

Extensions to the Comprehensive Framework for Saturation Theorem Proving

Jasmin Blanchette Sophie Tourret

May 26, 2024

Abstract

This Isabelle/HOL formalization extends the `Saturation_Framework` entry of the *Archive of Formal Proofs* with the following contributions:

- an application of the framework to prove Bachmair and Ganzinger’s resolution prover RP refutationally complete, which was formalized in a more ad hoc fashion by Schlichtkrull et al. in the *AFP* entry `Ordered_Resultion_Prover`;
- generalizations of various basic concepts formalized by Schlichtkrull et al., which were needed to verify RP and could be useful to formalize other calculi, such as superposition;
- alternative proofs of fairness (and hence saturation and ultimately refutational completeness) for the eager and lazy given clause procedures (GC and LGC) based on invariance.

Contents

1	Soundness	1
2	Counterexample-Reducing Inference Systems and the Standard Redundancy Criterion	2
2.1	Counterexample-Reducing Inference Systems	2
2.2	Compactness	3
2.3	The Finitary Standard Redundancy Criterion	4
2.4	The Standard Redundancy Criterion	6
2.5	Refutational Completeness	8
3	Clausal Calculi	9
3.1	Setup	9
3.2	Consequence Relation	9
3.3	Counterexample-Reducing Inference Systems	10
3.4	Counterexample-Reducing Calculi Equipped with a Standard Redundancy Criterion	10
4	Application of the Saturation Framework to Bachmair and Ganzinger’s RP	11
4.1	Setup	11
4.2	Library	11
4.3	Ground Layer	12
4.4	First-Order Layer	12
4.5	Labeled First-Order or Given Clause Layer	13

4.6	Resolution Prover Layer	14
4.7	Alternative Derivation of Previous RP Results	16
5	New Fairness Proofs for the Given Clause Prover Architectures	17
5.1	Given Clause Procedure	17
5.2	Lazy Given Clause	18

1 Soundness

theory *Soundness*

imports *Saturation-Framework.Calculus*

begin

Although consistency-preservation usually suffices, soundness is a more precise concept and is sometimes useful.

locale *sound-inference-system = inference-system + consequence-relation +*

assumes

sound: $\iota \in \text{Inf} \implies \text{set}(\text{prems-of } \iota) \models \{\text{concl-of } \iota\}$

begin

lemma *Inf-consist-preserving:*

assumes *n-cons: $\neg N \models \text{Bot}$*

shows *$\neg N \cup \text{concl-of } \iota \text{ Inf-from } N \models \text{Bot}$*

<proof>

end

The limit of a derivation based on a redundancy criterion is satisfiable if and only if the initial set is satisfiable. This material is partly based on Section 4.1 of Bachmair and Ganzinger's *Handbook* chapter, but adapted to the saturation framework of Waldmann et al.

context *calculus*

begin

The next three lemmas correspond to Lemma 4.2:

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

chain (\triangleright) $Ns \implies \text{Red-F}(\text{Sup-llist } Ns) \subseteq \text{Red-F}(\text{Liminf-llist } Ns)$

<proof>

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

chain (\triangleright) $Ns \implies \text{Red-I}(\text{Sup-llist } Ns) \subseteq \text{Red-I}(\text{Liminf-llist } Ns)$

<proof>

Proof idea due to Uwe Waldmann:

lemma *unsat-limit-iff:*

assumes

chain-red: chain (\triangleright) Ns and

chain-ent: chain (\models) Ns

shows *$\text{Liminf-llist } Ns \models \text{Bot} \iff \text{lhd } Ns \models \text{Bot}$*

<proof>

Some easy consequences:

lemma *Red-F-limit-Sup: chain (\triangleright) $Ns \implies \text{Red-F}(\text{Liminf-llist } Ns) = \text{Red-F}(\text{Sup-llist } Ns)$*

<proof>

lemma *Red-I-limit-Sup*: $chain (\triangleright) Ns \implies Red-I (Liminf-llist Ns) = Red-I (Sup-llist Ns)$
 ⟨proof⟩

end

end

2 Counterexample-Reducing Inference Systems and the Standard Redundancy Criterion

theory *Standard-Redundancy-Criterion*

imports

Saturation-Framework.Calculus

HOL-Library.Multiset-Order

begin

The standard redundancy criterion can be defined uniformly for all inference systems equipped with a compact consequence relation. The essence of the refutational completeness argument can be carried out abstractly for counterexample-reducing inference systems, which enjoy a “smallest counterexample” property. This material is partly based on Section 4.2 of Bachmair and Ganzinger’s *Handbook* chapter, but adapted to the saturation framework of Waldmann et al.

2.1 Counterexample-Reducing Inference Systems

abbreviation *main-prem-of* :: 'f inference \Rightarrow 'f **where**

main-prem-of $\iota \equiv last (prems-of \iota)$

abbreviation *side-prems-of* :: 'f inference \Rightarrow 'f list **where**

side-prems-of $\iota \equiv butlast (prems-of \iota)$

lemma *set-prems-of*:

set (prems-of ι) = (if prems-of $\iota = []$ then {} else {main-prem-of ι } \cup set (side-prems-of ι))

⟨proof⟩

locale *counterex-reducing-inference-system* = *inference-system Inf* + *consequence-relation*

for *Inf* :: 'f inference set +

fixes

I-of :: 'f set \Rightarrow 'f set **and**

less :: 'f \Rightarrow 'f \Rightarrow bool (**infix** $<$ 50)

assumes

wfP-less: *wfP* ($<$) **and**

Inf-counterex-reducing:

$N \cap Bot = \{\} \implies D \in N \implies \neg I-of N \models \{D\} \implies$

$(\bigwedge C. C \in N \implies \neg I-of N \models \{C\} \implies D < C \vee D = C) \implies$

$\exists \iota \in Inf. prems-of \iota \neq [] \wedge main-prem-of \iota = D \wedge set (side-prems-of \iota) \subseteq N \wedge$

$I-of N \models set (side-prems-of \iota) \wedge \neg I-of N \models \{concl-of \iota\} \wedge concl-of \iota < D$

begin

lemma *ex-min-counterex*:

fixes *N* :: 'f set

assumes $\neg I \models N$

shows $\exists C \in N. \neg I \models \{C\} \wedge (\forall D \in N. D \prec C \longrightarrow I \models \{D\})$
 ⟨proof⟩

end

Theorem 4.4 (generalizes Theorems 3.9 and 3.16):

locale *countereax-reducing-inference-system-with-trivial-redundancy* =
countereax-reducing-inference-system - - Inf + calculus - Inf - λ-. {} λ-. {}
for *Inf* :: 'f inference set +
assumes *less-total*: $\bigwedge C D. C \neq D \implies C \prec D \vee D \prec C$
begin

theorem *saturated-model*:

assumes
satur: *saturated N* **and**
bot-ni-n: $N \cap Bot = \{\}$
shows *I-of N* $\models N$
 ⟨proof⟩

An abstract version of Corollary 3.10 does not hold without some conditions, according to Nitpick:

corollary *saturated-complete*:

assumes
satur: *saturated N* **and**
unsat: $N \models Bot$
shows $N \cap Bot \neq \{\}$
 ⟨proof⟩

end

2.2 Compactness

locale *concl-compact-consequence-relation* = *consequence-relation* +
assumes
entails-concl-compact: $finite\ EE \implies CC \models EE \implies \exists CC' \subseteq CC. finite\ CC' \wedge CC' \models EE$
begin

lemma *entails-concl-compact-union*:

assumes
fin-e: *finite EE* **and**
cd-ent: $CC \cup DD \models EE$
shows $\exists CC' \subseteq CC. finite\ CC' \wedge CC' \cup DD \models EE$
 ⟨proof⟩

end

2.3 The Finitary Standard Redundancy Criterion

locale *finitary-standard-formula-redundancy* =
consequence-relation Bot (\models)
for
Bot :: 'f set **and**
entails :: 'f set \Rightarrow 'f set \Rightarrow bool (**infix** \models 50) +
fixes
less :: 'f \Rightarrow 'f \Rightarrow bool (**infix** \prec 50)

assumes

transp-less: $\text{transp } (\prec)$ **and**
wfP-less: $\text{wfP } (\prec)$

begin

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

$\text{Red-F } N = \{C. \exists DD \subseteq N. \text{finite } DD \wedge DD \models \{C\} \wedge (\forall D \in DD. D \prec C)\}$

The following results correspond to Lemma 4.5. The lemma *wlog-non-Red-F* generalizes the core of the argument.

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

<proof>

lemma *wlog-non-Red-F*:

assumes

dd0-fin: $\text{finite } DD0$ **and**
dd0-sub: $DD0 \subseteq N$ **and**
dd0-ent: $DD0 \cup CC \models \{E\}$ **and**
dd0-lt: $\forall D' \in DD0. D' \prec D$

shows $\exists DD \subseteq N - \text{Red-F } N. \text{finite } DD \wedge DD \cup CC \models \{E\} \wedge (\forall D' \in DD. D' \prec D)$

<proof>

lemma *Red-F-imp-ex-non-Red-F*:

assumes *c-in*: $C \in \text{Red-F } N$

shows $\exists CC \subseteq N - \text{Red-F } N. \text{finite } CC \wedge CC \models \{C\} \wedge (\forall C' \in CC. C' \prec C)$

<proof>

lemma *Red-F-sub-Red-F-diff-Red-F*: $\text{Red-F } N \subseteq \text{Red-F } (N - \text{Red-F } N)$

<proof>

lemma *Red-F-eq-Red-F-diff-Red-F*: $\text{Red-F } N = \text{Red-F } (N - \text{Red-F } N)$

<proof>

The following results correspond to Lemma 4.6.

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

<proof>

lemma *Red-F-model*: $M \models N - \text{Red-F } N \Longrightarrow M \models N$

<proof>

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

<proof>

end

locale *calculus-with-finitary-standard-redundancy* =

inference-system *Inf* + *finitary-standard-formula-redundancy* *Bot* (\models) (\prec)

for

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

Bot :: 'f set **and**

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

less :: 'f \Rightarrow 'f \Rightarrow bool (**infix** \prec 50) +

assumes

Inf-has-prem: $\iota \in \text{Inf} \Longrightarrow \text{prems-of } \iota \neq []$ **and**

Inf-reductive: $\iota \in \text{Inf} \Longrightarrow \text{concl-of } \iota \prec \text{main-prem-of } \iota$

begin

definition *redundant-infer* :: 'f set \Rightarrow 'f inference \Rightarrow bool **where**

redundant-infer $N \iota \longleftrightarrow$
($\exists DD \subseteq N$. finite $DD \wedge DD \cup \text{set}(\text{side-prems-of } \iota) \models \{\text{concl-of } \iota\} \wedge (\forall D \in DD. D \prec \text{main-prem-of } \iota)$)

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

Red-I $N = \{\iota \in \text{Inf. } \text{redundant-infer } N \iota\}$

The following results correspond to Lemma 4.6. It also uses *wlog-non-Red-F*.

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

<proof>

lemma *Red-I-sub-Red-I-diff-Red-F*: $\text{Red-I } N \subseteq \text{Red-I } (N - \text{Red-F } N)$

<proof>

lemma *Red-I-eq-Red-I-diff-Red-F*: $\text{Red-I } N = \text{Red-I } (N - \text{Red-F } N)$

<proof>

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

<proof>

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

<proof>

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

assumes

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

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

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

<proof>

The following corresponds to Theorems 4.7 and 4.8:

sublocale *calculus Bot Inf* (\models) *Red-I Red-F*

<proof>

end

2.4 The Standard Redundancy Criterion

locale *standard-formula-redundancy* =

concl-compact-consequence-relation Bot (\models)

for

Bot :: 'f set **and**

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

fixes

less :: 'f \Rightarrow 'f \Rightarrow bool (**infix** \prec 50)

assumes

transp-less: *transp* (\prec) **and**

wfP-less: *wfP* (\prec)

begin

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

Red-F $N = \{C. \exists DD \subseteq N. DD \models \{C\} \wedge (\forall D \in DD. D \prec C)\}$

Compactness of (\models) implies that $Red-F$ is equivalent to its finitary counterpart.

interpretation *fin-std-red-F*: *finitary-standard-formula-redundancy Bot* (\models) (\prec)
 $\langle proof \rangle$

lemma *Red-F-conv*: $Red-F = fin-std-red-F.Red-F$
 $\langle proof \rangle$

The results from *finitary-standard-formula-redundancy* can now be lifted.

The following results correspond to Lemma 4.5.

lemma *Red-F-of-subset*: $N \subseteq N' \implies Red-F N \subseteq Red-F N'$
 $\langle proof \rangle$

lemma *Red-F-imp-ex-non-Red-F*: $C \in Red-F N \implies \exists CC \subseteq N - Red-F N. CC \models \{C\} \wedge (\forall C' \in CC. C' \prec C)$
 $\langle proof \rangle$

lemma *Red-F-sub-Red-F-diff-Red-F*: $Red-F N \subseteq Red-F (N - Red-F N)$
 $\langle proof \rangle$

lemma *Red-F-eq-Red-F-diff-Red-F*: $Red-F N = Red-F (N - Red-F N)$
 $\langle proof \rangle$

The following results correspond to Lemma 4.6.

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

lemma *Red-F-model*: $M \models N - Red-F N \implies M \models N$
 $\langle proof \rangle$

lemma *Red-F-Bot*: $B \in Bot \implies N \models \{B\} \implies N - Red-F N \models \{B\}$
 $\langle proof \rangle$

end

locale *calculus-with-standard-redundancy* =
inference-system Inf + *standard-formula-redundancy Bot* (\models) (\prec)

for

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

Bot :: 'f set **and**

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

less :: 'f \Rightarrow 'f \Rightarrow bool (**infix** \prec 50) +

assumes

Inf-has-prem: $\iota \in Inf \implies prems-of \iota \neq []$ **and**

Inf-reductive: $\iota \in Inf \implies concl-of \iota \prec main-prem-of \iota$

begin

definition *redundant-infer* :: 'f set \Rightarrow 'f inference \Rightarrow bool **where**

redundant-infer $N \iota \longleftrightarrow$

$(\exists DD \subseteq N. DD \cup set (side-prems-of \iota) \models \{concl-of \iota\} \wedge (\forall D \in DD. D \prec main-prem-of \iota))$

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

Red-I $N = \{\iota \in Inf. redundant-infer N \iota\}$

Compactness of (\models) implies that *Red-I* is equivalent to its finitary counterpart.

interpretation *fin-std-red*: *calculus-with-finitary-standard-redundancy* *Inf Bot* (\models)
 ⟨*proof*⟩

lemma *redundant-infer-conv*: *redundant-infer* = *fin-std-red.redundant-infer*
 ⟨*proof*⟩

lemma *Red-I-conv*: *Red-I* = *fin-std-red.Red-I*
 ⟨*proof*⟩

The results from *calculus-with-finitary-standard-redundancy* can now be lifted.
 The following results correspond to Lemma 4.6.

lemma *Red-I-of-subset*: $N \subseteq N' \implies \text{Red-I } N \subseteq \text{Red-I } N'$
 ⟨*proof*⟩

lemma *Red-I-sub-Red-I-diff-Red-F*: $\text{Red-I } N \subseteq \text{Red-I } (N - \text{Red-F } N)$
 ⟨*proof*⟩

lemma *Red-I-eq-Red-I-diff-Red-F*: $\text{Red-I } N = \text{Red-I } (N - \text{Red-F } N)$
 ⟨*proof*⟩

lemma *Red-I-to-Inf*: $\text{Red-I } N \subseteq \text{Inf}$
 ⟨*proof*⟩

lemma *Red-I-of-Red-F-subset*: $N' \subseteq \text{Red-F } N \implies \text{Red-I } N \subseteq \text{Red-I } (N - N')$
 ⟨*proof*⟩

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

The following corresponds to Theorems 4.7 and 4.8:

sublocale *calculus Bot Inf* (\models) *Red-I Red-F*
 ⟨*proof*⟩

end

2.5 Refutational Completeness

locale *calculus-with-standard-inference-redundancy* = *calculus Bot Inf* (\models) *Red-I Red-F*

for *Bot* :: '*f set and Inf and entails* (*infix* \models 50) **and** *Red-I and Red-F* +

fixes

less :: '*f* \implies '*f* \implies *bool* (*infix* \prec 50)

assumes

Inf-has-prem: $\iota \in \text{Inf} \implies \text{prems-of } \iota \neq []$ **and**

Red-I-imp-redundant-infer: $\iota \in \text{Red-I } N \implies$

$(\exists DD \subseteq N. DD \cup \text{set } (\text{side-prems-of } \iota) \models \{\text{concl-of } \iota\} \wedge (\forall C \in DD. C \prec \text{main-prem-of } \iota))$

sublocale *calculus-with-finitary-standard-redundancy* \subseteq
calculus-with-standard-inference-redundancy Bot Inf (\models) *Red-I Red-F*
 ⟨*proof*⟩

sublocale *calculus-with-standard-redundancy* \subseteq
calculus-with-standard-inference-redundancy Bot Inf (\models) *Red-I Red-F*
 ⟨*proof*⟩


```

locale counterex-reducing-calculus-with-standard-inference-redundancy =
  calculus-with-standard-inference-redundancy Bot Inf ( $\models$ ) Red-I Red-F ( $\prec$ ) +
  counterex-reducing-inference-system Bot ( $\models$ ) Inf I-of ( $\prec$ )
for
  Bot :: 'f set and
  Inf :: 'f inference set and
  entails :: 'f set  $\Rightarrow$  'f set  $\Rightarrow$  bool (infix  $\models$  50) and
  Red-I :: 'f set  $\Rightarrow$  'f inference set and
  Red-F :: 'f set  $\Rightarrow$  'f set and
  I-of :: 'f set  $\Rightarrow$  'f set and
  less :: 'f  $\Rightarrow$  'f  $\Rightarrow$  bool (infix  $\prec$  50) +
assumes less-total:  $\bigwedge C D. C \neq D \Rightarrow C \prec D \vee D \prec C$ 
begin

```

The following result loosely corresponds to Theorem 4.9.

lemma *saturated-model*:

```

assumes
  satur: saturated N and
  bot-ni-n:  $N \cap Bot = \{\}$ 
shows I-of N  $\models$  N
<proof>

```

A more faithful abstract version of Theorem 4.9 does not hold without some conditions, according to Nitpick:

corollary *saturated-complete*:

```

assumes
  satur: saturated N and
  unsat:  $N \models Bot$ 
shows  $N \cap Bot \neq \{\}$ 
<proof>

```

end

end

3 Clausal Calculi

theory *Clausal-Calculus*

```

imports
  Ordered-Resolution-Prover.Unordered-Ground-Resolution
  Soundness
  Standard-Redundancy-Criterion

```

begin

Various results about consequence relations, counterexample-reducing inference systems, and the standard redundancy criteria are specialized and customized for clauses as opposed to arbitrary formulas.

3.1 Setup

To avoid confusion, we use the symbol \models (with or without subscripts) for the “models” and entailment relations on clauses and \models for the abstract concept of consequence.

abbreviation *true-lit-thick* :: 'a interp \Rightarrow 'a literal \Rightarrow bool (**infix** \models l 50) **where**

$I \models_l L \equiv I \models L$

abbreviation *true-cls-thick* :: 'a interp \Rightarrow 'a clause \Rightarrow bool (**infix** \models 50) **where**

$I \models C \equiv I \models C$

abbreviation *true-cls-thick* :: 'a interp \Rightarrow 'a clause set \Rightarrow bool (**infix** \models_s 50) **where**

$I \models_s C \equiv I \models_s C$

abbreviation *true-cls-mset-thick* :: 'a interp \Rightarrow 'a clause multiset \Rightarrow bool (**infix** \models_m 50) **where**

$I \models_m C \equiv I \models_m C$

no-notation *true-lit* (**infix** \models_l 50)

no-notation *true-cls* (**infix** \models 50)

no-notation *true-cls* (**infix** \models_s 50)

no-notation *true-cls-mset* (**infix** \models_m 50)

3.2 Consequence Relation

abbreviation *entails-cls* :: 'a clause set \Rightarrow 'a clause set \Rightarrow bool (**infix** \models_e 50) **where**

$N1 \models_e N2 \equiv \forall I. I \models_s N1 \longrightarrow I \models_s N2$

lemma *entails-iff-unsatisfiable-single*:

$CC \models_e \{E\} \longleftrightarrow \neg \text{satisfiable} (CC \cup \{\{\#- L\# \mid L. L \in \# E\}\})$ (**is** $- \longleftrightarrow - (- \cup ?NegD)$)
 $\langle \text{proof} \rangle$

lemma *entails-iff-unsatisfiable*:

$CC \models_e EE \longleftrightarrow (\forall E \in EE. \neg \text{satisfiable} (CC \cup \{\{\#- L\# \mid L. L \in \# E\}\}))$ (**is** $?lhs = ?rhs$)
 $\langle \text{proof} \rangle$

interpretation *consequence-relation* $\{\{\#\}\}$ (\models_e)

$\langle \text{proof} \rangle$

interpretation *concl-compact-consequence-relation* $\{\{\#\}\}$:: ('a :: wellorder) clause set (\models_e)

$\langle \text{proof} \rangle$

3.3 Counterexample-Reducing Inference Systems

definition *cls-of-interp* :: 'a set \Rightarrow 'a literal multiset set **where**

cls-of-interp $I = \{\{\#\}(\text{if } A \in I \text{ then Pos else Neg}) A\# \mid A. \text{True}\}$

lemma *true-cls-of-interp-iff-equal[simp]*: $J \models_s \text{cls-of-interp } I \longleftrightarrow J = I$

$\langle \text{proof} \rangle$

lemma *entails-iff-models[simp]*: $\text{cls-of-interp } I \models_e CC \longleftrightarrow I \models_s CC$

$\langle \text{proof} \rangle$

locale *clausal-counterex-reducing-inference-system* = *inference-system* *Inf*

for *Inf* :: ('a :: wellorder) clause inference set +

fixes *J-of* :: 'a clause set \Rightarrow 'a interp

assumes *clausal-Inf-counterex-reducing*:

$\{\#\} \notin N \Longrightarrow D \in N \Longrightarrow \neg J\text{-of } N \models D \Longrightarrow (\bigwedge C. C \in N \Longrightarrow \neg J\text{-of } N \models C \Longrightarrow D \leq C) \Longrightarrow$
 $\exists \iota \in \text{Inf}. \text{prems-of } \iota \neq [] \wedge \text{main-prem-of } \iota = D \wedge \text{set}(\text{side-prems-of } \iota) \subseteq N \wedge$
 $J\text{-of } N \models_s \text{set}(\text{side-prems-of } \iota) \wedge \neg J\text{-of } N \models \text{concl-of } \iota \wedge \text{concl-of } \iota < D$

begin

abbreviation *I-of* :: 'a clause set \Rightarrow 'a clause set **where**

$I\text{-of } N \equiv \text{cls-of-interp } (J\text{-of } N)$

lemma *Inf-counterex-reducing*:

assumes

bot-ni-n: $N \cap \{\#\} = \{\}$ **and**

d-in-n: $D \in N$ **and**

n-ent-d: $\neg I\text{-of } N \models_e \{D\}$ **and**

d-min: $\bigwedge C. C \in N \implies \neg I\text{-of } N \models_e \{C\} \implies D \leq C$

shows $\exists \iota \in \text{Inf}. \text{prems-of } \iota \neq \square \wedge \text{main-prem-of } \iota = D \wedge \text{set } (\text{side-prems-of } \iota) \subseteq N$

$\wedge I\text{-of } N \models_e \text{set } (\text{side-prems-of } \iota) \wedge \neg I\text{-of } N \models_e \{\text{concl-of } \iota\} \wedge \text{concl-of } \iota < D$

<proof>

sublocale *counterex-reducing-inference-system* $\{\#\}$ (\models_e) *Inf I-of*

< :: $'a \text{ clause} \implies 'a \text{ clause} \implies \text{bool}$

<proof>

end

3.4 Counterexample-Reducing Calculi Equipped with a Standard Redundancy Criterion

locale *clausal-counterex-reducing-calculus-with-standard-redundancy* =

calculus-with-standard-redundancy Inf $\{\#\}$ (\models_e) *<* :: $'a \text{ clause} \implies 'a \text{ clause} \implies \text{bool}$ +

clausal-counterex-reducing-inference-system Inf J-of

for

Inf :: $('a :: \text{wellorder}) \text{ clause inference set}$ **and**

J-of :: $'a \text{ clause set} \implies 'a \text{ set}$

begin

sublocale *counterex-reducing-calculus-with-standard-inference-redundancy* $\{\#\}$ *Inf* (\models_e) *Red-I*

Red-F I-of *<* :: $'a \text{ clause} \implies 'a \text{ clause} \implies \text{bool}$

<proof>

lemma *clausal-saturated-model*: $\text{saturated } N \implies \{\#\} \notin N \implies J\text{-of } N \models_s N$

<proof>

corollary *clausal-saturated-complete*: $\text{saturated } N \implies (\forall I. \neg I \models_s N) \implies \{\#\} \in N$

<proof>

end

end

4 Application of the Saturation Framework to Bachmair and Ganzinger's RP

theory *FO-Ordered-Resolution-Prover-Revisited*

imports

Ordered-Resolution-Prover.FO-Ordered-Resolution-Prover

Saturation-Framework.Given-Clause-Architectures

Clausal-Calculus

Soundness

begin

The main results about Bachmair and Ganzinger's RP prover, as established in Section 4.3

of their *Handbook* chapter and formalized by Schlichtkrull et al., are re-proved here using the saturation framework of Waldmann et al.

4.1 Setup

no-notation *true-lit* (**infix** \models_l 50)
no-notation *true-cls* (**infix** \models 50)
no-notation *true-cls* (**infix** \models_s 50)
no-notation *true-cls-mset* (**infix** \models_m 50)

hide-type (**open**) *Inference-System.inference*

hide-const (**open**) *Inference-System.Infer Inference-System.main-prem-of
Inference-System.side-prems-of Inference-System.prem-s-of Inference-System.concl-of
Inference-System.concls-of Inference-System.infer-from*

type-synonym 'a *old-inference* = 'a *Inference-System.inference*

abbreviation *old-Infer* \equiv *Inference-System.Infer*
abbreviation *old-side-prems-of* \equiv *Inference-System.side-prems-of*
abbreviation *old-main-prem-of* \equiv *Inference-System.main-prem-of*
abbreviation *old-concl-of* \equiv *Inference-System.concl-of*
abbreviation *old-prems-of* \equiv *Inference-System.prem-s-of*
abbreviation *old-concls-of* \equiv *Inference-System.concls-of*
abbreviation *old-infer-from* \equiv *Inference-System.infer-from*

lemmas *old-infer-from-def* = *Inference-System.infer-from-def*

4.2 Library

lemma *set-zip-replicate-right[simp]*:
set (zip xs (replicate (length xs) y)) = ($\lambda x. (x, y)$) ' set xs
<proof>

4.3 Ground Layer

context *FO-resolution-prover*
begin

no-notation *RP* (**infix** \rightsquigarrow 50)
notation *RP* (**infix** \rightsquigarrow_{RP} 50)

interpretation *gr*: *ground-resolution-with-selection S-M S M*
<proof>

definition *G-Inf* :: 'a *clause set* \Rightarrow 'a *clause inference set* **where**
G-Inf M = {Infer (CAs @ [DA]) E | CAs DA AAs As E. gr.ord-resolve M CAs DA AAs As E}

lemma *G-Inf-have-prems*: $\iota \in G-Inf M \Rightarrow$ *prems-of* $\iota \neq []$
<proof>

lemma *G-Inf-reductive*: $\iota \in G-Inf M \Rightarrow$ *concl-of* $\iota <$ *main-prem-of* ι
<proof>

interpretation *G*: *sound-inference-system G-Inf M {{#}} (\models_e)*

⟨proof⟩

interpretation G : clausal-counterex-reducing-inference-system $G\text{-Inf } M \text{ gr.INTERP } M$
⟨proof⟩

interpretation G : clausal-counterex-reducing-calculus-with-standard-redundancy $G\text{-Inf } M$
 $\text{gr.INTERP } M$
⟨proof⟩

interpretation G : statically-complete-calculus $\{\{\#\}\}$ $G\text{-Inf } M (\models_e) G\text{-Red-I } M G\text{-Red-F}$
⟨proof⟩

4.4 First-Order Layer

abbreviation $\mathcal{G}\text{-F} :: \langle 'a \text{ clause} \Rightarrow 'a \text{ clause set} \rangle$ **where**
 $\langle \mathcal{G}\text{-F} \equiv \text{grounding-of-cls} \rangle$

abbreviation $\mathcal{G}\text{-Fset} :: \langle 'a \text{ clause set} \Rightarrow 'a \text{ clause set} \rangle$ **where**
 $\langle \mathcal{G}\text{-Fset} \equiv \text{grounding-of-clss} \rangle$

lemmas $\mathcal{G}\text{-F-def} = \text{grounding-of-cls-def}$

lemmas $\mathcal{G}\text{-Fset-def} = \text{grounding-of-clss-def}$

definition $\mathcal{G}\text{-I} :: \langle 'a \text{ clause set} \Rightarrow 'a \text{ clause inference} \Rightarrow 'a \text{ clause inference set} \rangle$ **where**
 $\langle \mathcal{G}\text{-I } M \ \iota = \{ \text{Infer } (\text{prems-of } \iota \cdot \text{cl } \varrho\text{s}) (\text{concl-of } \iota \cdot \varrho) \mid \varrho \ \varrho\text{s}. \text{is-ground-subst-list } \varrho\text{s} \wedge \text{is-ground-subst } \varrho$
 $\wedge \text{Infer } (\text{prems-of } \iota \cdot \text{cl } \varrho\text{s}) (\text{concl-of } \iota \cdot \varrho) \in G\text{-Inf } M \} \rangle$

abbreviation

$\mathcal{G}\text{-I-opt} :: \langle 'a \text{ clause set} \Rightarrow 'a \text{ clause inference} \Rightarrow 'a \text{ clause inference set option} \rangle$

where

$\langle \mathcal{G}\text{-I-opt } M \ \iota \equiv \text{Some } (G\text{-I } M \ \iota) \rangle$

definition $F\text{-Inf} :: 'a \text{ clause inference set}$ **where**

$F\text{-Inf} = \{ \text{Infer } (CAs @ [DA]) E \mid CAs \ DA \ AAs \ As \ \sigma \ E. \text{ord-resolve-rename } S \ CAs \ DA \ AAs \ As \ \sigma \ E \}$

lemma $F\text{-Inf-have-prems}$: $\iota \in F\text{-Inf} \implies \text{prems-of } \iota \neq []$
⟨proof⟩

interpretation F : lifting-intersection $F\text{-Inf} \ \{\{\#\}\} \text{ UNIV } G\text{-Inf } \lambda N. (\models_e) G\text{-Red-I } \lambda N. G\text{-Red-F}$
 $\ \{\{\#\}\} \lambda N. \mathcal{G}\text{-F } \mathcal{G}\text{-I-opt } \lambda D \ C \ C'. \text{False}$
⟨proof⟩

notation $F.\text{entails-}\mathcal{G}$ (infix $\models_{\mathcal{G}} e$ 50)

lemma $F\text{-entails-}\mathcal{G}\text{-iff}$: $N1 \models_{\mathcal{G}} e N2 \iff \bigcup (G\text{-F } ' N1) \models_e \bigcup (G\text{-F } ' N2)$
⟨proof⟩

lemma $\text{true-Union-grounding-of-cls-iff}$:

$I \models_s (\bigcup C \in N. \{ C \cdot \sigma \mid \sigma. \text{is-ground-subst } \sigma \}) \iff (\forall \sigma. \text{is-ground-subst } \sigma \implies I \models_s N \cdot \text{cs } \sigma)$
⟨proof⟩

interpretation F : sound-inference-system $F\text{-Inf} \ \{\{\#\}\} (\models_{\mathcal{G}} e)$
⟨proof⟩

lemma $G\text{-Inf-overapprox-F-Inf}$: $\iota_0 \in G\text{-Inf-from } M (\bigcup (G\text{-F } ' M)) \implies \exists \iota \in F\text{-Inf-from } M. \iota_0 \in \mathcal{G}\text{-I}$

$M \iota$
 $\langle \text{proof} \rangle$

interpretation F : *statically-complete-calculus* $\{\{\#\}\}$ $F\text{-Inf}$ ($\models_{\mathcal{G}e}$) $F\text{-Red-I-}\mathcal{G}$ $F\text{-Red-F-}\mathcal{G}\text{-empty}$
 $\langle \text{proof} \rangle$

4.5 Labeled First-Order or Given Clause Layer

datatype $\text{label} = \text{New} \mid \text{Processed} \mid \text{Old}$

abbreviation $F\text{-Equiv} :: 'a \text{ clause} \Rightarrow 'a \text{ clause} \Rightarrow \text{bool}$ (**infix** \doteq 50) **where**
 $C \doteq D \equiv \text{generalizes } C D \wedge \text{generalizes } D C$

abbreviation $F\text{-Prec} :: 'a \text{ clause} \Rightarrow 'a \text{ clause} \Rightarrow \text{bool}$ (**infix** \prec 50) **where**
 $C \prec D \equiv \text{strictly-generalizes } C D$

fun $L\text{-Prec} :: \text{label} \Rightarrow \text{label} \Rightarrow \text{bool}$ (**infix** \sqsubset 50) **where**
 $\text{Old} \sqsubset l \longleftrightarrow l \neq \text{Old}$
 $\mid \text{Processed} \sqsubset l \longleftrightarrow l = \text{New}$
 $\mid \text{New} \sqsubset l \longleftrightarrow \text{False}$

lemma $\text{irrefl-}L\text{-Prec}: \neg l \sqsubset l$
 $\langle \text{proof} \rangle$

lemma $\text{trans-}L\text{-Prec}: l1 \sqsubset l2 \Longrightarrow l2 \sqsubset l3 \Longrightarrow l1 \sqsubset l3$
 $\langle \text{proof} \rangle$

lemma $\text{wf-}L\text{-Prec}: \text{wfP } (\sqsubset)$
 $\langle \text{proof} \rangle$

interpretation FL : *given-clause* $\{\{\#\}\}$ $F\text{-Inf}$ $\{\{\#\}\}$ $UNIV \lambda N. (\models_e)$ $G\text{-Inf}$ $G\text{-Red-I}$
 $\lambda N. G\text{-Red-F}$ $\lambda N. \mathcal{G}\text{-F}$ $\mathcal{G}\text{-I-opt}$ (\doteq) (\prec) (\sqsubset) Old
 $\langle \text{proof} \rangle$

notation $FL\text{-Prec-}FL$ (**infix** \sqsubset 50)
notation $FL\text{-entails-}\mathcal{G}\text{-L}$ (**infix** $\models_{\mathcal{G}Le}$ 50)
notation $FL\text{-derive}$ (**infix** \triangleright_L 50)
notation $FL\text{-step}$ (**infix** \sim_{GC} 50)

lemma $FL\text{-Red-F-eq}$:
 $FL\text{-Red-F } N =$
 $\{C. \forall D \in \mathcal{G}\text{-F } (fst C). D \in G\text{-Red-F } (\bigcup (\mathcal{G}\text{-F } 'fst' 'N)) \vee (\exists E \in N. E \sqsubset C \wedge D \in \mathcal{G}\text{-F } (fst E))\}$
 $\langle \text{proof} \rangle$

lemma $\text{mem-}FL\text{-Red-F-because-}G\text{-Red-F}$:
 $(\forall D \in \mathcal{G}\text{-F } (fst Cl). D \in G\text{-Red-F } (\bigcup (\mathcal{G}\text{-F } 'fst' 'N))) \Longrightarrow Cl \in FL\text{-Red-F } N$
 $\langle \text{proof} \rangle$

lemma $\text{mem-}FL\text{-Red-F-because-}Prec\text{-}FL$:
 $(\forall D \in \mathcal{G}\text{-F } (fst Cl). \exists El \in N. El \sqsubset Cl \wedge D \in \mathcal{G}\text{-F } (fst El)) \Longrightarrow Cl \in FL\text{-Red-F } N$
 $\langle \text{proof} \rangle$

4.6 Resolution Prover Layer

interpretation sq : *selection* $S\text{-Q}$ Sts
 $\langle \text{proof} \rangle$

interpretation *gd*: *ground-resolution-with-selection S-Q Sts*
 ⟨proof⟩

interpretation *src*: *standard-redundancy-criterion-counterex-reducing gd.ord-Γ Sts*
ground-resolution-with-selection.INTERP (S-Q Sts)
 ⟨proof⟩

definition *lclss-of-state* :: 'a state \Rightarrow ('a clause \times label) set **where**
lclss-of-state St =
 ($\lambda C. (C, \text{New})$) ' N-of-state St \cup ($\lambda C. (C, \text{Processed})$) ' P-of-state St
 \cup ($\lambda C. (C, \text{Old})$) ' Q-of-state St

lemma *image-hd-lclss-of-state[simp]*: *fst ' lclss-of-state St = clss-of-state St*
 ⟨proof⟩

lemma *insert-lclss-of-state[simp]*:
insert (C, New) (lclss-of-state (N, P, Q)) = lclss-of-state (N \cup {C}, P, Q)
insert (C, Processed) (lclss-of-state (N, P, Q)) = lclss-of-state (N, P \cup {C}, Q)
insert (C, Old) (lclss-of-state (N, P, Q)) = lclss-of-state (N, P, Q \cup {C})
 ⟨proof⟩

lemma *union-lclss-of-state[simp]*:
lclss-of-state (N1, P1, Q1) \cup lclss-of-state (N2, P2, Q2) =
lclss-of-state (N1 \cup N2, P1 \cup P2, Q1 \cup Q2)
 ⟨proof⟩

lemma *mem-lclss-of-state[simp]*:
(C, New) \in lclss-of-state (N, P, Q) \iff C \in N
(C, Processed) \in lclss-of-state (N, P, Q) \iff C \in P
(C, Old) \in lclss-of-state (N, P, Q) \iff C \in Q
 ⟨proof⟩

lemma *lclss-Liminf-commute*:
Liminf-list (lmap lclss-of-state Sts) = lclss-of-state (Liminf-state Sts)
 ⟨proof⟩

lemma *GC-tautology-step*:
assumes *tauto*: *Neg A \in # C Pos A \in # C*
shows *lclss-of-state (N \cup {C}, P, Q) \rightsquigarrow GC lclss-of-state (N, P, Q)*
 ⟨proof⟩

lemma *GC-subsumption-step*:
assumes
d-in: *Dl \in N* **and**
d-sub-c: *strictly-subsumes (fst Dl) (fst Cl) \vee subsumes (fst Dl) (fst Cl) \wedge snd Dl \sqsubseteq l snd Cl*
shows *N \cup {Cl} \rightsquigarrow GC N*
 ⟨proof⟩

lemma *GC-reduction-step*:
assumes
young: *snd Dl \neq Old* **and**
d-sub-c: *fst Dl \subset # fst Cl*
shows *N \cup {Cl} \rightsquigarrow GC N \cup {Dl}*
 ⟨proof⟩

lemma *GC-processing-step*: $N \cup \{(C, \text{New})\} \rightsquigarrow_{GC} N \cup \{(C, \text{Processed})\}$
 ⟨proof⟩

lemma *old-inferences-between-eq-new-inferences-between*:
 old-concl-of ‘inference-system.inferences-between (ord-FO- Γ S) N C =
 concl-of ‘F.Inf-between N {C} (is ?rp = ?f)
 ⟨proof⟩

lemma *GC-inference-step*:
assumes
 young: $l \neq \text{Old}$ **and**
 no-active: $FL.\text{active-subset } M = \{\}$ **and**
 m-sup: $\text{fst } M \supseteq \text{old-concl-of } \text{inference-system.inferences-between (ord-FO-}\Gamma \text{ S)}$
 ($\text{fst } FL.\text{active-subset } N$) C
shows $N \cup \{(C, l)\} \rightsquigarrow_{GC} N \cup \{(C, \text{Old})\} \cup M$
 ⟨proof⟩

lemma *RP-step-imp-GC-step*: $St \rightsquigarrow_{RP} St' \implies \text{lclss-of-state } St \rightsquigarrow_{GC} \text{lclss-of-state } St'$
 ⟨proof⟩

lemma *RP-derivation-imp-GC-derivation*: $\text{chain } (\rightsquigarrow_{RP}) \text{ Sts} \implies \text{chain } (\rightsquigarrow_{GC}) (\text{lmap } \text{lclss-of-state } \text{Sts})$
 ⟨proof⟩

lemma *RP-step-imp-derive-step*: $St \rightsquigarrow_{RP} St' \implies \text{lclss-of-state } St \triangleright_L \text{lclss-of-state } St'$
 ⟨proof⟩

lemma *RP-derivation-imp-derive-derivation*:
 $\text{chain } (\rightsquigarrow_{RP}) \text{ Sts} \implies \text{chain } (\triangleright_L) (\text{lmap } \text{lclss-of-state } \text{Sts})$
 ⟨proof⟩

theorem *RP-sound-new-statement*:
assumes
 deriv: $\text{chain } (\rightsquigarrow_{RP}) \text{ Sts}$ **and**
 bot-in: $\{\#\} \in \text{clss-of-state } (Liminf\text{-state } \text{Sts})$
shows $\text{clss-of-state } (\text{lhd } \text{Sts}) \models_{\mathcal{G}e} \{\#\}$
 ⟨proof⟩

theorem *RP-saturated-if-fair-new-statement*:
assumes
 deriv: $\text{chain } (\rightsquigarrow_{RP}) \text{ Sts}$ **and**
 init: $FL.\text{active-subset } (\text{lclss-of-state } (\text{lhd } \text{Sts})) = \{\}$ **and**
 final: $FL.\text{passive-subset } (Liminf\text{-llist } (\text{lmap } \text{lclss-of-state } \text{Sts})) = \{\}$
shows $FL.\text{saturated } (Liminf\text{-llist } (\text{lmap } \text{lclss-of-state } \text{Sts}))$
 ⟨proof⟩

corollary *RP-complete-if-fair-new-statement*:
assumes
 deriv: $\text{chain } (\rightsquigarrow_{RP}) \text{ Sts}$ **and**
 init: $FL.\text{active-subset } (\text{lclss-of-state } (\text{lhd } \text{Sts})) = \{\}$ **and**
 final: $FL.\text{passive-subset } (Liminf\text{-llist } (\text{lmap } \text{lclss-of-state } \text{Sts})) = \{\}$ **and**
 unsat: $\text{grounding-of-state } (\text{lhd } \text{Sts}) \models_e \{\#\}$
shows $\{\#\} \in Q\text{-of-state } (Liminf\text{-state } \text{Sts})$
 ⟨proof⟩

4.7 Alternative Derivation of Previous RP Results

lemma *old-fair-imp-new-fair*:

assumes

nnul: $\neg \text{lnull } Sts$ **and**

fair: *fair-state-seq* *Sts* **and**

empty-Q0: *Q-of-state* (*lhd Sts*) = {}

shows

FL.active-subset (*lclss-of-state* (*lhd Sts*)) = {} **and**

FL.passive-subset (*Liminf-llist* (*lmap lclss-of-state Sts*)) = {}

<proof>

lemma *old-redundant-infer-iff*:

src.redundant-infer *N* $\gamma \longleftrightarrow$

($\exists DD. DD \subseteq N \wedge DD \cup \text{set-mset}(\text{old-side-prems-of } \gamma) \Vdash_e \{\text{old-concl-of } \gamma\}$
 $\wedge (\forall D \in DD. D < \text{old-main-prem-of } \gamma)$)

(**is** ?lhs \longleftrightarrow ?rhs)

<proof>

definition *old-infer-of* :: 'a clause inference \Rightarrow 'a old-inference **where**

old-infer-of $\iota = \text{old-Infer}(\text{mset}(\text{side-prems-of } \iota))(\text{main-prem-of } \iota)(\text{concl-of } \iota)$

lemma *new-redundant-infer-imp-old-redundant-infer*:

G.redundant-infer *N* $\iota \Longrightarrow \text{src.redundant-infer } N(\text{old-infer-of } \iota)$

<proof>

lemma *saturated-imp-saturated-RP*:

assumes

satur: *FL.saturated* (*Liminf-llist* (*lmap lclss-of-state Sts*)) **and**

no-passive: *FL.passive-subset* (*Liminf-llist* (*lmap lclss-of-state Sts*)) = {}

shows *src.saturated-upto* *Sts* (*grounding-of-state* (*Liminf-state Sts*))

<proof>

theorem *RP-sound-old-statement*:

assumes

deriv: *chain* ($\rightsquigarrow RP$) *Sts* **and**

bot-in: {#} $\in \text{class-of-state}(\text{Liminf-state } Sts)$

shows $\neg \text{satisfiable}(\text{grounding-of-state}(\text{lhd } Sts))$

<proof>

The theorem below is stated differently than the original theorem in RP: The grounding of the limit might be a strict subset of the limit of the groundings. Because saturation is neither monotone nor antimonotone, the two results are incomparable. See also *grounding-of-state-Liminf-state-subseteq*.

theorem *RP-saturated-if-fair-old-statement-altered*:

assumes

deriv: *chain* ($\rightsquigarrow RP$) *Sts* **and**

fair: *fair-state-seq* *Sts* **and**

empty-Q0: *Q-of-state* (*lhd Sts*) = {}

shows *src.saturated-upto* *Sts* (*grounding-of-state* (*Liminf-state Sts*))

<proof>

corollary *RP-complete-if-fair-old-statement*:

assumes

deriv: *chain* ($\rightsquigarrow RP$) *Sts* **and**

fair: *fair-state-seq* *Sts* **and**

empty-Q0: *Q-of-state* (*lhd Sts*) = {} **and**

$unsat: \neg \text{satisfiable (grounding-of-state (lhd Sts))}$
shows $\{\#\} \in Q\text{-of-state (Liminf-state Sts)}$
 $\langle \text{proof} \rangle$

end

end

5 New Fairness Proofs for the Given Clause Prover Architectures

theory *Given-Clause-Architectures-Revisited*
imports *Saturation-Framework.Given-Clause-Architectures*
begin

The given clause and lazy given clause procedures satisfy key invariants. This provides an alternative way to prove fairness and hence saturation of the limit.

5.1 Given Clause Procedure

context *given-clause*
begin

definition $gc\text{-invar} :: ('f \times 'l) \text{ set llist} \Rightarrow \text{enat} \Rightarrow \text{bool}$ **where**
 $gc\text{-invar } Ns \ i \longleftrightarrow$
 $\text{Inf-from (active-subset (Liminf-upto-llist } Ns \ i)) \subseteq \text{Sup-upto-llist (lmap Red-I-G } Ns) \ i}$

lemma *gc-invar-infinity*:
assumes
 $nnil: \neg \text{lnull } Ns$ **and**
 $\text{invar}: \forall i. \text{enat } i < \text{llength } Ns \longrightarrow gc\text{-invar } Ns \ (\text{enat } i)$
shows $gc\text{-invar } Ns \ \infty$
 $\langle \text{proof} \rangle$

lemma *gc-invar-gc-init*:
assumes
 $\neg \text{lnull } Ns$ **and**
 $\text{active-subset (lhd } Ns) = \{\}$
shows $gc\text{-invar } Ns \ 0$
 $\langle \text{proof} \rangle$

lemma *gc-invar-gc-step*:
assumes
 $\text{Si-lt}: \text{enat } (\text{Suc } i) < \text{llength } Ns$ **and**
 $\text{invar}: gc\text{-invar } Ns \ i$ **and**
 $\text{step}: \text{lth } Ns \ i \rightsquigarrow GC \ \text{lth } Ns \ (\text{Suc } i)$
shows $gc\text{-invar } Ns \ (\text{Suc } i)$
 $\langle \text{proof} \rangle$

lemma *gc-invar-gc*:
assumes
 $gc: \text{chain } (\rightsquigarrow GC) \ Ns$ **and**
 $\text{init}: \text{active-subset (lhd } Ns) = \{\}$ **and**
 $i\text{-lt}: i < \text{llength } Ns$

shows *gc-invar* $Ns\ i$
 ⟨proof⟩

lemma *gc-fair-new-proof*:

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

end

5.2 Lazy Given Clause

context *lazy-given-clause*

begin

definition *from-F* :: '*f* inference \Rightarrow ('*f* \times '*l*) inference set **where**
from-F $\iota = \{\iota' \in \text{Inf-FL. to-F } \iota' = \iota\}$

definition *lgc-invar* :: ('*f* inference set \times ('*f* \times '*l*) set) *llist* \Rightarrow *enat* \Rightarrow *bool* **where**
lgc-invar $TNs\ i \iff$
Inf-from (*active-subset* (*Liminf*-*upto*-*llist* (*lmap* *snd* TNs) i))
 $\subseteq \bigcup$ (*from-F* '*Liminf*-*upto*-*llist* (*lmap* *fst* TNs) i) \cup *Sup*-*upto*-*llist* (*lmap* (*Red-I-G* \circ *snd*) TNs) i

lemma *lgc-invar-infinitary*:

assumes
nnil: \neg *lnull* TNs **and**
invar: $\forall i. \text{enat } i < \text{llength } TNs \longrightarrow \text{lgc-invar } TNs\ (\text{enat } i)$
shows *lgc-invar* $TNs\ \infty$
 ⟨proof⟩

lemma *lgc-invar-lgc-init*:

assumes
nnil: \neg *lnull* TNs **and**
n-init: *active-subset* (*snd* (*lhd* TNs)) = {} **and**
t-init: $\forall \iota \in \text{Inf-F. } \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst } (\text{lhd } TNs)$
shows *lgc-invar* $TNs\ 0$
 ⟨proof⟩

lemma *lgc-invar-lgc-step*:

assumes
Si-lt: *enat* (*Suc* i) $<$ *llength* TNs **and**
invar: *lgc-invar* $TNs\ i$ **and**
step: *lnth* $TNs\ i \rightsquigarrow LGC$ *lnth* $TNs\ (\text{Suc } i)$
shows *lgc-invar* $TNs\ (\text{Suc } i)$
 ⟨proof⟩

lemma *lgc-invar-lgc*:

assumes
lgc: *chain* ($\rightsquigarrow LGC$) TNs **and**
n-init: *active-subset* (*snd* (*lhd* TNs)) = {} **and**
t-init: $\forall \iota \in \text{Inf-F. } \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst } (\text{lhd } TNs)$ **and**
i-lt: $i < \text{llength } TNs$
shows *lgc-invar* $TNs\ i$

<proof>

lemma *lgc-fair-new-proof*:

assumes

lgc: *chain* (\rightsquigarrow *LGC*) *TNs* **and**

n-init: *active-subset* (*snd* (*lhd* *TNs*)) = {} **and**

n-lim: *passive-subset* (*Liminf-llist* (*lmap snd TNs*)) = {} **and**

t-init: $\forall \iota \in \text{Inf-}F. \text{prems-of } \iota = [] \longrightarrow \iota \in \text{fst} (\text{lhd } TNs)$ **and**

t-lim: *Liminf-llist* (*lmap fst TNs*) = {}

shows *fair* (*lmap snd TNs*)

<proof>

end

end