

A Comprehensive Framework for Saturation Theorem Proving

Sophie Touret

December 14, 2021

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. The names of the Isabelle lemmas and theorems corresponding to the results in the report are indicated in the margin of the report.

Contents

1	Calculi Based on a Redundancy Criterion	1
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
2	Calculi Based on the Intersection of Redundancy Criteria	6
2.1	Calculi with a Family of Redundancy Criteria	6
3	Variations on a Theme	7
4	Lifting to Non-ground Calculi	10
4.1	Standard Lifting	10
4.2	Strong Standard Lifting	11
4.3	Lifting with a Family of Tiebreaker Orderings	12
4.4	Lifting with a Family of Redundancy Criteria	14
5	Labeled Lifting to Non-Ground Calculi	17
5.1	Labeled Lifting with a Family of Tiebreaker Orderings	17
5.2	Labeled Lifting with a Family of Redundancy Criteria	18

6	Given Clause Prover Architectures	20
6.1	Basis of the Given Clause Prover Architectures	21
6.2	Given Clause Procedure	22
6.3	Lazy Given Clause Procedure	24

1 Calculi Based on a Redundancy Criterion

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

theory *Calculus*

imports

Ordered-Resolution-Prover.Lazy-List-Liminf

Ordered-Resolution-Prover.Lazy-List-Chain

begin

1.1 Consequence Relations

locale *consequence-relation* =

fixes

Bot :: 'f set **and**

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

assumes

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

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

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

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

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

begin

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

<proof>

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

<proof>

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

<proof>

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

<proof>

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

<proof>

end

1.2 Families of Consequence Relations

locale *consequence-relation-family* =

fixes

Bot :: 'f set **and**

Q :: 'q set **and**

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

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

lemma *bot-not-empty*: $\text{Bot} \neq \{\}$
 $\langle \text{proof} \rangle$

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

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

end

1.3 Inference Systems

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

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

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

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

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

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

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

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

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

end

1.4 Families of Inference Systems

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

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

1.5 Calculi Based on a Single Redundancy Criterion

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

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

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

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

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

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

⟨proof⟩

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

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

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

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

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

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

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

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

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

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

```

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

end

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

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

end

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

sublocale statically-complete-calculus
  ⟨proof⟩

end

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

end

```

2 Calculi Based on the Intersection of Redundancy Criteria

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

```

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

```

2.1 Calculi with a Family of Redundancy Criteria

```

locale intersection-calculus =

```

```

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

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

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

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

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

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

end

end

```

3 Variations on a Theme

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

```

theory Calculus-Variations
imports Calculus
begin

```

```

locale reduced-calculus = calculus Bot Inf entails Red-I Red-F
for
  Bot :: 'f set and
  Inf :: ⟨'f inference set⟩ and
  entails :: 'f set ⇒ 'f set ⇒ bool (infix |= 50) and
  Red-I :: 'f set ⇒ 'f inference set and

```

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

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

end

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

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

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

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

end

context *reduced-calculus*
begin

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

end

context *calculus*
begin

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

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

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

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

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

<proof>

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

<proof>

lemma *sat-imp-red-calc-sat: saturated N \implies reduc-calc.saturated N*

<proof>

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

<proof>

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

<proof>

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

<proof>

theorem *red-stat-red-is-stat-red:*

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

<proof>

theorem *red-stat-is-stat-red:*

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

<proof>

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

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

<proof>

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

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

<proof>

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

reduc-fair Ns \longleftrightarrow

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

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

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

<proof>

end

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

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

begin

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

end

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

context *calculus*

begin

lemma *dyn-equiv-stat: dynamically-complete-calculus Bot Inf entails Red-I Red-F =*
statically-complete-calculus Bot Inf entails Red-I Red-F
 $\langle proof \rangle$

lemma *red-dyn-equiv-red-stat:*
reducedly-dynamically-complete-calculus Bot Inf entails Red-I Red-F =
reducedly-statically-complete-calculus Bot Inf entails Red-I Red-F
 $\langle proof \rangle$

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

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

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

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

end

end

4 Lifting to Non-ground Calculi

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

theory *Lifting-to-Non-Ground-Calculi*

imports

Intersection-Calculus

Calculus-Variations

Well-Quasi-Orders.Minimal-Elements

begin

4.1 Standard Lifting

locale *standard-lifting* = *inference-system* *Inf-F* +
ground: calculus *Bot-G* *Inf-G* *entails-G* *Red-I-G* *Red-F-G*

for

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

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

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

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

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

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

+ **fixes**

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

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

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

assumes

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

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

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

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

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

begin

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

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

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

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

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

lemma *subs-Bot-G-entails*:

assumes

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

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

shows $\langle sB \models_G N \rangle$

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

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

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

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

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

end

4.2 Strong Standard Lifting

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

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

end

4.3 Lifting with a Family of Tiebreaker Orderings

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

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

lemma *Prec-trans*:

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

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

shows

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

assumes

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

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

shows

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

end

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

$\langle \text{proof} \rangle$

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

context *standard-lifting*
begin

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

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

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

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

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

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

assumes

saturated N **and**

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

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

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

⟨*proof*⟩

lemma *sat-imp-ground-sat*:

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

⟨*proof*⟩

theorem *stat-ref-comp-to-non-ground*:

assumes

stat-ref-G: *statically-complete-calculus Bot-G Inf-G entails-G Red-I-G Red-F-G* **and**

sat-n-imp: $\bigwedge N. \text{saturated } N \implies \text{ground-Inf-overapproximated } N$

shows

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

⟨*proof*⟩

end

context *tiebreaker-lifting*
begin

lemma *saturated-empty-order-equiv-saturated*:
saturated $N = \text{calc.}$ *saturated* N
 ⟨*proof*⟩

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

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

end

4.4 Lifting with a Family of Redundancy Criteria

locale *lifting-intersection = inference-system* $\text{Inf-F} +$
ground: inference-system-family $Q \text{ Inf-G-q} +$
ground: consequence-relation-family $\text{Bot-G } Q \text{ entails-q}$
for

$\text{Inf-F} :: 'f \text{ inference set and}$
 $\text{Bot-G} :: 'g \text{ set and}$
 $Q :: 'q \text{ set and}$
 $\text{Inf-G-q} :: \langle 'q \Rightarrow 'g \text{ inference set} \rangle \text{ and}$
 $\text{entails-q} :: 'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool and}$
 $\text{Red-I-q} :: 'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ inference set and}$
 $\text{Red-F-q} :: 'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ set}$

+ **fixes**

$\text{Bot-F} :: 'f \text{ set and}$
 $\mathcal{G}\text{-F-q} :: 'q \Rightarrow 'f \Rightarrow 'g \text{ set and}$
 $\mathcal{G}\text{-I-q} :: 'q \Rightarrow 'f \text{ inference} \Rightarrow 'g \text{ inference set option and}$
 $\text{Prec-F-g} :: 'g \Rightarrow 'f \Rightarrow 'f \Rightarrow \text{bool}$

assumes

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

begin

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

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

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

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

$E)\}$

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

lemma *red-crit-lifting-family*:

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

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

$\langle \text{proof} \rangle$

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

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

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

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

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

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

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

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

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

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

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

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

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

$\langle \text{proof} \rangle$

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

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

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

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

assumes

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

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

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

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

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

$\langle \text{proof} \rangle$

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

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

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

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

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

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

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

assumes

stat-ref-G:

∀ q ∈ Q. *statically-complete-calculus* *Bot-G* (*Inf-G-q* q) (*entails-q* q) (*Red-I-q* q)
 (*Red-F-q* q) **and**

sat-n-imp: ∧ N. *saturated* N ⇒ ∃ q ∈ Q. *ground-Inf-overapproximated* q N

shows

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

⟨*proof*⟩

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

lemma *static-empty-ord-inter-equiv-static-inter*:

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

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

⟨*proof*⟩

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

dynamically-complete-calculus *Bot-F* *Inf-F* *entails* *Red-I* *Red-F*

⟨*proof*⟩

end

end

5 Labeled Lifting to Non-Ground Calculi

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

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

imports *Lifting-to-Non-Ground-Calculi*

begin

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

⟨*proof*⟩

5.1 Labeled Lifting with a Family of Tiebreaker Orderings

locale *labeled-tiebreaker-lifting* = *no-labels*: *tiebreaker-lifting* *Bot-F* *Inf-F*

Bot-G *entails-G* *Inf-G* *Red-I-G* *Red-F-G* *G-F* *G-I* *Prec-F*

for

$Bot-F :: 'f \text{ set and}$
 $Inf-F :: 'f \text{ inference set and}$
 $Bot-G :: 'g \text{ set and}$
 $entails-G :: 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool (infix } \models_G 50) \text{ and}$
 $Inf-G :: 'g \text{ inference set and}$
 $Red-I-G :: 'g \text{ set} \Rightarrow 'g \text{ inference set and}$
 $Red-F-G :: 'g \text{ set} \Rightarrow 'g \text{ set and}$
 $\mathcal{G}-F :: 'f \Rightarrow 'g \text{ set and}$
 $\mathcal{G}-I :: 'f \text{ inference} \Rightarrow 'g \text{ inference set option and}$
 $Prec-F :: 'g \Rightarrow 'f \Rightarrow 'f \Rightarrow \text{bool (infix } \sqsubset 50)$

+ fixes

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

assumes

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

begin

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

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

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

$\langle Bot-FL \equiv Bot-F \times UNIV \rangle$

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

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

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

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

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

notation $entails-G \ (\text{infix } \models_G 50)$

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

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

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

lemma $\text{stat-ref-comp-to-labeled-sta-ref-comp}:$

assumes $\text{static}:$

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

shows $\text{statically-complete-calculus } Bot-FL \ Inf-FL \ (\models_G) \ Red-I-G \ Red-F-G$

<proof>

end

5.2 Labeled Lifting with a Family of Redundancy Criteria

locale *labeled-lifting-intersection = no-labels: lifting-intersection* *Inf-F*

Bot-G Q Inf-G-q entails-q Red-I-q Red-F-q Bot-F G-F-q G-I-q λg Cl Cl'. False

for

Bot-F :: 'f set **and**

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

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

Q :: 'q set **and**

entails-q :: 'q ⇒ 'g set ⇒ 'g set ⇒ bool **and**

Inf-G-q :: 'q ⇒ 'g inference set **and**

Red-I-q :: 'q ⇒ 'g set ⇒ 'g inference set **and**

Red-F-q :: 'q ⇒ 'g set ⇒ 'g set **and**

G-F-q :: 'q ⇒ 'f ⇒ 'g set **and**

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

+ fixes

Inf-FL :: ⟨('f × 'l) inference set⟩

assumes

Inf-F-to-Inf-FL:

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

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

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

begin

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *lifting-q*:

assumes $q \in Q$

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

lemma *lifted-q*:

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

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

$\langle proof \rangle$

lemma *ord-fam-lifted-q*:

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

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

$\langle proof \rangle$

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

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

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

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

lemma *all-lifted-red-crit*:

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

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

$\langle proof \rangle$

lemma *all-lifted-cons-rel*:

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

shows *consequence-relation* $Bot-FL$ ($entails-\mathcal{G}-L-q$ q)

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

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

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

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

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

$\langle proof \rangle$

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

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

theorem *labeled-static-ref*:

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

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

⟨proof⟩

end

end

6 Given Clause Prover Architectures

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

theory *Given-Clause-Architectures*

imports

Lambda-Free-RPOs.Lambda-Free-Util

Labeled-Lifting-to-Non-Ground-Calculi

begin

6.1 Basis of the Given Clause Prover Architectures

locale *given-clause-basis* = *std?*: *labeled-lifting-intersection Bot-F Inf-F Bot-G Q*
entails-q Inf-G-q Red-I-q Red-F-q G-F-q G-I-q Inf-FL

for

Bot-F :: 'f set

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

and *Bot-G* :: 'g set

and *Q* :: 'q set

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

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

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

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

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

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

and *Inf-FL* :: 'f \times 'l inference set

+ fixes

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

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

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

active :: 'l

assumes

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

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

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

compat-equiv-prec: $C1 \doteq D1 \implies C2 \doteq D2 \implies C1 \prec \cdot C2 \implies D1 \prec \cdot D2$ **and**
equiv-F-grounding: $q \in Q \implies C1 \doteq C2 \implies \mathcal{G}\text{-F-q } q \ C1 \subseteq \mathcal{G}\text{-F-q } q \ C2$ **and**
prec-F-grounding: $q \in Q \implies C2 \prec \cdot C1 \implies \mathcal{G}\text{-F-q } q \ C1 \subseteq \mathcal{G}\text{-F-q } q \ C2$ **and**
active-minimal: $l2 \neq \text{active} \implies \text{active} \sqsubset L \ l2$ **and**
at-least-two-labels: $\exists l2. \text{active} \sqsubset L \ l2$ **and**
inf-never-active: $\iota \in \text{Inf-FL} \implies \text{snd}(\text{concl-of } \iota) \neq \text{active}$ **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 *Prec-eq-F* :: $'f \Rightarrow 'f \Rightarrow \text{bool}$ (**infix** \preceq 50) **where**
 $C \preceq \cdot D \equiv C \doteq D \vee C \prec \cdot D$

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

lemma *irrefl-prec-F*: $\neg C \prec \cdot C$
 $\langle \text{proof} \rangle$

lemma *trans-prec-F*: $C1 \prec \cdot C2 \implies C2 \prec \cdot C3 \implies C1 \prec \cdot C3$
 $\langle \text{proof} \rangle$

lemma *wf-prec-FL*: *minimal-element* (\sqsubset) *UNIV*
 $\langle \text{proof} \rangle$

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

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

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

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

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

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

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

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

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

Bot-FL G-F-L-q G-I-L-q λg. Prec-FL
 ⟨proof⟩

notation *derive* (**infix** ▷L 50)

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

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

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

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

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

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

⟨proof⟩

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⟩

end

6.2 Given Clause Procedure

locale *given-clause = given-clause-basis Bot-F Inf-F Bot-G Q entails-q Inf-G-q Red-I-q*
Red-F-q G-F-q G-I-q Inf-FL 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**

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

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

Inf-FL :: \langle ('f \times 'l) inference set \rangle **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** \sqsubset L 50) **and**

active :: 'l +

assumes

inf-have-prems: $\iota F \in \text{Inf-F} \Longrightarrow \text{prems-of } \iota F \neq []$

begin

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

⟨proof⟩

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

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

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

lemma (in-) *all-ex-finite-set*: $(\forall (j::nat) \in \{0..<m\}. \exists (n::nat). P\ j\ n) \Rightarrow$
 $(\forall n1\ n2. \forall j \in \{0..<m\}. P\ j\ n1 \rightarrow P\ j\ n2 \rightarrow n1 = n2) \Rightarrow finite\ \{n. \exists j \in \{0..<m\}. P\ j\ n\}$ **for** m
 P
<proof>

lemma gc-fair:

assumes

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

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

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

shows *fair* Ns

<proof>

theorem gc-complete-Liminf:

assumes

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

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

final-state: $passive\text{-}subset (Liminf\text{-}l\text{list}\ Ns) = \{\}$ **and**

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

bot-entailed: $no\text{-}labels.entails\text{-}\mathcal{G} (fst \text{ ' } lhd\ Ns) \{B\}$

shows $\exists BL \in Bot-FL. BL \in Liminf\text{-}l\text{list}\ Ns$

<proof>

theorem gc-complete:

assumes

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

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

final-state: $passive\text{-}subset (Liminf\text{-}l\text{list}\ Ns) = \{\}$ **and**

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

bot-entailed: $no\text{-}labels.entails\text{-}\mathcal{G} (fst \text{ ' } lhd\ Ns) \{B\}$

shows $\exists i. enat\ i < llength\ Ns \wedge (\exists BL \in Bot-FL. BL \in lnth\ Ns\ i)$

<proof>

end

6.3 Lazy Given Clause Procedure

locale *lazy-given-clause* = *given-clause-basis* $Bot-F\ Inf-F\ Bot-G\ Q\ entails\text{-}q\ Inf-G\ q\ Red-I\ q$

Red-F-q G-F-q G-I-q Inf-FL Equiv-F Prec-F Prec-L active

for

Bot-F :: 'f set **and**
Inf-F :: 'f inference set **and**
Bot-G :: 'g set **and**
Q :: 'q set **and**
entails-q :: 'q \Rightarrow 'g set \Rightarrow 'g set \Rightarrow bool **and**
Inf-G-q :: <'q \Rightarrow 'g inference set> **and**
Red-I-q :: 'q \Rightarrow 'g set \Rightarrow 'g inference set **and**
Red-F-q :: 'q \Rightarrow 'g set \Rightarrow 'g set **and**
G-F-q :: 'q \Rightarrow 'f \Rightarrow 'g set **and**
G-I-q :: 'q \Rightarrow 'f inference \Rightarrow 'g inference set option **and**
Inf-FL :: <'f \times 'l inference set> **and**
Equiv-F :: 'f \Rightarrow 'f \Rightarrow bool (**infix** \doteq 50) **and**
Prec-F :: 'f \Rightarrow 'f \Rightarrow bool (**infix** \prec 50) **and**
Prec-L :: 'l \Rightarrow 'l \Rightarrow bool (**infix** \sqsubseteq L 50) **and**
active :: 'l

begin

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

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

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

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

lemma *lgc-fair*:

assumes

deriv: $\text{chain } (\rightsquigarrow$ LGC) Ns **and**
init-state: $\text{active-subset } (\text{snd } (\text{lhd } Ns)) = \{\}$ **and**
final-state: $\text{passive-subset } (\text{Liminf-list } (\text{lmap snd } Ns)) = \{\}$ **and**
no-prems-init-active: $\forall \iota \in \text{Inf-F}. \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst } (\text{lhd } Ns)$ **and**
final-schedule: $\text{Liminf-list } (\text{lmap fst } Ns) = \{\}$

shows *fair* $(\text{lmap snd } Ns)$

<proof>

theorem *lgc-complete-Liminf*:

assumes

deriv: $\text{chain } (\rightsquigarrow$ LGC) Ns **and**

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

theorem *lgc-complete:*

assumes

deriv: $\text{chain} (\sim \text{LGC}) Ns$ **and**

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

final-state: passive-subset ($\text{Liminf-list} (\text{lmap } \text{snd } Ns) = \{\}$) **and**

no-prems-init-active: $\forall \iota \in \text{Inf-F. } \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst} (\text{lhd } Ns)$ **and**

final-schedule: $\text{Liminf-list} (\text{lmap } \text{fst } Ns) = \{\}$ **and**

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

bot-entailed: $\text{no-labels.entails-}\mathcal{G} (\text{fst } ' \text{snd} (\text{lhd } Ns)) \{B\}$

shows $\exists i. \text{enat } i < \text{llength } Ns \wedge (\exists BL \in \text{Bot-FL. } BL \in \text{snd} (\text{lth } Ns \ i))$

<proof>

end

end