail\text{-}set\text{-}all\text{-}formulas\text{-}singletonI$ )

**next**  
**fix**  $N1\ N2\ N3$   
**assume**  
 $\langle N1 \models_{\mathcal{G}} N2 \rangle$  **and**  $\langle N2 \models_{\mathcal{G}} N3 \rangle$   
**then show**  $\langle N1 \models_{\mathcal{G}} N3 \rangle$  **using** *ground. entails-trans* **by** *blast*  
**qed**

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

**definition** *Red-F-G* :: '*f set*  $\Rightarrow$  '*f set* **where**  
 $\langle \text{Red-F-G } N = \{C. \forall D \in \mathcal{G-F } C. D \in \text{Red-F-G } (\mathcal{G-Fset } N)\} \rangle$   
**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*  $\Rightarrow$  '*g set*  $\Rightarrow$  *bool* (**infix**  $\models_{\mathcal{G}}$  50) **and**  
*Red-I-G* :: '*g set*  $\Rightarrow$  '*g inference set* **and**  
*Red-F-G* :: '*g set*  $\Rightarrow$  '*g set*

+ **fixes**  
*Bot-F* :: '*f set* **and**  
*G-F* :: '*f*  $\Rightarrow$  '*g set* **and**  
*G-I* :: '*f inference*  $\Rightarrow$  '*g inference set option*

**assumes**  
*Bot-F-not-empty*:  $\text{Bot-F} \neq \{\}$  **and**  
*Bot-map-not-empty*:  $\langle B \in \text{Bot-F} \Longrightarrow \mathcal{G-F } B \neq \{\} \rangle$  **and**  
*Bot-map*:  $\langle B \in \text{Bot-F} \Longrightarrow \mathcal{G-F } B \subseteq \text{Bot-G} \rangle$  **and**  
*Bot-cond*:  $\langle \mathcal{G-F } C \cap \text{Bot-G} \neq \{\} \longrightarrow C \in \text{Bot-F} \rangle$  **and**  
*strong-inf-map*:  $\langle \iota \in \text{Inf-F} \Longrightarrow \mathcal{G-I } \iota \neq \text{None} \Longrightarrow \text{concl-of } \iota \subseteq (\mathcal{G-F } (\text{concl-of } \iota)) \rangle$  **and**  
*inf-map-in-Inf*:  $\langle \iota \in \text{Inf-F} \Longrightarrow \mathcal{G-I } \iota \neq \text{None} \Longrightarrow \text{the } (\mathcal{G-I } \iota) \subseteq \text{Inf-G} \rangle$

**begin**

**sublocale** *standard-lifting* *Inf-F* *Bot-G* *Inf-G* ( $\models_{\mathcal{G}}$ ) *Red-I-G* *Red-F-G* *Bot-F* *G-F* *G-I*

**proof**

**show**  $\text{Bot-F} \neq \{\}$  **using** *Bot-F-not-empty* .

**next**

**fix**  $B$

**assume** *b-in*:  $B \in \text{Bot-F}$

**show**  $\mathcal{G-F } B \neq \{\}$  **using** *Bot-map-not-empty*[*OF* *b-in*] .

**next**

**fix**  $B$

**assume** *b-in*:  $B \in \text{Bot-F}$

**show**  $\mathcal{G-F } B \subseteq \text{Bot-G}$  **using** *Bot-map*[*OF* *b-in*] .

**next**

**show**  $\bigwedge C. \mathcal{G-F } C \cap \text{Bot-G} \neq \{\} \longrightarrow C \in \text{Bot-F}$  **using** *Bot-cond* .

**next**

**fix**  $\iota$

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

*some-g*:  $\mathcal{G-I } \iota \neq \text{None}$

**show** the  $(\mathcal{G}\text{-I } \iota) \subseteq \text{Red-I-G } (\mathcal{G}\text{-F } (\text{concl-of } \iota))$   
**proof**  
 fix  $\iota G$   
 assume  $ig\text{-in1}: \iota G \in \text{the } (\mathcal{G}\text{-I } \iota)$   
 then have  $ig\text{-in2}: \iota G \in \text{Inf-G}$  using  $\text{inf-map-in-Inf}[OF i\text{-in some-g}]$  by blast  
 show  $\iota G \in \text{Red-I-G } (\mathcal{G}\text{-F } (\text{concl-of } \iota))$   
 using  $\text{strong-inf-map}[OF i\text{-in some-g}]$  ground.Red-I-of-Inf-to-N[OF  $ig\text{-in2}$ ]  
 $ig\text{-in1}$  by blast  
 qed  
 qed  
 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* ::  $\langle 'f \text{ set} \rangle$  and  
*Inf-F* ::  $\langle 'f \text{ inference set} \rangle$  and  
*Bot-G* ::  $\langle 'g \text{ set} \rangle$  and  
*entails-G* ::  $\langle 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool} \rangle$  (infix  $\models_G 50$ ) and  
*Inf-G* ::  $\langle 'g \text{ inference set} \rangle$  and  
*Red-I-G* ::  $\langle 'g \text{ set} \Rightarrow 'g \text{ inference set} \rangle$  and  
*Red-F-G* ::  $\langle 'g \text{ set} \Rightarrow 'g \text{ set} \rangle$  and  
*G-F* ::  $\langle 'f \Rightarrow 'g \text{ set} \rangle$  and  
*G-I* ::  $\langle 'f \text{ inference} \Rightarrow 'g \text{ inference set option} \rangle$   
 + **fixes**  
*Prec-F-g* ::  $\langle 'g \Rightarrow 'f \Rightarrow 'f \Rightarrow \text{bool} \rangle$   
**assumes**  
*all-wf: minimal-element (Prec-F-g g) UNIV*  
**begin**

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

**lemma** *Prec-trans:*

**assumes**  
 $\langle \text{Prec-F-g } D A B \rangle$  and  
 $\langle \text{Prec-F-g } D B C \rangle$   
**shows**  
 $\langle \text{Prec-F-g } D A C \rangle$   
**using** *minimal-element.po assms unfolding po-on-def transp-on-def* by (smt UNIV-I all-wf)

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

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

$\langle \text{Red-F-G } N = \{C. \forall Di \in \mathcal{G}\text{-F } C. Di \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \vee (\exists E \in (N - \text{Red-F-G } N). \text{Prec-F-g } Di E C \wedge Di \in \mathcal{G}\text{-F } E)\} \rangle$   
**proof** (rule; clarsimp)  
 fix  $C D$   
**assume**  
 $C\text{-in}: \langle C \in \text{Red-F-G } N \rangle$  and



*D-in*:  $\langle D \in \mathcal{G}\text{-}F\ \mathcal{C} \rangle$  **and**  
*not-sec-case*:  $\langle \forall E \in N - \text{Red-}F\text{-}\mathcal{G}\ N. \text{Prec-}F\text{-}g\ D\ E\ C \longrightarrow D \notin \mathcal{G}\text{-}F\ E \rangle$   
**have** *C-in-unfolded*:  $\langle C \in \{C. \forall Di \in \mathcal{G}\text{-}F\ C. Di \in \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \vee (\exists E \in N. \text{Prec-}F\text{-}g\ Di\ E\ C \wedge Di \in \mathcal{G}\text{-}F\ E)\} \rangle$   
**using** *C-in unfolding Red-F-G-def* .  
**have** *neg-not-sec-case*:  $\langle \neg (\exists E \in N - \text{Red-}F\text{-}\mathcal{G}\ N. \text{Prec-}F\text{-}g\ D\ E\ C \wedge D \in \mathcal{G}\text{-}F\ E) \rangle$   
**using** *not-sec-case by clarsimp*  
**have** *unfol-C-D*:  $\langle D \in \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \vee (\exists E \in N. \text{Prec-}F\text{-}g\ D\ E\ C \wedge D \in \mathcal{G}\text{-}F\ E) \rangle$   
**using** *prop-nested-in-set*[of  $D\ \mathcal{G}\text{-}F\ C\ \lambda x. x \in \text{Red-}F\text{-}G\ (\bigcup (\mathcal{G}\text{-}F\ 'N))$   
 $\lambda x\ y. \exists E \in N. \text{Prec-}F\text{-}g\ y\ E\ x \wedge y \in \mathcal{G}\text{-}F\ E, OF\ D\text{-}in\ C\text{-}in\text{-}unfolded]$  **by** *blast*  
**show**  $\langle D \in \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \rangle$   
**proof** (*rule ccontr*)  
**assume** *contrad*:  $\langle D \notin \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \rangle$   
**have** *non-empty*:  $\langle \exists E \in N. \text{Prec-}F\text{-}g\ D\ E\ C \wedge D \in \mathcal{G}\text{-}F\ E \rangle$  **using** *contrad unfol-C-D* **by** *auto*  
**define** *B* **where**  $\langle B = \{E \in N. \text{Prec-}F\text{-}g\ D\ E\ C \wedge D \in \mathcal{G}\text{-}F\ E\} \rangle$   
**then have** *B-non-empty*:  $\langle B \neq \{\} \rangle$  **using** *non-empty* **by** *auto*  
**interpret** *minimal-element Prec-F-g D UNIV* **using** *all-wf*[of *D*] .  
**obtain** *F* :: '*f* **where** *F*:  $\langle F = \text{min-elt}\ B \rangle$  **by** *auto*  
**then have** *D-in-F*:  $\langle D \in \mathcal{G}\text{-}F\ F \rangle$  **unfolding** *B-def* **using** *non-empty*  
**by** (*smt Sup-UNIV Sup-upper UNIV-I contra-subsetD empty-iff empty-subsetI mem-Collect-eq min-elt-mem unfol-C-D*)  
**have** *F-prec*:  $\langle \text{Prec-}F\text{-}g\ D\ F\ C \rangle$  **using** *F min-elt-mem*[of *B*, *OF - B-non-empty*] **unfolding** *B-def*  
**by** *auto*  
**have** *F-not-in*:  $\langle F \notin \text{Red-}F\text{-}\mathcal{G}\ N \rangle$   
**proof**  
**assume** *F-in*:  $\langle F \in \text{Red-}F\text{-}\mathcal{G}\ N \rangle$   
**have** *unfol-F-D*:  $\langle D \in \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \vee (\exists G \in N. \text{Prec-}F\text{-}g\ D\ G\ F \wedge D \in \mathcal{G}\text{-}F\ G) \rangle$   
**using** *F-in D-in-F* **unfolding** *Red-F-G-def* **by** *auto*  
**then have**  $\langle \exists G \in N. \text{Prec-}F\text{-}g\ D\ G\ F \wedge D \in \mathcal{G}\text{-}F\ G \rangle$  **using** *contrad D-in* **unfolding** *Red-F-G-def*  
**by** *auto*  
**then obtain** *G* **where** *G-in*:  $\langle G \in N \rangle$  **and** *G-prec*:  $\langle \text{Prec-}F\text{-}g\ D\ G\ F \rangle$  **and** *G-map*:  $\langle D \in \mathcal{G}\text{-}F\ G \rangle$   
**by** *auto*  
**have**  $\langle \text{Prec-}F\text{-}g\ D\ G\ C \rangle$  **using** *G-prec F-prec Prec-trans* **by** *blast*  
**then have**  $\langle G \in B \rangle$  **unfolding** *B-def* **using** *G-in G-map* **by** *auto*  
**then show**  $\langle \text{False} \rangle$  **using** *F G-prec min-elt-minimal*[of *B G*, *OF - B-non-empty*] **by** *auto*  
**qed**  
**have**  $\langle F \in N \rangle$  **using** *F* **by** (*metis B-def B-non-empty mem-Collect-eq min-elt-mem top-greatest*)  
**then have**  $\langle F \in N - \text{Red-}F\text{-}\mathcal{G}\ N \rangle$  **using** *F-not-in* **by** *auto*  
**then show**  $\langle \text{False} \rangle$   
**using** *D-in-F neg-not-sec-case F-prec* **by** *blast*  
**qed**  
**next**  
**fix** *C*  
**assume** *only-if*:  $\langle \forall D \in \mathcal{G}\text{-}F\ C. D \in \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \vee (\exists E \in N - \text{Red-}F\text{-}\mathcal{G}\ N. \text{Prec-}F\text{-}g\ D\ E\ C \wedge D \in \mathcal{G}\text{-}F\ E) \rangle$   
**show**  $\langle C \in \text{Red-}F\text{-}\mathcal{G}\ N \rangle$  **unfolding** *Red-F-G-def* **using** *only-if* **by** *auto*  
**qed**

**lemma** *not-red-map-in-map-not-red*:  $\langle \mathcal{G}\text{-}F\text{set}\ N - \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \subseteq \mathcal{G}\text{-}F\text{set}\ (N - \text{Red-}F\text{-}\mathcal{G}\ N) \rangle$   
**proof**  
**fix** *D*  
**assume**  
*D-hyp*:  $\langle D \in \mathcal{G}\text{-}F\text{set}\ N - \text{Red-}F\text{-}G\ (\mathcal{G}\text{-}F\text{set}\ N) \rangle$   
**interpret** *minimal-element Prec-F-g D UNIV* **using** *all-wf*[of *D*] .

```

have D-in:  $\langle D \in \mathcal{G}\text{-Fset } N \rangle$  using D-hyp by blast
have D-not-in:  $\langle D \notin \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \rangle$  using D-hyp by blast
have exist-C:  $\langle \exists C. C \in N \wedge D \in \mathcal{G}\text{-F } C \rangle$  using D-in by auto
define B where  $\langle B = \{C \in N. D \in \mathcal{G}\text{-F } C\} \rangle$ 
obtain C where C:  $\langle C = \text{min-elt } B \rangle$  by auto
have C-in-N:  $\langle C \in N \rangle$ 
  using exist-C by (metis B-def C empty-iff mem-Collect-eq min-elt-mem top-greatest)
have D-in-C:  $\langle D \in \mathcal{G}\text{-F } C \rangle$ 
  using exist-C by (metis B-def C empty-iff mem-Collect-eq min-elt-mem top-greatest)
have C-not-in:  $\langle C \notin \text{Red-F-G } N \rangle$ 
proof
  assume C-in:  $\langle C \in \text{Red-F-G } N \rangle$ 
  have  $\langle D \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \vee (\exists E \in N. \text{Prec-F-g } D \ E \ C \wedge D \in \mathcal{G}\text{-F } E) \rangle$ 
    using C-in D-in-C unfolding Red-F-G-def by auto
  then show  $\langle \text{False} \rangle$ 
    proof
      assume  $\langle D \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \rangle$ 
      then show  $\langle \text{False} \rangle$  using D-not-in by simp
    next
      assume  $\langle \exists E \in N. \text{Prec-F-g } D \ E \ C \wedge D \in \mathcal{G}\text{-F } E \rangle$ 
      then show  $\langle \text{False} \rangle$ 
        using C by (metis (no-types, lifting) B-def UNIV-I empty-iff mem-Collect-eq
          min-elt-minimal top-greatest)
    qed
  qed
show  $\langle D \in \mathcal{G}\text{-Fset } (N - \text{Red-F-G } N) \rangle$  using D-in-C C-not-in C-in-N by blast
qed

```

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

**proof** –

```

fix B N
assume
  B-in:  $\langle B \in \text{Bot-F} \rangle$  and
  N-entails:  $\langle N \models_{\mathcal{G}} \{B\} \rangle$ 
then have to-bot:  $\langle \mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \models_{\mathcal{G}} \mathcal{G}\text{-F } B \rangle$ 
  using ground.Red-F-Bot Bot-map
  by (smt cSup-singleton ground. entail-set-all-formulas image-insert image-is-empty subsetCE)
have from-f:  $\langle \mathcal{G}\text{-Fset } (N - \text{Red-F-G } N) \models_{\mathcal{G}} \mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \rangle$ 
  using ground.subset-entailed[OF not-red-map-in-map-not-red] by blast
then have  $\langle \mathcal{G}\text{-Fset } (N - \text{Red-F-G } N) \models_{\mathcal{G}} \mathcal{G}\text{-F } B \rangle$  using to-bot ground.entails-trans by blast
then show  $\langle N - \text{Red-F-G } N \models_{\mathcal{G}} \{B\} \rangle$  using Bot-map by simp
qed

```

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

**using** *ground.Red-F-of-subset* **unfolding** *Red-F-G-def* **by** *clarsimp* (*meson* *G-subset* *subsetD*)

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

**using** *Collect-mono* *G-subset* *subset-iff* *ground.Red-I-of-subset* **unfolding** *Red-I-G-def* **by** (*smt* *ground.Red-F-of-subset* *Un-iff*)

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

**proof**

**fix**  $N N' C$

**assume**

$N'-in-Red-F-N$ :  $\langle N' \subseteq Red-F-G N \rangle$  **and**

$C-in-red-F-N$ :  $\langle C \in Red-F-G N \rangle$

**have**  $lem8$ :  $\langle \forall D \in \mathcal{G}-F C. D \in Red-F-G (\mathcal{G}-Fset N) \vee (\exists E \in (N - Red-F-G N). Prec-F-g D E C \wedge D \in \mathcal{G}-F E) \rangle$

**using**  $Red-F-G-equiv-def C-in-red-F-N$  **by**  $blast$

**show**  $\langle C \in Red-F-G (N - N') \rangle$  **unfolding**  $Red-F-G-def$

**proof** ( $rule,rule$ )

**fix**  $D$

**assume**  $\langle D \in \mathcal{G}-F C \rangle$

**then have**  $\langle D \in Red-F-G (\mathcal{G}-Fset N) \vee (\exists E \in (N - Red-F-G N). Prec-F-g D E C \wedge D \in \mathcal{G}-F E) \rangle$

**using**  $lem8$  **by**  $auto$

**then show**  $\langle D \in Red-F-G (\mathcal{G}-Fset (N - N')) \vee (\exists E \in N - N'. Prec-F-g D E C \wedge D \in \mathcal{G}-F E) \rangle$

**proof**

**assume**  $\langle D \in Red-F-G (\mathcal{G}-Fset N) \rangle$

**then have**  $\langle D \in Red-F-G (\mathcal{G}-Fset N - Red-F-G (\mathcal{G}-Fset N)) \rangle$

**using**  $ground.Red-F-of-Red-F-subset$ [of  $Red-F-G (\mathcal{G}-Fset N) \mathcal{G}-Fset N$ ] **by**  $auto$

**then have**  $\langle D \in Red-F-G (\mathcal{G}-Fset (N - Red-F-G N)) \rangle$

**using**  $ground.Red-F-of-subset$ [OF  $not-red-map-in-map-not-red$ [of  $N$ ]] **by**  $auto$

**then have**  $\langle D \in Red-F-G (\mathcal{G}-Fset (N - N')) \rangle$

**using**  $N'-in-Red-F-N \mathcal{G}-subset$ [of  $N - Red-F-G N N - N'$ ]

**by** ( $smt DiffE DiffI ground.Red-F-of-subset subsetCE subsetI$ )

**then show**  $?thesis$  **by**  $blast$

**next**

**assume**  $\langle \exists E \in N - Red-F-G N. Prec-F-g D E C \wedge D \in \mathcal{G}-F E \rangle$

**then obtain**  $E$  **where**

$E-in$ :  $\langle E \in N - Red-F-G N \rangle$  **and**

$E-prec-C$ :  $\langle Prec-F-g D E C \rangle$  **and**

$D-in$ :  $\langle D \in \mathcal{G}-F E \rangle$

**by**  $auto$

**have**  $\langle E \in N - N' \rangle$  **using**  $E-in N'-in-Red-F-N$  **by**  $blast$

**then show**  $?thesis$  **using**  $E-prec-C D-in$  **by**  $blast$

**qed**

**qed**

**qed**

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

**proof**

**fix**  $N N' \iota$

**assume**

$N'-in-Red-F-N$ :  $\langle N' \subseteq Red-F-G N \rangle$  **and**

$i-in-Red-I-N$ :  $\langle \iota \in Red-I-G N \rangle$

**have**  $i-in$ :  $\langle \iota \in Inf-F \rangle$  **using**  $i-in-Red-I-N$  **unfolding**  $Red-I-G-def$  **by**  $blast$

**{**

**assume**  $not-none$ :  $\mathcal{G}-I \iota \neq None$

**have**  $\langle \forall \iota' \in the (\mathcal{G}-I \iota). \iota' \in Red-I-G (\mathcal{G}-Fset N) \rangle$

**using**  $not-none i-in-Red-I-N$  **unfolding**  $Red-I-G-def$  **by**  $auto$

**then have**  $\langle \forall \iota' \in the (\mathcal{G}-I \iota). \iota' \in Red-I-G (\mathcal{G}-Fset N - Red-F-G (\mathcal{G}-Fset N)) \rangle$

**using**  $not-none ground.Red-I-of-Red-F-subset$  **by**  $blast$

**then have**  $ip-in-Red-I-G$ :  $\langle \forall \iota' \in the (\mathcal{G}-I \iota). \iota' \in Red-I-G (\mathcal{G}-Fset (N - Red-F-G N)) \rangle$

**using**  $not-none ground.Red-I-of-subset$ [OF  $not-red-map-in-map-not-red$ [of  $N$ ]] **by**  $auto$

**then have**  $not-none-in$ :  $\langle \forall \iota' \in the (\mathcal{G}-I \iota). \iota' \in Red-I-G (\mathcal{G}-Fset (N - N')) \rangle$

**using** *not-none*  $N'$ -in-Red-F-N  
**by** (*meson* *Diff-mono* *ground.Red-I-of-subset*  $\mathcal{G}$ -subset *subset-iff* *subset-refl*)  
**then have** the  $(\mathcal{G}\text{-I } \iota) \subseteq \text{Red-I-G } (\mathcal{G}\text{-Fset } (N - N'))$  **by** *blast*  
**}**  
**moreover** {  
**assume** *none*:  $\mathcal{G}\text{-I } \iota = \text{None}$   
**have** *ground-concl-subs*:  $\mathcal{G}\text{-F } (\text{concl-of } \iota) \subseteq (\mathcal{G}\text{-Fset } N \cup \text{Red-F-G } (\mathcal{G}\text{-Fset } N))$   
**using** *none* *i-in-Red-I-N* **unfolding** *Red-I-G-def* **by** *blast*  
**then have** *d-in-imp12*:  $D \in \mathcal{G}\text{-F } (\text{concl-of } \iota) \implies D \in \mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \vee D \in$   
 $\text{Red-F-G } (\mathcal{G}\text{-Fset } N)$   
**by** *blast*  
**have** *d-in-imp1*:  $D \in \mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \implies D \in \mathcal{G}\text{-Fset } (N - N')$   
**using** *not-red-map-in-map-not-red*  $N'$ -in-Red-F-N **by** *blast*  
**have** *d-in-imp-d-in*:  $D \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \implies D \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N))$   
**using** *ground.Red-F-of-Red-F-subset*[of  $\text{Red-F-G } (\mathcal{G}\text{-Fset } N)$   $\mathcal{G}\text{-Fset } N]$  **by** *blast*  
**have** *g-subs1*:  $\mathcal{G}\text{-Fset } N - \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \subseteq \mathcal{G}\text{-Fset } (N - \text{Red-F-G } N)$   
**using** *not-red-map-in-map-not-red* **unfolding** *Red-F-G-def* **by** *auto*  
**have** *g-subs2*:  $\mathcal{G}\text{-Fset } (N - \text{Red-F-G } N) \subseteq \mathcal{G}\text{-Fset } (N - N')$   
**using**  $N'$ -in-Red-F-N **by** *blast*  
**have** *d-in-imp2*:  $D \in \text{Red-F-G } (\mathcal{G}\text{-Fset } N) \implies D \in \text{Red-F-G } (\mathcal{G}\text{-Fset } (N - N'))$   
**using** *ground.Red-F-of-subset* *ground.Red-F-of-subset*[OF *g-subs1*]  
*ground.Red-F-of-subset*[OF *g-subs2*] *d-in-imp-d-in* **by** *blast*  
**have**  $\mathcal{G}\text{-F } (\text{concl-of } \iota) \subseteq (\mathcal{G}\text{-Fset } (N - N') \cup \text{Red-F-G } (\mathcal{G}\text{-Fset } (N - N')))$   
**using** *d-in-imp12* *d-in-imp1* *d-in-imp2*  
**by** (*smt* *ground.Red-F-of-Red-F-subset* *ground.Red-F-of-subset* *UnCI* *UnE* *Un-Diff-cancel2*  
*ground-concl-subs* *g-subs1* *g-subs2* *subset-iff*)  
**}**  
**ultimately show**  $\langle \iota \in \text{Red-I-G } (N - N') \rangle$  **using** *i-in* **unfolding** *Red-I-G-def* **by** *auto*  
**qed**

**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**

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

**proof** –

**have**  $\langle \iota \in \text{Inf-F} \implies \mathcal{G}\text{-I } \iota \neq \text{None} \implies \text{the } (\mathcal{G}\text{-I } \iota) \subseteq \text{Red-I-G } (\mathcal{G}\text{-F } (\text{concl-of } \iota)) \rangle$  **using** *inf-map* **by** *simp*

**moreover have**  $\langle \text{Red-I-G } (\mathcal{G}\text{-F } (\text{concl-of } \iota)) \subseteq \text{Red-I-G } (\mathcal{G}\text{-Fset } N) \rangle$

**using** *concl-i-in* *ground.Red-I-of-subset* **by** *blast*

**moreover have**  $\langle \iota \in \text{Inf-F} \implies \mathcal{G}\text{-I } \iota = \text{None} \implies \text{concl-of } \iota \in N \implies \mathcal{G}\text{-F } (\text{concl-of } \iota) \subseteq \mathcal{G}\text{-Fset } N$

**by** *blast*

**ultimately show** *?thesis* **using** *i-in* *concl-i-in* **unfolding** *Red-I-G-def* **by** *auto*

**qed**

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

**proof**

**fix**  $B$   $N$   $N'$   $\iota$

**show**  $\langle \text{Red-I-G } N \subseteq \text{Inf-F} \rangle$  **unfolding** *Red-I-G-def* **by** *blast*

**show**  $\langle B \in \text{Bot-F} \implies N \models_{\mathcal{G}} \{B\} \implies N - \text{Red-F-G } N \models_{\mathcal{G}} \{B\} \rangle$  **using** *Red-F-Bot-F* **by** *simp*

**show**  $\langle N \subseteq N' \implies \text{Red-F-G } N \subseteq \text{Red-F-G } N' \rangle$  **using** *Red-F-of-subset-F* **by** *simp*

**show**  $\langle N \subseteq N' \implies \text{Red-I-G } N \subseteq \text{Red-I-G } N' \rangle$  **using** *Red-I-of-subset-F* **by** *simp*

**show**  $\langle N' \subseteq \text{Red-F-G } N \implies \text{Red-F-G } N \subseteq \text{Red-F-G } (N - N') \rangle$  **using** *Red-F-of-Red-F-subset-F* **by**  
*simp*  
**show**  $\langle N' \subseteq \text{Red-F-G } N \implies \text{Red-I-G } N \subseteq \text{Red-I-G } (N - N') \rangle$  **using** *Red-I-of-Red-F-subset-F* **by** *simp*  
**show**  $\langle \iota \in \text{Inf-F} \implies \text{concl-of } \iota \in N \implies \iota \in \text{Red-I-G } N \rangle$  **using** *Red-I-of-Inf-to-N-F* **by** *simp*  
**qed**

**end**

**lemma** *wf-empty-rel: minimal-element* ( $\lambda - . \text{False}$ ) *UNIV*  
**by** (*simp add: minimal-element.intro po-on-def transp-onI wfp-on-imp-irreflp-on*)

**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** –

**have** *tiebreaker-lifting-axioms* ( $\lambda g C C'. \text{False}$ )  
**unfolding** *tiebreaker-lifting-axioms-def* **using** *wf-empty-rel* **by** *simp*  
**then show** *?thesis*  
**unfolding** *standard-lifting-def tiebreaker-lifting-def* **by** *blast*  
**qed**

**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*  $\lambda g C C'. \text{False}$   
**using** *standard-empty-tiebreaker-equiv* **using** *standard-lifting-axioms* **by** *blast*

**lemma** *red-f-equiv: empt-ord.Red-F-G = Red-F-G*  
**unfolding** *Red-F-G-def empt-ord.Red-F-G-def* **by** *simp*

**sublocale** *calc?: calculus* *Bot-F Inf-F entails-G* *Red-I-G Red-F-G*  
**using** *empt-ord.calculus-axioms red-f-equiv* **by** *fastforce*

**lemma** *grounded-inf-in-ground-inf:  $\iota \in \text{Inf-F} \implies \mathcal{G}\text{-I } \iota \neq \text{None} \implies \text{the } (\mathcal{G}\text{-I } \iota) \subseteq \text{Inf-G}$*   
**using** *inf-map ground.Red-I-to-Inf* **by** *blast*

**abbreviation** *ground-Inf-overapproximated*  $:: 'f \text{ set} \Rightarrow \text{bool}$  **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**  $\iota \in \text{Red-I-G } (\mathcal{G}\text{-Fset } N)$   
**using** *assms Red-I-G-def* **unfolding** *saturated-def* **by** *auto*

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

*saturated*  $N \implies \text{ground-Inf-overapproximated } N \implies \text{ground.saturated } (\mathcal{G}\text{-Fset } N)$   
**unfolding** *ground.saturated-def* **using** *sat-inf-imp-ground-red* **by** *auto*

**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**

**fix** *B N*

**assume**

*b-in*:  $B \in \text{ Bot-F}$  **and**

*sat-n*: *saturated N* **and**

*n-entails-bot*:  $N \models_{\mathcal{G}} \{B\}$

**have** *ground-n-entails*:  $\mathcal{G}\text{-Fset } N \models_{\mathcal{G}} \mathcal{G}\text{-F } B$

**using** *n-entails-bot* **by** *simp*

**then obtain** *BG* **where** *bg-in1*:  $BG \in \mathcal{G}\text{-F } B$

**using** *Bot-map-not-empty[OF b-in]* **by** *blast*

**then have** *bg-in*:  $BG \in \text{ Bot-G}$

**using** *Bot-map[OF b-in]* **by** *blast*

**have** *ground-n-entails-bot*:  $\mathcal{G}\text{-Fset } N \models_{\mathcal{G}} \{BG\}$

**using** *ground-n-entails bg-in1 ground.entail-set-all-formulas* **by** *blast*

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

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

**using** *sat-n-imp[OF sat-n]* .

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

**using** *sat-imp-ground-sat[OF sat-n sat-n-imp[OF sat-n]]* .

**then have**  $\exists BG' \in \text{ Bot-G}. BG' \in (\mathcal{G}\text{-Fset } N)$

**using** *stat-ref-G ground.calculus-axioms bg-in ground-n-entails-bot*

**unfolding** *statically-complete-calculus-def statically-complete-calculus-axioms-def*

**by** *blast*

**then show**  $\exists B' \in \text{ Bot-F}. B' \in N$

**using** *bg-in Bot-cond Bot-map-not-empty Bot-cond*

**by** *blast*

**qed**

**end**

**context** *tiebreaker-lifting*

**begin**

**lemma** *saturated-empty-order-equiv-saturated*:

*saturated N = calc.saturated N*

**by** *(rule refl)*

**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*

**unfolding** *statically-complete-calculus-def*

**by** *(rule iffI) (standard,(standard)[],simp)+*

**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*  
**using** *dyn-equiv-stat static-empty-order-equiv-static*  
**by** *blast*

**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:*  
 $\forall q \in Q. \text{tiebreaker-lifting Bot-F Inf-F Bot-G (entails-q q) (Inf-G-q q) (Red-I-q q)}$   
 $(\text{Red-F-q q}) (\text{G-F-q q}) (\text{G-I-q q}) \text{Prec-F-g}$

**begin**

**abbreviation** *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*

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

**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*

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

**abbreviation** *entails-G-q :: 'q ⇒ 'f set ⇒ 'f set ⇒ bool where*

*entails-G-q q N1 N2 ≡ entails-q q (G-Fset-q q N1) (G-Fset-q q N2)*

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

**assumes** *q-in: q ∈ Q*

**shows** *calculus Bot-F Inf-F (entails-G-q q) (Red-I-G-q q) (Red-F-G-q q)*

**proof** –

**interpret** *wf-lift:*

*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*

**using** *standard-lifting-family q-in* **by** *metis*  
**have**  $Red-I-G-q \ q = wf-lift.Red-I-G$   
**unfolding**  $Red-I-G-q-def \ wf-lift.Red-I-G-def$  **by** *blast*  
**moreover have**  $Red-F-G-q \ q = wf-lift.Red-F-G$   
**unfolding**  $Red-F-G-q-def \ wf-lift.Red-F-G-def$  **by** *blast*  
**ultimately show** *?thesis*  
**using** *wf-lift.calculus-axioms* **by** *simp*  
**qed**

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

**assumes** *q-in: q ∈ Q*

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

**proof** –

**interpret** *wf-lift:*

*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*

**using** *standard-lifting-family q-in* **by** *metis*

**have**  $Red-I-G-q \ q = wf-lift.Red-I-G$

**unfolding**  $Red-I-G-q-def \ wf-lift.Red-I-G-def$  **by** *blast*

**moreover have**  $Red-F-G-empty-q \ q = wf-lift.empty-ord.Red-F-G$

**unfolding**  $Red-F-G-empty-q-def \ wf-lift.empty-ord.Red-F-G-def$  **by** *blast*

**ultimately show** *?thesis*

**using** *wf-lift.calc.calculus-axioms* **by** *simp*

**qed**

**sublocale** *consequence-relation-family Bot-F Q entails-G-q*

**proof** (*unfold-locales; (intro ballI) ?*)

**show**  $Q \neq \{\}$

**by** (*rule ground.Q-nonempty*)

**next**

**fix** *qi*

**assume** *qi-in: qi ∈ Q*

**interpret** *lift: tiebreaker-lifting Bot-F Inf-F Bot-G entails-q qi Inf-G-q qi*

*Red-I-q qi Red-F-q qi G-F-q qi G-I-q qi Prec-F-g*

**using** *qi-in* **by** (*metis standard-lifting-family*)

**show** *consequence-relation Bot-F (entails-G-q qi)*

**by** *unfold-locales*

**qed**

**sublocale** *intersection-calculus Bot-F Inf-F Q entails-G-q Red-I-G-q Red-F-G-q*

**by** *unfold-locales (auto simp: Q-nonempty red-crit-lifting-family)*

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

$(\models_{\cap G}) \equiv entails$

**abbreviation**  $Red-I-G :: 'f \ set \Rightarrow 'f \ inference \ set$  **where**

$Red-I-G \equiv Red-I$

**abbreviation**  $Red-F-G :: 'f \ set \Rightarrow 'f \ set$  **where**

$Red-F-G \equiv Red-F$

**lemmas**  $entails-G-def = entails-def$

**lemmas**  $Red-I-G-def = Red-I-def$



**lemmas**  $Red-F-G-def = Red-F-def$

**sublocale**  $empty-ord$ : *intersection-calculus Bot-F Inf-F Q entails-G-q Red-I-G-q Red-F-G-empty-q*  
**by** *unfold-locales (auto simp: Q-nonempty red-crit-lifting-family-empty-ord)*

**abbreviation**  $Red-F-G-empty :: 'f set \Rightarrow 'f set$  **where**  
 $Red-F-G-empty \equiv empty-ord.Red-F$

**lemmas**  $Red-F-G-empty-def = empty-ord.Red-F-def$

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

**assumes**

$sat-n$ : *saturated N* **and**

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

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

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

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

**proof** –

**have**  $\iota' \in Red-I-G-q\ q\ N$

**using**  $sat-n\ i'-in\ q-in$  *all-red-crit calculus.saturated-def sat-int-to-sat-q*

**by** *blast*

**then have**  $the\ (\mathcal{G}-I-q\ q\ \iota') \subseteq Red-I-q\ q\ (\mathcal{G}-Fset-q\ q\ N)$

**by** *(simp add: Red-I-G-q-def grounding)*

**then show** *?thesis*

**using** *grounding* **by** *blast*

**qed**

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

$ground-Inf-overapproximated\ q\ N \equiv$

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

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

**abbreviation**  $ground-saturated :: 'q \Rightarrow 'f set \Rightarrow bool$  **where**

$ground-saturated\ q\ N \equiv ground.Inf-from-q\ q\ (\mathcal{G}-Fset-q\ q\ N) \subseteq Red-I-q\ q\ (\mathcal{G}-Fset-q\ q\ N)$

**lemma**  $sat-imp-ground-sat-fam-inter$ :

$saturated\ N \Longrightarrow q \in Q \Longrightarrow ground-Inf-overapproximated\ q\ N \Longrightarrow ground-saturated\ q\ N$

**using**  $sat-inf-imp-ground-red-fam-inter$  **by** *auto*

**theorem**  $stat-ref-comp-to-non-ground-fam-inter$ :

**assumes**

$stat-ref-G$ :

$\forall q \in Q. statically-complete-calculus\ Bot-G\ (Inf-G-q\ q)\ (entails-q\ q)\ (Red-I-q\ q)$

$(Red-F-q\ q)$  **and**

$sat-n-imp$ :  $\bigwedge N. saturated\ N \Longrightarrow \exists q \in Q. ground-Inf-overapproximated\ q\ N$

**shows**

$statically-complete-calculus\ Bot-F\ Inf-F\ entails-G\ Red-I-G\ Red-F-G-empty$

**using**  $empty-ord.calculus-axioms$  **unfolding**  $statically-complete-calculus-def$

$statically-complete-calculus-axioms-def$

**proof** *(standard, clarify)*

**fix**  $B\ N$

**assume**

$b-in$ :  $B \in Bot-F$  **and**

$sat-n$ : *saturated N* **and**

*entails-bot*:  $N \models_{\cap \mathcal{G}} \{B\}$   
**then obtain**  $q$  **where**  
*q-in*:  $q \in Q$  **and**  
*inf-sub*:  $\text{ground. Inf-from-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N) \subseteq$   
 $\{\iota. \exists \iota' \in \text{Inf-from } N. \mathcal{G}\text{-I-}q \ q \ \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I-}q \ q \ \iota')\}$   
 $\cup \text{Red-I-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$   
**using** *sat-n-imp*[of  $N$ ] **by** *blast*  
**interpret** *q-calc*: *calculus Bot-F Inf-F entails- $\mathcal{G}$ -q q Red-I- $\mathcal{G}$ -q q Red-F- $\mathcal{G}$ -q q*  
**using** *all-red-crit*[*rule-format*, *OF q-in*].  
**have** *n-q-sat*: *q-calc.saturated N*  
**using** *q-in sat-int-to-sat-q sat-n* **by** *simp*  
**interpret** *lifted-q-calc*:  
*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*  
**using** *q-in* **by** (*simp add: standard-lifting-family*)  
**have** *n-lift-sat*: *lifted-q-calc.calc.saturated N*  
**using** *n-q-sat unfolding Red-I- $\mathcal{G}$ -q-def lifted-q-calc.Red-I- $\mathcal{G}$ -def*  
*lifted-q-calc.saturated-def q-calc.saturated-def* **by** *auto*  
**have** *ground-sat-n*: *lifted-q-calc.ground.saturated ( $\mathcal{G}$ -Fset- $q \ q \ N$ )*  
**by** (*rule lifted-q-calc.sat-imp-ground-sat[OF n-lift-sat]*)  
*(use n-lift-sat inf-sub ground. Inf-from-q-def in auto)*  
**have** *ground-n-entails-bot*: *entails- $\mathcal{G}$ -q q N {B}*  
**using** *q-in entails-bot unfolding entails- $\mathcal{G}$ -def* **by** *simp*  
**interpret** *statically-complete-calculus Bot-G Inf-G-q q entails-q q Red-I-q q*  
*Red-F-q q*  
**using** *stat-ref- $\mathcal{G}$* [*rule-format*, *OF q-in*].  
**obtain**  $BG$  **where** *bg-in*:  $BG \in \mathcal{G}\text{-F-}q \ q \ B$   
**using** *lifted-q-calc.Bot-map-not-empty[OF b-in]* **by** *blast*  
**then have**  $BG \in \text{Bot-G}$  **using** *lifted-q-calc.Bot-map[OF b-in]* **by** *blast*  
**then have**  $\exists BG' \in \text{Bot-G}. BG' \in \mathcal{G}\text{-Fset-}q \ q \ N$   
**using** *ground-sat-n ground-n-entails-bot statically-complete[of BG, OF - ground-sat-n]*  
*bg-in lifted-q-calc.ground.entail-set-all-formulas[of  $\mathcal{G}$ -Fset- $q \ q \ N \ \mathcal{G}$ -Fset- $q \ q \ \{B\}$ ]*  
**by** *simp*  
**then show**  $\exists B' \in \text{Bot-F}. B' \in N$  **using** *lifted-q-calc.Bot-cond* **by** *blast*  
**qed**

**lemma** *sat-eq-sat-empty-order*: *saturated N = empty-ord.saturated N*  
**by** (*rule refl*)

**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- $\mathcal{G}$ -empty*  
**unfolding** *statically-complete-calculus-def*  
**by** (*simp add: empty-ord.calculus-axioms calculus-axioms*)

**theorem** *stat-eq-dyn-ref-comp-fam-inter*: *statically-complete-calculus Bot-F Inf-F*  
*entails Red-I Red-F- $\mathcal{G}$ -empty =*  
*dynamically-complete-calculus Bot-F Inf-F entails Red-I Red-F*  
**using** *dyn-equiv-stat static-empty-ord-inter-equiv-static-inter*  
**by** *blast*

**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* ( $\lambda$ - . *False*) *UNIV*  
**unfolding** *po-on-def irreflp-on-def transp-on-def* **by** *auto*

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

**begin**

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

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

**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*

**proof**

**show** *Bot-FL*  $\neq$   $\{\}$

**using** *no-labels.Bot-F-not-empty* **by** *simp*

**next**  
**show**  $B \in \text{Bot-FL} \implies \mathcal{G}\text{-F-L } B \neq \{\}$  **for**  $B$   
**using** *no-labels.Bot-map-not-empty* **by** *auto*  
**next**  
**show**  $B \in \text{Bot-FL} \implies \mathcal{G}\text{-F-L } B \subseteq \text{Bot-G}$  **for**  $B$   
**using** *no-labels.Bot-map* **by** *force*  
**next**  
**fix**  $CL$   
**show**  $\mathcal{G}\text{-F-L } CL \cap \text{Bot-G} \neq \{\} \longrightarrow CL \in \text{Bot-FL}$   
**using** *no-labels.Bot-cond* **by** (*metis SigmaE UNIV-I UNIV-Times-UNIV mem-Sigma-iff prod.sel(1)*)  
**next**  
**fix**  $\iota$   
**assume**  
*i-in:  $\langle \iota \in \text{Inf-FL} \rangle$  and*  
*ground-not-none:  $\langle \mathcal{G}\text{-I-L } \iota \neq \text{None} \rangle$*   
**then show** *the  $(\mathcal{G}\text{-I-L } \iota) \subseteq \text{Red-I-G } (\mathcal{G}\text{-F-L } (\text{concl-of } \iota))$*   
**unfolding** *to-F-def* **using** *no-labels.inf-map Inf-FL-to-Inf-F* **by** *fastforce*  
**qed**

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

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

**lemma** *red-inf-impl:  $\iota \in \text{Red-I-G } NL \implies \text{to-F } \iota \in \text{no-labels.Red-I-G } (\text{fst } 'NL)$*   
**unfolding** *Red-I-G-def no-labels.Red-I-G-def* **using** *Inf-FL-to-Inf-F* **by** (*auto simp: to-F-def*)

**lemma** *labeled-saturation-lifting:  $\text{saturated } NL \implies \text{no-labels.saturated } (\text{fst } 'NL)$*   
**unfolding** *saturated-def no-labels.saturated-def Inf-from-def no-labels.Inf-from-def*

**proof** *clarify*

**fix**  $\iota$   
**assume**  
*subs-Red-I:  $\{\iota \in \text{Inf-FL}. \text{set } (\text{prems-of } \iota) \subseteq NL\} \subseteq \text{Red-I-G } NL$  and*  
*i-in:  $\iota \in \text{Inf-F}$  and*  
*i-prems:  $\text{set } (\text{prems-of } \iota) \subseteq \text{fst } 'NL$*   
**define** *Lli* **where** *Lli*  $i = (\text{SOME } x. ((\text{prems-of } \iota)!i, x) \in NL)$  **for**  $i$   
**have** [*simp*]:  $(\text{prems-of } \iota)!i, \text{Lli } i \in NL$  **if**  $i < \text{length } (\text{prems-of } \iota)$  **for**  $i$   
**using** *that i-prems* **unfolding** *Lli-def* **by** (*metis nth-mem someI-ex DomainE Domain-fst subset-eq*)  
**define** *Ll* **where** *Ll*  $= \text{map } \text{Lli } [0..<\text{length } (\text{prems-of } \iota)]$   
**have** *Ll-length*:  $\text{length } Ll = \text{length } (\text{prems-of } \iota)$  **unfolding** *Ll-def* **by** *auto*  
**have** *subs-NL*:  $\text{set } (\text{zip } (\text{prems-of } \iota) Ll) \subseteq NL$  **unfolding** *Ll-def* **by** (*auto simp: in-set-zip*)  
**obtain** *L0* **where** *L0*:  $\text{Infer } (\text{zip } (\text{prems-of } \iota) Ll) (\text{concl-of } \iota, L0) \in \text{Inf-FL}$   
**using** *Inf-F-to-Inf-FL[OF i-in Ll-length]* **..**  
**define**  $\iota\text{-FL}$  **where**  $\iota\text{-FL} = \text{Infer } (\text{zip } (\text{prems-of } \iota) Ll) (\text{concl-of } \iota, L0)$   
**then have**  $\text{set } (\text{prems-of } \iota\text{-FL}) \subseteq NL$  **using** *subs-NL* **by** *simp*  
**then have**  $\iota\text{-FL} \in \{\iota \in \text{Inf-FL}. \text{set } (\text{prems-of } \iota) \subseteq NL\}$  **unfolding**  $\iota\text{-FL-def}$  **using** *L0* **by** *blast*  
**then have**  $\iota\text{-FL} \in \text{Red-I-G } NL$  **using** *subs-Red-I* **by** *fast*  
**moreover have**  $\iota = \text{to-F } \iota\text{-FL}$  **unfolding** *to-F-def  $\iota\text{-FL-def}$*  **using** *Ll-length* **by** (*cases  $\iota$* ) *auto*  
**ultimately show**  $\iota \in \text{no-labels.Red-I-G } (\text{fst } 'NL)$  **by** (*auto intro: red-inf-impl*)  
**qed**

**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 ( $\models_{\mathcal{GL}}$ ) Red-I- $\mathcal{G}$  Red-F- $\mathcal{G}$*

**proof**

**fix** *Bl* ::  $\langle 'f \times 'l \rangle$  **and** *Nl* ::  $\langle ('f \times 'l) \text{ set} \rangle$

**assume**

*Bl-in*:  $\langle Bl \in \text{Bot-FL} \rangle$  **and**

*Nl-sat*:  $\langle \text{saturated } Nl \rangle$  **and**

*Nl-entails-Bl*:  $\langle Nl \models_{\mathcal{GL}} \{Bl\} \rangle$

**define** *B* **where**  $B = \text{fst } Bl$

**have** *B-in*:  $B \in \text{Bot-F}$  **using** *Bl-in* *B-def* *SigmaE* **by** *force*

**define** *N* **where**  $N = \text{fst } 'Nl$

**have** *N-sat*: *no-labels.saturated* *N*

**using** *N-def* *Nl-sat* *labeled-saturation-lifting* **by** *blast*

**have** *N-entails-B*:  $N \models_{\mathcal{G}} \{B\}$

**using** *Nl-entails-Bl* **unfolding** *labeled-entailment-lifting* *N-def* *B-def* **by** *force*

**have**  $\exists B' \in \text{Bot-F}. B' \in N$  **using** *B-in* *N-sat* *N-entails-B*

**using** *static[unfolded statically-complete-calculus-def*  
*statically-complete-calculus-axioms-def]* **by** *blast*

**then obtain** *B'* **where** *in-Bot*:  $B' \in \text{Bot-F}$  **and** *in-N*:  $B' \in N$  **by** *force*

**then have**  $B' \in \text{fst } ' \text{Bot-FL}$  **by** *fastforce*

**obtain** *Bl'* **where** *in-Nl*:  $Bl' \in Nl$  **and** *fst-Bl'*:  $\text{fst } Bl' = B'$

**using** *in-N* **unfolding** *N-def* **by** *blast*

**have**  $Bl' \in \text{Bot-FL}$  **using** *fst-Bl'* *in-Bot* *vimage-fst* **by** *fastforce*

**then show**  $\langle \exists Bl' \in \text{Bot-FL}. Bl' \in Nl \rangle$  **using** *in-Nl* **by** *blast*

**qed**

**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*  $\lambda q$  *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* ::  $'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool}$  **and**

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

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

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

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

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

**+ fixes**

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

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

**begin**

**definition** *to-F* ::  $\langle ('f \times 'l) \text{ inference} \Rightarrow 'f \text{ inference} \rangle$  **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**  $\text{Bot-FL} :: \langle 'f \times 'l \text{ set} \rangle$  **where**  
 $\langle \text{Bot-FL} \equiv \text{Bot-F} \times \text{UNIV} \rangle$

**abbreviation**  $\mathcal{G}\text{-F-L-q} :: \langle 'q \Rightarrow ('f \times 'l) \Rightarrow 'g \text{ set} \rangle$  **where**  
 $\langle \mathcal{G}\text{-F-L-q } q \text{ CL} \equiv \mathcal{G}\text{-F-q } q \text{ (fst CL)} \rangle$

**abbreviation**  $\mathcal{G}\text{-I-L-q} :: \langle 'q \Rightarrow ('f \times 'l) \text{ inference} \Rightarrow 'g \text{ inference set option} \rangle$  **where**  
 $\langle \mathcal{G}\text{-I-L-q } q \iota_{FL} \equiv \mathcal{G}\text{-I-q } q \text{ (to-F } \iota_{FL}) \rangle$

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

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

**abbreviation**  $\text{Red-I-G-L} :: ('f \times 'l) \text{ set} \Rightarrow ('f \times 'l) \text{ inference set}$  **where**  
 $\text{Red-I-G-L } N \equiv (\bigcap q \in Q. \text{Red-I-G-L-q } q \text{ N})$

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

**lemma**  $\text{lifting-q}$ :

**assumes**  $q \in Q$

**shows**  $\text{labeled-tiebreaker-lifting Bot-F Inf-F Bot-G (entails-q } q) \text{ (Inf-G-q } q) \text{ (Red-I-q } q)$   
 $\text{ (Red-F-q } q) \text{ } (\mathcal{G}\text{-F-q } q) \text{ } (\mathcal{G}\text{-I-q } q) \text{ } (\lambda g \text{ Cl Cl'. False) Inf-FL}$

**using**  $\text{assms no-labels.standard-lifting-family Inf-F-to-Inf-FL Inf-FL-to-Inf-F}$

**by**  $(\text{simp add: labeled-tiebreaker-lifting-axioms-def labeled-tiebreaker-lifting-def})$

**lemma**  $\text{lifted-q}$ :

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

**shows**  $\text{standard-lifting Inf-FL Bot-G (Inf-G-q } q) \text{ (entails-q } q) \text{ (Red-I-q } q) \text{ (Red-F-q } q)$   
 $\text{ Bot-FL } (\mathcal{G}\text{-F-L-q } q) \text{ } (\mathcal{G}\text{-I-L-q } q)$

**proof** –

**interpret**  $q\text{-lifting}: \text{labeled-tiebreaker-lifting Bot-F Inf-F Bot-G entails-q } q \text{ Inf-G-q } q$   
 $\text{ Red-I-q } q \text{ Red-F-q } q \text{ } \mathcal{G}\text{-F-q } q \text{ } \mathcal{G}\text{-I-q } q \text{ } \lambda g \text{ Cl Cl'. False Inf-FL}$

**using**  $\text{lifting-q}[OF q\text{-in}]$ .

**have**  $\mathcal{G}\text{-I-L-q } q = q\text{-lifting.}\mathcal{G}\text{-I-L}$

**unfolding**  $\text{to-F-def } q\text{-lifting.to-F-def}$  **by**  $\text{simp}$

**then show**  $?thesis$

**using**  $q\text{-lifting.standard-lifting-axioms}$  **by**  $\text{simp}$

**qed**

**lemma**  $\text{ord-fam-lifted-q}$ :

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

**shows**  $\text{tiebreaker-lifting Bot-FL Inf-FL Bot-G (entails-q } q) \text{ (Inf-G-q } q) \text{ (Red-I-q } q)$   
 $\text{ (Red-F-q } q) \text{ } (\mathcal{G}\text{-F-L-q } q) \text{ } (\mathcal{G}\text{-I-L-q } q) \text{ } (\lambda g \text{ Cl Cl'. False)}$

**proof** –

**interpret**  $\text{standard-q-lifting}: \text{standard-lifting Inf-FL Bot-G Inf-G-q } q \text{ entails-q } q$   
 $\text{ Red-I-q } q \text{ Red-F-q } q \text{ Bot-FL } \mathcal{G}\text{-F-L-q } q \text{ } \mathcal{G}\text{-I-L-q } q$

**using**  $\text{lifted-q}[OF q\text{-in}]$ .

**have**  $\text{minimal-element } (\lambda \text{Cl Cl'. False) UNIV}$

by (*simp add: minimal-element.intro po-on-def transp-onI wfp-on-imp-irreflp-on*)  
 then show *?thesis*  
 using *standard-q-lifting.standard-lifting-axioms*  
 by (*simp add: tiebreaker-lifting-axioms-def tiebreaker-lifting-def*)  
 qed

**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*  $\in$  *Red-F-q* *q* ( $\mathcal{G}\text{-Fset-L-q } q$  *N*)  $\vee$   
 ( $\exists E \in N$ . *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*  $\equiv$  ( $\bigcap q \in Q$ . *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)*

**proof** –

**interpret** *ord-q-lifting: 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  $\lambda g$  Cl Cl'. False*

using *ord-fam-lifted-q[OF q-in]* .

have *Red-I-G-L-q q = ord-q-lifting.Red-I-G*

unfolding *Red-I-G-L-q-def ord-q-lifting.Red-I-G-def* **by** *simp*

moreover have *Red-F-G-empty-L-q q = ord-q-lifting.Red-F-G*

unfolding *Red-F-G-empty-L-q-def ord-q-lifting.Red-F-G-def* **by** *simp*

ultimately show *?thesis*

using *ord-q-lifting.calculus-axioms* **by** *argo*

qed

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

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

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

using *all-lifted-red-crit calculus-def q-in* **by** *blast*

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

using *all-lifted-cons-rel* **by** (*simp add: consequence-relation-family.intro no-labels.Q-nonempty*)

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

using *intersection-calculus.intro[OF consequence-relation-family-axioms]*

**by** (*simp add: all-lifted-red-crit intersection-calculus-axioms-def no-labels.Q-nonempty*)

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

**by** (*simp add: Inf-FL-to-Inf-F to-F-def*)

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

unfolding *Inf-from-def no-labels.Inf-from-def to-F-def* **by** (*auto intro: Inf-FL-to-Inf-F*)

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

**abbreviation** *entails-G-L* :: ('*f*  $\times$  '*l*) set  $\Rightarrow$  ('*f*  $\times$  '*l*) set  $\Rightarrow$  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*:  $NL1 \models_{\mathcal{G}L} NL2 \iff \text{fst ' } NL1 \models_{\mathcal{G}} \text{fst ' } NL2$

**unfolding** *no-labels.entails-G-def entails-G-L-def* **by** *force*

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

**unfolding** *no-labels.Red-I-G-def Red-I-def*

**proof** *clarify*

**fix** *X Xa q*

**assume**

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

*i-in-inter*:  $\iota \in (\bigcap q \in Q. \text{Red-I-G-L-q } q \text{ NL})$

**have** *i-in-q*:  $\iota \in \text{Red-I-G-L-q } q \text{ NL}$  **using** *q-in i-in-inter image-eqI* **by** *blast*

**then have** *i-in*:  $\iota \in \text{Inf-FL}$  **unfolding** *Red-I-G-L-q-def* **by** *blast*

**have** *to-F-in*:  $\text{to-F } \iota \in \text{Inf-F}$  **unfolding** *to-F-def* **using** *Inf-FL-to-Inf-F[OF i-in]* .

**have** *rephrase1*:  $(\bigcup CL \in \text{NL}. \mathcal{G}\text{-F-q } q \text{ (fst CL)}) = (\bigcup (\mathcal{G}\text{-F-q } q \text{ ' fst ' NL}))$  **by** *blast*

**have** *rephrase2*:  $\text{fst (concl-of } \iota) = \text{concl-of (to-F } \iota)$

**unfolding** *concl-of-def to-F-def* **by** *simp*

**have** *subs-red*:  $(\mathcal{G}\text{-I-L-q } q \text{ } \iota \neq \text{None} \wedge \text{the } (\mathcal{G}\text{-I-L-q } q \text{ } \iota) \subseteq \text{Red-I-q } q \text{ } (\mathcal{G}\text{-Fset-L-q } q \text{ NL}))$

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

**using** *i-in-q* **unfolding** *Red-I-G-L-q-def* **by** *blast*

**then have** *to-F-subs-red*:  $(\mathcal{G}\text{-I-q } q \text{ (to-F } \iota) \neq \text{None} \wedge$

$\text{the } (\mathcal{G}\text{-I-q } q \text{ (to-F } \iota)) \subseteq \text{Red-I-q } q \text{ (no-labels.}\mathcal{G}\text{-Fset-q } q \text{ (fst ' NL))})$

$\vee (\mathcal{G}\text{-I-q } q \text{ (to-F } \iota) = \text{None} \wedge$

$\mathcal{G}\text{-F-q } q \text{ (concl-of (to-F } \iota))$

$\subseteq \text{no-labels.}\mathcal{G}\text{-Fset-q } q \text{ (fst ' NL)} \cup \text{Red-F-q } q \text{ (no-labels.}\mathcal{G}\text{-Fset-q } q \text{ (fst ' NL))})$

**using** *rephrase1 rephrase2* **by** *metis*

**then show**  $\text{to-F } \iota \in \text{no-labels.Red-I-G-q } q \text{ (fst ' NL)}$

**using** *to-F-in* **unfolding** *no-labels.Red-I-G-q-def* **by** *simp*

**qed**

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

**unfolding** *saturated-def no-labels.saturated-def Inf-from-def no-labels.Inf-from-def*

**proof** *clarify*

**fix**  $\iota F$

**assume**

*labeled-sat*:  $\{\iota \in \text{Inf-FL}. \text{set (prems-of } \iota) \subseteq \text{NL}\} \subseteq \text{Red-I NL}$  **and**

*iF-in*:  $\iota F \in \text{Inf-F}$  **and**

*iF-prems*:  $\text{set (prems-of } \iota F) \subseteq \text{fst ' NL}$

**define** *Lli* **where**  $Lli \ i = (\text{SOME } x. ((\text{prems-of } \iota F)!i, x) \in \text{NL})$  **for** *i*

**have** [*simp*]:  $((\text{prems-of } \iota F)!i, Lli \ i) \in \text{NL}$  **if**  $i < \text{length (prems-of } \iota F)$  **for** *i*

**using** *that iF-prems nth-mem someI-ex* **unfolding** *Lli-def* **by** (*metis DomainE Domain-fst subset-eq*)

**define** *Ll* **where**  $Ll = \text{map } Lli \ [0..<\text{length (prems-of } \iota F)]$

**have** *Ll-length*:  $\text{length } Ll = \text{length (prems-of } \iota F)$  **unfolding** *Ll-def* **by** *auto*

**have** *subs-NL*:  $\text{set (zip (prems-of } \iota F) Ll) \subseteq \text{NL}$  **unfolding** *Ll-def* **by** (*auto simp:in-set-zip*)

**obtain** *L0* **where**  $L0: \text{Infer (zip (prems-of } \iota F) Ll) (\text{concl-of } \iota F, L0) \in \text{Inf-FL}$

**using** *Inf-F-to-Inf-FL[OF iF-in Ll-length]* ..

**define**  $\iota FL$  **where**  $\iota FL = \text{Infer (zip (prems-of } \iota F) Ll) (\text{concl-of } \iota F, L0)$

**then have**  $\text{set (prems-of } \iota FL) \subseteq \text{NL}$  **using** *subs-NL* **by** *simp*

**then have**  $\iota FL \in \{\iota \in \text{Inf-FL}. \text{set (prems-of } \iota) \subseteq \text{NL}\}$  **unfolding**  $\iota FL\text{-def}$  **using** *L0* **by** *blast*

**then have**  $\iota FL \in \text{Red-I NL}$  **using** *labeled-sat* **by** *fast*

**moreover have**  $\iota F = \text{to-F } \iota FL$  **unfolding** *to-F-def*  $\iota FL\text{-def}$  **using** *Ll-length* **by** (*cases*  $\iota F$ ) *auto*

**ultimately show**  $\iota F \in \text{no-labels.Red-I-G (fst ' NL)}$

**by** (*auto intro: red-inf-impl*)

**qed**



**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**

**fix** *Bl* ::  $\langle 'f \times 'l \rangle$  **and** *Nl* ::  $\langle ('f \times 'l) \text{ set} \rangle$

**assume**

*Bl-in*:  $\langle Bl \in Bot-FL \rangle$  **and**

*Nl-sat*:  $\langle \text{saturated } Nl \rangle$  **and**

*Nl-entails-Bl*:  $\langle Nl \models \cap \mathcal{G}L \{Bl\} \rangle$

**define** *B* **where**  $B = \text{fst } Bl$

**have** *B-in*:  $B \in Bot-F$  **using** *Bl-in B-def SigmaE* **by** *force*

**define** *N* **where**  $N = \text{fst } 'Nl$

**have** *N-sat*: *no-labels.saturated N*

**using** *N-def Nl-sat labeled-family-saturation-lifting* **by** *blast*

**have** *N-entails-B*:  $N \models \cap \mathcal{G} \{B\}$

**using** *Nl-entails-Bl unfolding labeled-entailment-lifting N-def B-def* **by** *force*

**have**  $\exists B' \in Bot-F. B' \in N$  **using** *B-in N-sat N-entails-B*

*calc*[*unfolded statically-complete-calculus-def*  
*statically-complete-calculus-axioms-def*]

**by** *blast*

**then obtain** *B'* **where** *in-Bot*:  $B' \in Bot-F$  **and** *in-N*:  $B' \in N$  **by** *force*

**then have**  $B' \in \text{fst } 'Bot-FL$  **by** *fastforce*

**obtain** *Bl'* **where** *in-Nl*:  $Bl' \in Nl$  **and** *fst-Bl'*:  $\text{fst } Bl' = B'$

**using** *in-N unfolding N-def* **by** *blast*

**have**  $Bl' \in Bot-FL$  **using** *fst-Bl' in-Bot vimage-fst* **by** *fastforce*

**then show**  $\langle \exists Bl' \in Bot-FL. Bl' \in Nl \rangle$  **using** *in-Nl* **by** *blast*

**qed**

**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*

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

**for**

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

**and** *Inf-F* ::  $'f \text{ inference set}$

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

**and** *Q* ::  $'q \text{ set}$

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

```

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
  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
  equiv-equiv-F: equivp (≐) and
  wf-prec-F: minimal-element (<·) UNIV and
  wf-prec-L: minimal-element (⊆L) UNIV and
  compat-equiv-prec:  $C1 ≐ D1 \implies C2 ≐ D2 \implies C1 <· C2 \implies D1 <· D2$  and
  equiv-F-grounding:  $q \in Q \implies C1 ≐ C2 \implies \mathcal{G}\text{-F}\text{-}q\ q\ C1 \subseteq \mathcal{G}\text{-F}\text{-}q\ q\ C2$  and
  prec-F-grounding:  $q \in Q \implies C2 <· C1 \implies \mathcal{G}\text{-F}\text{-}q\ q\ C1 \subseteq \mathcal{G}\text{-F}\text{-}q\ q\ C2$  and
  active-minimal:  $l2 \neq \text{active} \implies \text{active} \subseteqL l2$  and
  at-least-two-labels:  $\exists l2. \text{active} \subseteqL l2$  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 × 'l) inference set where
  Inf-FL ≡ {ιFL. Infer (map fst (prems-of ιFL)) (fst (concl-of ιFL)) ∈ Inf-F}

abbreviation Prec-eq-F :: 'f ⇒ 'f ⇒ bool (infix ≐· 50) where
   $C \preceq· D \equiv C \doteq D \vee C <· D$ 

definition Prec-FL :: ('f × 'l) ⇒ ('f × 'l) ⇒ bool (infix ⊆ 50) where
   $Cl1 \subseteq Cl2 \iff \text{fst } Cl1 <· \text{fst } Cl2 \vee (\text{fst } Cl1 \doteq \text{fst } Cl2 \wedge \text{snd } Cl1 \subseteqL \text{snd } Cl2)$ 

lemma irrefl-prec-F:  $\neg C <· C$ 
by (simp add: minimal-element.po[OF wf-prec-F, unfolded po-on-def irreflp-on-def])

lemma trans-prec-F:  $C1 <· C2 \implies C2 <· C3 \implies C1 <· C3$ 
by (auto intro: minimal-element.po[OF wf-prec-F, unfolded po-on-def transp-on-def, THEN conjunct2, simplified, rule-format])

lemma wf-prec-FL: minimal-element (⊆) UNIV
proof
  show po-on (⊆) UNIV unfolding po-on-def
  proof
    show irreflp (⊆) unfolding irreflp-on-def Prec-FL-def
    proof
      fix Cl
      assume a-in:  $Cl \in (\text{UNIV}::('f \times 'l)\ \text{set})$ 
      have  $\neg (\text{fst } Cl <· \text{fst } Cl)$  using wf-prec-F minimal-element.min-elt-ex by force
      moreover have  $\neg (\text{snd } Cl \subseteqL \text{snd } Cl)$  using wf-prec-L minimal-element.min-elt-ex by force
      ultimately show  $\neg (\text{fst } Cl <· \text{fst } Cl \vee \text{fst } Cl \doteq \text{fst } Cl \wedge \text{snd } Cl \subseteqL \text{snd } Cl)$  by blast
    qed
  next
    show transp (⊆) unfolding Prec-FL-def
    proof (rule transpI)
      fix Cl1 Cl2 Cl3

```

```

assume trans-hyps:
  (fst Cl1 <· fst Cl2 ∨ fst Cl1 ≐ fst Cl2 ∧ snd Cl1 ⊆L snd Cl2)
  (fst Cl2 <· fst Cl3 ∨ fst Cl2 ≐ fst Cl3 ∧ snd Cl2 ⊆L snd Cl3)
have fst Cl1 <· fst Cl2 ⇒ fst Cl2 ≐ fst Cl3 ⇒ fst Cl1 <· fst Cl3
  using compat-equiv-prec by (metis equiv-equiv-F equivp-def)
moreover have fst Cl1 ≐ fst Cl2 ⇒ fst Cl2 <· fst Cl3 ⇒ fst Cl1 <· fst Cl3
  using compat-equiv-prec by (metis equiv-equiv-F equivp-def)
moreover have snd Cl1 ⊆L snd Cl2 ⇒ snd Cl2 ⊆L snd Cl3 ⇒ snd Cl1 ⊆L snd Cl3
  using wf-prec-L unfolding minimal-element-def po-on-def transp-def by (meson UNIV-I)
moreover have fst Cl1 ≐ fst Cl2 ⇒ fst Cl2 ≐ fst Cl3 ⇒ fst Cl1 ≐ fst Cl3
  using equiv-equiv-F by (meson equivp-transp)
ultimately show fst Cl1 <· fst Cl3 ∨ fst Cl1 ≐ fst Cl3 ∧ snd Cl1 ⊆L snd Cl3
  using trans-hyps trans-prec-F by blast
qed
qed
next
show wfp-on (⊆) UNIV unfolding wfp-on-def
proof
  assume contra: ∃f. ∀i. f i ∈ UNIV ∧ f (Suc i) ⊆ f i
  then obtain f where
    f-suc: ∀i. f (Suc i) ⊆ f i
    by blast

  define R :: (('f × 'l) × ('f × 'l)) set where
    R = {(Cl1, Cl2). fst Cl1 <· fst Cl2}
  define S :: (('f × 'l) × ('f × 'l)) set where
    S = {(Cl1, Cl2). fst Cl1 ≐ fst Cl2 ∧ snd Cl1 ⊆L snd Cl2}

  obtain k where
    f-chain: ∀i. (f (Suc (i + k)), f (i + k)) ∈ S
  proof (atomize-elim, rule wf-infinite-down-chain-compatible[of R f S])
    show wf R
      unfolding R-def using wf-app[OF wf-prec-F[unfolded minimal-element-def, THEN conjunct2,
        unfolded wfp-on-UNIV wfP-def]]
      by force
    next
      show ∀i. (f (Suc i), f i) ∈ R ∪ S
      using f-suc unfolding R-def S-def Prec-FL-def by blast
    next
      show R ∩ S ⊆ R
      unfolding R-def S-def using compat-equiv-prec equiv-equiv-F equivp-reflp by fastforce
    qed

  define g where
    ∧i. g i = f (i + k)

  have g-chain: ∀i. (g (Suc i), g i) ∈ S
    unfolding g-def using f-chain by simp
  have wf-s: wf S
    unfolding S-def
    by (rule wf-subset[OF wf-app[OF wf-prec-L[unfolded minimal-element-def, THEN conjunct2,
      unfolded wfp-on-UNIV wfP-def], of snd]])
    fast
  show False
    using g-chain[unfolded S-def]

```

$wf\text{-}s[\text{unfolded } S\text{-def}, \text{folded } wfP\text{-def } wfp\text{-on-UNIV}, \text{unfolded } wfp\text{-on-def}]$   
**by auto**  
**qed**  
**qed**

**definition** *active-subset* :: ('f × 'l) set ⇒ ('f × 'l) set **where**  
*active-subset* M = {CL ∈ M. snd CL = active}

**definition** *passive-subset* :: ('f × 'l) set ⇒ ('f × 'l) set **where**  
*passive-subset* M = {CL ∈ M. snd CL ≠ active}

**lemma** *active-subset-insert[simp]*:  
*active-subset* (insert Cl N) = (if snd Cl = active then {Cl} else {}) ∪ *active-subset* N  
**unfolding** *active-subset-def* **by auto**

**lemma** *active-subset-union[simp]*: *active-subset* (M ∪ N) = *active-subset* M ∪ *active-subset* N  
**unfolding** *active-subset-def* **by auto**

**lemma** *passive-subset-insert[simp]*:  
*passive-subset* (insert Cl N) = (if snd Cl ≠ active then {Cl} else {}) ∪ *passive-subset* N  
**unfolding** *passive-subset-def* **by auto**

**lemma** *passive-subset-union[simp]*: *passive-subset* (M ∪ N) = *passive-subset* M ∪ *passive-subset* N  
**unfolding** *passive-subset-def* **by auto**

**sublocale** *std?*: *statically-complete-calculus Bot-FL Inf-FL (|=∩GL) Red-I Red-F*  
**using** *labeled-static-ref[OF static-ref-comp]* .

**lemma** *labeled-tiebreaker-lifting*:  
**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) (G-F-L-q q) (G-I-L-q q) (λg. Prec-FL)*  
**proof** –  
**have** *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) (λg Cl Cl'. False)*  
**using** *ord-fam-lifted-q[OF q-in]* .  
**then have** *standard-lifting Inf-FL Bot-G (Inf-G-q q) (entails-q q) (Red-I-q q)*  
*(Red-F-q q) Bot-FL (G-F-L-q q) (G-I-L-q q)*  
**using** *lifted-q[OF q-in]* **by blast**  
**then show** *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) (λg. Prec-FL)*  
**using** *wf-prec-FL* **by** (*simp add: tiebreaker-lifting.intro tiebreaker-lifting-axioms.intro*)  
**qed**

**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*  
**using** *labeled-tiebreaker-lifting* **unfolding** *lifting-intersection-def*  
**by** (*simp add: lifting-intersection-axioms.intro*  
*no-labels.ground.consequence-relation-family-axioms*  
*no-labels.ground.inference-system-family-axioms*)

**notation** *derive* (**infix** ▷L 50)

**lemma** *std-Red-I-eq*: *std.Red-I* = *Red-I-G*  
**unfolding** *Red-I-G-q-def Red-I-G-L-q-def* **by simp**

**lemma** *std-Red-F-eq*:  $std.Red-F = Red-F-G-empty$   
**unfolding** *Red-F-G-empty-q-def Red-F-G-empty-L-q-def* **by** *simp*

**sublocale** *statically-complete-calculus Bot-FL Inf-FL ( $\models \cap \mathcal{G}L$ ) Red-I Red-F*  
**by** *unfold-locale* (use *statically-complete std-Red-I-eq* **in** *auto*)

**lemma** *labeled-red-inf-eq-red-inf*:

**assumes** *i-in*:  $\iota \in Inf-FL$

**shows**  $\iota \in Red-I N \longleftrightarrow to-F \iota \in no-labels.Red-I-G (fst \ ' N)$

**proof**

**assume** *i-in2*:  $\iota \in Red-I N$

**then have**  $X \in Red-I-G-q \ ' Q \implies \iota \in X N$  **for**  $X$

**unfolding** *Red-I-def* **by** *blast*

**obtain**  $X0$  **where**  $X0 \in Red-I-G-q \ ' Q$

**using** *Q-nonempty* **by** *blast*

**then obtain**  $q0$  **where**  $x0-is: X0 N = Red-I-G-q q0 N$  **by** *blast*

**then obtain**  $Y0$  **where**  $y0-is: Y0 (fst \ ' N) = to-F \ ' (X0 N)$  **by** *auto*

**have**  $Y0 (fst \ ' N) = no-labels.Red-I-G-q q0 (fst \ ' N)$

**unfolding** *y0-is*

**proof**

**show**  $to-F \ ' X0 N \subseteq no-labels.Red-I-G-q q0 (fst \ ' N)$

**proof**

**fix**  $\iota0$

**assume** *i0-in*:  $\iota0 \in to-F \ ' X0 N$

**then have** *i0-in2*:  $\iota0 \in to-F \ ' Red-I-G-q q0 N$

**using** *x0-is* **by** *argo*

**then obtain**  $\iota0-FL$  **where** *i0-FL-in*:  $\iota0-FL \in Inf-FL$  **and** *i0-to-i0-FL*:  $\iota0 = to-F \ \iota0-FL$  **and**

*subs1*:  $((\mathcal{G}-I-L-q q0 \ \iota0-FL) \neq None \wedge$

$the (\mathcal{G}-I-L-q q0 \ \iota0-FL) \subseteq Red-I-q q0 (\mathcal{G}-Fset-q q0 N))$

$\vee ((\mathcal{G}-I-L-q q0 \ \iota0-FL = None) \wedge$

$\mathcal{G}-F-L-q q0 (concl-of \ \iota0-FL) \subseteq \mathcal{G}-Fset-q q0 N \cup Red-F-q q0 (\mathcal{G}-Fset-q q0 N))$

**unfolding** *Red-I-G-q-def* **by** *blast*

**have** *concl-swap*:  $fst (concl-of \ \iota0-FL) = concl-of \ \iota0$

**unfolding** *concl-of-def i0-to-i0-FL to-F-def* **by** *simp*

**have** *i0-in3*:  $\iota0 \in Inf-F$

**using** *i0-to-i0-FL Inf-FL-to-Inf-F[OF i0-FL-in]* **unfolding** *to-F-def* **by** *blast*

{

**assume**

*not-none*:  $\mathcal{G}-I-q q0 \ \iota0 \neq None$  **and**

*the*  $(\mathcal{G}-I-q q0 \ \iota0) \neq \{\}$

**then obtain**  $\iota1$  **where** *i1-in*:  $\iota1 \in the (\mathcal{G}-I-q q0 \ \iota0)$  **by** *blast*

**have**  $the (\mathcal{G}-I-q q0 \ \iota0) \subseteq Red-I-q q0 (no-labels.\mathcal{G}-Fset-q q0 (fst \ ' N))$

**using** *subs1 i0-to-i0-FL not-none* **by** *auto*

}

**moreover** {

**assume**

*is-none*:  $\mathcal{G}-I-q q0 \ \iota0 = None$

**then have**  $\mathcal{G}-F-q q0 (concl-of \ \iota0) \subseteq no-labels.\mathcal{G}-Fset-q q0 (fst \ ' N)$

$\cup Red-F-q q0 (no-labels.\mathcal{G}-Fset-q q0 (fst \ ' N))$

**using** *subs1 i0-to-i0-FL concl-swap* **by** *simp*

}

**ultimately show**  $\iota0 \in no-labels.Red-I-G-q q0 (fst \ ' N)$

**unfolding** *no-labels.Red-I-G-q-def* **using** *i0-in3* **by** *auto*

```

qed
next
show  $no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q0\ (fst\ 'N) \subseteq to\text{-}F\ 'X0\ N$ 
proof
  fix  $\iota0$ 
  assume  $i0\text{-}in: \iota0 \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q0\ (fst\ 'N)$ 
  then have  $i0\text{-}in2: \iota0 \in Inf\text{-}F$ 
    unfolding  $no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\text{-}def$  by blast
  obtain  $\iota0\text{-}FL$  where  $i0\text{-}FL\text{-}in: \iota0\text{-}FL \in Inf\text{-}FL$  and  $i0\text{-}to\text{-}i0\text{-}FL: \iota0 = to\text{-}F\ \iota0\text{-}FL$ 
    using  $Inf\text{-}F\text{-}to\text{-}Inf\text{-}FL[OF\ i0\text{-}in2]$  unfolding  $to\text{-}F\text{-}def$ 
    by (smt (verit)  $Ex\text{-}list\text{-}of\text{-}length\ fst\text{-}conv\ inference.\text{exhaust}\text{-}sel\ inference.\text{sel}(1)$ 
       $inference.\text{sel}(2)\ map\text{-}fst\text{-}zip$ )
  have  $concl\text{-}swap: fst\ (concl\text{-}of\ \iota0\text{-}FL) = concl\text{-}of\ \iota0$ 
    unfolding  $concl\text{-}of\text{-}def\ i0\text{-}to\text{-}i0\text{-}FL\ to\text{-}F\text{-}def$  by simp
  have  $subs1: ((\mathcal{G}\text{-}I\text{-}L\text{-}q\ q0\ \iota0\text{-}FL) \neq None \wedge$ 
     $the\ (\mathcal{G}\text{-}I\text{-}L\text{-}q\ q0\ \iota0\text{-}FL) \subseteq Red\text{-}I\text{-}q\ q0\ (\mathcal{G}\text{-}Fset\text{-}q\ q0\ N))$ 
     $\vee ((\mathcal{G}\text{-}I\text{-}L\text{-}q\ q0\ \iota0\text{-}FL = None) \wedge$ 
     $\mathcal{G}\text{-}F\text{-}L\text{-}q\ q0\ (concl\text{-}of\ \iota0\text{-}FL) \subseteq (\mathcal{G}\text{-}Fset\text{-}q\ q0\ N \cup Red\text{-}F\text{-}q\ q0\ (\mathcal{G}\text{-}Fset\text{-}q\ q0\ N)))$ 
    using  $i0\text{-}in\ i0\text{-}to\text{-}i0\text{-}FL\ concl\text{-}swap$  unfolding  $no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\text{-}def$  by simp
  then have  $\iota0\text{-}FL \in Red\text{-}I\mathcal{G}\text{-}q\ q0\ N$ 
    using  $i0\text{-}FL\text{-}in$  unfolding  $Red\text{-}I\mathcal{G}\text{-}q\text{-}def$  by simp
  then show  $\iota0 \in to\text{-}F\ 'X0\ N$ 
    using  $x0\text{-}is\ i0\text{-}to\text{-}i0\text{-}FL\ i0\text{-}in2$  by blast
qed
qed
then have  $Y \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ 'Q \implies to\text{-}F\ \iota \in Y\ (fst\ 'N)$  for  $Y$ 
  using  $i\text{-}in2\ no\text{-}labels.Red\text{-}I\text{-}def\ std\text{-}Red\text{-}I\text{-}eq\ red\text{-}inf\text{-}impl$  by force
then show  $to\text{-}F\ \iota \in no\text{-}labels.Red\text{-}I\mathcal{G}\ (fst\ 'N)$ 
  unfolding  $Red\text{-}I\text{-}def\ no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}def$  by blast
next
assume  $to\text{-}F\text{-}in: to\text{-}F\ \iota \in no\text{-}labels.Red\text{-}I\mathcal{G}\ (fst\ 'N)$ 
have  $imp\text{-}to\text{-}F: X \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ 'Q \implies to\text{-}F\ \iota \in X\ (fst\ 'N)$  for  $X$ 
  using  $to\text{-}F\text{-}in$  unfolding  $no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}def$  by blast
then have  $to\text{-}F\text{-}in2: to\text{-}F\ \iota \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q\ (fst\ 'N)$  if  $q \in Q$  for  $q$ 
  using  $that$  by auto
have  $Red\text{-}I\mathcal{G}\text{-}q\ q\ N = \{\iota0\text{-}FL \in Inf\text{-}FL.\ to\text{-}F\ \iota0\text{-}FL \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q\ (fst\ 'N)\}$  for  $q$ 
proof
  show  $Red\text{-}I\mathcal{G}\text{-}q\ q\ N \subseteq \{\iota0\text{-}FL \in Inf\text{-}FL.\ to\text{-}F\ \iota0\text{-}FL \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q\ (fst\ 'N)\}$ 
  proof
    fix  $q0\ \iota1$ 
    assume
       $i1\text{-}in: \iota1 \in Red\text{-}I\mathcal{G}\text{-}q\ q0\ N$ 
    have  $i1\text{-}in2: \iota1 \in Inf\text{-}FL$ 
      using  $i1\text{-}in$  unfolding  $Red\text{-}I\mathcal{G}\text{-}q\text{-}def$  by blast
    then have  $to\text{-}F\text{-}i1\text{-}in: to\text{-}F\ \iota1 \in Inf\text{-}F$ 
      using  $Inf\text{-}FL\text{-}to\text{-}Inf\text{-}F$  unfolding  $to\text{-}F\text{-}def$  by blast
    have  $concl\text{-}swap: fst\ (concl\text{-}of\ \iota1) = concl\text{-}of\ (to\text{-}F\ \iota1)$ 
      unfolding  $concl\text{-}of\text{-}def\ to\text{-}F\text{-}def$  by simp
    then have  $i1\text{-}to\text{-}F\text{-}in: to\text{-}F\ \iota1 \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q0\ (fst\ 'N)$ 
      using  $i1\text{-}in\ to\text{-}F\text{-}i1\text{-}in$  unfolding  $Red\text{-}I\mathcal{G}\text{-}q\text{-}def\ no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\text{-}def$  by force
    show  $\iota1 \in \{\iota0\text{-}FL \in Inf\text{-}FL.\ to\text{-}F\ \iota0\text{-}FL \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q0\ (fst\ 'N)\}$ 
      using  $i1\text{-}in2\ i1\text{-}to\text{-}F\text{-}in$  by blast
  qed
next
show  $\{\iota0\text{-}FL \in Inf\text{-}FL.\ to\text{-}F\ \iota0\text{-}FL \in no\text{-}labels.Red\text{-}I\mathcal{G}\text{-}q\ q\ (fst\ 'N)\} \subseteq Red\text{-}I\mathcal{G}\text{-}q\ q\ N$ 

```

**proof**

**fix**  $q0 \ \iota1$

**assume**

$i1\text{-in}: \iota1 \in \{\iota0\text{-FL} \in \text{Inf-FL}. \text{to-F } \iota0\text{-FL} \in \text{no-labels.Red-I-G-q } q0 \text{ (fst ' N)}\}$

**then have**  $i1\text{-in}2: \iota1 \in \text{Inf-FL}$  **by** *blast*

**then have**  $\text{to-F-}i1\text{-in}: \text{to-F } \iota1 \in \text{Inf-F}$

**using**  $\text{Inf-FL-to-Inf-F}$  **unfolding**  $\text{to-F-def}$  **by** *blast*

**have**  $\text{concl-swap}: \text{fst}(\text{concl-of } \iota1) = \text{concl-of}(\text{to-F } \iota1)$

**unfolding**  $\text{concl-of-def}$   $\text{to-F-def}$  **by** *simp*

**then have**  $((\mathcal{G}\text{-I-L-q } q0 \ \iota1) \neq \text{None} \wedge \text{the}(\mathcal{G}\text{-I-L-q } q0 \ \iota1) \subseteq \text{Red-I-q } q0 \ (\mathcal{G}\text{-Fset-q } q0 \ N))$

$\vee (\mathcal{G}\text{-I-L-q } q0 \ \iota1 = \text{None} \wedge$

$\mathcal{G}\text{-F-L-q } q0 \ (\text{concl-of } \iota1) \subseteq \mathcal{G}\text{-Fset-q } q0 \ N \cup \text{Red-F-q } q0 \ (\mathcal{G}\text{-Fset-q } q0 \ N))$

**using**  $i1\text{-in}$  **unfolding**  $\text{no-labels.Red-I-G-q-def}$  **by** *auto*

**then show**  $\iota1 \in \text{Red-I-G-q } q0 \ N$

**using**  $i1\text{-in}2$  **unfolding**  $\text{Red-I-G-q-def}$  **by** *blast*

**qed**

**qed**

**then have**  $\iota \in \text{Red-I-G-q } q \ N$  **if**  $q \in Q$  **for**  $q$

**using**  $\text{that to-F-in}2$   $i\text{-in}$  **unfolding**  $\text{Red-I-G-q-def}$   $\text{no-labels.Red-I-G-q-def}$  **by** *auto*

**then show**  $\iota \in \text{Red-I-G } N$

**unfolding**  $\text{Red-I-G-def}$  **by** *blast*

**qed**

**lemma** *red-labeled-clauses:*

**assumes**  $\langle C \in \text{no-labels.Red-F-G-empty}(\text{fst ' N}) \vee$

$(\exists C' \in \text{fst ' N}. C' \prec \cdot C) \vee (\exists (C', L') \in N. L' \sqsubset L \wedge C' \preceq \cdot C) \rangle$

**shows**  $\langle (C, L) \in \text{Red-F } N \rangle$

**proof** –

**note** *assms*

**moreover have**  $i: \langle C \in \text{no-labels.Red-F-G-empty}(\text{fst ' N}) \implies (C, L) \in \text{Red-F } N \rangle$

**proof** –

**assume**  $C \in \text{no-labels.Red-F-G-empty}(\text{fst ' N})$

**then have**  $C \in \text{no-labels.Red-F-G-empty-q } q \ (\text{fst ' N})$  **if**  $q \in Q$  **for**  $q$

**unfolding**  $\text{no-labels.Red-F-G-empty-def}$  **using**  $\text{that}$  **by** *fast*

**then have**  $g\text{-in-red}: \mathcal{G}\text{-F-q } q \ C \subseteq \text{Red-F-q } q \ (\text{no-labels.G-Fset-q } q \ (\text{fst ' N}))$  **if**  $q \in Q$  **for**  $q$

**unfolding**  $\text{no-labels.Red-F-G-empty-q-def}$  **using**  $\text{that}$  **by** *blast*

**have**  $\mathcal{G}\text{-F-L-q } q \ (C, L) \subseteq \text{Red-F-q } q \ (\mathcal{G}\text{-Fset-q } q \ N)$  **if**  $q \in Q$  **for**  $q$

**using**  $\text{that } g\text{-in-red}$  **by** *simp*

**then show** *?thesis*

**unfolding**  $\text{Red-F-def}$   $\text{Red-F-G-q-def}$  **by** *blast*

**qed**

**moreover have**  $ii: \langle \exists C' \in \text{fst ' N}. C' \prec \cdot C \implies (C, L) \in \text{Red-F } N \rangle$

**proof** –

**assume**  $\exists C' \in \text{fst ' N}. C' \prec \cdot C$

**then obtain**  $C'$  **where**  $c'\text{-in}: C' \in \text{fst ' N}$  **and**  $c\text{-prec-}c': C' \prec \cdot C$  **by** *blast*

**obtain**  $L'$  **where**  $c'\text{-l'-in}: (C', L') \in N$  **using**  $c'\text{-in}$  **by** *auto*

**have**  $c'\text{-l'-prec}: (C', L') \sqsubset (C, L)$

**using**  $c\text{-prec-}c'$  **unfolding**  $\text{Prec-FL-def}$  **by** *simp*

**have**  $c\text{-in-}c'\text{-g}: \mathcal{G}\text{-F-q } q \ C \subseteq \mathcal{G}\text{-F-q } q \ C'$  **if**  $q \in Q$  **for**  $q$

**using**  $\text{prec-F-grounding}[OF \text{ that } c\text{-prec-}c']$  **by** *presburger*

**then have**  $\mathcal{G}\text{-F-L-q } q \ (C, L) \subseteq \mathcal{G}\text{-F-L-q } q \ (C', L')$  **if**  $q \in Q$  **for**  $q$

**using**  $\text{that}$  **by** *auto*

**then have**  $(C, L) \in \text{Red-F-G-q } q \ N$  **if**  $q \in Q$  **for**  $q$

**unfolding**  $\text{Red-F-G-q-def}$  **using**  $\text{that } c'\text{-l'-in } c'\text{-l'-prec}$  **by** *blast*

**then show** *?thesis*  
    **unfolding** *Red-F-def* **by** *blast*  
**qed**  
**moreover have** *iii:  $\langle \exists (C', L') \in N. L' \sqsubseteq L \wedge C' \preceq C \implies (C, L) \in \text{Red-F } N \rangle$*   
**proof** –  
    **assume**  $\exists (C', L') \in N. L' \sqsubseteq L \wedge C' \preceq C$   
    **then obtain**  $C' L'$  **where** *c'-l'-in:  $(C', L') \in N$*  **and** *l'-sub-l:  $L' \sqsubseteq L$*  **and** *c'-sub-c:  $C' \preceq C$*   
    **by** *fast*  
    **have**  $(C, L) \in \text{Red-F } N$  **if**  $C' \prec C$   
    **using** *that c'-l'-in ii by fastforce*  
    **moreover** {  
        **assume** *equiv-c-c':  $C \doteq C'$*   
        **then have** *equiv-c'-c:  $C' \doteq C$*   
        **using** *equiv-equiv-F by (simp add: equivp-symp)*  
        **then have** *c'-l'-prec:  $(C', L') \sqsubseteq (C, L)$*   
        **using** *l'-sub-l unfolding Prec-FL-def by simp*  
        **have**  $\mathcal{G}\text{-F-q } q \ C = \mathcal{G}\text{-F-q } q \ C'$  **if**  $q \in Q$  **for**  $q$   
        **using** *that equiv-F-grounding equiv-c-c' equiv-c'-c by (simp add: set-eq-subset)*  
        **then have**  $\mathcal{G}\text{-F-L-q } q \ (C, L) = \mathcal{G}\text{-F-L-q } q \ (C', L')$  **if**  $q \in Q$  **for**  $q$   
        **using** *that by auto*  
        **then have**  $(C, L) \in \text{Red-F-}\mathcal{G}\text{-q } q \ N$  **if**  $q \in Q$  **for**  $q$   
        **unfolding** *Red-F-}\mathcal{G}\text{-q-def using that c'-l'-in c'-l'-prec by blast*  
        **then have** *?thesis*  
        **unfolding** *Red-F-def* **by** *blast*  
    }  
    **ultimately show** *?thesis*  
    **using** *c'-sub-c equiv-equiv-F equivp-symp by fastforce*  
**qed**  
**ultimately show** *?thesis*  
    **by** *blast*  
**qed**  
**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 }-F-q }-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  $\implies$  'g set  $\implies$  'g set  $\implies$  bool and*  
    *Inf-G-q :: 'q  $\implies$  'g inference set and*  
    *Red-I-q :: 'q  $\implies$  'g set  $\implies$  'g inference set and*  
    *Red-F-q :: 'q  $\implies$  'g set  $\implies$  'g set and*  
    *}-F-q :: 'q  $\implies$  'f  $\implies$  'g set and*  
    *}-I-q :: 'q  $\implies$  'f inference  $\implies$  'g inference set option and*  
    *Equiv-F :: 'f  $\implies$  'f  $\implies$  bool (infix  $\doteq$  50) and*  
    *Prec-F :: 'f  $\implies$  'f  $\implies$  bool (infix  $\prec$  50) and*  
    *Prec-L :: 'l  $\implies$  'l  $\implies$  bool (infix  $\sqsubseteq L$  50) and*  
    *active :: 'l +*  
**assumes**  
    *inf-have-prems:  $\iota F \in \text{Inf-F} \implies \text{prems-of } \iota F \neq []$*   
**begin**



**lemma** *labeled-inf-have-prems*:  $\iota \in \text{Inf-FL} \implies \text{prems-of } \iota \neq []$   
**using** *inf-have-prems* **by** *fastforce*

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

**lemma** *one-step-equiv*:  $N1 \rightsquigarrow_{GC} N2 \implies N1 \triangleright_L N2$

**proof** (*cases*  $N1$   $N2$  *rule*: *step.cases*)

**show**  $N1 \rightsquigarrow_{GC} N2 \implies N1 \rightsquigarrow_{GC} N2$  **by** *blast*

**next**

**fix**  $N M M'$

**assume**

*gc-step*:  $N1 \rightsquigarrow_{GC} N2$  **and**

*n1-is*:  $N1 = N \cup M$  **and**

*n2-is*:  $N2 = N \cup M'$  **and**

*m-red*:  $M \subseteq \text{Red-F } (N \cup M')$  **and**

*active-empty*: *active-subset*  $M' = \{\}$

**have**  $N1 - N2 \subseteq \text{Red-F } N2$

**using** *n1-is* *n2-is* *m-red* **by** *auto*

**then show**  $N1 \triangleright_L N2$

**unfolding** *derive.simps* **by** *blast*

**next**

**fix**  $N C L M$

**assume**

*gc-step*:  $N1 \rightsquigarrow_{GC} N2$  **and**

*n1-is*:  $N1 = N \cup \{(C, L)\}$  **and**

*not-active*:  $L \neq \text{active}$  **and**

*n2-is*:  $N2 = N \cup \{(C, \text{active})\} \cup M$  **and**

*active-empty*: *active-subset*  $M = \{\}$

**have**  $(C, \text{active}) \in N2$  **using** *n2-is* **by** *auto*

**moreover have**  $C \preceq C$  **using** *equiv-equiv-F* **by** (*metis equivp-def*)

**moreover have**  $\text{active} \sqsubseteq_L L$  **using** *active-minimal*[*OF not-active*].

**ultimately have**  $\{(C, L)\} \subseteq \text{Red-F } N2$

**using** *red-labeled-clauses* **by** *blast*

**moreover have**  $N1 - N2 = \{\} \vee N1 - N2 = \{(C, L)\}$  **using** *n1-is* *n2-is* **by** *blast*

**ultimately have**  $N1 - N2 \subseteq \text{Red-F } N2$

**using** *std-Red-F-eq* **by** *blast*

**then show**  $N1 \triangleright_L N2$

**unfolding** *derive.simps* **by** *blast*

**qed**

**lemma** *gc-to-red*:  $\text{chain } (\rightsquigarrow_{GC}) Ns \implies \text{chain } (\triangleright_L) Ns$

**using** *one-step-equiv Lazy-List-Chain.chain-mono* **by** *blast*

**lemma** (**in**  $-$ ) *all-ex-finite-set*:  $(\forall (j::\text{nat}) \in \{0..<m\}. \exists (n::\text{nat}). P j n) \implies$

$(\forall n1 n2. \forall j \in \{0..<m\}. P j n1 \longrightarrow P j n2 \longrightarrow n1 = n2) \implies \text{finite } \{n. \exists j \in \{0..<m\}. P j n\}$  **for**  $m$

$P$

**proof** –

**fix**  $m::nat$  **and**  $P:: nat \Rightarrow nat \Rightarrow bool$

**assume**

$allj\text{-}exn: \forall j \in \{0..<m\}. \exists n. P\ j\ n$  **and**

$uniq\text{-}n: \forall n1\ n2. \forall j \in \{0..<m\}. P\ j\ n1 \longrightarrow P\ j\ n2 \longrightarrow n1 = n2$

**have**  $\{n. \exists j \in \{0..<m\}. P\ j\ n\} = (\bigcup((\lambda j. \{n. P\ j\ n\}) \text{ ‘ } \{0..<m\}))$  **by** *blast*

**then have**  $imp\text{-}finite: (\forall j \in \{0..<m\}. finite\ \{n. P\ j\ n\}) \Longrightarrow finite\ \{n. \exists j \in \{0..<m\}. P\ j\ n\}$

**using**  $finite\text{-}UN$ [of  $\{0..<m\}$   $\lambda j. \{n. P\ j\ n\}$ ] **by** *simp*

**have**  $\forall j \in \{0..<m\}. \exists! n. P\ j\ n$  **using**  $allj\text{-}exn\ uniq\text{-}n$  **by** *blast*

**then have**  $\forall j \in \{0..<m\}. finite\ \{n. P\ j\ n\}$  **by** (*metis*  $bounded\text{-}nat\text{-}set\text{-}is\text{-}finite\ lessI\ mem\text{-}Collect\text{-}eq$ )

**then show**  $finite\ \{n. \exists j \in \{0..<m\}. P\ j\ n\}$  **using**  $imp\text{-}finite$  **by** *simp*

**qed**

**lemma** *gc-fair*:

**assumes**

$deriv: chain\ (\sim GC)\ Ns$  **and**

$init\text{-}state: active\text{-}subset\ (lhd\ Ns) = \{\}$  **and**

$final\text{-}state: passive\text{-}subset\ (Liminf\text{-}llist\ Ns) = \{\}$

**shows**  $fair\ Ns$

**unfolding** *fair-def*

**proof**

**fix**  $\iota$

**assume**  $i\text{-}in: \iota \in Inf\text{-}from\ (Liminf\text{-}llist\ Ns)$

**note**  $lhd\text{-}is = lhd\text{-}conv\text{-}lnth$ [*OF*  $chain\text{-}not\text{-}lnull$ [*OF*  $deriv$ ]]

**have**  $i\text{-}in\text{-}inf\text{-}fl: \iota \in Inf\text{-}FL$  **using**  $i\text{-}in$  **unfolding** *Inf-from-def* **by** *blast*

**have**  $Liminf\text{-}llist\ Ns = active\text{-}subset\ (Liminf\text{-}llist\ Ns)$

**using**  $final\text{-}state$  **unfolding**  $passive\text{-}subset\text{-}def\ active\text{-}subset\text{-}def$  **by** *blast*

**then have**  $i\text{-}in2: \iota \in Inf\text{-}from\ (active\text{-}subset\ (Liminf\text{-}llist\ Ns))$  **using**  $i\text{-}in$  **by** *simp*

**define**  $m$  **where**  $m = length\ (prems\text{-}of\ \iota)$

**then have**  $m\text{-}def\text{-}F: m = length\ (prems\text{-}of\ (to\text{-}F\ \iota))$  **unfolding** *to-F-def* **by** *simp*

**have**  $i\text{-}in\text{-}F: to\text{-}F\ \iota \in Inf\text{-}F$

**using**  $i\text{-}in\ Inf\text{-}FL\text{-}to\text{-}Inf\text{-}F$  **unfolding** *Inf-from-def to-F-def* **by** *blast*

**then have**  $m\text{-}pos: m > 0$  **using**  $m\text{-}def\text{-}F$  **using**  $inf\text{-}have\text{-}prems$  **by** *blast*

**have**  $exist\text{-}nj: \forall j \in \{0..<m\}. (\exists nj. enat\ (Suc\ nj) < llength\ Ns \wedge$

$prems\text{-}of\ \iota ! j \notin active\text{-}subset\ (lnth\ Ns\ nj) \wedge$

$(\forall k. k > nj \longrightarrow enat\ k < llength\ Ns \longrightarrow prems\text{-}of\ \iota ! j \in active\text{-}subset\ (lnth\ Ns\ k)))$

**proof** *clarify*

**fix**  $j$

**assume**  $j\text{-}in: j \in \{0..<m\}$

**then obtain**  $C$  **where**  $c\text{-}is: (C, active) = prems\text{-}of\ \iota ! j$

**using**  $i\text{-}in2$  **unfolding**  $m\text{-}def\ Inf\text{-}from\text{-}def\ active\text{-}subset\text{-}def$

**by** (*smt*  $Collect\text{-}mem\text{-}eq\ Collect\text{-}mono\text{-}iff\ atLeastLessThan\text{-}iff\ nth\text{-}mem\ old.\text{prod.}\text{exhaust}\ snd\text{-}conv$ )

**then have**  $(C, active) \in Liminf\text{-}llist\ Ns$

**using**  $j\text{-}in\ i\text{-}in$  **unfolding**  $m\text{-}def\ Inf\text{-}from\text{-}def$  **by** *force*

**then obtain**  $nj$  **where**  $nj\text{-}is: enat\ nj < llength\ Ns$  **and**

$c\text{-}in2: (C, active) \in \bigcap\ (lnth\ Ns\ \text{ ‘ } \{k. nj \leq k \wedge enat\ k < llength\ Ns\})$

**unfolding**  $Liminf\text{-}llist\text{-}def$  **using**  $init\text{-}state$  **by** *blast*

**then have**  $c\text{-}in3: \forall k. k \geq nj \longrightarrow enat\ k < llength\ Ns \longrightarrow (C, active) \in lnth\ Ns\ k$  **by** *blast*

**have**  $nj\text{-}pos: nj > 0$  **using**  $init\text{-}state\ c\text{-}in2\ nj\text{-}is$

**unfolding**  $active\text{-}subset\text{-}def\ lhd\text{-}is$  **by** *force*

**obtain**  $nj\text{-}min$  **where**  $nj\text{-}min\text{-}is: nj\text{-}min = (LEAST\ nj. enat\ nj < llength\ Ns \wedge$

$(C, active) \in \bigcap\ (lnth\ Ns\ \text{ ‘ } \{k. nj \leq k \wedge enat\ k < llength\ Ns\}))$  **by** *blast*

**then have**  $in\text{-}allk: \forall k. k \geq nj\text{-}min \longrightarrow enat\ k < llength\ Ns \longrightarrow (C, active) \in (lnth\ Ns\ k)$

**using**  $c\text{-}in3\ nj\text{-}is\ c\text{-}in2$

by (metis (mono-tags, lifting) INT-E LeastI-ex mem-Collect-eq)  
 have *njm-smaller-D*: enat *nj-min* < llength *Ns*  
 using *nj-min-is*  
 by (smt LeastI-ex <math>\langle \bigwedge thesis. (\bigwedge nj. \llbracket enat\ nj < llength\ Ns; (C, active) \in \bigcap (lnth\ Ns\ ' \{k. nj \leq k \wedge enat\ k < llength\ Ns\}) \rrbracket \implies thesis) \implies thesis \rangle</math>)  
 have *nj-min* > 0  
 using *nj-is c-in2 nj-pos nj-min-is lhd-is*  
 by (metis (mono-tags, lifting) Collect-empty-eq <math>\langle (C, active) \in Liminf-llist\ Ns \rangle</math>  
 <math>\langle Liminf-llist\ Ns = active-subset\ (Liminf-llist\ Ns) \rangle</math>  
 <math>\langle \forall k \geq nj-min. enat\ k < llength\ Ns \longrightarrow (C, active) \in lnth\ Ns\ k \rangle active-subset-def\ init-state</math>  
*linorder-not-less mem-Collect-eq zero-enat-def chain-length-pos[OF deriv]*)  
 then obtain *njm-prec* where *nj-prec-is*: Suc *njm-prec* = *nj-min* using *gr0-conv-Suc* by auto  
 then have *njm-prec-njm*: *njm-prec* < *nj-min* by blast  
 then have *njm-prec-njm-enat*: enat *njm-prec* < enat *nj-min* by simp  
 have *njm-prec-smaller-d*: *njm-prec* < llength *Ns*  
 by (rule less-trans[OF *njm-prec-njm-enat nj-smaller-D*])  
 have *njm-prec-all-suc*:  $\forall k > nj-prec. enat\ k < llength\ Ns \longrightarrow (C, active) \in lnth\ Ns\ k$   
 using *nj-prec-is in-allk* by simp  
 have *notin-njm-prec*:  $(C, active) \notin lnth\ Ns\ nj-prec$   
 proof (rule ccontr)  
 assume  $\neg (C, active) \notin lnth\ Ns\ nj-prec$   
 then have *absurd-hyp*:  $(C, active) \in lnth\ Ns\ nj-prec$  by simp  
 have *prec-smaller*: enat *njm-prec* < llength *Ns* using *nj-min-is nj-prec-is*  
 by (smt LeastI-ex Suc-leD <math>\langle \bigwedge thesis. (\bigwedge nj. \llbracket enat\ nj < llength\ Ns; (C, active) \in \bigcap (lnth\ Ns\ ' \{k. nj \leq k \wedge enat\ k < llength\ Ns\}) \rrbracket \implies thesis) \implies thesis \rangle</math>  
 enat-ord-simps(1) le-eq-less-or-eq le-less-trans)  
 have  $(C, active) \in \bigcap (lnth\ Ns\ ' \{k. nj-prec \leq k \wedge enat\ k < llength\ Ns\})$   
 proof -  
 {  
 fix *k*  
 assume *k-in*: *njm-prec*  $\leq k \wedge enat\ k < llength\ Ns$   
 have  $k = nj-prec \implies (C, active) \in lnth\ Ns\ k$  using *absurd-hyp* by simp  
 moreover have *njm-prec* < *k*  $\implies (C, active) \in lnth\ Ns\ k$   
 using *nj-prec-is in-allk k-in* by simp  
 ultimately have  $(C, active) \in lnth\ Ns\ k$  using *k-in* by fastforce  
 }  
 then show  $(C, active) \in \bigcap (lnth\ Ns\ ' \{k. nj-prec \leq k \wedge enat\ k < llength\ Ns\})$  by blast  
 qed  
 then have enat *njm-prec* < llength *Ns*  $\wedge$   
 $(C, active) \in \bigcap (lnth\ Ns\ ' \{k. nj-prec \leq k \wedge enat\ k < llength\ Ns\})$   
 using *prec-smaller* by blast  
 then show *False*  
 using *nj-min-is nj-prec-is Orderings.wellorder-class.not-less-Least njm-prec-njm* by blast  
 qed  
 then have *notin-active-subs-njm-prec*:  $(C, active) \notin active-subset\ (lnth\ Ns\ nj-prec)$   
 unfolding *active-subset-def* by blast  
 then show  $\exists nj. enat\ (Suc\ nj) < llength\ Ns \wedge prems-of\ \iota ! j \notin active-subset\ (lnth\ Ns\ nj) \wedge$   
 $(\forall k. k > nj \longrightarrow enat\ k < llength\ Ns \longrightarrow prems-of\ \iota ! j \in active-subset\ (lnth\ Ns\ k))$   
 using *c-is njm-prec-all-suc njm-prec-smaller-d* by (metis (mono-tags, lifting)  
*active-subset-def mem-Collect-eq nj-prec-is njm-smaller-D snd-conv*)  
 qed  
 define *nj-set* where *nj-set* =  $\{nj. (\exists j \in \{0..<m\}. enat\ (Suc\ nj) < llength\ Ns \wedge$   
 $prems-of\ \iota ! j \notin active-subset\ (lnth\ Ns\ nj) \wedge$   
 $(\forall k. k > nj \longrightarrow enat\ k < llength\ Ns \longrightarrow prems-of\ \iota ! j \in active-subset\ (lnth\ Ns\ k)))\}$   
 then have *nj-not-empty*: *nj-set*  $\neq \{\}$

**proof** –

**have** *zero-in*:  $0 \in \{0..<m\}$  **using** *m-pos* **by** *simp*  
**then obtain** *n0* **where** *enat* (*Suc* *n0*) < *llength* *Ns* **and**  
*prems-of*  $\iota ! 0 \notin \text{active-subset } (\text{lth } Ns \ n0)$  **and**  
 $\forall k > n0. \text{enat } k < \text{llength } Ns \longrightarrow \text{prems-of } \iota ! 0 \in \text{active-subset } (\text{lth } Ns \ k)$   
**using** *exist-nj* **by** *fast*  
**then have**  $n0 \in \text{nj-set}$  **unfolding** *nj-set-def* **using** *zero-in* **by** *blast*  
**then show**  $\text{nj-set} \neq \{\}$  **by** *auto*

**qed**

**have** *nj-finite*: *finite* *nj-set*  
**using** *all-ex-finite-set*[*OF exist-nj*]  
**by** (*metis* (*no-types*, *lifting*) *Suc-ile-eq* *dual-order.strict-implies-order*  
*linorder-neqE-nat* *nj-set-def*)

**have**  $\exists n \in \text{nj-set}. \forall nj \in \text{nj-set}. nj \leq n$

**using** *nj-not-empty* *nj-finite* **using** *Max-ge* *Max-in* **by** *blast*

**then obtain** *n* **where** *n-in*:  $n \in \text{nj-set}$  **and** *n-bigger*:  $\forall nj \in \text{nj-set}. nj \leq n$  **by** *blast*

**then obtain** *j0* **where** *j0-in*:  $j0 \in \{0..<m\}$  **and** *suc-n-length*: *enat* (*Suc* *n*) < *llength* *Ns* **and**

*j0-notin*: *prems-of*  $\iota ! j0 \notin \text{active-subset } (\text{lth } Ns \ n)$  **and**

*j0-allin*:  $(\forall k. k > n \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow \text{prems-of } \iota ! j0 \in \text{active-subset } (\text{lth } Ns \ k))$

**unfolding** *nj-set-def* **by** *blast*

**obtain** *C0* **where** *C0-is*: *prems-of*  $\iota ! j0 = (C0, \text{active})$  **using** *j0-in*

**using** *i-in2* **unfolding** *m-def* *Inf-from-def* *active-subset-def*

**by** (*smt* *Collect-mem-eq* *Collect-mono-iff* *atLeastLessThan-iff* *nth-mem* *old.prod.exhaust* *snd-conv*)

**then have** *C0-prems-i*:  $(C0, \text{active}) \in \text{set } (\text{prems-of } \iota)$  **using** *in-set-conv-nth* *j0-in* *m-def* **by** *force*

**have** *C0-in*:  $(C0, \text{active}) \in (\text{lth } Ns \ (\text{Suc } n))$

**using** *C0-is* *j0-allin* *suc-n-length* **by** (*simp* *add*: *active-subset-def*)

**have** *C0-notin*:  $(C0, \text{active}) \notin (\text{lth } Ns \ n)$  **using** *C0-is* *j0-notin* **unfolding** *active-subset-def* **by** *simp*

**have** *step-n*:  $\text{lth } Ns \ n \sim_{GC} \text{lth } Ns \ (\text{Suc } n)$

**using** *deriv* *chain-lth-rel* *n-in* **unfolding** *nj-set-def* **by** *blast*

**have**  $\exists N \ C \ L \ M. (\text{lth } Ns \ n = N \cup \{(C, L)\} \wedge$

$\text{lth } Ns \ (\text{Suc } n) = N \cup \{(C, \text{active})\} \cup M \wedge L \neq \text{active} \wedge \text{active-subset } M = \{\} \wedge$

$\text{no-labels.} \text{Inf-between } (\text{fst } ' \text{active-subset } N) \ \{C\}$

$\subseteq \text{no-labels.Red-I } (\text{fst } '(N \cup \{(C, \text{active})\} \cup M)))$ )

**proof** –

**have** *proc-or-infer*:  $(\exists N1 \ N \ M \ N2 \ M'. \text{lth } Ns \ n = N1 \wedge \text{lth } Ns \ (\text{Suc } n) = N2 \wedge N1 = N \cup M \wedge$

$N2 = N \cup M' \wedge M \subseteq \text{Red-F } (N \cup M') \wedge \text{active-subset } M' = \{\}) \vee$

$(\exists N1 \ N \ C \ L \ N2 \ M. \text{lth } Ns \ n = N1 \wedge \text{lth } Ns \ (\text{Suc } n) = N2 \wedge N1 = N \cup \{(C, L)\} \wedge$

$N2 = N \cup \{(C, \text{active})\} \cup M \wedge L \neq \text{active} \wedge \text{active-subset } M = \{\} \wedge$

$\text{no-labels.} \text{Inf-between } (\text{fst } ' \text{active-subset } N) \ \{C\} \subseteq$

$\text{no-labels.Red-I } (\text{fst } '(N \cup \{(C, \text{active})\} \cup M)))$ )

**using** *step.simps*[*of* *lth* *Ns* *n* *lth* *Ns* (*Suc* *n*)] *step-n* **by** *blast*

**show** *?thesis*

**using** *C0-in* *C0-notin* *proc-or-infer* *j0-in* *C0-is*

**by** (*smt* *Un-iff* *active-subset-def* *mem-Collect-eq* *snd-conv* *sup-bot.right-neutral*)

**qed**

**then obtain** *N* *M* *L* **where** *inf-from-subs*:

*no-labels.} \text{Inf-between } (\text{fst } ' \text{active-subset } N) \ \{C0\}*

$\subseteq \text{no-labels.Red-I } (\text{fst } '(N \cup \{(C0, \text{active})\} \cup M))$  **and**

*nth-d-is*:  $\text{lth } Ns \ n = N \cup \{(C0, L)\}$  **and**

*suc-nth-d-is*:  $\text{lth } Ns \ (\text{Suc } n) = N \cup \{(C0, \text{active})\} \cup M$  **and**

*l-not-active*:  $L \neq \text{active}$

**using** *C0-in* *C0-notin* *j0-in* *C0-is* **using** *active-subset-def* **by** *fastforce*

**have**  $j \in \{0..<m\} \implies \text{prems-of } \iota ! j \neq \text{prems-of } \iota ! j0 \implies \text{prems-of } \iota ! j \in (\text{active-subset } N)$  **for** *j*

**proof** –

**fix**  $j$   
**assume**  $j\text{-in}$ :  $j \in \{0..<m\}$  **and**  
 $j\text{-not-}j0$ :  $\text{prems-of } \iota ! j \neq \text{prems-of } \iota ! j0$   
**obtain**  $nj$  **where**  $nj\text{-len}$ :  $\text{enat } (Suc\ nj) < \text{llength } Ns$  **and**  
 $nj\text{-prems}$ :  $\text{prems-of } \iota ! j \notin \text{active-subset } (\text{lth } Ns\ nj)$  **and**  
 $nj\text{-greater}$ :  $(\forall k. k > nj \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow \text{prems-of } \iota ! j \in \text{active-subset } (\text{lth } Ns\ k))$   
**using**  $\text{exist-nj } j\text{-in}$  **by**  $\text{blast}$   
**then have**  $nj \in nj\text{-set}$  **unfolding**  $nj\text{-set-def}$  **using**  $j\text{-in}$  **by**  $\text{blast}$   
**moreover have**  $nj \neq n$   
**proof** ( $\text{rule } c\text{contr}$ )  
**assume**  $\neg nj \neq n$   
**then have**  $\text{prems-of } \iota ! j = (C0, \text{active})$   
**using**  $C0\text{-in } C0\text{-notin } \text{step.simps}[of\ \text{lth } Ns\ n\ \text{lth } Ns\ (Suc\ n)]\ \text{step-n}$   
**by** ( $\text{smt } Un\text{-iff } nth\text{-d-is } suc\text{-nth-d-is } l\text{-not-active } \text{active-subset-def } \text{insertCI } \text{insertE } \text{lessI}$   
 $\text{mem-Collect-eq } nj\text{-greater } nj\text{-prems } \text{snd-conv } \text{suc-n-length}$ )  
**then show**  $False$  **using**  $j\text{-not-}j0$   $C0\text{-is}$  **by**  $\text{simp}$   
**qed**  
**ultimately have**  $nj < n$  **using**  $n\text{-bigger}$  **by**  $\text{force}$   
**then have**  $\text{prems-of } \iota ! j \in (\text{active-subset } (\text{lth } Ns\ n))$   
**using**  $nj\text{-greater } n\text{-in } \text{Suc-ile-eq } \text{dual-order.strict-implies-order}$  **unfolding**  $nj\text{-set-def}$  **by**  $\text{blast}$   
**then show**  $\text{prems-of } \iota ! j \in (\text{active-subset } N)$   
**using**  $nth\text{-d-is } l\text{-not-active}$  **unfolding**  $\text{active-subset-def}$  **by**  $\text{force}$   
**qed**  
**then have**  $\text{set } (\text{prems-of } \iota) \subseteq \text{active-subset } N \cup \{(C0, \text{active})\}$   
**using**  $C0\text{-prems-i } C0\text{-is } m\text{-def}$   
**by** ( $\text{metis } Un\text{-iff } \text{atLeast0LessThan } \text{in-set-conv-nth } \text{insertCI } \text{lessThan-iff } \text{subrelI}$ )  
**moreover have**  $\neg (\text{set } (\text{prems-of } \iota) \subseteq \text{active-subset } N - \{(C0, \text{active})\})$  **using**  $C0\text{-prems-i}$  **by**  $\text{blast}$   
**ultimately have**  $\iota \in \text{Inf-between } (\text{active-subset } N) \{(C0, \text{active})\}$   
**using**  $i\text{-in-inf-fl}$  **unfolding**  $\text{Inf-between-def } \text{Inf-from-def}$  **by**  $\text{blast}$   
**then have**  $\text{to-F } \iota \in \text{no-labels.Inf-between } (\text{fst } ' \text{active-subset } N) \{C0\}$   
**unfolding**  $\text{to-F-def } \text{Inf-between-def } \text{Inf-from-def}$   
 $\text{no-labels.Inf-between-def } \text{no-labels.Inf-from-def}$  **using**  $\text{Inf-FL-to-Inf-F}$   
**by**  $\text{force}$   
**then have**  $\text{to-F } \iota \in \text{no-labels.Red-I } (\text{fst } ' (\text{lth } Ns\ (Suc\ n)))$   
**using**  $\text{suc-nth-d-is } \text{inf-from-subs}$  **by**  $\text{fastforce}$   
**then have**  $\forall q \in Q. (\mathcal{G}\text{-I-q } q (\text{to-F } \iota) \neq \text{None} \wedge$   
 $\text{the } (\mathcal{G}\text{-I-q } q (\text{to-F } \iota)) \subseteq \text{Red-I-q } q (\bigcup (\mathcal{G}\text{-F-q } q ' \text{fst } ' \text{lth } Ns\ (Suc\ n))))$   
 $\vee (\mathcal{G}\text{-I-q } q (\text{to-F } \iota) = \text{None} \wedge$   
 $\mathcal{G}\text{-F-q } q (\text{concl-of } (\text{to-F } \iota)) \subseteq \bigcup (\mathcal{G}\text{-F-q } q ' \text{fst } ' \text{lth } Ns\ (Suc\ n)) \cup$   
 $\text{Red-F-q } q (\bigcup (\mathcal{G}\text{-F-q } q ' \text{fst } ' \text{lth } Ns\ (Suc\ n))))$ )  
**unfolding**  $\text{to-F-def } \text{no-labels.Red-I-def } \text{no-labels.Red-I-}\mathcal{G}\text{-q-def}$  **by**  $\text{blast}$   
**then have**  $\iota \in \text{Red-I-}\mathcal{G} (\text{lth } Ns\ (Suc\ n))$   
**using**  $i\text{-in-inf-fl}$  **unfolding**  $\text{Red-I-}\mathcal{G}\text{-def } \text{Red-I-}\mathcal{G}\text{-q-def}$  **by** ( $\text{simp add: to-F-def}$ )  
**then show**  $\iota \in \text{Sup-llist } (\text{lmap } \text{Red-I-}\mathcal{G} Ns)$   
**unfolding**  $\text{Sup-llist-def}$  **using**  $\text{suc-n-length}$  **by**  $\text{auto}$   
**qed**

**theorem**  $gc\text{-complete-Liminf}$ :

**assumes**

$\text{deriv}$ :  $\text{chain } (\rightsquigarrow GC) Ns$  **and**

$\text{init-state}$ :  $\text{active-subset } (\text{lhd } Ns) = \{\}$  **and**

$\text{final-state}$ :  $\text{passive-subset } (\text{Liminf-llist } Ns) = \{\}$  **and**

$b\text{-in}$ :  $B \in \text{Bot-F}$  **and**

$\text{bot-entailed}$ :  $\text{no-labels.entails-}\mathcal{G} (\text{fst } ' \text{lhd } Ns) \{B\}$

**shows**  $\exists BL \in \text{Bot-FL}. BL \in \text{Liminf-llist } Ns$

**proof** –

**note**  $lhd-is = lhd-conv-lnth[OF\ chain-not-lnull[OF\ deriv]]$   
**have**  $labeled-b-in: (B, active) \in Bot-FL$  **using**  $b-in$  **by**  $simp$   
**have**  $labeled-bot-entailed: entails-\mathcal{G}-L (lhd\ Ns) \{(B, active)\}$   
**using**  $labeled-entailment-lifting\ bot-entailed\ lhd-is$  **by**  $fastforce$   
**have**  $fair: fair\ Ns$  **using**  $gc-fair[OF\ deriv\ init-state\ final-state]$  .  
**then show**  $?thesis$   
**using**  $dynamically-complete-Liminf[OF\ labeled-b-in\ gc-to-red[OF\ deriv]\ fair$   
 $labeled-bot-entailed]$   
**by**  $blast$   
**qed**

**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** –

**note**  $lhd-is = lhd-conv-lnth[OF\ chain-not-lnull[OF\ deriv]]$   
**have**  $\exists BL \in Bot-FL. BL \in Liminf-llist\ Ns$   
**using**  $assms$  **by**  $(rule\ gc-complete-Liminf)$   
**then show**  $?thesis$   
**unfolding**  $Liminf-llist-def$  **by**  $auto$   
**qed**

**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**  
 $entails-q :: 'q \Rightarrow 'g\ set \Rightarrow 'g\ set \Rightarrow bool$  **and**  
 $Inf-G-q :: \langle 'q \Rightarrow 'g\ inference\ set \rangle$  **and**  
 $Red-I-q :: 'q \Rightarrow 'g\ set \Rightarrow 'g\ inference\ set$  **and**  
 $Red-F-q :: 'q \Rightarrow 'g\ set \Rightarrow 'g\ set$  **and**  
 $\mathcal{G}-F-q :: 'q \Rightarrow 'f \Rightarrow 'g\ set$  **and**  
 $\mathcal{G}-I-q :: 'q \Rightarrow 'f\ inference \Rightarrow 'g\ inference\ set\ option$  **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**  
 $process: N1 = N \cup M \Longrightarrow N2 = N \cup M' \Longrightarrow M \subseteq Red-F (N \cup M') \Longrightarrow$   
 $active-subset\ M' = \{\} \Longrightarrow (T, N1) \rightsquigarrow LGC (T, N2) \mid$

*schedule-infer*:  $T2 = T1 \cup T' \implies N1 = N \cup \{(C, L)\} \implies N2 = N \cup \{(C, \text{active})\} \implies$   
 $L \neq \text{active} \implies T' = \text{no-labels.Inf-between } (\text{fst } ' \text{ active-subset } N) \{C\} \implies$   
 $(T1, N1) \rightsquigarrow \text{LGC } (T2, N2) \mid$   
*compute-infer*:  $T1 = T2 \cup \{\iota\} \implies N2 = N1 \cup M \implies \text{active-subset } M = \{\} \implies$   
 $\iota \in \text{no-labels.Red-I } (\text{fst } ' (N1 \cup M)) \implies (T1, N1) \rightsquigarrow \text{LGC } (T2, N2) \mid$   
*delete-orphan-infers*:  $T1 = T2 \cup T' \implies$   
 $T' \cap \text{no-labels.Inf-from } (\text{fst } ' \text{ active-subset } N) = \{\} \implies (T1, N) \rightsquigarrow \text{LGC } (T2, N)$

**lemma** *premise-free-inf-always-from*:  $\iota \in \text{Inf-F} \implies \text{prems-of } \iota = [] \implies \iota \in \text{no-labels.Inf-from } N$   
**unfolding** *no-labels.Inf-from-def* **by** *simp*

**lemma** *one-step-equiv*:  $(T1, N1) \rightsquigarrow \text{LGC } (T2, N2) \implies N1 \triangleright L N2$

**proof** (*cases*  $(T1, N1)$   $(T2, N2)$ ) *rule: step.cases*

**show**  $(T1, N1) \rightsquigarrow \text{LGC } (T2, N2) \implies (T1, N1) \rightsquigarrow \text{LGC } (T2, N2)$  **by** *blast*

**next**

**fix**  $N M M'$

**assume**

*n1-is*:  $N1 = N \cup M$  **and**

*n2-is*:  $N2 = N \cup M'$  **and**

*m-red*:  $M \subseteq \text{Red-F } (N \cup M')$

**have**  $N1 - N2 \subseteq \text{Red-F } N2$

**using** *n1-is n2-is m-red* **by** *auto*

**then show**  $N1 \triangleright L N2$

**unfolding** *derive.simps* **by** *blast*

**next**

**fix**  $N C L M$

**assume**

*n1-is*:  $N1 = N \cup \{(C, L)\}$  **and**

*not-active*:  $L \neq \text{active}$  **and**

*n2-is*:  $N2 = N \cup \{(C, \text{active})\}$

**have**  $(C, \text{active}) \in N2$  **using** *n2-is* **by** *auto*

**moreover have**  $C \preceq C$  **by** (*metis equivp-def equiv-equiv-F*)

**moreover have**  $\text{active} \sqsubseteq L L$  **using** *active-minimal[OF not-active]* .

**ultimately have**  $\{(C, L)\} \subseteq \text{Red-F } N2$

**using** *red-labeled-clauses* **by** *blast*

**then have**  $N1 - N2 \subseteq \text{Red-F } N2$

**using** *std-Red-F-eq* **using** *n1-is n2-is* **by** *blast*

**then show**  $N1 \triangleright L N2$

**unfolding** *derive.simps* **by** *blast*

**next**

**fix**  $M$

**assume**

*n2-is*:  $N2 = N1 \cup M$

**have**  $N1 - N2 \subseteq \text{Red-F } N2$

**using** *n2-is* **by** *blast*

**then show**  $N1 \triangleright L N2$

**unfolding** *derive.simps* **by** *blast*

**next**

**assume** *n2-is*:  $N2 = N1$

**have**  $N1 - N2 \subseteq \text{Red-F } N2$

**using** *n2-is* **by** *blast*

**then show**  $N1 \triangleright L N2$

**unfolding** *derive.simps* **by** *blast*

**qed**

**lemma** *lgc-to-red*:  $\text{chain } (\rightsquigarrow \text{LGC}) \text{ } Ns \implies \text{chain } (\triangleright L) (\text{lmap snd } Ns)$   
**using** *one-step-equiv Lazy-List-Chain.chain-mono* **by** (*smt chain-lmap prod.collapse*)

**lemma** *lgc-fair*:

**assumes**

*deriv*:  $\text{chain } (\rightsquigarrow \text{LGC}) \text{ } Ns$  **and**

*init-state*:  $\text{active-subset } (\text{snd } (\text{lhd } Ns)) = \{\}$  **and**

*final-state*:  $\text{passive-subset } (\text{Liminf-llist } (\text{lmap snd } Ns)) = \{\}$  **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*:  $\text{Liminf-llist } (\text{lmap fst } Ns) = \{\}$

**shows** *fair* ( $\text{lmap snd } Ns$ )

**unfolding** *fair-def*

**proof**

**fix**  $\iota$

**assume** *i-in*:  $\iota \in \text{Inf-from } (\text{Liminf-llist } (\text{lmap snd } Ns))$

**note** *lhd-is* =  $\text{lhd-conv-lnth}[\text{OF chain-not-lnull}[\text{OF deriv}]]$

**have** *i-in-inf-fl*:  $\iota \in \text{Inf-FL}$  **using** *i-in* **unfolding** *Inf-from-def* **by** *blast*

**have**  $\text{Liminf-llist } (\text{lmap snd } Ns) = \text{active-subset } (\text{Liminf-llist } (\text{lmap snd } Ns))$

**using** *final-state* **unfolding** *passive-subset-def active-subset-def* **by** *blast*

**then have** *i-in2*:  $\iota \in \text{Inf-from } (\text{active-subset } (\text{Liminf-llist } (\text{lmap snd } Ns)))$

**using** *i-in* **by** *simp*

**define** *m* **where**  $m = \text{length } (\text{prems-of } \iota)$

**then have** *m-def-F*:  $m = \text{length } (\text{prems-of } (\text{to-F } \iota))$  **unfolding** *to-F-def* **by** *simp*

**have** *i-in-F*:  $\text{to-F } \iota \in \text{Inf-F}$

**using** *i-in Inf-FL-to-Inf-F* **unfolding** *Inf-from-def to-F-def* **by** *blast*

**have** *exist-nj*:  $\forall j \in \{0..<m\}. (\exists nj. \text{enat } (\text{Suc } nj) < \text{llength } Ns \wedge$

$\text{prems-of } \iota ! j \notin \text{active-subset } (\text{snd } (\text{lnth } Ns \ nj)) \wedge$

$(\forall k. k > nj \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow \text{prems-of } \iota ! j \in \text{active-subset } (\text{snd } (\text{lnth } Ns \ k))))$

**proof** *clarify*

**fix**  $j$

**assume** *j-in*:  $j \in \{0..<m\}$

**then obtain** *C* **where** *c-is*:  $(C, \text{active}) = \text{prems-of } \iota ! j$

**using** *i-in2* **unfolding** *m-def Inf-from-def active-subset-def*

**by** (*smt Collect-mem-eq Collect-mono-iff atLeastLessThan-iff nth-mem old.prod.exhaust snd-conv*)

**then have**  $(C, \text{active}) \in \text{Liminf-llist } (\text{lmap snd } Ns)$

**using** *j-in i-in* **unfolding** *m-def Inf-from-def* **by** *force*

**then obtain** *nj* **where** *nj-is*:  $\text{enat } nj < \text{llength } Ns$  **and**

*c-in2*:  $(C, \text{active}) \in \bigcap (\text{snd } ' (\text{lnth } Ns \ ' \{k. nj \leq k \wedge \text{enat } k < \text{llength } Ns\}))$

**unfolding** *Liminf-llist-def* **using** *init-state* **by** *fastforce*

**then have** *c-in3*:  $\forall k. k \geq nj \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow (C, \text{active}) \in \text{snd } (\text{lnth } Ns \ k)$  **by** *blast*

**have** *nj-pos*:  $nj > 0$  **using** *init-state c-in2 nj-is* **unfolding** *active-subset-def lhd-is* **by** *fastforce*

**obtain** *nj-min* **where** *nj-min-is*:  $nj\text{-min} = (\text{LEAST } nj. \text{enat } nj < \text{llength } Ns \wedge$

$(C, \text{active}) \in \bigcap (\text{snd } ' (\text{lnth } Ns \ ' \{k. nj \leq k \wedge \text{enat } k < \text{llength } Ns\})))$  **by** *blast*

**then have** *in-allk*:  $\forall k. k \geq nj\text{-min} \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow (C, \text{active}) \in \text{snd } (\text{lnth } Ns \ k)$

**using** *c-in3 nj-is c-in2 INT-E LeastI-ex*

[*of*  $\lambda n. \text{enat } n < \text{llength } Ns$

$\wedge (C, \text{active}) \in \bigcap (\text{snd } ' (\text{lnth } Ns \ ' \{na. n \leq na \wedge \text{enat } na < \text{llength } Ns\}))$ ]

**by** *blast*

**have** *njm-smaller-D*:  $\text{enat } nj\text{-min} < \text{llength } Ns$

**using** *nj-min-is*

**by** (*smt LeastI-ex <math>\wedgethesis. ( $\bigwedge nj. \llbracket \text{enat } nj < \text{llength } Ns;$*

$(C, \text{active}) \in \bigcap (\text{snd } ' (\text{lnth } Ns \ ' \{k. nj \leq k \wedge \text{enat } k < \text{llength } Ns\})) \rrbracket \implies \text{thesis}) \implies \text{thesis}$ )

**have**  $nj\text{-min} > 0$



**using** *nj-is c-in2 nj-pos nj-min-is lhd-is*  
**by** (*metis (mono-tags, lifting) active-subset-def emptyE in-alkk init-state mem-Collect-eq not-less snd-conv zero-enat-def chain-length-pos[OF deriv]*)  
**then obtain** *njm-prec* **where** *nj-prec-is*:  $Suc\ njm-prec = nj-min$  **using** *gr0-conv-Suc* **by** *auto*  
**then have** *njm-prec-njm*:  $njm-prec < nj-min$  **by** *blast*  
**then have** *njm-prec-njm-enat*:  $enat\ njm-prec < enat\ nj-min$  **by** *simp*  
**have** *njm-prec-smaller-d*:  $njm-prec < llength\ Ns$   
**by** (*rule less-trans[OF njm-prec-njm-enat njm-smaller-D]*)  
**have** *njm-prec-all-suc*:  $\forall k > njm-prec. enat\ k < llength\ Ns \longrightarrow (C, active) \in snd\ (lnth\ Ns\ k)$   
**using** *nj-prec-is in-alkk* **by** *simp*  
**have** *notin-njm-prec*:  $(C, active) \notin snd\ (lnth\ Ns\ njm-prec)$   
**proof** (*rule ccontr*)  
**assume**  $\neg (C, active) \notin snd\ (lnth\ Ns\ njm-prec)$   
**then have** *absurd-hyp*:  $(C, active) \in snd\ (lnth\ Ns\ njm-prec)$  **by** *simp*  
**have** *prec-smaller*:  $enat\ njm-prec < llength\ Ns$  **using** *nj-min-is nj-prec-is*  
**by** (*smt LeastI-ex Suc-leD ‹ $\wedge$ thesis. ( $\wedge nj. \llbracket enat\ nj < llength\ Ns; (C, active) \in \bigcap (snd\ ' (lnth\ Ns\ ' \{k. nj \leq k \wedge enat\ k < llength\ Ns\}) \rrbracket \implies thesis) \implies thesis$ ›*)  
*enat-ord-simps(1) le-eq-less-or-eq le-less-trans*)  
**have**  $(C, active) \in \bigcap (snd\ ' (lnth\ Ns\ ' \{k. njm-prec \leq k \wedge enat\ k < llength\ Ns\}))$   
**proof** –  
{  
  **fix** *k*  
  **assume** *k-in*:  $njm-prec \leq k \wedge enat\ k < llength\ Ns$   
  **have**  $k = njm-prec \implies (C, active) \in snd\ (lnth\ Ns\ k)$  **using** *absurd-hyp* **by** *simp*  
  **moreover have**  $njm-prec < k \implies (C, active) \in snd\ (lnth\ Ns\ k)$   
  **using** *nj-prec-is in-alkk k-in* **by** *simp*  
  **ultimately have**  $(C, active) \in snd\ (lnth\ Ns\ k)$  **using** *k-in* **by** *fastforce*  
}
**then show**  $(C, active) \in \bigcap (snd\ ' (lnth\ Ns\ ' \{k. njm-prec \leq k \wedge enat\ k < llength\ Ns\}))$   
**by** *blast*  
**qed**  
**then have**  $enat\ njm-prec < llength\ Ns \wedge (C, active) \in \bigcap (snd\ ' (lnth\ Ns\ ' \{k. njm-prec \leq k \wedge enat\ k < llength\ Ns\}))$   
**using** *prec-smaller* **by** *blast*  
**then show** *False*  
**using** *nj-min-is nj-prec-is Orderings.wellorder-class.not-less-Least njm-prec-njm* **by** *blast*  
**qed**  
**then have** *notin-active-subs-njm-prec*:  $(C, active) \notin active-subset\ (snd\ (lnth\ Ns\ njm-prec))$   
**unfolding** *active-subset-def* **by** *blast*  
**then show**  $\exists nj. enat\ (Suc\ nj) < llength\ Ns \wedge prems-of\ \iota ! j \notin active-subset\ (snd\ (lnth\ Ns\ nj)) \wedge (\forall k. k > nj \longrightarrow enat\ k < llength\ Ns \longrightarrow prems-of\ \iota ! j \in active-subset\ (snd\ (lnth\ Ns\ k)))$   
**using** *c-is njm-prec-all-suc njm-prec-smaller-d* **by** (*metis (mono-tags, lifting) active-subset-def mem-Collect-eq nj-prec-is njm-smaller-D snd-conv*)  
**qed**  
**define** *nj-set* **where**  $nj-set = \{nj. (\exists j \in \{0..<m\}. enat\ (Suc\ nj) < llength\ Ns \wedge prems-of\ \iota ! j \notin active-subset\ (snd\ (lnth\ Ns\ nj)) \wedge (\forall k. k > nj \longrightarrow enat\ k < llength\ Ns \longrightarrow prems-of\ \iota ! j \in active-subset\ (snd\ (lnth\ Ns\ k))))\}$   
{  
  **assume** *m-null*:  $m = 0$   
  **then have**  $enat\ 0 < llength\ Ns \wedge to-F\ \iota \in fst\ (lhd\ Ns)$   
  **using** *no-prems-init i-in-F m-def-F zero-enat-def chain-length-pos[OF deriv]* **by** *auto*  
  **then have**  $\exists n. enat\ n < llength\ Ns \wedge to-F\ \iota \in fst\ (lnth\ Ns\ n)$   
  **unfolding** *lhd-is* **by** *blast*  
}
**moreover** {

**assume**  $m\text{-pos}$ :  $m > 0$   
**have**  $nj\text{-not-empty}$ :  $nj\text{-set} \neq \{\}$   
**proof** –  
**have**  $zero\text{-in}$ :  $0 \in \{0..<m\}$  **using**  $m\text{-pos}$  **by** *simp*  
**then obtain**  $n0$  **where**  $enat (Suc n0) < llength\ Ns$  **and**  
 $prems\text{-of } \iota ! 0 \notin active\text{-subset} (snd (lnth\ Ns\ n0))$  **and**  
 $\forall k > n0. enat\ k < llength\ Ns \longrightarrow prems\text{-of } \iota ! 0 \in active\text{-subset} (snd (lnth\ Ns\ k))$   
**using**  $exist\text{-nj}$  **by** *fast*  
**then have**  $n0 \in nj\text{-set}$  **unfolding**  $nj\text{-set-def}$  **using**  $zero\text{-in}$  **by** *blast*  
**then show**  $nj\text{-set} \neq \{\}$  **by** *auto*  
**qed**  
**have**  $nj\text{-finite}$ : *finite nj-set*  
**using**  $all\text{-ex-finite-set}[OF\ exist\text{-nj}]$  **by** (*metis (no-types, lifting) Suc-ile-eq dual-order.strict-implies-order linorder-neqE-nat nj-set-def*)  
**have**  $\exists n \in nj\text{-set}. \forall nj \in nj\text{-set}. nj \leq n$   
**using**  $nj\text{-not-empty } nj\text{-finite}$  **using**  $Max\text{-ge } Max\text{-in}$  **by** *blast*  
**then obtain**  $n$  **where**  $n\text{-in}$ :  $n \in nj\text{-set}$  **and**  $n\text{-bigger}$ :  $\forall nj \in nj\text{-set}. nj \leq n$  **by** *blast*  
**then obtain**  $j0$  **where**  $j0\text{-in}$ :  $j0 \in \{0..<m\}$  **and**  $suc\text{-n-length}$ :  $enat (Suc\ n) < llength\ Ns$  **and**  
 $j0\text{-notin}$ :  $prems\text{-of } \iota ! j0 \notin active\text{-subset} (snd (lnth\ Ns\ n))$  **and**  
 $j0\text{-allin}$ :  $(\forall k. k > n \longrightarrow enat\ k < llength\ Ns \longrightarrow$   
 $prems\text{-of } \iota ! j0 \in active\text{-subset} (snd (lnth\ Ns\ k)))$   
**unfolding**  $nj\text{-set-def}$  **by** *blast*  
**obtain**  $C0$  **where**  $C0\text{-is}$ :  $prems\text{-of } \iota ! j0 = (C0, active)$   
**using**  $j0\text{-in } i\text{-in}2$  **unfolding**  $m\text{-def } Inf\text{-from-def } active\text{-subset-def}$   
**by** (*smt Collect-mem-eq Collect-mono-iff atLeastLessThan-iff nth-mem old.prod.exhaust snd-conv*)  
**then have**  $C0\text{-prems-i}$ :  $(C0, active) \in set (prems\text{-of } \iota)$  **using**  $in\text{-set-conv-nth } j0\text{-in } m\text{-def}$  **by** *force*  
**have**  $C0\text{-in}$ :  $(C0, active) \in (snd (lnth\ Ns (Suc\ n)))$   
**using**  $C0\text{-is } j0\text{-allin } suc\text{-n-length}$  **by** (*simp add: active-subset-def*)  
**have**  $C0\text{-notin}$ :  $(C0, active) \notin (snd (lnth\ Ns\ n))$   
**using**  $C0\text{-is } j0\text{-notin}$  **unfolding**  $active\text{-subset-def}$  **by** *simp*  
**have**  $step\text{-n}$ :  $lnth\ Ns\ n \sim LGC\ lnth\ Ns (Suc\ n)$   
**using**  $deriv\ chain\text{-lnth-rel } n\text{-in}$  **unfolding**  $nj\text{-set-def}$  **by** *blast*  
**have**  $is\text{-scheduled}$ :  $\exists T2\ T1\ T'\ N1\ N\ C\ L\ N2. lnth\ Ns\ n = (T1, N1) \wedge lnth\ Ns (Suc\ n) = (T2, N2)$   
 $\wedge$   
 $T2 = T1 \cup T' \wedge N1 = N \cup \{(C, L)\} \wedge N2 = N \cup \{(C, active)\} \wedge L \neq active \wedge$   
 $T' = no\text{-labels}.Inf\text{-between} (fst\ 'active\text{-subset } N) \{C\}$   
**using**  $step.simps[of\ lnth\ Ns\ n\ lnth\ Ns (Suc\ n)]$   $step\text{-n } C0\text{-in } C0\text{-notin}$   
**unfolding**  $active\text{-subset-def}$  **by** *fastforce*  
**then obtain**  $T2\ T1\ T'\ N1\ N\ L\ N2$  **where**  $nth\text{-d-is}$ :  $lnth\ Ns\ n = (T1, N1)$  **and**  
 $suc\text{-nth-d-is}$ :  $lnth\ Ns (Suc\ n) = (T2, N2)$  **and**  $t2\text{-is}$ :  $T2 = T1 \cup T'$  **and**  
 $n1\text{-is}$ :  $N1 = N \cup \{(C0, L)\}$   $N2 = N \cup \{(C0, active)\}$  **and**  
 $l\text{-not-active}$ :  $L \neq active$  **and**  
 $tp\text{-is}$ :  $T' = no\text{-labels}.Inf\text{-between} (fst\ 'active\text{-subset } N) \{C0\}$   
**using**  $C0\text{-in } C0\text{-notin } j0\text{-in } C0\text{-is}$  **using**  $active\text{-subset-def}$  **by** *fastforce*  
**have**  $j \in \{0..<m\} \implies prems\text{-of } \iota ! j \neq prems\text{-of } \iota ! j0 \implies prems\text{-of } \iota ! j \in (active\text{-subset } N)$   
**for**  $j$   
**proof** –  
**fix**  $j$   
**assume**  $j\text{-in}$ :  $j \in \{0..<m\}$  **and**  
 $j\text{-not-j0}$ :  $prems\text{-of } \iota ! j \neq prems\text{-of } \iota ! j0$   
**obtain**  $nj$  **where**  $nj\text{-len}$ :  $enat (Suc\ nj) < llength\ Ns$  **and**  
 $nj\text{-prems}$ :  $prems\text{-of } \iota ! j \notin active\text{-subset} (snd (lnth\ Ns\ nj))$  **and**  
 $nj\text{-greater}$ :  $(\forall k. k > nj \longrightarrow enat\ k < llength\ Ns \longrightarrow$   
 $prems\text{-of } \iota ! j \in active\text{-subset} (snd (lnth\ Ns\ k)))$   
**using**  $exist\text{-nj } j\text{-in}$  **by** *blast*

**then have**  $nj \in nj\text{-set}$  **unfolding**  $nj\text{-set-def}$  **using**  $j\text{-in}$  **by**  $blast$   
**moreover have**  $nj \neq n$   
**proof** (*rule ccontr*)  
**assume**  $\neg nj \neq n$   
**then have**  $prems\text{-of } \iota ! j = (C0, active)$   
**using**  $C0\text{-in } C0\text{-notin } step.simps[of\ lnth\ Ns\ n\ lnth\ Ns\ (Suc\ n)]\ step\text{-n}$   
 $active\text{-subset-def } is\text{-scheduled } nj\text{-greater } nj\text{-prems } suc\text{-n-length}$  **by**  $auto$   
**then show**  $False$  **using**  $j\text{-not-}j0\ C0\text{-is}$  **by**  $simp$   
**qed**  
**ultimately have**  $nj < n$  **using**  $n\text{-bigger}$  **by**  $force$   
**then have**  $prems\text{-of } \iota ! j \in (active\text{-subset } (snd\ (lnth\ Ns\ n)))$   
**using**  $nj\text{-greater } n\text{-in } Suc\text{-ile-eq } dual\text{-order.strict-implies-order}$   
**unfolding**  $nj\text{-set-def}$  **by**  $blast$   
**then show**  $prems\text{-of } \iota ! j \in (active\text{-subset } N)$   
**using**  $nth\text{-d-is } l\text{-not-active } n1\text{-is}$  **unfolding**  $active\text{-subset-def}$  **by**  $force$   
**qed**  
**then have**  $prems\text{-i-active: set } (prems\text{-of } \iota) \subseteq active\text{-subset } N \cup \{(C0, active)\}$   
**using**  $C0\text{-prems-i } C0\text{-is } m\text{-def}$   
**by** (*metis Un-iff atLeast0LessThan in-set-conv-nth insertCI lessThan-iff subrelI*)  
**moreover have**  $\neg (set\ (prems\text{-of } \iota) \subseteq active\text{-subset } N - \{(C0, active)\})$  **using**  $C0\text{-prems-i}$  **by**  $blast$   
**ultimately have**  $\iota \in Inf\text{-between } (active\text{-subset } N) \{(C0, active)\}$   
**using**  $i\text{-in-inf-fl } prems\text{-i-active}$  **unfolding**  $Inf\text{-between-def } Inf\text{-from-def}$  **by**  $blast$   
**then have**  $to\text{-F } \iota \in no\text{-labels.Inf-between } (fst\ 'active\text{-subset } N) \{C0\}$   
**unfolding**  $to\text{-F-def } Inf\text{-between-def } Inf\text{-from-def}$   
 $no\text{-labels.Inf-between-def } no\text{-labels.Inf-from-def}$   
**using**  $Inf\text{-FL-to-Inf-F}$  **by**  $force$   
**then have**  $i\text{-in-t2: } to\text{-F } \iota \in T2$  **using**  $tp\text{-is } t2\text{-is}$  **by**  $simp$   
**have**  $j \in \{0..<m\} \implies (\forall k. k > n \implies enat\ k < llength\ Ns \implies$   
 $prems\text{-of } \iota ! j \in active\text{-subset } (snd\ (lnth\ Ns\ k)))$  **for**  $j$   
**proof** (*cases j = j0*)  
**case**  $True$   
**assume**  $j = j0$   
**then show**  $(\forall k. k > n \implies enat\ k < llength\ Ns \implies$   
 $prems\text{-of } \iota ! j \in active\text{-subset } (snd\ (lnth\ Ns\ k)))$  **using**  $j0\text{-allin}$  **by**  $simp$   
**next**  
**case**  $False$   
**assume**  $j\text{-in: } j \in \{0..<m\}$  **and**  
 $j \neq j0$   
**obtain**  $nj$  **where**  $nj\text{-len: } enat\ (Suc\ nj) < llength\ Ns$  **and**  
 $nj\text{-prems: } prems\text{-of } \iota ! j \notin active\text{-subset } (snd\ (lnth\ Ns\ nj))$  **and**  
 $nj\text{-greater: } (\forall k. k > nj \implies enat\ k < llength\ Ns \implies$   
 $prems\text{-of } \iota ! j \in active\text{-subset } (snd\ (lnth\ Ns\ k)))$   
**using**  $exist\text{-nj } j\text{-in}$  **by**  $blast$   
**then have**  $nj \in nj\text{-set}$  **unfolding**  $nj\text{-set-def}$  **using**  $j\text{-in}$  **by**  $blast$   
**then show**  $(\forall k. k > n \implies enat\ k < llength\ Ns \implies$   
 $prems\text{-of } \iota ! j \in active\text{-subset } (snd\ (lnth\ Ns\ k)))$   
**using**  $nj\text{-greater } n\text{-bigger}$  **by**  $auto$   
**qed**  
**then have**  $allj\text{-allk: } (\forall c \in set\ (prems\text{-of } \iota). (\forall k. k > n \implies enat\ k < llength\ Ns \implies$   
 $c \in active\text{-subset } (snd\ (lnth\ Ns\ k))))$   
**using**  $m\text{-def}$  **by** (*metis atLeast0LessThan in-set-conv-nth lessThan-iff*)  
**have**  $\forall c \in set\ (prems\text{-of } \iota). snd\ c = active$   
**using**  $prems\text{-i-active}$  **unfolding**  $active\text{-subset-def}$  **by**  $auto$   
**then have**  $ex\text{-n-i-in: } \exists n. enat\ (Suc\ n) < llength\ Ns \wedge to\text{-F } \iota \in fst\ (lnth\ Ns\ (Suc\ n)) \wedge$   
 $(\forall c \in set\ (prems\text{-of } \iota). snd\ c = active) \wedge$

$(\forall c \in \text{set } (\text{prems-of } \iota). (\forall k. k > n \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow$   
 $c \in \text{active-subset } (\text{snd } (\text{lnth } Ns \ k))))$   
**using** *allj-allk i-in-t2 suc-nth-d-is fstI n-in nj-set-def*  
**by** *auto*  
**then have**  $\exists n. \text{enat } n < \text{llength } Ns \wedge \text{to-F } \iota \in \text{fst } (\text{lnth } Ns \ n) \wedge$   
 $(\forall c \in \text{set } (\text{prems-of } \iota). \text{snd } c = \text{active}) \wedge (\forall c \in \text{set } (\text{prems-of } \iota). (\forall k. k \geq n \longrightarrow$   
 $\text{enat } k < \text{llength } Ns \longrightarrow c \in \text{active-subset } (\text{snd } (\text{lnth } Ns \ k))))$   
**by** *auto*  
**}**  
**ultimately obtain**  $n \ T2 \ N2$  **where** *i-in-suc-n: to-F  $\iota \in \text{fst } (\text{lnth } Ns \ n)$  and*  
*all-prems-active-after:  $m > 0 \implies (\forall c \in \text{set } (\text{prems-of } \iota). (\forall k. k \geq n \longrightarrow \text{enat } k < \text{llength } Ns \longrightarrow$*   
 $c \in \text{active-subset } (\text{snd } (\text{lnth } Ns \ k))))$  **and**  
*suc-n-length:  $\text{enat } n < \text{llength } Ns$  and suc-nth-d-is:  $\text{lnth } Ns \ n = (T2, N2)$*   
**by** *(metis less-antisym old.prod.exhaust zero-less-Suc)*  
**then have** *i-in-t2: to-F  $\iota \in T2$  by simp*  
**have**  $\exists p \geq n. \text{enat } (\text{Suc } p) < \text{llength } Ns \wedge \text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ p)) \wedge \text{to-F } \iota \notin (\text{fst } (\text{lnth } Ns \ (\text{Suc } p)))$   
**proof** *(rule ccontr)*  
**assume**  
 $\text{contra: } \neg (\exists p \geq n. \text{enat } (\text{Suc } p) < \text{llength } Ns \wedge \text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ p)) \wedge$   
 $\text{to-F } \iota \notin (\text{fst } (\text{lnth } Ns \ (\text{Suc } p))))$   
**then have** *i-in-suc:  $p0 \geq n \implies \text{enat } (\text{Suc } p0) < \text{llength } Ns \implies \text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ p0)) \implies$*   
 $\text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ (\text{Suc } p0)))$  **for**  $p0$   
**by** *blast*  
**have**  $p0 \geq n \implies \text{enat } p0 < \text{llength } Ns \implies \text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ p0))$  **for**  $p0$   
**proof** *(induction rule: nat-induct-at-least)*  
**case** *base*  
**then show** *?case using i-in-t2 suc-nth-d-is*  
**by** *simp*  
**next**  
**case**  $(\text{Suc } p0)$   
**assume** *p-bigger-n:  $n \leq p0$  and*  
*induct-hyp:  $\text{enat } p0 < \text{llength } Ns \implies \text{to-F } \iota \in \text{fst } (\text{lnth } Ns \ p0)$  and*  
*sucsuc-smaller-d:  $\text{enat } (\text{Suc } p0) < \text{llength } Ns$*   
**have** *suc-p-bigger-n:  $n \leq p0$  using p-bigger-n by simp*  
**have** *suc-smaller-d:  $\text{enat } p0 < \text{llength } Ns$*   
**using** *sucsuc-smaller-d Suc-ile-eq dual-order.strict-implies-order by blast*  
**then have**  $\text{to-F } \iota \in \text{fst } (\text{lnth } Ns \ p0)$  **using** *induct-hyp by blast*  
**then show** *?case using i-in-suc[OF suc-p-bigger-n sucsuc-smaller-d] by blast*  
**qed**  
**then have** *i-in-all-bigger-n:  $\forall j. j \geq n \wedge \text{enat } j < \text{llength } Ns \longrightarrow \text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ j))$*   
**by** *presburger*  
**have**  $\text{llength } (\text{lmap } \text{fst } Ns) = \text{llength } Ns$  **by** *force*  
**then have**  $\text{to-F } \iota \in \bigcap (\text{lnth } (\text{lmap } \text{fst } Ns) \ \{j. n \leq j \wedge \text{enat } j < \text{llength } (\text{lmap } \text{fst } Ns)\})$   
**using** *i-in-all-bigger-n using Suc-le-D by auto*  
**then have**  $\text{to-F } \iota \in \text{Liminf-llist } (\text{lmap } \text{fst } Ns)$   
**unfolding** *Liminf-llist-def using suc-n-length by auto*  
**then show** *False using final-schedule by fast*  
**qed**  
**then obtain**  $p$  **where** *p-greater-n:  $p \geq n$  and p-smaller-d:  $\text{enat } (\text{Suc } p) < \text{llength } Ns$  and*  
*i-in-p:  $\text{to-F } \iota \in (\text{fst } (\text{lnth } Ns \ p))$  and i-notin-suc-p:  $\text{to-F } \iota \notin (\text{fst } (\text{lnth } Ns \ (\text{Suc } p)))$*   
**by** *blast*  
**have** *p-neq-n:  $\text{Suc } p \neq n$  using i-notin-suc-p i-in-suc-n by blast*  
**have** *step-p:  $\text{lnth } Ns \ p \rightsquigarrow \text{LGC } \text{lnth } Ns \ (\text{Suc } p)$  using deriv p-smaller-d chain-lnth-rel by blast*  
**then have**  $\exists T1 \ T2 \ \iota \ N2 \ N1 \ M. \text{lnth } Ns \ p = (T1, N1) \wedge \text{lnth } Ns \ (\text{Suc } p) = (T2, N2) \wedge$   
 $T1 = T2 \cup \{\iota\} \wedge N2 = N1 \cup M \wedge \text{active-subset } M = \{\}$  **and**

$\iota \in \text{no-labels.Red-I-G } (\text{fst } ' (N1 \cup M))$   
**proof** –  
**have** *ci-or-do*:  $(\exists T1 T2 \iota N2 N1 M. \text{lnth } Ns p = (T1, N1) \wedge \text{lnth } Ns (Suc p) = (T2, N2) \wedge$   
 $T1 = T2 \cup \{\iota\} \wedge N2 = N1 \cup M \wedge \text{active-subset } M = \{\} \wedge$   
 $\iota \in \text{no-labels.Red-I-G } (\text{fst } ' (N1 \cup M))) \vee$   
 $(\exists T1 T2 T' N. \text{lnth } Ns p = (T1, N) \wedge \text{lnth } Ns (Suc p) = (T2, N) \wedge$   
 $T1 = T2 \cup T' \wedge T' \cap \text{no-labels.Inf-from } (\text{fst } ' \text{active-subset } N) = \{\})$   
**using** *step.simps*[of *lnth Ns p lnth Ns (Suc p)*] *step-p i-in-p i-notin-suc-p* **by** *fastforce*  
**then have** *p-greater-n-strict*:  $n < Suc p$   
**using** *suc-nth-d-is p-greater-n i-in-t2 i-notin-suc-p le-eq-less-or-eq* **by** *force*  
**have**  $m > 0 \implies j \in \{0..<m\} \implies \text{prems-of } (to-F \iota) ! j \in \text{fst } ' \text{active-subset } (\text{snd } (\text{lnth } Ns p))$   
**for**  $j$   
**proof** –  
**fix**  $j$   
**assume**  
 $m\text{-pos}: m > 0$  **and**  
 $j\text{-in}: j \in \{0..<m\}$   
**then have** *prems-of*  $\iota ! j \in (\text{active-subset } (\text{snd } (\text{lnth } Ns p)))$   
**using** *all-prems-active-after*[*OF m-pos*] *p-smaller-d m-def p-greater-n p-neq-n*  
**by** (*meson Suc-ile-eq atLeastLessThan-iff dual-order.strict-implies-order nth-mem*  
*p-greater-n-strict*)  
**then have** *fst* (*prems-of*  $\iota ! j$ )  $\in \text{fst } ' \text{active-subset } (\text{snd } (\text{lnth } Ns p))$   
**by** *blast*  
**then show** *prems-of*  $(to-F \iota) ! j \in \text{fst } ' \text{active-subset } (\text{snd } (\text{lnth } Ns p))$   
**unfolding** *to-F-def* **using** *j-in m-def* **by** *simp*  
**qed**  
**then have** *prems-i-active-p*:  $m > 0 \implies$   
 $to-F \iota \in \text{no-labels.Inf-from } (\text{fst } ' \text{active-subset } (\text{snd } (\text{lnth } Ns p)))$   
**using** *i-in-F unfolding no-labels.Inf-from-def*  
**by** (*smt atLeast0LessThan in-set-conv-nth lessThan-iff m-def-F mem-Collect-eq subsetI*)  
**have**  $m = 0 \implies (\exists T1 T2 \iota N2 N1 M. \text{lnth } Ns p = (T1, N1) \wedge \text{lnth } Ns (Suc p) = (T2, N2) \wedge$   
 $T1 = T2 \cup \{\iota\} \wedge N2 = N1 \cup M \wedge \text{active-subset } M = \{\} \wedge$   
 $\iota \in \text{no-labels.Red-I-G } (\text{fst } ' (N1 \cup M)))$   
**using** *ci-or-do premise-free-inf-always-from*[of *to-F \iota fst ' active-subset -, OF i-in-F*]  
*m-def i-in-p i-notin-suc-p m-def-F* **by** *auto*  
**then show**  $(\exists T1 T2 \iota N2 N1 M. \text{lnth } Ns p = (T1, N1) \wedge \text{lnth } Ns (Suc p) = (T2, N2) \wedge$   
 $T1 = T2 \cup \{\iota\} \wedge N2 = N1 \cup M \wedge \text{active-subset } M = \{\} \wedge$   
 $\iota \in \text{no-labels.Red-I-G } (\text{fst } ' (N1 \cup M)))$   
**using** *ci-or-do i-in-p i-notin-suc-p prems-i-active-p unfolding active-subset-def* **by** *force*  
**qed**  
**then obtain**  $T1p T2p N1p N2p Mp$  **where**  $\text{lnth } Ns p = (T1p, N1p)$  **and**  
 $\text{suc-p-is}: \text{lnth } Ns (Suc p) = (T2p, N2p)$  **and**  $T1p = T2p \cup \{to-F \iota\}$  **and**  $T2p \cap \{to-F \iota\} = \{\}$  **and**  
 $n2p\text{-is}: N2p = N1p \cup Mp$  **and**  $\text{active-subset } Mp = \{\}$  **and**  
 $i\text{-in-red-inf}: to-F \iota \in \text{no-labels.Red-I-G } (\text{fst } ' (N1p \cup Mp))$   
**using** *i-in-p i-notin-suc-p* **by** *fastforce*  
**have**  $to-F \iota \in \text{no-labels.Red-I } (\text{fst } ' (\text{snd } (\text{lnth } Ns (Suc p))))$   
**using** *i-in-red-inf suc-p-is n2p-is* **by** *fastforce*  
**then have**  $\forall q \in Q. (\mathcal{G}\text{-I-q } q (to-F \iota) \neq \text{None} \wedge$   
 $\text{the } (\mathcal{G}\text{-I-q } q (to-F \iota)) \subseteq \text{Red-I-q } q (\bigcup (\mathcal{G}\text{-F-q } q ' \text{fst } ' \text{snd } (\text{lnth } Ns (Suc p))))))$   
 $\vee (\mathcal{G}\text{-I-q } q (to-F \iota) = \text{None} \wedge$   
 $\mathcal{G}\text{-F-q } q (\text{concl-of } (to-F \iota)) \subseteq \bigcup (\mathcal{G}\text{-F-q } q ' \text{fst } ' \text{snd } (\text{lnth } Ns (Suc p))) \cup$   
 $\text{Red-F-q } q (\bigcup (\mathcal{G}\text{-F-q } q ' \text{fst } ' \text{snd } (\text{lnth } Ns (Suc p))))))$   
**unfolding** *to-F-def no-labels.Red-I-def no-labels.Red-I-G-q-def* **by** *blast*  
**then have**  $\iota \in \text{Red-I-G } (\text{snd } (\text{lnth } Ns (Suc p)))$

**using** *i-in-inf-fl unfolding Red-I-G-def Red-I-G-q-def* **by** (*simp add: to-F-def*)  
**then show**  $\iota \in \text{Sup-llist} (\text{lmap Red-I-G} (\text{lmap snd Ns}))$   
**unfolding** *Sup-llist-def* **using** *suc-n-length p-smaller-d* **by** *auto*  
**qed**

**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-G* (*fst* ' *snd* (*lhd* *Ns*)) {*B*}

**shows**  $\exists BL \in \text{Bot-FL}. BL \in \text{Liminf-llist} (\text{lmap snd } Ns)$

**proof** –

**have** *labeled-b-in*:  $(B, \text{active}) \in \text{Bot-FL}$  **using** *b-in* **by** *simp*

**have** *simp-snd-lmap*: *lhd* (*lmap snd Ns*) = *snd* (*lhd Ns*)

**by** (*rule* *llist.map-sel(1)[OF chain-not-lnull[OF deriv]]*)

**have** *labeled-bot-entailed*: *entails-G-L* (*snd* (*lhd* *Ns*)) {(*B*, *active*)}

**using** *labeled-entailment-lifting bot-entailed* **by** *fastforce*

**have** *fair* (*lmap snd Ns*)

**using** *lgc-fair[OF deriv init-state final-state no-prems-init final-schedule]* .

**then show** *?thesis*

**using** *dynamically-complete-Liminf labeled-b-in lgc-to-red[OF deriv]*

*labeled-bot-entailed simp-snd-lmap std-Red-I-eq*

**by** *presburger*

**qed**

**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-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** –

**have**  $\exists BL \in \text{Bot-FL}. BL \in \text{Liminf-llist} (\text{lmap snd } Ns)$

**using** *assms* **by** (*rule* *lgc-complete-Liminf*)

**then show** *?thesis*

**unfolding** *Liminf-llist-def* **by** *auto*

**qed**

**end**

**end**