Formalization of a Comprehensive Framework for Saturation Theorem Proving in Isabelle/HOL

Sophie Tourret

April 20, 2020

Abstract

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

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

Contents

1 Consequence Relations and Inference Systems 2
  1.1 Consequence Relations .................................................. 2
  1.2 Families of Consequence Relations ..................................... 2
  1.3 Inference Systems .......................................................... 3

2 Calculi 3
  2.1 Calculi with a Redundancy Criterion ................................. 3
  2.2 Calculi with a Family of Redundancy Criteria ...................... 6
  2.3 Variations on a Theme ..................................................... 7

3 Lifting to Non-ground Calculi 10
  3.1 Standard Lifting ............................................................ 10
  3.2 Strong Standard Lifting ................................................... 11
  3.3 Lifting with a Family of Well-founded Orderings .................. 11
  3.4 Lifting with a Family of Redundancy Criteria ....................... 14

4 Labeled Liftings 16
  4.1 Labeled Lifting with a Family of Well-founded Orderings ........ 16
  4.2 Labeled Lifting with a Family of Redundancy Criteria ............ 18
5 Prover Architectures

5.1 Basis of the Prover Architectures

5.2 Given Clause Architecture

5.3 Lazy Given Clause Architecture

1 Consequence Relations and Inference Systems

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. This corresponds to section 2.1 of the report.

theory Consequence-Relations-and-Inference-Systems
imports Main
begin

1.1 Consequence Relations

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

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

lemma entail-union: N | N1 ∧ N | N2 ↔ N | N1 ∪ N2
⟨proof⟩

lemma entail-unions: (∀ i ∈ I. N | Ni i) ↔ N | ∪ (Ni ‘ I)
⟨proof⟩

lemma entail-all-bot: (∃ B ∈ Bot. N | {B}) ⇒ (∀ B’ ∈ Bot. N | {B'})
⟨proof⟩

end

1.2 Families of Consequence Relations

locale consequence-relation-family =
  fixes Bot :: 'f set and
  Q :: 'q itself and
  entails-q :: 'q ⇒ ('f set ⇒ 'f set ⇒ bool)
  assumes Bot-not-empty: Bot ≠ {} and
  q-cons-rel: consequence-relation Bot (entails-q q)
begin


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

lemma intersect-cons-rel-family: consequence-relation Bot entails-Q
(proof)
end

1.3 Inference Systems
datatype 'f inference =
  Infer (prems-of: 'f list) (concl-of: 'f)
locale inference-system =
  fixes Inf :: ('f inference set)
begin
definition Inf-from :: 'f set ⇒ 'f inference set where
  Inf-from N = {ι ∈ Inf. set (prems-of ι) ⊆ N}
definition Inf-from2 :: 'f set ⇒ 'f set ⇒ 'f inference set where
  Inf-from2 N M = Inf-from (N ∪ M) − Inf-from (N − M)
end
end

2 Calculi

In this section, the section 2.2 to 2.4 of the report are covered. This includes results on calculi equipped with a redundancy criterion or with a family of redundancy criteria, as well as a proof that various notions of redundancy are equivalent

theory Calculi
  imports
    Consequence-Relations-and-Inference-Systems
    Ordered-Resolution-Prover.Lazy-List-Liminf
    Ordered-Resolution-Prover.Lazy-List-Chain
begin

2.1 Calculi with a Redundancy Criterion
locale calculus-with-red-crit = inference-system Inf + consequence-relation Bot entails
  for
    Bot :: 'f set and
    Inf :: ('f inference set) and
    entails :: 'f set ⇒ 'f set ⇒ bool (infix ⊨ 50)
  + fixes
    Red-Inf :: 'f set ⇒ 'f inference set and
    Red-F :: 'f set ⇒ 'f set
  assumes
    Red-Inf-to-Inf: Red-Inf N ⊆ Inf and
Red-F-Bot: $B \in \text{Bot} \Rightarrow N = \{B\} \Rightarrow N - \text{Red-F} N = \{B\}$ and

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

Red-Inf-of-subset: $N \subseteq N' \Rightarrow \text{Red-Inf} N \subseteq \text{Red-Inf} N'$ and

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

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

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

begin

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

(proof)

lemma red-concl-to-red-inf:

assumes

\begin{align*}
\iota - \text{in} & : \iota \in \text{Inf} \text{ and} \\
\text{concl} & : \text{concl-of} \iota \in \text{Red-F} N \\
shows & : \iota \in \text{Red-Inf} N
\end{align*}

(proof)

definition saturated :: 'f set $\Rightarrow$ bool

saturated $N \equiv \text{Inf-from} N \subseteq \text{Red-Inf} N$

definition reduc-saturated :: 'f set $\Rightarrow$ bool

reduc-saturated $N \equiv \text{Inf-from} (N - \text{Red-F} N) \subseteq \text{Red-Inf} N$

lemma Red-Inf-without-red-F:

\text{Red-Inf} (N - \text{Red-F} N) = \text{Red-Inf} N

(proof)

lemma saturated-without-red-F:

assumes saturated: saturated $N$

shows saturated $(N - \text{Red-F} N)$

(proof)

definition Sup-Red-Inf-lst :: 'f set llist $\Rightarrow$ 'f inference set

$\text{Sup-Red-Inf-lst} \equiv ( \bigcup \iota \in \{i. \text{enat} i < \text{llength} D\}. \text{Red-Inf} (\text{lnth} D i))$

lemma Sup-Red-Inf-unit: $\text{Sup-Red-Inf-lst} (LCons X LNil) = \text{Red-Inf} X$

(proof)

definition fair :: 'f set llist $\Rightarrow$ bool

$\text{fair} \equiv \text{Inf-from} (\text{Liminf-lst} D) \subseteq \text{Sup-Red-Inf-lst} D$

inductive derive :: 'f set $\Rightarrow$ 'f set $\Rightarrow$ bool ($\text{infix} \triangleright \text{Red} 50$) where

derive: $M - N \subseteq \text{Red-F} N \Rightarrow M \triangleright \text{Red} N$


lemma (in –) elem-Sup-lst-imp-Sup-upto-lst':

\begin{align*}
x \in \text{Sup-lst} Xs & \Rightarrow \exists j < \text{llength} Xs. x \in \text{Sup-upto-lst} Xs j \\
\end{align*}

(proof)

lemma gt-Max-notin: ($\text{finite} A \Rightarrow A \neq \{\} \Rightarrow x > \text{Max} A \Rightarrow x \notin A$)

(proof)

lemma equiv-Sup-Liminf:

assumes
in-Sup: $C \in \text{Sup-list } D$ and
not-in-Liminf: $C \notin \text{Liminf-list } D$
shows
$\exists \ i \in \{ i. \ \text{enat} (\text{Suc } i) < \text{llength } D \}. \ C \in \text{lnth } D \ i - \text{lnth } D (\text{Suc } i)$
\(\langle\text{proof}\rangle\)

**lemma** Red-in-Sup:
\text{assumes deriv: chain (Red) } D
\text{shows Sup-list } D - \text{Liminf-list } D \subseteq \text{Red-F (Sup-list } D)\n\(\langle\text{proof}\rangle\)

**lemma** Red-Inf-subset-Liminf:
\text{assumes deriv: chain (Red) } D \text{ and}
i: \text{enat } i < \text{llength } D
\text{shows Red-Inf (lnth } D \ i) \subseteq \text{Red-Inf (Liminf-list } D)\n\(\langle\text{proof}\rangle\)

**lemma** Red-F-subset-Liminf:
\text{assumes deriv: chain (Red) } D \text{ and}
i: \text{enat } i < \text{llength } D
\text{shows Red-F (lnth } D \ i) \subseteq \text{Red-F (Liminf-list } D)\n\(\langle\text{proof}\rangle\)

**lemma** i-in-Liminf-or-Red-F:
\text{assumes deriv: chain (Red) } D \text{ and}
i: \text{enat } i < \text{llength } D
\text{shows lnth } D \ i \subseteq \text{Red-F (Liminf-list } D) \cup \text{Liminf-list } D\n\(\langle\text{proof}\rangle\)

**lemma** fair-implies-Liminf-saturated:
\text{assumes deriv: chain (Red) } D \text{ and}
fair: \text{fair } D
\text{shows saturated (Liminf-list } D)\n\(\langle\text{proof}\rangle\)

end

locale static-refutational-complete-calculus = calculus-with-red-crit +
\text{assumes static-refutational-complete: } B \in \text{Bot} \implies \text{saturated } N \implies N \models \{ B \} \implies \exists B' \in \text{Bot}. \ B' \in N

locale dynamic-refutational-complete-calculus = calculus-with-red-crit +
\text{assumes dynamic-refutational-complete: } B \in \text{Bot} \implies \text{chain (Red) } D \implies \text{fair } D
\implies \text{lnth } D \ 0 \models \{ B \} \implies \exists i \in \{ i. \ \text{enat } i < \text{llength } D \}. \ \exists B' \in \text{Bot}. \ B' \in \text{lnth } D \ i

begin

sublocale static-refutational-complete-calculus
sublocale static-refutational-complete-calculus ⊆ dynamic-refutational-complete-calculus
(proof)

2.2 Calculi with a Family of Redundancy Criteria
locale calculus-with-red-crit-family = inference-system Inf + consequence-relation-family Bot Q entails-q for
Bot :: 'f set and
Inf :: ('f inference set) and
Q :: 'q itself and
teracts-qr :: 'q ⇒ ('f set ⇒ 'f set ⇒ bool)
+ fixes
Red-Inf-qr :: 'q ⇒ ('f set ⇒ 'f inference set) and
Red-F-qr :: 'q ⇒ ('f set ⇒ 'f set)
assumes
all-red-crit: calculus-with-red-crit Bot Inf (entails-qr q) (Red-Inf-qr q) (Red-F-qr q)
begin
definition Red-Inf-Q :: 'f set ⇒ 'f inference set where
Red-Inf-Q N = ∩ {X N | X ∈ (Red-Inf-qr ' UNIV)}
definition Red-F-Qr :: 'f set ⇒ 'f set where
Red-F-Qr N = ∩ {X N | X ∈ (Red-F-qr ' UNIV)}
lemma inter-red-crit: calculus-with-red-crit Bot Inf entails-qr Red-Inf-Q Red-F-Qr
(proof)
sublocale inter-red-crit-calculus: calculus-with-red-crit
where Bot=Bot
and Inf=Inf
and entails=entails-qr
and Red-Inf=Red-Inf-Qr
and Red-F=Red-F-Qr
(proof)
lemma sat-int-to-sat-qr: calculus-with-red-crit.saturated Inf Red-Inf-Q r for N
(proof)
lemma stat-ref-comp-from-bot-in-sat:
∀ N. (calculus-with-red-crit.saturated Inf Red-Inf-Q r N) →
(3 B ∈ Bot. B ∈ N)
(proof)
end
2.3 Variations on a Theme

locale calculus-with-reduced-red-crit = calculus-with-red-crit Bot Inf entails Red-Inf Red-F
for
Bot :: 'f set and
Inf :: 'f inference set and
entails :: 'f set ⇒ 'f set ⇒ bool (infix ⟨=⟩ 50) and
Red-Inf :: 'f set ⇒ 'f inference set and
Red-F :: 'f set ⇒ 'f set
+ assumes
inf-in-red-inf: Inf-from2 UNIV (Red-F N) ⊆ Red-Inf N
begin
lemma sat-eq-reduc-sat: saturated N ⇔ reduc-saturated N
⟨proof⟩
end
locale reduc-static-refutational-complete-calculus = calculus-with-red-crit +
  assumes reduc-static-refutational-complete: B ∈ Bot ⇒ reduc-saturated N ⇒ N | {B} ⇒
  ∃ B' ∈ Bot. B' ∈ N
locale reduc-static-refutational-complete-reduc-calculus = calculus-with-reduced-red-crit +
  assumes reduc-static-refutational-complete: B ∈ Bot ⇒ reduc-saturated N ⇒ N | {B} ⇒
  ∃ B' ∈ Bot. B' ∈ N
begin
sublocale reduc-static-refutational-complete-calculus
⟨proof⟩
sublocale static-refutational-complete-calculus
⟨proof⟩
end
context calculus-with-reduced-red-crit
begin
lemma stat-ref-comp-imp-red-stat-ref-comp: static-refutational-complete-calculus Bot Inf entails Red-Inf Red-F
  ⇒ reduc-static-refutational-complete-calculus Bot Inf entails Red-Inf Red-F
⟨proof⟩
end
context calculus-with-red-crit
begin
definition Red-Red-Inf :: 'f set ⇒ 'f inference set where
  Red-Red-Inf N = Red-Inf N ∪ Inf-from2 UNIV (Red-F N)
lemma reduced-calc-is-calc: calculus-with-red-crit Bot Inf entails Red-Red-Inf Red-F
⟨proof⟩
lemma inf-subs-reduced-red-inf: Inf-from2 UNIV (Red-F N) ⊆ Red-Red-Inf N

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: calculus-with-reduced-red-crit Bot Inf entails Red-Red-Inf Red-F**

**interpretation reduc-calc : calculus-with-reduced-red-crit Bot Inf entails Red-Red-Inf Red-F**

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

**lemma red-sat-eq-red-calc-sat: reduc-saturated N \iff reduc-calc.saturated N**

**lemma red-sat-eq-sat: reduc-saturated N \iff saturated (N − Red-F N)**

**theorem stat-is-stat-red: static-refutational-complete-calculus Bot Inf entails Red-Inf Red-F \iff static-refutational-complete-calculus Bot Inf entails Red-Red-Inf Red-F**


**definition Sup-Red-F-llist :: \forall f set llist \Rightarrow f set where Sup-Red-F-llist D = (\bigcup i \in \{i. enat i < llength D\}. Red-F (lnth D i))**

**lemma Sup-Red-F-unit: Sup-Red-F-llist (LCons X LNil) = Red-F X**

**lemma sup-red-f-in-red-liminf: chain derive D \implies Sup-Red-F-llist D \subseteq Red-F (Liminf-llist D)**

**lemma sup-red-inf-in-red-liminf: chain derive D \implies Sup-Red-Inf-llist D \subseteq Red-Inf (Liminf-llist D)**

**definition reduc-fair :: \forall f set llist \Rightarrow bool where reduc-fair D \equiv Inf-from (Liminf-llist D − (Sup-Red-F-llist D)) \subseteq Sup-Red-Inf-llist D**
lemma reduc-fair-imp-Liminf-reduc-sat: chain derive $D \implies \text{reduc-fair } D \implies \text{reduc-saturated} (\text{Liminf-list } D)$

(proof)

end

locale reduc-dynamic-refutational-complete-calculus = calculus-with-red-crit +

assumes reduc-dynamic-refutational-complete: $B \in \text{Bot} \implies \text{chain derive } D \implies \text{reduc-fair } D \implies \text{lnth } D \ 0 \models \{B\} \implies \exists i \in \{i. \text{enat } i < \text{llength } D\}. ~ \exists B' \in \text{Bot}. ~ B' \in \text{lnth } D \ i$

begin

sublocale reduc-static-refutational-complete-calculus

(proof)

end

sublocale reduc-static-refutational-complete-calculus \subseteq reduc-dynamic-refutational-complete-calculus

(proof)

context calculus-with-red-crit

begin

lemma dyn-equiv-stat: dynamic-refutational-complete-calculus Bot Inf entails Red-Inf Red-F =

static-refutational-complete-calculus Bot Inf entails Red-Inf Red-F

(proof)

lemma red-dyn-equiv-red-stat: reduc-dynamic-refutational-complete-calculus Bot Inf entails Red-Inf Red-F =

reduc-static-refutational-complete-calculus Bot Inf entails Red-Inf Red-F

(proof)

interpretation reduc-calc : calculus-with-reduced-red-crit Bot Inf entails Red-Red-Inf Red-F

(proof)


(proof)


(proof)

end
3 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 Calculi Well-Quasi-Orders.Minimal-Elements
begin

3.1 Standard Lifting

locale standard-lifting =
  Non-ground: inference-system Inf-F +
for
  Bot-F :: ⟨f set⟩ and
  Inf-F :: ⟨f inference set⟩ and
  Bot-G :: ⟨g set⟩ and
  Inf-G :: ⟨g inference set⟩ and
  entails-G :: ⟨g set ⇒ bool⟩ (infix ⊢ G 50) and
  Red-Inf-G :: ⟨g set ⇒ g inference set⟩ and
  Red-F-G :: ⟨g set ⇒ g set⟩
+ fixes
  G-F :: ⟨f ⇒ g set⟩ and
  G-Inf :: ⟨f inference ⇒ g inference set option⟩
assumes
  Bot-F-not-empty: Bot-F ≠ {} and
  Bot-map-not-empty: B ∈ Bot-F ⊢ G-F B ≠ {} and
  Bot-map: ⟨B ∈ Bot-F ⊢ G-F B ⊆ Bot-G⟩ and
  Bot-cond: ⟨G-F C ∩ Bot-G ≠ {} ⊢ C ∈ Bot-F⟩ and
  inf-map: i ∈ Inf-F ⊢ G-Inf i ≠ None ⊢ the (G-Inf i) ⊆ Red-Inf-G (G-F (concl-of i))
begin

abbreviation G-set :: ⟨f set ⇒ g set⟩ where
  ⟨G-set N ≡ ⋃ (G-F · N)⟩

lemma G-subset: ⟨N1 ⊆ N2 ⊢ G-set N1 ⊆ G-set N2⟩ (proof)

definition entails-G :: ⟨f set ⇒ f set ⇒ bool⟩ (infix ⊢ G 50) where
  ⟨N1 ⊢ G N2 ≡ G-set N1 ⊢ G G-set N2⟩

lemma subs-Bot-G-entails:
  assumes
    not-empty: ⟨sB ≠ {}⟩ and
    in-bot: ⟨sB ⊆ Bot-G⟩
  shows ⟨sB ⊢ G N⟩
  (proof)

sublocale lifted-consequence-relation: consequence-relation
  where Bot=Bot-F and entails=entails-G
3.2 Strong Standard Lifting

locale strong-standard-lifting =

Non-ground: inference-system Inf-F +


for

Bot-F :: ⟨'f set⟩ and
Inf-F :: ⟨'f inference set⟩ and
Bot-G :: ⟨'g set⟩ and
Inf-G :: ⟨'g inference set⟩ and

entails-G :: ⟨'g set ⇒ 'g set ⇒ bool⟩ (infix ⊨ G 50) and
Red-Inf-G :: ⟨'g set ⇒ 'g inference set⟩ and
Red-F-G :: ⟨'g set ⇒ 'g set⟩

+ fixes

G-F :: ⟨'f ⇒ 'g set⟩ and

G-Inf :: ⟨'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-cond: B ∈ Bot-F ⇒ G-F B ⊆ Bot-G and

strong-inf-map: ι ∈ Inf-F ⇒ G-Inf ι ≠ None ⇒ concl-of ' (the (G-Inf ι)) ⊆ (G-F (concl-of ι))

and

inf-map-in-Inf: ι ∈ Inf-F ⇒ G-Inf ι ≠ None ⇒ the (G-Inf ι) ⊆ Inf-G

begin

sublocale standard-lifting
(proof)

end

3.3 Lifting with a Family of Well-founded Orderings

locale lifting-with-wf-ordering-family =


for

Bot-F :: ⟨'f set⟩ and
Inf-F :: ⟨'f inference set⟩ and
Bot-G :: ⟨'g set⟩ and

entails-G :: ⟨'g set ⇒ 'g set ⇒ bool⟩ (infix ⊨ G 50) and
Inf-G :: ⟨'g inference set⟩ and
Red-Inf-G :: ⟨'g set ⇒ 'g inference set⟩ and
Red-F-G :: ⟨'g set ⇒ 'g set⟩ and

G-F :: 'f ⇒ 'g set and

G-Inf :: '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-Inf-G :: ⟨'f set ⇒ 'f inference set⟩ where

Red-Inf-G N = {ι ∈ Inf-F. (G-Inf ι ≠ None ∧ the (G-Inf ι) ⊆ Red-Inf-G (G-set N))}
\[ G - \text{Inf} = \text{None} \land G - F \ (\text{concl-of } \iota) \subseteq (G - \text{set } N \cup \text{Red-F-G} (G - \text{set } N)) \]

**Definition** Red-F-G :: 'f set \Rightarrow 'f set where
\[ \text{Red-F-G} N = \{ C. \forall D \in G - F C. D \in \text{Red-F-G} (G - \text{set } N) \lor (\exists E \in N. \text{Prec-F-g} D E C) \} \]

**Lemma** Prec-trans:

assumes
\[ \langle \text{Prec-F-g} D A B \rangle \text{ and } \langle \text{Prec-F-g} D B C \rangle \]

shows
\[ \langle \text{Prec-F-g} D A C \rangle \]

**Proof**

**Lemma** prop-nested-in-set:

\[ D \in P C = \Rightarrow C \in \{ C. \forall D \in P C. A D \lor B C D \} \Rightarrow A D \lor B C D \]

**Proof**

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

\[ \text{Red-F-G} N = \{ C. \forall D \in G - F C. D \in \text{Red-F-G} (G - \text{set } N) \lor (\exists E \in (N - \text{Red-F-G} N). \text{Prec-F-g} D E C \land D \in G - F E) \} \]

**Proof**

**Lemma** not-red-map-in-map-not-red: \( G - \text{set } N - \text{Red-F-G} (G - \text{set } N) \subseteq G - \text{set } (N - \text{Red-F-G} N) \)

**Proof**

**Lemma** Red-F-Bot-F: \( B \in \text{Bot-F} = \Rightarrow N |={ G \{ B \} } \Rightarrow N - \text{Red-F-G} N |={ G \{ B \} } \)

**Proof**

**Lemma** Red-F-of-subset-F: \( \forall N \subseteq N' = \Rightarrow \text{Red-F-G} N \subseteq \text{Red-F-G} N' \)

**Proof**

**Lemma** Red-Inf-of-subset-F: \( \forall N \subseteq N' = \Rightarrow \text{Red-Inf-G} N \subseteq \text{Red-Inf-G} N' \)

**Proof**

**Lemma** Red-F-of-Red-F-subset-F: \( \forall N' \subseteq \text{Red-F-G} N = \Rightarrow \text{Red-F-G} N \subseteq \text{Red-F-G} (N - N') \)

**Proof**

**Lemma** Red-Inf-of-Red-F-subset-F: \( \forall N' \subseteq \text{Red-F-G} N = \Rightarrow \text{Red-Inf-G} N \subseteq \text{Red-Inf-G} (N - N') \)

**Proof**

**Lemma** Red-Inf-of-Inf-to-N-F:

assumes
\[ \iota - \text{in: } \iota \in \text{Inf-F} \text{ and } \text{concl-i-in: } (\text{concl-of } \iota \in N) \]

shows
\[ \iota \in \text{Red-Inf-G} N \]

**Proof**
sublocale lifted-calculus-with-red-crit: calculus-with-red-crit
where
  Bot = Bot-F and Inf = Inf-F and entails = entails-\mathcal{G} and
  Red-Inf = Red-Inf-\mathcal{G} and Red-F = Red-F-\mathcal{G}
⟨proof⟩
lemma lifted-calc-is-calc: calculus-with-red-crit Bot-F Inf-F entails-\mathcal{G} Red-Inf-\mathcal{G} Red-F-\mathcal{G}
⟨proof⟩
lemma grounded-inf-in-ground-inf: \iota \in Inf-F \Rightarrow G-INF \iota \neq None \Rightarrow \text{the } (G-INF \iota) \subseteq Inf-G
⟨proof⟩
lemma sat-imp-ground-sat: lifted-calculus-with-red-crit. \text{saturated } N \Rightarrow \text{Ground-INF from } (G-set N) \subseteq
  (\{\iota. \exists \iota' \in \text{Non-ground-INF from } N. G-INF \iota' \neq None \land \iota \in \text{the } (G-INF \iota')\} \cup \text{Red-INF-}G (G-set N))
⟨proof⟩
theorem stat-ref-comp-to-non-ground:
  assumes
    stat-ref-G: static-refutational-complete-calculus Bot-G Inf-G entails-G Red-Inf-G Red-F-G and
    sat-n-imp: \forall N. \text{lifted-calculus-with-red-crit.} \text{saturated } N \Rightarrow \text{Ground-INF from } (G-set N) \subseteq
      (\{\iota. \exists \iota' \in \text{Non-ground-INF from } N. G-INF \iota' \neq None \land \iota \in \text{the } (G-INF \iota')\} \cup \text{Red-INF-}G (G-set N))
  shows
    static-refutational-complete-calculus Bot-F Inf-F entails-G Red-Inf-\mathcal{G} Red-F-\mathcal{G}
⟨proof⟩
end
abbreviation Empty-Order where
  Empty-Order C1 C2 ≡ False
lemma any-to-empty-order-lifting:
  G-Inf Prec-F-g \Rightarrow lifting-with-wf-ordering-family Bot-F Inf-F Bot-G entails-G Inf-G Red-Inf-G
  Red-F-\mathcal{G} G-\mathcal{F} G-\text{Inf } (\lambda g. \text{Empty-Order})
⟨proof⟩
locale lifting-equivalence-with-empty-order =
  Red-F-\mathcal{G} G-\mathcal{F} G-\text{Inf Prec-F-g} +
  Red-F-\mathcal{G} G-\mathcal{F} G-\text{Inf } \lambda g. \text{Empty-Order}
for
  G-F :: ('f ⇒ 'g set) and
  G-Inf :: ('f inference ⇒ 'g inference set option) and
  Bot-F :: ('f set) and
  Inf-F :: ('f inference set) and
  Bot-G :: ('g set) and
  Inf-G :: ('g inference set) and
  entails-G :: ('g set ⇒ 'g set ⇒ bool \textbf{infix} \models G 50) and
  Red-Inf-G :: ('g set ⇒ 'g inference set) and
The text contains formal logic expressions and proofs, discussing concepts of lifting with red-crit criteria in a formal proof assistant context. The content is structured to include definitions, lemmas, theorems, and sublocales, each with detailed logical expressions and proofs to support the claims made. The text is precise and adheres to the syntax and semantics of the formal proof assistant used.
\[G\text{-}set\text{-}q \; q \; N \equiv \bigcup \{G\text{-}F\text{-}q \; q \; \ast \; N\}\]

definition Red-Inf-G\text{-}q :: \(\forall q \Rightarrow \{\text{set} \Rightarrow \{\text{inference} \text{ set} \} \} \text{ where} \\
\text{Red-Inf-G}\text{-}q \; q \; N = \{ i \in \text{Inf-F}. \ (G\text{-}Inf-q \; q \; i) \subseteq \text{Red-Inf-q} \; q \; (G\text{-}set-q \; q \; N) \} \lor (G\text{-}Inf-q \; q \; i = \text{None} \land G\text{-}F-q \; q \; \text{(concl-of} \; i) \subseteq \left(G\text{-}set-q \; q \; N \cup \text{Red-F-q} \; q \; (G\text{-}set-q \; q \; N)\right))\}

definition Red-Inf-G\text{-}Q :: \(\forall \text{ set} \Rightarrow \{\text{inference} \text{ set} \} \text{ where} \\
\text{Red-Inf-G}\text{-}Q \; N = \bigcap \{X \; N \mid X \in (\text{Red-Inf-G}\text{-}q \; q \; \ast \; \text{UNIV})\}\}

definition Red-F-G\text{-}empty\text{-}q :: \(\forall q \Rightarrow \{\text{set} \Rightarrow \{\text{inference} \text{ set} \} \} \text{ where} \\
\text{Red-F-G}\text{-}empty\text{-}q \; q \; N = \{ C. \forall D \in G\text{-}F-q \; q \; C. \; D \in \text{Red-F-q} \; q \; (G\text{-}set-q \; q \; N) \lor (\exists E \in N. \; \text{Empty-Order} \; E \; C \land D \in G\text{-}F-q \; q \; E)\}\}

definition Red-F-G\text{-}empty :: \(\forall \text{ set} \Rightarrow \{\text{set} \Rightarrow \{\text{inference} \text{ set} \} \} \text{ where} \\
\text{Red-F-G}\text{-}empty \; N = \bigcap \{X \; N \mid X \in (\text{Red-F-G}\text{-}empty\text{-}q \; q \; \ast \; \text{UNIV})\}\}

definition Red-F-G\text{-}q \; q \; g :: \(\forall q \Rightarrow \{\text{set} \Rightarrow \{\text{set} \} \} \text{ where} \\
\text{Red-F-G}\text{-}q \; q \; g \; N = \{ C. \forall D \in G\text{-}F-q \; q \; C. \; D \in \text{Red-F-q} \; q \; (G\text{-}set-q \; q \; N) \lor (\exists E \in N. \; \text{Prec-F-g} \; D \; E \; C \land D \in G\text{-}F-q \; q \; E)\}\}

definition Red-F-G\text{-}g :: \(\forall \text{ set} \Rightarrow \{\text{set} \Rightarrow \{\text{set} \} \} \text{ where} \\
\text{Red-F-G}\text{-}g \; N = \bigcap \{X \; N \mid X \in (\text{Red-F-G}\text{-}q \; q \; \ast \; \text{UNIV})\}\}

definition entails-G\text{-}q :: \(\forall q \Rightarrow \{\text{set} \Rightarrow \{\text{set} \} \} \text{ where} \\
\text{entails-G}\text{-}q \; q \; N \equiv \forall \text{ q}. \; \text{entails-G}\text{-}q \; q \; \ast \; N \equiv N \equiv \forall \text{ q}. \; \text{entails-G}\text{-}q \; q \; N\}

definition entails-G\text{-}Q :: \(\forall \text{ set} \Rightarrow \{\text{set} \Rightarrow \{\text{set} \} \} \text{ where} \\
\text{entails-G}\text{-}Q \; N \equiv \forall \text{ q}. \; \text{entails-G}\text{-}q \; q \; N \equiv N \equiv \forall \text{ q}. \; \text{entails-G}\text{-}q \; q \; N\}

\text{lemma red-crit-lifting-family:} \\
\text{calculus-with-red-crit} \; \text{Bot-F Inf-F} \; (\text{entails-G}\text{-}q \; q) \; (\text{Red-Inf-G}\text{-}q \; q) \; (\text{Red-F-G}\text{-}q \; q \; g) \\
\text{(proof)}

\text{lemma red-crit-lifting-family-empty-ord:} \\
\text{calculus-with-red-crit} \; \text{Bot-F Inf-F} \; (\text{entails-G}\text{-}q \; q) \; (\text{Red-Inf-G}\text{-}q \; q) \; (\text{Red-F-G}\text{-}empty\text{-}q \; q) \\
\text{(proof)}

\text{lemma cons-rel-fam-Q-lem:} \; \text{consequence-relation-family} \; \text{Bot-F entails-G}\text{-}q \\
\text{(proof)}

\text{interpretation cons-rel-Q:} \; \text{consequence-relation} \; \text{Bot-F entails-G}\text{-}Q \\
\text{(proof)}

\text{sublocale lifted-calc-w-red-crit-family:} \\
\text{calculus-with-red-crit-family} \; \text{Bot-F Inf-F} \; Q \; \text{entails-G}\text{-}q \; \text{Red-Inf-G}\text{-}q \; \text{Red-F-G}\text{-}q \; g \\
\text{(proof)}

\text{lemma lifted-calc-family-is-calc:} \; \text{calculus-with-red-crit} \; \text{Bot-F Inf-F} \; \text{entails-G}\text{-}Q \; \text{Red-Inf-G}\text{-}Q \; \text{Red-F-G}\text{-}g \\
\text{(proof)}

\text{sublocale empty-ord-lifted-calc-w-red-crit-family:} \\
\text{calculus-with-red-crit-family} \; \text{Bot-F Inf-F} \; Q \; \text{entails-G}\text{-}q \; \text{Red-Inf-G}\text{-}q \; \text{Red-F-G}\text{-}empty-q \\
\text{(proof)}

\text{lemma inter-calc:} \; \text{calculus-with-red-crit} \; \text{Bot-F Inf-F} \; \text{entails-G}\text{-}Q \; \text{Red-Inf-G}\text{-}Q \; \text{Red-F-G}\text{-}empty
proof

\textbf{theorem} stat-ref-comp-to-non-ground-fam-inter:
\textbf{assumes}
\begin{align*}
\text{stat-ref-G: } & \forall q. \text{static-refutational-complete-calculus } \Bot-G \text{ Inf-G } \text{ (entails-q q) (Red-Inf-q q) (Red-F-q q) } \\
\text{sat-n-imp: } & \forall N. \text{(empty-ord-lifted-calc-w-red-crit-family, inter-red-crit-calculus, saturated N } \implies \exists q. \text{ Ground-family, Inf-from } (\Bot-G \text{-set-q q N}) \subseteq \\
& \{ \{ t. \exists t' \in \text{Non-ground, Inf-from N, } \Bot-G \text{-Inf-q q t'} \neq \text{None } \land t \in \text{the } (\Bot-G \text{-Inf-q q t'}) \} \cup \text{Red-Inf-q q} \\
(\Bot-G \text{-set-q q N} ) ) \\
\text{shows}
\end{align*}
\text{static-refutational-complete-calculus } \Bot-F \text{ Inf-F } \text{ entails-Q } \Bot-F \text{ Inf-F } \text{ entail-G-Q Red-Inf-G-Q Red-F-G-empty}
\langle \text{proof} \rangle

\langle \text{proof} \rangle

\textbf{lemma} static-empty-ord-inter-equiv-static-inter:
\text{static-refutational-complete-calculus } \Bot-F \text{ Inf-F } \text{ lifted-calc-w-red-crit-family, entails-Q } \\
\text{lifted-calc-w-red-crit-family, Red-Inf-Q } \text{ lifted-calc-w-red-crit-family, Red-F-Q } = \\
\text{static-refutational-complete-calculus } \Bot-F \text{ Inf-F } \text{ lifted-calc-w-red-crit-family, entails-Q } \\
\text{empty-ord-lifted-calc-w-red-crit-family, Red-Inf-Q } \text{ empty-ord-lifted-calc-w-red-crit-family, Red-F-Q}
\langle \text{proof} \rangle

\textbf{theorem} stat-eq-dyn-ref-comp-fam-inter: static-refutational-complete-calculus Bot-F Inf-F lifted-calc-w-red-crit-family, entails-Q \\
\text{empty-ord-lifted-calc-w-red-crit-family, Red-Inf-Q empty-ord-lifted-calc-w-red-crit-family, Red-F-Q } = \\
\text{dynamic-refutational-complete-calculus } \Bot-F \text{ Inf-F } \text{ lifted-calc-w-red-crit-family, entails-Q } \\
\text{lifted-calc-w-red-crit-family, Red-Inf-Q lifted-calc-w-red-crit-family, Red-F-Q } (\text{is ?static=?dynamic})
\langle \text{proof} \rangle

end

end

4 Labeled Liftings

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

\textbf{theory} Labeled-Lifting-to-Non-Ground-Calculi
\textbf{imports} Lifting-to-Non-Ground-Calculi
\begin{align*}
\text{begin} \\
\end{align*}
\begin{align*}
\text{4.1 Labeled Lifting with a Family of Well-founded Orderings} \\
\text{locale} labeled-lifting-w-wf-ord-family = \\
lifting-with-wf-ordering-family Bot-F Inf-F Bot-G \text{ entails-G Inf-G Red-Inf-G Red-F-G G-F G-Inf Prec-F } \\
\text{for} \\
\Bot-F :: \text{'f set and} \\
\text{Inf-F :: } \text{'f inference set and}
\end{align*}
Bot-G :: 'g set and
entails-G :: 'g set ⇒ 'g set ⇒ bool (infix = G 50) and
Inf-G :: 'g inference set and
Red-Inf-G :: 'g set ⇒ 'g inference set and
Red-F-G :: 'g set ⇒ 'g set and
G-F :: 'f ⇒ 'g set and
G-Inf :: 'f inference ⇒ 'g inference set option and
Prec-F :: 'g ⇒ 'f ⇒ 'f ⇒ bool (infix < 50)
+ fixes
l :: 'l itself and
Inf-FL :: ⟨'f × 'l⟩ inference set
assumes
Inf-F-to-Inf-FL: u_F ∈ Inf-F =⇒ length (Ll :: 'l list) = length (prems-of u_F) =⇒
∃L0. Infer (zip (prems-of u_F) Ll) (concl-of u_F, L0) ∈ Inf-FL and
Inf-FL-to-Inf-F: u_FL ∈ Inf-FL =⇒ Infer (map fst (prems-of u_FL)) (fst (concl-of u_FL)) ∈ Inf-F

begin

definition to-F :: ⟨'f × 'l⟩ inference ⇒ 'f inference where
to-F u_FL = Infer (map fst (prems-of u_FL)) (fst (concl-of u_FL))

definition Bot-FL :: ⟨'f × 'l⟩ set where (Bot-FL = Bot-F × UNIV)

definition G-F-L :: ⟨'f × 'l⟩ ⇒ 'g set where (G-F-L CL = G-F (fst CL))

definition G-Inf-L :: ⟨'f × 'l⟩ inference ⇒ 'g inference set option where (G-Inf-L u_FL = G-Inf (to-F u_FL))

sublocale labeled-standard-lifting: standard-lifting
where
Bot-F = Bot-FL and
Inf-F = Inf-FL and
G-F = G-F-L and
G-Inf = G-Inf-L
⟨proof⟩

abbreviation Labeled-Empty-Order :: ⟨'f × 'l⟩ ⇒ ⟨'f × 'l⟩ ⇒ bool where
Labeled-Empty-Order C1 C2 ≡ False

sublocale labeled-lifting-w-empty-ord-family :
lifting-with-wf-ordering-family Bot-FL Inf-FL Bot-G entails-G Inf-G Red-Inf-G Red-F-G
G-F-L G-Inf-L λg. Labeled-Empty-Order
⟨proof⟩

notation labeled-standard-lifting. entails-G (infix = G 50)

lemma labeled-entailment-lifting: NL1 ⊨ G NL2 =⇒ fst ⌈ NL1 ⌉ = G fst ⌈ NL2 ⌉
⟨proof⟩

lemma (in ¬) subset-fst: A ⊆ fst ⌈ AB ⌉ =⇒ ∀x ∈ A. ∃y. (x, y) ∈ AB ⟨proof⟩

lemma red-inf-impl: i ∈ labeled-lifting-w-empty-ord-family.Red-Inf-G NL =⇒ to-F i ∈ Red-Inf-G (fst ⌈ NL) ⟨proof⟩
lemma labeled-saturation-lifting:
labeled-lifting-w-empty-ord-family.lifted-calculus-with-red-crit.saturated NL ⋱
empty-order-lifting.lifted-calculus-with-red-crit.saturated (fst ⋯ NL)
(proof)

lemma stat-ref-comp-to-labeled-sta-ref-comp: static-refutational-complete-calculus Bot-F Inf-F (|=G) Red-Inf-G
Red-F-G ⋱
static-refutational-complete-calculus Bot-FL Inf-FL (|=GL)
(proof)

end

4.2 Labeled Lifting with a Family of Redundancy Criteria
locale labeled-lifting-with-red-crit-family = no-labels: standard-lifting-with-red-crit-family Inf-F
Bot-G Inf-G Q entails-q Red-Inf-q Red-F-q Bot-F G-F-q G-Inf-q λg. Empty-Order
for
Bot-F :: 'f set and
Inf-F :: 'f inference set and
Bot-G :: 'g set and
Q :: 'q itself and
entails-q :: 'q ⇒ 'g set ⇒ 'g set ⇒ bool and
Inf-G :: 'g inference set and
Red-Inf-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-Inf-q :: 'q ⇒ 'f inference ⇒ 'g inference set option
+ fixes
l :: 'l itself and
Inf-FL :: ('f × 'l) inference set
assumes
Inf-F-to-Inf-FL: ∀F ∈ Inf-F ⇒ length (Ll :: 'l list) = length (prems-of F) ⇒ ∃L0. Infer (zip (prems-of F) L) (concl-of F, L0) ∈ Inf-FL) and
Inf-FL-to-Inf-F: ∀FL ∈ Inf-FL ⇒ Infer (map fst (prems-of FL)) (fst (concl-of FL)) ∈ Inf-F
begin

definition to-F :: (('f × 'l) inference ⇒ 'f inference) where
F to-F = Infer (map fst (prems-of FL)) (fst (concl-of FL))

definition Bot-FL :: ('f × 'l) set where Bot-FL = Bot-F × UNIV

definition G-F-L-q :: 'q ⇒ (('f × 'l) ⇒ 'g set) where G-F-L-q q CL = G-F-q q (fst CL)

definition G-Inf-L-q :: 'q ⇒ ('f × 'l) inference ⇒ 'g inference set option) where
G-Inf-L-q q FL = G-Inf-q q (to-F FL)

definition G-set-L-q :: 'q ⇒ ('f × 'l) set ⇒ 'g set where
G-set-L-q q N ≡ ∪ (G-F-L-q q ′ N)

definition Red-Inf-G-L-q :: 'q ⇒ ('f × 'l) set ⇒ ('f × 'l) inference set where
Red-Inf-G-L-q q N = {i ∈ Inf-FL. ((G-Inf-L-q q i) ≠ None ∧ the (G-Inf-L-q q i) ⊆ Red-Inf-q q (G-set-L-q q N))}
\[ \forall \ (\text{G-Inf-L-q } q \ i = \text{None}) \land \text{G-F-L-q } q \ (\text{concl-of } i) \subseteq (\text{G-set-L-q } q \ N \cup \text{Red-F-q } q \ (\text{G-set-L-q } q \ N)) \]

**Definition** Red-Inf-G-L-q :: ('f × 'l) set ⇒ ('f × 'l) inference set where
\[ \text{Red-Inf-G-L-q } N = \bigcap \{ X \ N \mid X \in (\text{Red-Inf-G-L-q } q \ \text{UNIV}) \} \]

**Definition** Labeled-Empty-Order :: ('f × 'l) ⇒ ('f × 'l) ⇒ bool where
\[ \text{Labeled-Empty-Order } C1\ C2 = \text{False} \]

**Definition** Red-F-G-empty-L-q :: 'q ⇒ ('f × 'l) set ⇒ ('f × 'l) set where
\[ \text{Red-F-G-empty-L-q } N = \{ C. \forall D \in \text{G-F-L-q } q \ C. \ D \in \text{Red-F-q } q \ (\text{G-set-L-q } q \ N) \lor \ (
\exists E \in N. \ \text{Labeled-Empty-Order } E \ C \land D \in \text{G-F-L-q } q \ E) \} \]

**Definition** Red-F-G-empty-L-q :: ('f × 'l) set ⇒ ('f × 'l) set where
\[ \text{Red-F-G-empty-L-q } N = \bigcap \{ X \ N \mid X \in (\text{Red-F-G-empty-L-q } q \ \text{UNIV}) \} \]

**Definition** entails-G-L-q :: 'q ⇒ ('f × 'l) set ⇒ ('f × 'l) set ⇒ bool where
\[ \text{entails-G-L-q } q \ N1\ N2 = \text{entails-q } q \ (\text{G-set-L-q } q \ N1) \ (\text{G-set-L-q } q \ N2) \]

**Definition** entails-G-L-q :: ('f × 'l) set ⇒ ('f × 'l) set ⇒ bool (infix \(\models L 50\)) where
\[ \text{entails-G-L-q } N1\ N2 = \forall q. \ \text{entails-G-L-q } q \ N1\ N2 \]

**Lemma** lifting-q: labeled-lifting-w-wf-ord-family Bot-F Inf-F Bot-G (entails-q) Inf-G (Red-Inf-q) (Red-F-q) (G-F-q) (G-Inf-q) \((\lambda g. \text{Empty-Order})\) Inf-FL
\[ \text{proof} \]

**Lemma** lifted-q: standard-lifting Bot-FL Inf-FL Bot-G Inf-G (entails-q) (Red-Inf-q) (Red-F-q) (G-F-q) (G-Inf-q)
\[ \text{proof} \]

**Lemma** ord-fam-lifted-q: lifting-with-wf-ordering-family Bot-FL Inf-FL Bot-G (entails-q) Inf-G (Red-Inf-q) (Red-F-q) (G-F-q) (G-Inf-q)
\[ \text{proof} \]

**Lemma** all-lifted-red-crit: calculus-with-red-crit Bot-FL Inf-FL (entails-G-L-q) (Red-Inf-G-L-q)
\[ \text{proof} \]

**Lemma** all-lifted-cons-rel: consequence-relation Bot-FL (entails-G-L-q)
\[ \text{proof} \]

**Sublocale** labeled-cons-rel-family: consequence-relation-family Bot-FL Q entails-G-L-q
\[ \text{proof} \]

**Sublocale** with-labels: calculus-with-red-crit-family Bot-FL Inf-FL Q entails-G-L-q Red-Inf-G-L-q
\[ \text{proof} \]

**Notation** no-labels.\text{entails-G-L-q} \text{(infix }\models\text{L 50})

**Lemma** labeled-entailment-lifting: NL1 \(\models L\) NL2 \(
\iff\) \(\text{fst}'\ NL1 \(\models L\) \(\text{fst}'\ NL2\)
\[ \text{proof} \]

**Lemma** subset-fst: A \(\subseteq\) \(\text{fst}'\ AB \iff \forall x \in A. \exists y. (x,y) \in AB\)
\[ \text{proof} \]

19
lemma red-inf-impl: \( \iota \in \text{with-labels.Red-Inf-Q \ NL} \implies \text{to-F} \ \iota \in \text{no-labels.empty-ord-lifted-calc-w-red-crit-family.Red-Inf-Q (fst \ ' \ NL)} \)
  \(\langle \text{proof} \rangle\)

lemma labeled-family-saturation-lifting: \( \text{with-labels.inter-red-crit-calculus.saturated \ NL} \implies \text{no-labels.lifted-calc-w-red-crit-family.inter-red-crit-calculus.saturated (fst \ ' \ NL)} \)
  \(\langle \text{proof} \rangle\)

theorem labeled-static-ref: \( \text{static-refutational-complete-calculus Bot-F Inf-F (|=\cap)} \)
  \(\text{no-labels.empty-ord-lifted-calc-w-red-crit-family.Red-Inf-Q} \)
  \(\text{no-labels.empty-ord-lifted-calc-w-red-crit-family.Red-F-Q} \)
  \(\implies \text{static-refutational-complete-calculus Bot-FL Inf-FL (|=\cap L)} \ \text{with-labels.Red-Inf-Q with-labels.Red-F-Q} \)
  \(\langle \text{proof} \rangle\)

end
end

5 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 Prover-Architectures
  imports Labeled-Lifting-to-Non-Ground-Calculi
begin

5.1 Basis of the Prover Architectures

locale Prover-Architecture-Basis = labeled-lifting-with-red-crit-family Bot-F Inf-F Bot-G Q entails-q Inf-G
  Red-Inf-q Red-F-q G-F-q G-Inf-q l Inf-FL
  for
    Bot-F :: 'f set
  and Inf-F :: 'f inference set
  and Bot-G :: 'g set
  and Q :: 'q itself
  and entails-q :: 'q \Rightarrow ('g set \Rightarrow 'g set \Rightarrow \text{bool})
  and Inf-G :: ('g inference set)
  and Red-Inf-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-Inf-q :: 'q \Rightarrow 'f inference \Rightarrow 'g inference set option
  and l :: 'l itself
  and Inf-FL :: ('f \times 'l) inference set;
+ fixes
  Equiv-F :: ('f \times 'f) set and
  Prec-F :: 'f \Rightarrow 'f \Rightarrow \text{bool (infix \(\succapprox\) 50)} and
  Prec-l :: 'l \Rightarrow 'l \Rightarrow \text{bool (infix \(\sqsubseteq\) 50)}

assumes
  equiv-F-is-equiv-rel: equiv UNIV Equiv-F and
  wf-prec-F: minimal-element (Prec-F) UNIV and
  wf-prec-l: minimal-element (Prec-l) UNIV and

20
compat-eqv-prec: \((C_1, D_1) \in \text{equiv-F} \implies (C_2, D_2) \in \text{equiv-F} \implies C_1 \triangleright C_2 \implies D_1 \triangleright D_2\) and
equiv-F-grounding: \((C_1, C_2) \in \text{equiv-F} \implies G-F-q \ q C_1 = G-F-q \ q C_2\) and
prec-F-grounding: \(C_1 \triangleright C_2 \implies G-F-q \ q C_1 \subseteq G-F-q \ q C_2\) and

static-ref-comp: static-refutational-complete-calculus Bot-F Inf-F \(\vdash\) \(\subseteq\)

no-labels. empty-ord-lifted-calc-w-red-crit-family. Red-Inf-Q

no-labels. empty-ord-lifted-calc-w-red-crit-family. Red-F-Q

begin

definition equiv-F-fun :: \('f \Rightarrow 'f\) \Rightarrow \text{bool} \ (\text{infix} \ 50)\) where
equiv-F-fun \(C \ D\) \(\equiv\) \((C, D) \in \text{Equiv-F}\)

definition Prec-eq-F :: \('f \Rightarrow 'f\) \Rightarrow \text{bool} \ (\text{infix} \ '\preceq\' \ 50)\) where
Prec-eq-F \(C \ D\) \(\equiv\) \((\((C, D) \in \text{Equiv-F} \lor C \triangleright D\)\)

definition Prec-FL :: \('f \times 'l\) \Rightarrow \text{bool} \ (\text{infix} \ '\prec\' \ 50)\) where
Prec-FL \(Cl_1 \ Cl_2\) \(\equiv\) \((\text{fst Cl}_1 \triangleright \text{fst Cl}_2) \lor (\text{fst Cl}_1 \preceq \text{fst Cl}_2 \land \text{snd Cl}_1 \preceq \text{snd Cl}_2)\)

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

lemma labeled-static-ref-comp:
static-refutational-complete-calculus Bot-FL Inf-FL (\(\vdash\)L) with-labels. Red-Inf-Q with-labels. Red-F-Q
\(\langle\text{proof}\rangle\)

lemma standard-labeled-lifting-family: lifting-with-wf-ordering-family Bot-FL Inf-FL Bot-G
(\text{entails-q q}) Inf-G (Red-Inf-q q) (Red-F-q q) (G-F-L-q q) (G-Inf-L-q q) (\lambda g. Prec-FL)
\(\langle\text{proof}\rangle\)

entails-q Red-Inf-q Red-F-q
Bot-FL G-F-L-q G-Inf-L-q \(\lambda g.\) Prec-FL
\(\langle\text{proof}\rangle\)

lemma entail-equiv:
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family. entails-Q \(N_1 \ N_2\) = \((N_1 \vdash\)L \(N_2)\)
\(\langle\text{proof}\rangle\)

lemma entail-equiv2: labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family. entails-Q = (\(\vdash\)L)
\(\langle\text{proof}\rangle\)

\(\langle\text{proof}\rangle\)

\(\langle\text{proof}\rangle\)

\(\langle\text{proof}\rangle\)

\(\langle\text{proof}\rangle\)
lemma labeled-ordered-static-ref-comp:
static-refutational-complete-calculus Bot-FL Inf-FL
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.entails-Q
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-Inf-Q
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-F-Q
⟨proof⟩

interpretation stat-ref-calc: static-refutational-complete-calculus Bot-FL Inf-FL
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.entails-Q
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-Inf-Q
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-F-Q
⟨proof⟩

lemma labeled-ordered-dynamic-ref-comp:
dynamic-refutational-complete-calculus Bot-FL Inf-FL
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.entails-Q
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-Inf-Q
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-F-Q
⟨proof⟩

lemma labeled-red-inf-eq-red-inf: ι ∈ Inf-FL ⇒
labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-Inf-Q N ≡
(to-F ι) ∈ no-labels.empty-ord-lifted-calc-w-red-crit-family.Red-Inf-Q (fst ι) for ι
⟨proof⟩

lemma red-labeled-clauses: C ∈ no-labels.Red-F-G-empty (fst ι) ∨ (∃ C′ ∈ (fst ι). C ⪰ C′) ∨
(∃ (C′, L′) ∈ N. (L′ ⊑ l L ∧ C ·≻ C′)) ⇒
(C, L) ∈ labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-F-Q N
⟨proof⟩

end

5.2 Given Clause Architecture

locale Given-Clause = Prover-Architecture-Basis Bot-F Inf-F Bot-G Q entails-q Inf-G Red-Inf-q
Red-F-q G-F-q G-Inf-q l Inf-FL Equiv-F Prec-F Prec-l
for
Bot-F :: 'f set and
Inf-F :: 'f inference set and
Bot-G :: 'g set and
Q :: 'q itself and
entails-q :: 'q ⇒ ('g set ⇒ 'g set ⇒ bool) and
Inf-G :: 'g inference set and
Red-Inf-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-Inf-q :: 'q ⇒ 'f inference ⇒ 'g inference set option and
l :: 'l itself and
Inf-FL :: ('f × 'l) inference set and
Equiv-F :: ('f × 'f) set and
Prec-F :: 'f ⇒ 'f ⇒ bool (infix ·≻ 50) and
Prec-l :: 'l ⇒ 'l ⇒ bool (infix ⊑ l 50)
+ fixes

22
active :: 'l

assumes
inf-have-premises: \( iF \in \text{Inf}-F \implies \text{length (prems-of } iF) > 0 \) and
active-minimal: \( l_2 \neq \text{active} \implies \text{active} \sqsubseteq l_2 \) and
at-least-two-labels: \( \exists l_2. \text{ active} \sqsubseteq l_2 \) and
inf-never-active: \( \iota \in \text{Inf-FL} \implies \text{snd (concl-of } \iota) \neq \text{active} \)

begin

lemma labeled-inf-have-premises: \( \iota \in \text{Inf-FL} \implies \text{set (prems-of } \iota) \neq \{ \} \)
(proof)

definition active-subset :: \( ('f \times 'l) \rightarrow ('f \times 'l) \) set where
active-subset \( M = \{ \text{CL} \in M, \text{snd CL = active} \} \)

definition non-active-subset :: \( ('f \times 'l) \rightarrow ('f \times 'l) \) set where
non-active-subset \( M = \{ \text{CL} \in M, \text{snd CL \neq active} \} \)

inductive Given-Clause-step :: \( ('f \times 'l) \rightarrow ('f \times 'l) \rightarrow \text{bool (infix \( \implies \) GC 50)} \) where
process: \( \text{N1} = N \cup M \implies \text{N2} = N \cup M' \implies N \cap M = \{ \} \implies \)
\( \text{M} \subseteq \text{labeled-ord-red-crit-fam.\text{lifsted-calc-w-red-crit-family.\text{Red-F-Q (N \cup M')}} \implies \)
active-subset \( M' = \{ \} \implies \text{N1} \implies \text{GC N2} \mid \)
infer: \( \text{N1} = N \cup \{(C,L)\} \implies \{(C,L)\} \cap N = \{ \} \implies \text{N2} = N \cup \{(C,\text{active})\} \cup M \implies L \neq \text{active} \implies \)
active-subset \( M = \{ \} \implies \)
no-labels.Non-ground.Inf-from2 (fst (active-subset N)) \( \{C\} \subseteq \)
no-labels.lifsted-calc-w-red-crit-family.Red-Inf-Q (fst (N \cup \{(C,\text{active})\} \cup M)) \implies \)
\( \text{N1} \implies \text{GC N2} \)

abbreviation derive :: \( ('f \times 'l) \rightarrow ('f \times 'l) \rightarrow \text{bool (infix \( \triangleright \) RedL 50)} \) where
derive \( \equiv \text{labeled-ord-red-crit-fam.lifsted-calc-w-red-crit-family.inter-red-crit-calculus.derrive} \)

lemma one-step-equiv: \( \text{N1} \implies \text{GC N2} \implies \text{N1} \triangleright \text{RedL N2} \)
(proof)

abbreviation fair :: \( ('f \times 'l) \rightarrow \text{llist} \rightarrow \text{bool} \) where
fair \( \equiv \text{labeled-ord-red-crit-fam.lifsted-calc-w-red-crit-family.inter-red-crit-calculus.fair} \)

lemma gc-to-red: \( \text{chain (\( \implies \) GC) D \implies \text{chain (\( \triangleright \) RedL) D} } \)
(proof)

lemma (in-) all-ex-finite-set: \( (\forall (j::\text{nat}) \in \{0..<m\}. \exists (n::\text{nat}). \, P \, j \, n) \implies \)
\( (\forall n1 \, n2. \forall j \in \{0..<m\}. \, P \, j \, n1 \implies P \, j \, n2 \implies n1 = n2) \implies \text{finite} \{ n. \exists j \in \{0..<m\}. \, P \, j \, n \} \) for m
\( P \)
(proof)

lemma gc-fair: \( \text{chain (\( \implies \) GC) D \implies \text{llength D} > 0 \implies \text{active-subset (lnth D 0) = \{\}} \implies \)
non-active-subset \( \text{LimInf-llist D} = \{\} \implies \text{fair D} \)
(proof)

theorem gc-complete: \( \text{chain (\( \implies \) GC) D \implies \text{llength D} > 0 \implies \text{active-subset (lnth D 0) = \{\}} \implies \)
non-active-subset \( \text{LimInf-llist D} = \{\} \implies B \in \text{Bot-F} \implies \)
no-labels.entails-G-Q (fst (lnth D 0)) \( \{B\} \implies \)

23
\[ \exists i. \text{enat } i < \text{length } D \land (\exists BL \in \text{Bot-FL}. \ BL \in (\text{lnth } D \ i)) \]

(proof)

5.3 Lazy Given Clause Architecture

locale Lazy-Given-Clause = Prover-Architecture-Basis Bot-F Inf-F Bot-G Q entails-q Inf-G Red-Inf-q Red-F-q G-F-q G-Inf-q \ Inf-FL Equiv-F Prec-F Prec-l

for
Bot-F :: 'f set and
Inf-F :: 'f inference set and
Bot-G :: 'g set and
Q :: 'q itself and
entails-q :: 'q \Rightarrow ('g set \Rightarrow 'g set \Rightarrow \text{bool}) and
Inf-G :: 'g inference set and
Red-Inf-q :: 'q \Rightarrow ('g set \Rightarrow 'q 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-Inf-q :: 'q \Rightarrow 'f inference \Rightarrow 'q inference set option and
l :: 'l itself and
Inf-FL :: ('f \times 'l) inference set and
Equiv-F :: ('f \times 'f) set and
Prec-F :: 'f \Rightarrow 'f \Rightarrow \text{bool} (\text{infix } \Rightarrow 50) and
Prec-l :: 'l \Rightarrow 'l \Rightarrow \text{bool} (\text{infix } \square 50)
+ fixes
active :: 'l

assumes
active-minimal: \text{l2} \neq \text{active} \implies \text{active } \subseteq \text{l2} and
at-least-two-labels: \exists \text{l2}. \text{active } \subseteq \text{l2} and
inf-never-active: i \in \text{Inf-FL} \implies \text{snd (concl-of } i) \neq \text{active}

begin

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

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

inductive Lazy-Given-Clause-step :: ('f inference set) \times ('f \times 'l) set \Rightarrow
('f inference set) \times ('f \times 'l) set \Rightarrow \text{bool} (\text{infix } \Rightarrow \text{LGC 50}) where

process: N1 = N \cup M \implies N2 = N \cup M' \implies N \cap M = \{\} \implies
M \subseteq \text{labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.Red-F-Q } (N \cup M') \implies
active-subset M' = \{\} \implies (T,N1) \Rightarrow \text{LGC } (T,N2)

schedule-infer: T2 = T1 \cup T' \implies N1 = N \cup \{(C,L)\} \implies \{(C,L)\} \cap N = \{\} \implies N2 = N \cup \{(C,active)\} \implies
L \neq \text{active} \implies T' = \text{no-labels.Non-ground.Inf-from2 } (\text{fst } \cdot \text{ (active-subset } N)) \{C\} \implies
(T1,N1) \Rightarrow \text{LGC } (T2,N2)

compute-infer: T1 = T2 \cup \{i\} \implies T2 \cap \{i\} = \{\} \implies N2 = N1 \cup M \implies active-subset M = \{\} \implies
i \in \text{no-labels.lifted-calc-w-red-crit-family.Red-Inf-Q } (\text{fst } \cdot \text{ (active-subset } N)) \implies
(T1,N1) \Rightarrow \text{LGC } (T2,N2)

delete-orphans: T1 = T2 \cup T' \implies T2 \cap T' = \{\} \implies
T' \cap \text{no-labels.Non-ground.Inf-from } (\text{fst } \cdot \text{ (active-subset } N)) = \{\} \implies (T1,N) \Rightarrow \text{LGC } (T2,N)

abbreviation derive :: ('f \times 'l) set \Rightarrow ('f \times 'l) set \Rightarrow \text{bool} (\text{infix } \Rightarrow \text{RedL 50}) where
derive \equiv \text{labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.inter-red-crit-calculus.derive}
lemma premise-free-inf-always-from: \( \iota \in \text{Inf-F} \implies \text{length (prems-of } \iota) = 0 \implies \iota \in \text{no-labels. Non-ground. Inf-from } N \)
(proof)

lemma one-step-eqv: \((T1,N1) \implies \text{LGC} \) \( (T2,N2) \implies N1 \triangleright \text{RedL} N2 \)
(proof)

abbreviation fair :: ('f × 'l) set llist ⇒ bool where
fair ≡ labeled-ord-red-crit-fam.lifted-calc-w-red-crit-family.inter-red-crit-calculus.fair

lemma lgc-to-red: chain (⇒\text{LGC}) \( D \implies \text{chain (⇒RedL)} (\text{lmap snd } D) \)
(proof)

lemma lgc-fair: chain (⇒\text{LGC}) \( D \implies l\text{length } D > 0 \implies \text{active-subset (snd (lnth } D 0)) = \{\} \implies \text{non-active-subset (Liminf-llist (lmap snd } D)) = \{\} \implies (\forall \iota \in \text{Inf-F. length (prems-of } \iota) = 0 \implies \iota \in (\text{fst (lnth } D 0)))) \implies \text{Liminf-llist (lmap fst } D) = \{\} \implies \text{fair (lmap snd } D) \)
(proof)

theorem lgc-complete: chain (⇒\text{LGC}) \( D \implies l\text{length } D > 0 \implies \text{active-subset (snd (lnth } D 0)) = \{\} \implies \text{non-active-subset (Liminf-llist (lmap snd } D)) = \{\} \implies (\forall \iota \in \text{Inf-F. length (prems-of } \iota) = 0 \implies \iota \in (\text{fst (lnth } D 0)))) \implies \text{Liminf-llist (lmap fst } D) = \{\} \implies B \in \text{Bot-F} \implies \text{no-labels.entails-G-Q (fst ' (snd (lnth } D 0))} \{B\}
(proof)

end

end