A Formalization of the SCL(FOL) Calculus: Simple Clause Learning for First-Order Logic

Martin Desharnais

March 19, 2025

Abstract

This Isabelle/HOL formalization covers the unexecutable specification of Simple Clause Learning for first-order logic without equality [2, 1]: SCL(FOL). The main results are formal proofs of soundness, non-redundancy of learned clauses, termination, and refutational completeness. Compared to the unformalized version, the formalized calculus is simpler, a number of results were generalized, and the non-redundancy statement was strengthened. We found and corrected one bug in a previously published version of the SCL Backtrack rule. Compared to related formalizations, we introduce a new technique for showing termination based on non-redundant clause learning.

Contents

theory *Abstract-Renaming-Apart*
  imports *Main*
begin

1 Abstract Renaming

locale *renaming-apart* =
  fixes
    $renaming :: 'a\ set \Rightarrow 'a \Rightarrow 'a$
  assumes
    renaming-correct: $finite\ V \Longrightarrow renaming\ V\ x \notin V$ **and**

inj-renaming: *finite* $V \Longrightarrow$ *inj* (*renaming* V)

1.1 Interpretation to Prove That Assumptions Are Consistent

experiment begin

definition *renaming-apart-nats* **where**
renaming-apart-nats V = (*let* m = *Max* V *in* (λx. *Suc* ($x + m$)))

interpretation *renaming-apart-nats*: *renaming-apart renaming-apart-nats*
⟨*proof*⟩

end

end
theory *Ordered-Resolution-Prover-Extra*
imports
Ordered-Resolution-Prover.Abstract-Substitution
begin

2 Abstract Substitution Extra

lemma (**in** *substitution-ops*) *subst-atm-of-eqI*:
$L \cdot l\ \sigma_L = K \cdot l\ \sigma_K \Longrightarrow$ *atm-of* $L \cdot a\ \sigma_L$ = *atm-of* $K \cdot a\ \sigma_K$
⟨*proof*⟩

lemma (**in** *substitution-ops*) *set-mset-subst-cls-conv*: *set-mset* ($C \cdot \sigma$) = (λL. $L \cdot l$ σ) ‘ *set-mset* C
⟨*proof*⟩

end
theory *SCL-FOL*
imports
Main
HOL−Library.FSet
Saturation-Framework.Calculus
Saturation-Framework-Extensions.Clausal-Calculus
Ordered-Resolution-Prover.Clausal-Logic
Ordered-Resolution-Prover.Abstract-Substitution
Ordered-Resolution-Prover.Herbrand-Interpretation
First-Order-Terms.Subsumption
First-Order-Terms.Term
First-Order-Terms.Unification
Abstract-Renaming-Apart
Ordered-Resolution-Prover-Extra
begin

3 Extra Lemmas

3.1 Set Extra

lemma *not-in-iff*: $L \notin xs \longleftrightarrow (\forall y{\in}xs.\ L \neq y)$
$\langle proof \rangle$

lemma *disjoint-iff'*: $A \cap B = \{\} \longleftrightarrow (\forall a \in A.\ a \notin B) \wedge (\forall b \in B.\ b \notin A)$
$\langle proof \rangle$

lemma *set-filter-insert-conv*:
$\{x \in \mathit{insert}\ y\ S.\ P\ x\} = (\mathit{if}\ P\ y\ \mathit{then}\ \mathit{insert}\ y\ \mathit{else}\ \mathit{id})\ \{x \in S.\ P\ x\}$
$\langle proof \rangle$

lemma *not-empty-if-mem*: $x \in X \Longrightarrow X \neq \{\}$
$\langle proof \rangle$

3.2 Finite Set Extra

lemma *finite-induct'* [*case-names empty singleton insert-insert*, *induct set*: *finite*]:
— Discharging $x \notin F$ entails extra work.
assumes *finite F*
assumes $P\ \{\}$
and *singleton*: $\bigwedge x.\ P\ \{x\}$
and *insert-insert*: $\bigwedge x\ y\ F.\ \mathit{finite}\ F \Longrightarrow x \neq y \Longrightarrow x \notin F \Longrightarrow y \notin F \Longrightarrow P\ (\mathit{insert}\ y\ F) \Longrightarrow P\ (\mathit{insert}\ x\ (\mathit{insert}\ y\ F))$
shows $P\ F$
$\langle proof \rangle$

3.3 Product Type Extra

lemma *insert-Times*: $\mathit{insert}\ a\ A \times B = \mathit{Pair}\ a\ \text{‘}\ B \cup A \times B$
$\langle proof \rangle$

lemma *Times-insert*: $A \times \mathit{insert}\ b\ B = (\lambda x.\ (x,\ b))\ \text{‘}\ A \cup A \times B$
$\langle proof \rangle$

lemma *insert-Times-insert'*:
$\mathit{insert}\ a\ A \times \mathit{insert}\ b\ B = \mathit{insert}\ (a,\ b)\ ((\mathit{Pair}\ a\ \text{‘}\ B) \cup ((\lambda x.\ (x,\ b))\ \text{‘}\ A) \cup (A \times B))$
(**is** *?lhs* = *?rhs*)
$\langle proof \rangle$

3.4 List Extra

lemma *lt-lengthD*:
assumes *i-lt-xs*: $i < \mathit{length}\ xs$
shows $\exists\, xs1\ xi\ xs2.\ xs = xs1\ @\ xi\ \#\ xs2 \wedge \mathit{length}\ xs1 = i$
$\langle proof \rangle$

lemma *lt-lt-lengthD*:
assumes *i-lt-xs*: $i < \mathit{length}\ xs$ **and** *j-lt-xs*: $j < \mathit{length}\ xs$ **and**
i-lt-j: $i < j$
shows $\exists\, xs1\ xi\ xs2\ xj\ xs3.\ xs = xs1\ @\ xi\ \#\ xs2\ @\ xj\ \#\ xs3 \wedge \mathit{length}\ xs1 = i \wedge \mathit{length}\ (xs1\ @\ xi\ \#\ xs2) = j$
⟨*proof*⟩

3.5 Sublist Extra

lemma *not-mem-strict-suffix*:
shows $\mathit{strict\text{-}suffix}\ xs\ (y\ \#\ ys) \Longrightarrow y \notin \mathit{set}\ ys \Longrightarrow y \notin \mathit{set}\ xs$
⟨*proof*⟩

lemma *not-mem-strict-suffix′*:
shows $\mathit{strict\text{-}suffix}\ xs\ (y\ \#\ ys) \Longrightarrow f\ y \notin f\ \text{‘}\ \mathit{set}\ ys \Longrightarrow f\ y \notin f\ \text{‘}\ \mathit{set}\ xs$
⟨*proof*⟩

3.6 Multiset Extra

lemma *multp$_{DM}$-implies-one-step*:
$\mathit{multp}_{DM}\ R\ M\ N \Longrightarrow \exists\, I\ J\ K.\ N = I + J \wedge M = I + K \wedge J \neq \{\#\} \wedge (\forall\, k{\in}\#K.\ \exists\, x{\in}\#J.\ R\ k\ x)$
⟨*proof*⟩

lemma *multp$_{HO}$-implies-one-step*:
$\mathit{multp}_{HO}\ R\ M\ N \Longrightarrow \exists\, I\ J\ K.\ N = I + J \wedge M = I + K \wedge J \neq \{\#\} \wedge (\forall\, k{\in}\#K.\ \exists\, x{\in}\#J.\ R\ k\ x)$
⟨*proof*⟩

lemma *Multiset-Bex-plus-iff*: $(\exists\, x \in\#\ (M1 + M2).\ P\ x) \longleftrightarrow (\exists\, x \in\#\ M1.\ P\ x) \vee (\exists\, x \in\#\ M2.\ P\ x)$
⟨*proof*⟩

lemma *multp-singleton-rightD*:
assumes $\mathit{multp}\ R\ M\ \{\#x\#\}$ **and** $\mathit{transp}\ R$
shows $y \in\#\ M \Longrightarrow R\ y\ x$
⟨*proof*⟩

3.6.1 Calculus Extra

lemma (**in** *consequence-relation*) *entails-one-formula*: $N \models U \Longrightarrow D \in U \Longrightarrow N \models \{D\}$
⟨*proof*⟩

3.7 Clausal Calculus Extra

3.7.1 Clausal Calculus Only

lemma *true-cls-iff-set-mset-eq*: $\mathit{set\text{-}mset}\ C = \mathit{set\text{-}mset}\ D \Longrightarrow I \models C \longleftrightarrow I \models D$
⟨*proof*⟩

lemma *true-clss-if-set-mset-eq*: $(\forall D \in \mathcal{D}.\ \exists C \in \mathcal{C}.\ \textit{set-mset}\ D = \textit{set-mset}\ C) \Longrightarrow I \models s\ \mathcal{C} \Longrightarrow I \models s\ \mathcal{D}$
⟨*proof*⟩

lemma *entails-clss-insert*: $N \models e\ \textit{insert}\ C\ U \longleftrightarrow N \models e\ \{C\} \wedge N \models e\ U$
⟨*proof*⟩

lemma *Collect-lits-from-atms-conv*: $\{L.\ P\ (\textit{atm-of}\ L)\} = (\bigcup x \in \{x.\ P\ x\}.\ \{\textit{Pos}\ x,\ \textit{Neg}\ x\})$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

3.7.2 Clausal Calculus and Abstract Substitution

lemma (**in** *substitution*) *is-ground-lit-Pos*[*simp*]: *is-ground-atm atm* $\Longrightarrow$ *is-ground-lit* (*Pos atm*)
⟨*proof*⟩

lemma (**in** *substitution*) *is-ground-lit-Neg*[*simp*]: *is-ground-atm atm* $\Longrightarrow$ *is-ground-lit* (*Neg atm*)
⟨*proof*⟩

3.8 First Order Terms Extra

3.8.1 First Order Terms Only

lemma *atm-of-eq-uminus-if-lit-eq*: $L = -\ K \Longrightarrow \textit{atm-of}\ L = \textit{atm-of}\ K$
⟨*proof*⟩

lemma *subst-subst-eq-subst-subst-if-subst-eq-substI*:
assumes $t \cdot \sigma = u \cdot \delta$ **and**
t-inter-δ-empty: $\textit{vars-term}\ t \cap \textit{subst-domain}\ \delta = \{\}$ **and**
u-inter-σ-empty: $\textit{vars-term}\ u \cap \textit{subst-domain}\ \sigma = \{\}$
shows
$\textit{range-vars}\ \sigma \cap \textit{subst-domain}\ \delta = \{\} \Longrightarrow t \cdot \sigma \cdot \delta = u \cdot \sigma \cdot \delta$
$\textit{range-vars}\ \delta \cap \textit{subst-domain}\ \sigma = \{\} \Longrightarrow t \cdot \delta \cdot \sigma = u \cdot \delta \cdot \sigma$
⟨*proof*⟩

lemma *subst-compose-in-unifiersI*:
assumes $t \cdot \sigma = u \cdot \delta$ **and**
$\textit{vars-term}\ t \cap \textit{subst-domain}\ \delta = \{\}$ **and**
$\textit{vars-term}\ u \cap \textit{subst-domain}\ \sigma = \{\}$
shows
$\textit{range-vars}\ \sigma \cap \textit{subst-domain}\ \delta = \{\} \Longrightarrow \sigma \circ_s \delta \in \textit{unifiers}\ \{(t,\ u)\}$
$\textit{range-vars}\ \delta \cap \textit{subst-domain}\ \sigma = \{\} \Longrightarrow \delta \circ_s \sigma \in \textit{unifiers}\ \{(t,\ u)\}$
⟨*proof*⟩

lemma *subst-ident-if-not-in-domain*: $x \notin \textit{subst-domain}\ \mu \Longrightarrow \mu\ x = \textit{Var}\ x$
⟨*proof*⟩

lemma *is-renaming* $(\mathit{Var}(x := \mathit{Var}\ x'))$
$\langle \mathit{proof} \rangle$

lemma *ex-mgu-if-subst-eq-subst-and-disj-vars*:
fixes $t\ u$ :: $('f,\ 'v)\ \mathit{Term.term}$ **and** $\sigma_t\ \sigma_u$ :: $('f,\ 'v)\ \mathit{subst}$
assumes $t \cdot \sigma_t = u \cdot \sigma_u$ **and** $\mathit{vars\text{-}term}\ t \cap \mathit{vars\text{-}term}\ u = \{\}$
shows $\exists\, \mu :: ('f,\ 'v)\ \mathit{subst}.\ \mathit{Unification.mgu}\ t\ u = \mathit{Some}\ \mu$
$\langle \mathit{proof} \rangle$

lemma *restrict-subst-domain-subst-composition*:
fixes $\sigma_A\ \sigma_B\ A\ B$
assumes
distinct-domains: $A \cap B = \{\}$ **and**
distinct-range: $\forall\, x \in A.\ \mathit{vars\text{-}term}\ (\sigma_A\ x) \cap \mathit{subst\text{-}domain}\ \sigma_B = \{\}$
defines $\sigma \equiv \mathit{restrict\text{-}subst\text{-}domain}\ A\ \sigma_A \circ_s \sigma_B$
shows $x \in A \Longrightarrow \sigma\ x = \sigma_A\ x\ x \in B \Longrightarrow \sigma\ x = \sigma_B\ x$
$\langle \mathit{proof} \rangle$

lemma *merge-substs-on-disjoint-domains*:
fixes $\sigma_A\ \sigma_B\ A\ B$
assumes *distinct-domains*: $A \cap B = \{\}$
defines $\sigma \equiv (\lambda x.\ \mathit{if}\ x \in A\ \mathit{then}\ \sigma_A\ x\ \mathit{else}\ \mathit{if}\ x \in B\ \mathit{then}\ \sigma_B\ x\ \mathit{else}\ \mathit{Var}\ x)$
shows
$x \in A \Longrightarrow \sigma\ x = \sigma_A\ x$
$x \in B \Longrightarrow \sigma\ x = \sigma_B\ x$
$x \notin A \cup B \Longrightarrow \sigma\ x = \mathit{Var}\ x$
$\langle \mathit{proof} \rangle$

definition *is-grounding-merge* **where**
is-grounding-merge $\gamma\ A\ \gamma_A\ B\ \gamma_B \longleftrightarrow$
$A \cap B = \{\} \longrightarrow (\forall\, x \in A.\ \mathit{vars\text{-}term}\ (\gamma_A\ x) = \{\}) \longrightarrow (\forall\, x \in B.\ \mathit{vars\text{-}term}$
$(\gamma_B\ x) = \{\}) \longrightarrow$
$(\forall\, x \in A.\ \gamma\ x = \gamma_A\ x) \wedge (\forall\, x \in B.\ \gamma\ x = \gamma_B\ x)$

lemma *is-grounding-merge-if-mem-then-else*[*simp*]:
fixes $\gamma_A\ \gamma_B\ A\ B$
defines $\gamma \equiv (\lambda x.\ \mathit{if}\ x \in A\ \mathit{then}\ \gamma_A\ x\ \mathit{else}\ \gamma_B\ x)$
shows *is-grounding-merge* $\gamma\ A\ \gamma_A\ B\ \gamma_B$
$\langle \mathit{proof} \rangle$

lemma *is-grounding-merge-restrict-subst-domain-comp*[*simp*]:
fixes $\gamma_A\ \gamma_B\ A\ B$
defines $\gamma \equiv \mathit{restrict\text{-}subst\text{-}domain}\ A\ \gamma_A \circ_s \gamma_B$

shows *is-grounding-merge* γ A γ_A B γ_B
⟨*proof*⟩

3.8.2 First Order Terms And Abstract Substitution

no-notation *subst-apply-term* (**infixl** ‹·› *67*)
no-notation *subst-compose* (**infixl** ‹$\circ_s$› *75*)

global-interpretation *substitution-ops subst-apply-term Var subst-compose* ⟨*proof*⟩

notation *subst-atm-abbrev* (**infixl** ‹·*a*› *67*)
notation *subst-atm-list* (**infixl** ‹·*al*› *67*)
notation *subst-lit* (**infixl** ‹·*l*› *67*)
notation *subst-cls* (**infixl** ‹·› *67*)
notation *subst-clss* (**infixl** ‹·*cs*› *67*)
notation *subst-cls-list* (**infixl** ‹·*cl*› *67*)
notation *subst-cls-lists* (**infixl** ‹··*cl*› *67*)
notation *comp-subst-abbrev* (**infixl** ‹$\odot$› *67*)

abbreviation *vars-lit* :: ($'f$, $'v$) *Term.term literal* ⇒ $'v$ *set* **where**
vars-lit $L \equiv$ *vars-term* (*atm-of* L)

definition *vars-cls* :: ($'f$, $'v$) *term clause* ⇒ $'v$ *set* **where**
vars-cls C = *Union* (*set-mset* (*image-mset vars-lit* C))

definition *vars-clss* :: ($'f$, $'v$) *term clause set* ⇒ $'v$ *set* **where**
vars-clss N = ($\bigcup C \in N$. *vars-cls* C)

lemma *vars-clss-empty*[*simp*]: *vars-clss* {} = {}
⟨*proof*⟩

lemma *vars-clss-insert*[*simp*]: *vars-clss* (*insert* C N) = *vars-cls* C ∪ *vars-clss* N
⟨*proof*⟩

lemma *vars-clss-union*[*simp*]: *vars-clss* (*CC* ∪ *DD*) = *vars-clss CC* ∪ *vars-clss DD*
⟨*proof*⟩

lemma *vars-cls-empty*[*simp*]: *vars-cls* {#} = {}
⟨*proof*⟩

lemma *finite-vars-cls*[*simp*]: *finite* (*vars-cls* C)
⟨*proof*⟩

lemma *vars-cls-plus-iff*: *vars-cls* (C + D) = *vars-cls* C ∪ *vars-cls* D
⟨*proof*⟩

lemma *vars-cls-subset-vars-cls-if-subset-mset*: $C \subseteq\# D \Longrightarrow$ *vars-cls* $C \subseteq$ *vars-cls* D

$\langle proof \rangle$

lemma *is-ground-atm-iff-vars-empty*: *is-ground-atm* $t \longleftrightarrow$ *vars-term* $t = \{\}$
$\langle proof \rangle$

lemma *is-ground-lit-iff-vars-empty*: *is-ground-lit* $L \longleftrightarrow$ *vars-lit* $L = \{\}$
$\langle proof \rangle$

lemma *is-ground-cls-iff-vars-empty*: *is-ground-cls* $C \longleftrightarrow$ *vars-cls* $C = \{\}$
$\langle proof \rangle$

lemma *is-ground-atm-is-ground-on-var*:
assumes *is-ground-atm* $(A \cdot a\ \sigma)$ **and** $v \in$ *vars-term* A
shows *is-ground-atm* $(\sigma\ v)$
$\langle proof \rangle$

lemma *is-ground-lit-is-ground-on-var*:
assumes *ground-lit*: *is-ground-lit* (*subst-lit* $L\ \sigma$) **and** *v-in-L*: $v \in$ *vars-lit* L
shows *is-ground-atm* $(\sigma\ v)$
$\langle proof \rangle$

lemma *is-ground-cls-is-ground-on-var*:
assumes
ground-clause: *is-ground-cls* (*subst-cls* $C\ \sigma$) **and**
v-in-C: $v \in$ *vars-cls* C
shows *is-ground-atm* $(\sigma\ v)$
$\langle proof \rangle$

lemma *vars-atm-subset-subst-domain-if-grounding*:
assumes *is-ground-atm* $(t \cdot a\ \gamma)$
shows *vars-term* $t \subseteq$ *subst-domain* γ
$\langle proof \rangle$

lemma *vars-lit-subset-subst-domain-if-grounding*:
assumes *is-ground-lit* $(L \cdot l\ \gamma)$
shows *vars-lit* $L \subseteq$ *subst-domain* γ
$\langle proof \rangle$

lemma *vars-cls-subset-subst-domain-if-grounding*:
assumes *is-ground-cls* $(C \cdot \sigma)$
shows *vars-cls* $C \subseteq$ *subst-domain* σ
$\langle proof \rangle$

lemma *same-on-vars-lit*:
assumes $\forall\, v \in$ *vars-lit* $L.\ \sigma\ v = \tau\ v$
shows *subst-lit* $L\ \sigma =$ *subst-lit* $L\ \tau$
$\langle proof \rangle$

lemma *same-on-vars-clause*:
assumes $\forall\, v \in$ *vars-cls* $S.\ \sigma\ v = \tau\ v$
shows *subst-cls* $S\ \sigma =$ *subst-cls* $S\ \tau$
⟨*proof*⟩

lemma *same-on-lits-clause*:
assumes $\forall\, L \in\#\ C.$ *subst-lit* $L\ \sigma =$ *subst-lit* $L\ \tau$
shows *subst-cls* $C\ \sigma =$ *subst-cls* $C\ \tau$
⟨*proof*⟩

global-interpretation *substitution* $(\cdot a)$ *Var* :: - $\Rightarrow$ $('f,\ 'v)$ *term* $(\odot)$
⟨*proof*⟩

lemma *vars-subst-lit-eq-vars-subst-atm*: *vars-lit* $(L \cdot l\ \sigma) =$ *vars-term* (*atm-of* $L \cdot a\ \sigma)$
⟨*proof*⟩

lemma *vars-subst-lit-eq*:
vars-lit $(L \cdot l\ \sigma) = (\bigcup x \in$ *vars-lit* $L.$ *vars-term* $(\sigma\ x))$
⟨*proof*⟩

lemma *vars-subst-cls-eq*:
vars-cls $(C \cdot \sigma) = (\bigcup x \in$ *vars-cls* $C.$ *vars-term* $(\sigma\ x))$
⟨*proof*⟩

lemma *vars-subst-lit-subset*: *vars-lit* $(L \cdot l\ \sigma) \subseteq$ *vars-lit* L $-$ *subst-domain* $\sigma\ \cup$ *range-vars* σ
⟨*proof*⟩

lemma *vars-subst-cls-subset*: *vars-cls* $(C \cdot \sigma) \subseteq$ *vars-cls* C $-$ *subst-domain* $\sigma\ \cup$ *range-vars* σ
⟨*proof*⟩

lemma *vars-subst-cls-subset-weak*: *vars-cls* $(C \cdot \sigma) \subseteq$ *vars-cls* $C\ \cup$ *range-vars* σ
⟨*proof*⟩

lemma *vars-cls-plus*[*simp*]: *vars-cls* $(C + D) =$ *vars-cls* $C\ \cup$ *vars-cls* D
⟨*proof*⟩

lemma *vars-cls-add-mset*[*simp*]: *vars-cls* (*add-mset* $L\ C) =$ *vars-lit* $L\ \cup$ *vars-cls* C
⟨*proof*⟩

lemma *UN-vars-term-atm-of-cls*[*simp*]: $(\bigcup T \in \{$*atm-of* ‘ *set-mset* $C\}.\ \bigcup$ (*vars-term* ‘ $T)) =$ *vars-cls* C
⟨*proof*⟩

lemma *vars-lit-subst-subset-vars-cls-substI*[*intro*]:
vars-lit $L \subseteq$ *vars-cls* $C \Longrightarrow$ *vars-lit* $(L \cdot l\ \sigma) \subseteq$ *vars-cls* $(C \cdot \sigma)$
⟨*proof*⟩

lemma *vars-subst-cls-subset-vars-cls-subst*:
$\quad$ *vars-cls* $C \subseteq$ *vars-cls* $D \Longrightarrow$ *vars-cls* $(C \cdot \sigma) \subseteq$ *vars-cls* $(D \cdot \sigma)$
$\quad\langle proof\rangle$

lemma *vars-cls-subst-subset*:
$\quad$ **assumes** *range-vars-η*: *range-vars* $\eta \subseteq$ *vars-lit* $L \cup$ *vars-lit* L'
$\quad$ **shows** *vars-cls* (*add-mset* $L\ D \cdot \eta) \subseteq$ *vars-cls* (*add-mset* L' (*add-mset* $L\ D$))
$\langle proof\rangle$

definition *disjoint-vars* **where**
$\quad$ *disjoint-vars* $C\ D \longleftrightarrow$ *vars-cls* $C \cap$ *vars-cls* $D = \{\}$

lemma *disjoint-vars-iff-inter-empty*: *disjoint-vars* $C\ D \longleftrightarrow$ *vars-cls* $C \cap$ *vars-cls* $D = \{\}$
$\quad\langle proof\rangle$

hide-fact *disjoint-vars-def*

lemma *disjoint-vars-sym*: *disjoint-vars* $C\ D \longleftrightarrow$ *disjoint-vars* $D\ C$
$\quad\langle proof\rangle$

lemma *disjoint-vars-plus-iff*: *disjoint-vars* $(C + D)\ E \longleftrightarrow$ *disjoint-vars* $C\ E\ \wedge$ *disjoint-vars* $D\ E$
$\quad\langle proof\rangle$

lemma *disjoint-vars-subset-mset*: *disjoint-vars* $C\ D \Longrightarrow E \subseteq\#\ C \Longrightarrow$ *disjoint-vars* $E\ D$
$\quad\langle proof\rangle$

lemma *disjoint-vars-subst-clsI*:
$\quad$ *disjoint-vars* $C\ D \Longrightarrow$ *range-vars* $\sigma \cap$ *vars-cls* $D = \{\} \Longrightarrow$ *disjoint-vars* $(C \cdot \sigma)$ D
$\quad\langle proof\rangle$

lemma *is-renaming-iff*: *is-renaming* $\varrho \longleftrightarrow (\forall x.$ *is-Var* $(\varrho\ x)) \wedge$ *inj* ϱ
$\quad$(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
$\langle proof\rangle$

lemma *subst-cls-idem-if-disj-vars*: *subst-domain* $\sigma \cap$ *vars-cls* $C = \{\} \Longrightarrow C \cdot \sigma = C$
$\quad\langle proof\rangle$

lemma *subst-lit-idem-if-disj-vars*: *subst-domain* $\sigma \cap$ *vars-lit* $L = \{\} \Longrightarrow L \cdot l\ \sigma = L$
$\quad\langle proof\rangle$

lemma *subst-lit-restrict-subst-domain*: *vars-lit* $L \subseteq V \Longrightarrow L \cdot l$ *restrict-subst-domain* $V\ \sigma = L \cdot l\ \sigma$

$\langle proof \rangle$

lemma *subst-cls-restrict-subst-domain*: *vars-cls* $C \subseteq V \Longrightarrow C \cdot$ *restrict-subst-domain* $V\ \sigma = C \cdot \sigma$
$\langle proof \rangle$

lemma *subst-clss-insert*[*simp*]: *insert* $C\ U \cdot cs\ \eta =$ *insert* $(C \cdot \eta)\ (U \cdot cs\ \eta)$
$\langle proof \rangle$

lemma *valid-grounding-of-renaming*:
assumes *is-renaming* ϱ
shows $I \models s$ *grounding-of-cls* $(C \cdot \varrho) \longleftrightarrow I \models s$ *grounding-of-cls* C
$\langle proof \rangle$

lemma *is-unifier-iff-mem-unifiers-Times*:
assumes *fin-AA*: *finite AA*
shows *is-unifier* υ $AA \longleftrightarrow \upsilon \in$ *unifiers* $(AA \times AA)$
$\langle proof \rangle$

lemma *is-mgu-singleton-iff-Unifiers-is-mgu-Times*:
assumes *fin*: *finite AA*
shows *is-mgu* υ $\{AA\} \longleftrightarrow$ *Unifiers.is-mgu* υ $(AA \times AA)$
$\langle proof \rangle$

lemma *is-imgu-singleton-iff-Unifiers-is-imgu-Times*:
assumes *fin*: *finite AA*
shows *is-imgu* υ $\{AA\} \longleftrightarrow$ *Unifiers.is-imgu* υ $(AA \times AA)$
$\langle proof \rangle$

lemma *unifiers-without-refl*: *unifiers* $E =$ *unifiers* $\{e \in E.\ fst\ e \neq snd\ e\}$
(**is** *?lhs = ?rhs*)
$\langle proof \rangle$

lemma *subst-lit-renaming-subst-adapted*:
assumes *ren-*ϱ: *is-renaming* ϱ **and** *vars-L*: *vars-lit* $L \subseteq$ *subst-domain* σ
shows $L \cdot l\ \varrho \cdot l$ *rename-subst-domain* $\varrho\ \sigma = L \cdot l\ \sigma$
$\langle proof \rangle$

lemma *subst-renaming-subst-adapted*:
assumes *ren-*ϱ: *is-renaming* ϱ **and** *vars-D*: *vars-cls* $D \subseteq$ *subst-domain* σ
shows $D \cdot \varrho \cdot$ *rename-subst-domain* $\varrho\ \sigma = D \cdot \sigma$
$\langle proof \rangle$

lemma *subst-domain-rename-subst-domain-subset'*:
assumes *is-var-*ϱ: $\forall x.$ *is-Var* $(\varrho\ x)$
shows *subst-domain* (*rename-subst-domain* $\varrho\ \sigma$) $\subseteq (\bigcup x \in$ *subst-domain* σ. *vars-term* $(\varrho\ x))$
$\langle proof \rangle$

lemma *range-vars-eq-empty-if-is-ground*:
is-ground-cls $(C \cdot \gamma) \Longrightarrow$ *subst-domain* $\gamma \subseteq$ *vars-cls* $C \Longrightarrow$ *range-vars* $\gamma = \{\}$
⟨*proof*⟩

3.8.3 Minimal, Idempotent Most General Unifier

lemma *is-imgu-if-mgu-eq-Some*:
assumes *mgu*: *Unification.mgu t u* = *Some* μ
shows *is-imgu* μ $\{\{t, u\}\}$
⟨*proof*⟩

primrec *pairs* **where**
pairs [] = [] |
pairs (*x* # *xs*) = (*x*, *x*) # *map* (*Pair x*) *xs* @ *map* ($\lambda y.$ (*y*, *x*)) *xs* @ *pairs xs*

lemma *set* (*pairs* [*a*, *b*, *c*, *d*]) =
{(*a*, *a*), (*a*, *b*), (*a*, *c*), (*a*, *d*),
(*b*, *a*), (*b*, *b*), (*b*, *c*), (*b*, *d*),
(*c*, *a*), (*c*, *b*), (*c*, *c*), (*c*, *d*),
(*d*, *a*), (*d*, *b*), (*d*, *c*), (*d*, *d*)}
⟨*proof*⟩

lemma *set-pairs*: *set* (*pairs xs*) = *set xs* $\times$ *set xs*
⟨*proof*⟩

Reflexive and symmetric pairs are not necessary to computing the MGU, but it makes the set of the resulting list equivalent to $\{(x, y).\ x \in xs \wedge y \in ys\}$, which is necessary for the following properties.

lemma *pair-in-set-pairs*: $a \in$ *set as* $\Longrightarrow b \in$ *set as* $\Longrightarrow (a, b) \in$ *set* (*pairs as*)
⟨*proof*⟩

lemma *fst-pair-in-set-if-pair-in-pairs*: $p \in$ *set* (*pairs as*) $\Longrightarrow$ *fst p* $\in$ *set as*
⟨*proof*⟩

lemma *snd-pair-in-set-if-pair-in-pairs*: $p \in$ *set* (*pairs as*) $\Longrightarrow$ *snd p* $\in$ *set as*
⟨*proof*⟩

lemma *vars-mset-mset-pairs*:
vars-mset (*mset* (*pairs as*)) = ($\bigcup b \in$ *set as*. $\bigcup a \in$ *set as*. *vars-term a* $\cup$ *vars-term b*)
⟨*proof*⟩

definition *mgu-sets* **where**
mgu-sets μ *AAA* $\longleftrightarrow$ ($\exists$ *ass*. *set* (*map set ass*) = *AAA* $\wedge$
map-option subst-of (*unify* (*concat* (*map pairs ass*)) []) = *Some* μ)

lemma *is-imgu-if-mgu-sets*:
assumes *mgu-AAA*: *mgu-sets* μ *AAA*

shows *is-imgu* μ *AAA*
⟨*proof*⟩

3.8.4 Renaming Extra

context *renaming-apart* **begin**

lemma *inj-Var-comp-renaming*: *finite V* $\Longrightarrow$ *inj* (*Var* $\circ$ *renaming V*)
⟨*proof*⟩

lemma *is-renaming-Var-comp-renaming*: *finite V* $\Longrightarrow$ *Term.is-renaming* (*Var* $\circ$ *renaming V*)
⟨*proof*⟩

lemma *vars-term-subst-term-Var-comp-renaming-disj*:
assumes *fin-V*: *finite V*
shows *vars-term* (*t* $\cdot a$ (*Var* $\circ$ *renaming V*)) $\cap$ *V* = {}
⟨*proof*⟩

lemma *vars-term-subst-term-Var-comp-renaming-disj'*:
assumes *fin-V*: *finite V1* **and** *sub*: *V2* $\subseteq$ *V1*
shows *vars-term* (*t* $\cdot a$ (*Var* $\circ$ *renaming V1*)) $\cap$ *V2* = {}
⟨*proof*⟩

lemma *vars-lit-subst-renaming-disj*:
assumes *fin-V*: *finite V*
shows *vars-lit* (*L* $\cdot l$ (*Var* $\circ$ *renaming V*)) $\cap$ *V* = {}
⟨*proof*⟩

lemma *vars-cls-subst-renaming-disj*:
assumes *fin-V*: *finite V*
shows *vars-cls* (*C* $\cdot$ (*Var* $\circ$ *renaming V*)) $\cap$ *V* = {}
⟨*proof*⟩

abbreviation *renaming-wrt* :: ('*f*, -) *Term.term clause set* $\Rightarrow$ - $\Rightarrow$ ('*f*, -) *Term.term*
where
renaming-wrt N $\equiv$ *Var* $\circ$ *renaming* (*vars-clss N*)

lemma *is-renaming-renaming-wrt*: *finite N* $\Longrightarrow$ *is-renaming* (*renaming-wrt N*)
⟨*proof*⟩

lemma *ex-renaming-to-disjoint-vars*:
fixes *C* :: ('*f*, '*a*) *Term.term clause* **and** *N* :: ('*f*, '*a*) *Term.term clause set*
assumes *fin*: *finite N*
shows $\exists\,\varrho$. *is-renaming* ϱ $\wedge$ *vars-cls* (*C* $\cdot$ ϱ) $\cap$ *vars-clss N* = {}
⟨*proof*⟩

end

4 SCL State

type-synonym $('f, 'v)$ *closure* = $('f, 'v)$ *term clause* × $('f, 'v)$ *subst*
type-synonym $('f, 'v)$ *closure-with-lit* =
$('f, 'v)$ *term clause* × $('f, 'v)$ *term literal* × $('f, 'v)$ *subst*
type-synonym $('f, 'v)$ *trail* = $(('f, 'v)$ *term literal* × $('f, 'v)$ *closure-with-lit option*) *list*
type-synonym $('f, 'v)$ *state* =
$('f, 'v)$ *trail* × $('f, 'v)$ *term clause fset* × $('f, 'v)$ *closure option*

Note that, in contrast to Bromberger, Schwarz, and Weidenbach, the level is not part of the state. It would be redundant because it can always be computed from the trail.

abbreviation *initial-state* :: $('f, 'v)$ *state* **where**
initial-state ≡ ([], {||}, *None*)

definition *state-trail* :: $('f, 'v)$ *state* ⇒ $('f, 'v)$ *trail* **where**
state-trail S = *fst* S

lemma *state-trail-simp*[*simp*]: *state-trail* $(\Gamma, U, u) = \Gamma$
⟨*proof*⟩

definition *state-learned* :: $('f, 'v)$ *state* ⇒ $('f, 'v)$ *term clause fset* **where**
state-learned S = *fst* (*snd* S)

lemma *state-learned-simp*[*simp*]: *state-learned* $(\Gamma, U, u) = U$
⟨*proof*⟩

definition *state-conflict* :: $('f, 'v)$ *state* ⇒ $('f, 'v)$ *closure option* **where**
state-conflict S = *snd* (*snd* S)

lemma *state-conflict-simp*[*simp*]: *state-conflict* $(\Gamma, U, u) = u$
⟨*proof*⟩

lemmas *state-proj-simp* = *state-trail-simp state-learned-simp state-conflict-simp*

lemma *state-simp*[*simp*]: (*state-trail* S, *state-learned* S, *state-conflict* S) = S
⟨*proof*⟩

fun *clss-of-trail-lit* **where**
clss-of-trail-lit (-, *None*) = {||} |
clss-of-trail-lit (-, *Some* $(C, L, -)$) = {|*add-mset* L C|}

primrec *clss-of-trail* :: $('f, 'v)$ *trail* ⇒ $('f, 'v)$ *term clause fset* **where**
clss-of-trail [] = {||} |
clss-of-trail $(Ln \# \Gamma)$ = *clss-of-trail-lit* Ln |∪| *clss-of-trail* Γ

hide-fact *clss-of-trail-def*

lemma *clss-of-trail-append*: *clss-of-trail* $(\Gamma_0 @ \Gamma_1) =$ *clss-of-trail* Γ_0 $|\cup|$ *clss-of-trail* Γ_1
$\langle proof \rangle$

fun *clss-of-closure* **where**
clss-of-closure None $= \{||\}$ |
clss-of-closure (*Some* $(C, -)) = \{|C|\}$

definition *propagate-lit* **where**
propagate-lit $L\ C\ \gamma = (L \cdot l\ \gamma,$ *Some* $(C, L, \gamma))$

abbreviation *trail-propagate* ::
$('f, 'v)$ *trail* $\Rightarrow$ $('f, 'v)$ *term literal* $\Rightarrow$ $('f, 'v)$ *term clause* $\Rightarrow$ $('f, 'v)$ *subst* $\Rightarrow$
$('f, 'v)$ *trail* **where**
trail-propagate $\Gamma\ L\ C\ \gamma \equiv$ *propagate-lit* $L\ C\ \gamma\ \#\ \Gamma$

lemma *fst-propagate-lit*[*simp*]: *fst* (*propagate-lit* $L\ C\ \sigma) = L \cdot l\ \sigma$
$\langle proof \rangle$

lemma *suffix-trail-propagate*[*simp*]: *suffix* Γ (*trail-propagate* $\Gamma\ L\ C\ \delta$)
$\langle proof \rangle$

lemma *clss-of-trail-trail-propagate*[*simp*]:
clss-of-trail (*trail-propagate* $\Gamma\ L\ C\ \gamma$) = *finsert* (*add-mset* $L\ C$) (*clss-of-trail* Γ)
$\langle proof \rangle$

definition *decide-lit* **where**
decide-lit $L = (L,$ *None*$)$

abbreviation *trail-decide* :: $('f, 'v)$ *trail* $\Rightarrow$ $('f, 'v)$ *term literal* $\Rightarrow$ $('f, 'v)$ *trail* **where**
trail-decide $\Gamma\ L \equiv$ *decide-lit* $L\ \#\ \Gamma$

lemma *fst-decide-lit*[*simp*]: *fst* (*decide-lit* L) $= L$
$\langle proof \rangle$

lemma *clss-of-trail-trail-decide*[*simp*]:
clss-of-trail (*trail-decide* $\Gamma\ L$) = *clss-of-trail* Γ
$\langle proof \rangle$

definition *is-decision-lit*
:: $('f, 'v)$ *term literal* $\times$ $('f, 'v)$ *closure-with-lit option* $\Rightarrow$ *bool* **where**
is-decision-lit $Ln \longleftrightarrow$ *snd* $Ln =$ *None*

definition *trail-interp* :: - *list* $\Rightarrow$ - *interp* **where**
trail-interp $\Gamma = \bigcup((\lambda L.$ *case* L *of Pos* $A \Rightarrow \{A\}$ | *Neg* $A \Rightarrow \{\})$ ‘ *fst* ‘ *set* $\Gamma)$

lemma

$trail\text{-}interp\ [] = \{\}$
$trail\text{-}interp\ ((Pos\ A,\ ann)\ \#\ \Gamma) = insert\ A\ (trail\text{-}interp\ \Gamma)$
$trail\text{-}interp\ ((Neg\ A,\ ann)\ \#\ \Gamma) = trail\text{-}interp\ \Gamma$
⟨*proof*⟩

lemma *trail-interp-eq-Union*:
$trail\text{-}interp\ \Gamma = (\bigcup Ln \in set\ \Gamma.\ case\ fst\ Ln\ of\ Pos\ t \Rightarrow \{t\}\ |\ Neg\ t \Rightarrow \{\})$
⟨*proof*⟩

definition *trail-true-lit* :: (- *literal* × - *option*) *list* ⇒ - *literal* ⇒ *bool* **where**
$trail\text{-}true\text{-}lit\ \Gamma\ L \longleftrightarrow L \in fst\ `\ set\ \Gamma$

definition *trail-false-lit* :: (- *literal* × - *option*) *list* ⇒ - *literal* ⇒ *bool* **where**
$trail\text{-}false\text{-}lit\ \Gamma\ L \longleftrightarrow -\ L \in fst\ `\ set\ \Gamma$

definition *trail-true-cls* :: (- *literal* × - *option*) *list* ⇒ - *clause* ⇒ *bool* **where**
$trail\text{-}true\text{-}cls\ \Gamma\ C \longleftrightarrow (\exists L \in\#\ C.\ trail\text{-}true\text{-}lit\ \Gamma\ L)$

definition *trail-false-cls* :: (- *literal* × - *option*) *list* ⇒ - *clause* ⇒ *bool* **where**
$trail\text{-}false\text{-}cls\ \Gamma\ C \longleftrightarrow (\forall L \in\#\ C.\ trail\text{-}false\text{-}lit\ \Gamma\ L)$

definition *trail-true-clss* :: ('*f*, '*v*) *trail* ⇒ ('*f*, '*v*) *term clause set* ⇒ *bool* **where**
$trail\text{-}true\text{-}clss\ \Gamma\ N \longleftrightarrow (\forall C \in N.\ trail\text{-}true\text{-}cls\ \Gamma\ C)$

definition *trail-defined-lit* :: (- *literal* × - *option*) *list* ⇒ - *literal* ⇒ *bool* **where**
$trail\text{-}defined\text{-}lit\ \Gamma\ L \longleftrightarrow (L \in fst\ `\ set\ \Gamma \lor -\ L \in fst\ `\ set\ \Gamma)$

definition *trail-defined-cls* :: (- *literal* × - *option*) *list* ⇒ - *clause* ⇒ *bool* **where**
$trail\text{-}defined\text{-}cls\ \Gamma\ C \longleftrightarrow (\forall L \in\#\ C.\ trail\text{-}defined\text{-}lit\ \Gamma\ L)$

lemma *trail-defined-lit-iff-true-or-false*:
$trail\text{-}defined\text{-}lit\ \Gamma\ L \longleftrightarrow trail\text{-}true\text{-}lit\ \Gamma\ L \lor trail\text{-}false\text{-}lit\ \Gamma\ L$
⟨*proof*⟩

lemma *trail-true-or-false-cls-if-defined*:
$trail\text{-}defined\text{-}cls\ \Gamma\ C \Longrightarrow trail\text{-}true\text{-}cls\ \Gamma\ C \lor trail\text{-}false\text{-}cls\ \Gamma\ C$
⟨*proof*⟩

lemma *trail-false-cls-mempty*[*simp*]: $trail\text{-}false\text{-}cls\ \Gamma\ \{\#\}$
⟨*proof*⟩

lemma *trail-false-cls-add-mset*:
$trail\text{-}false\text{-}cls\ \Gamma\ (add\text{-}mset\ L\ C) \longleftrightarrow trail\text{-}false\text{-}lit\ \Gamma\ L \land trail\text{-}false\text{-}cls\ \Gamma\ C$
⟨*proof*⟩

lemma *trail-false-cls-plus*:
$trail\text{-}false\text{-}cls\ \Gamma\ (C + D) \longleftrightarrow trail\text{-}false\text{-}cls\ \Gamma\ C \land trail\text{-}false\text{-}cls\ \Gamma\ D$
⟨*proof*⟩

lemma *not-trail-true-Nil*[*simp*]:
  $\neg$ *trail-true-lit* [] L
  $\neg$ *trail-true-cls* [] C
  $N \neq \{\} \Longrightarrow \neg$ *trail-true-clss* [] N
  ⟨*proof*⟩

lemma *not-trail-false-Nil*[*simp*]:
  $\neg$ *trail-false-lit* [] L
  trail-false-cls [] $C \longleftrightarrow C = \{\#\}$
  ⟨*proof*⟩

lemma *not-trail-defined-lit-Nil*[*simp*]: $\neg$ *trail-defined-lit* [] L
  ⟨*proof*⟩

lemma *trail-defined-lit-if-trail-defined-suffix*:
  suffix Γ' $\Gamma \Longrightarrow$ *trail-defined-lit* Γ' $K \Longrightarrow$ *trail-defined-lit* Γ K
  ⟨*proof*⟩

lemma *trail-defined-cls-if-trail-defined-suffix*:
  suffix Γ' $\Gamma \Longrightarrow$ *trail-defined-cls* Γ' $C \Longrightarrow$ *trail-defined-cls* Γ C
  ⟨*proof*⟩

lemma *trail-false-lit-if-trail-false-suffix*:
  suffix Γ' $\Gamma \Longrightarrow$ *trail-false-lit* Γ' $K \Longrightarrow$ *trail-false-lit* Γ K
  ⟨*proof*⟩

lemma *trail-false-cls-if-trail-false-suffix*:
  suffix Γ' $\Gamma \Longrightarrow$ *trail-false-cls* Γ' $C \Longrightarrow$ *trail-false-cls* Γ C
  ⟨*proof*⟩

lemma *trail-interp-Cons*: *trail-interp* $(Ln \# \Gamma) =$ *trail-interp* $[Ln] \cup$ *trail-interp* Γ
  ⟨*proof*⟩

lemma *trail-interp-Cons'*: *trail-interp* $(Ln \# \Gamma) = ($*case fst Ln of Pos A* $\Rightarrow \{A\}$ |
Neg A $\Rightarrow \{\}) \cup$ *trail-interp* Γ
  ⟨*proof*⟩

lemma *true-lit-thick-unionD*: $(I1 \cup I2) \Vdash\!\!= l\ L \Longrightarrow I1 \Vdash\!\!= l\ L \vee I2 \Vdash\!\!= l\ L$
  ⟨*proof*⟩

lemma *subtrail-falseI*:
  assumes *tr-false*: *trail-false-cls* $((L, Cl) \# \Gamma)$ C **and** *L-not-in*: $-L \notin\# C$
  shows *trail-false-cls* Γ C
  ⟨*proof*⟩

lemma *trail-false-cls-ignores-duplicates*:
  set-mset $C =$ *set-mset* $D \Longrightarrow$ *trail-false-cls* Γ $C \longleftrightarrow$ *trail-false-cls* Γ D
  ⟨*proof*⟩

lemma *ball-trail-propagate-is-ground-lit*:
  assumes $\forall x \in set\ \Gamma.\ \textit{is-ground-lit}\ (\textit{fst}\ x)$ **and** $\textit{is-ground-lit}\ (L \cdot l\ \sigma)$
  shows $\forall x \in set\ (\textit{trail-propagate}\ \Gamma\ L\ C\ \sigma).\ \textit{is-ground-lit}\ (\textit{fst}\ x)$
  ⟨*proof*⟩

lemma *ball-trail-decide-is-ground-lit*:
  assumes $\forall x \in set\ \Gamma.\ \textit{is-ground-lit}\ (\textit{fst}\ x)$ **and** $\textit{is-ground-lit}\ L$
  shows $\forall x \in set\ (\textit{trail-decide}\ \Gamma\ L).\ \textit{is-ground-lit}\ (\textit{fst}\ x)$
  ⟨*proof*⟩

lemma *trail-false-cls-subst-mgu-before-grounding*:
  fixes $\Gamma :: ('f, 'v)\ \textit{trail}$
  assumes *tr-false-cls*: $\textit{trail-false-cls}\ \Gamma\ ((D + \{\#L,\ L'\#\}) \cdot \sigma)$ **and**
    imgu-μ: $\textit{is-imgu}\ \mu\ \{\{\textit{atm-of}\ L,\ \textit{atm-of}\ L'\}\}$ **and**
    unif-σ: $\textit{is-unifiers}\ \sigma\ \{\{\textit{atm-of}\ L,\ \textit{atm-of}\ L'\}\}$
  shows $\textit{trail-false-cls}\ \Gamma\ ((D + \{\#L\#\}) \cdot \mu \cdot \sigma)$
  ⟨*proof*⟩

lemma *trail-defined-lit-iff-defined-uminus*: $\textit{trail-defined-lit}\ \Gamma\ L \longleftrightarrow \textit{trail-defined-lit}\ \Gamma\ (-L)$
  ⟨*proof*⟩

lemma *trail-defined-lit-iff*: $\textit{trail-defined-lit}\ \Gamma\ L \longleftrightarrow \textit{atm-of}\ L \in \textit{atm-of}\ `\ \textit{fst}\ `\ \textit{set}\ \Gamma$
  ⟨*proof*⟩

lemma *trail-interp-conv*: $\textit{trail-interp}\ \Gamma = \textit{atm-of}\ `\ \{L \in \textit{fst}\ `\ \textit{set}\ \Gamma.\ \textit{is-pos}\ L\}$
⟨*proof*⟩

lemma *not-in-trail-interp-if-not-in-trail*: $t \notin \textit{atm-of}\ `\ \textit{fst}\ `\ \textit{set}\ \Gamma \Longrightarrow t \notin \textit{trail-interp}\ \Gamma$
  ⟨*proof*⟩

inductive *trail-consistent* **where**
  Nil[*simp*]: *trail-consistent* [] |
  Cons: $\neg\ \textit{trail-defined-lit}\ \Gamma\ L \Longrightarrow \textit{trail-consistent}\ \Gamma \Longrightarrow \textit{trail-consistent}\ ((L,\ u)\ \#\ \Gamma)$

lemma *distinct-atm-of-trail-if-trail-consistent*:
  $\textit{trail-consistent}\ \Gamma \Longrightarrow \textit{distinct}\ (\textit{map}\ (\textit{atm-of} \circ \textit{fst})\ \Gamma)$
  ⟨*proof*⟩

lemma *trail-consistent-appendD*: $\textit{trail-consistent}\ (\Gamma\ @\ \Gamma') \Longrightarrow \textit{trail-consistent}\ \Gamma'$
  ⟨*proof*⟩

lemma *trail-consistent-if-suffix*:
  $\textit{trail-consistent}\ \Gamma \Longrightarrow \textit{suffix}\ \Gamma'\ \Gamma \Longrightarrow \textit{trail-consistent}\ \Gamma'$
  ⟨*proof*⟩

lemma *trail-interp-lit-if-trail-true*:
  shows *trail-consistent* $\Gamma \Longrightarrow$ *trail-true-lit* Γ $L \Longrightarrow$ *trail-interp* $\Gamma \models l$ L
$\langle proof \rangle$

lemma *trail-interp-cls-if-trail-true*:
  assumes *trail-consistent* Γ **and** *trail-true-cls* Γ C
  shows *trail-interp* $\Gamma \models C$
$\langle proof \rangle$

lemma *trail-true-cls-iff-trail-interp-entails*:
  assumes *trail-consistent* Γ $\forall L \in\# C$. *trail-defined-lit* Γ L
  shows *trail-true-cls* Γ $C \longleftrightarrow$ *trail-interp* $\Gamma \models C$
$\langle proof \rangle$

lemma *trail-false-cls-iff-not-trail-interp-entails*:
  assumes *trail-consistent* Γ $\forall L \in\# C$. *trail-defined-lit* Γ L
  shows *trail-false-cls* Γ $C \longleftrightarrow \neg$ *trail-interp* $\Gamma \models C$
$\langle proof \rangle$

inductive *trail-closures-false* **where**
  Nil[*simp*]: *trail-closures-false* [] |
  Cons:
    $(\forall D\ K\ \gamma.\ Kn =$ *propagate-lit* $K\ D\ \gamma \longrightarrow$ *trail-false-cls* $\Gamma\ (D \cdot \gamma)) \Longrightarrow$
    trail-closures-false $\Gamma \Longrightarrow$ *trail-closures-false* $(Kn \# \Gamma)$

lemma *trail-closures-false-ConsD*: *trail-closures-false* $(Ln \# \Gamma) \Longrightarrow$ *trail-closures-false* Γ
  $\langle proof \rangle$

lemma *trail-closures-false-appendD*: *trail-closures-false* $(\Gamma @ \Gamma') \Longrightarrow$ *trail-closures-false* Γ'
  $\langle proof \rangle$

lemma *is-ground-lit-if-true-in-ground-trail*:
  assumes $\forall L \in$ *fst* ‘ *set* Γ. *is-ground-lit* L
  shows *trail-true-lit* Γ $L \Longrightarrow$ *is-ground-lit* L
  $\langle proof \rangle$

lemma *is-ground-lit-if-false-in-ground-trail*:
  assumes $\forall L \in$ *fst* ‘ *set* Γ. *is-ground-lit* L
  shows *trail-false-lit* Γ $L \Longrightarrow$ *is-ground-lit* L
  $\langle proof \rangle$

lemma *is-ground-lit-if-defined-in-ground-trail*:
  assumes $\forall L \in$ *fst* ‘ *set* Γ. *is-ground-lit* L
  shows *trail-defined-lit* Γ $L \Longrightarrow$ *is-ground-lit* L
  $\langle proof \rangle$

lemma *is-ground-cls-if-false-in-ground-trail*:

assumes $\forall L \in \mathit{fst}$ ‘ $\mathit{set}\ \Gamma.\ \mathit{is\text{-}ground\text{-}lit}\ L$
shows $\mathit{trail\text{-}false\text{-}cls}\ \Gamma\ C \Longrightarrow \mathit{is\text{-}ground\text{-}cls}\ C$
⟨*proof*⟩

5 SCL(FOL) Calculus

locale $\mathit{scl\text{-}fol\text{-}calculus} = \mathit{renaming\text{-}apart}\ \mathit{renaming\text{-}vars}$
for $\mathit{renaming\text{-}vars} :: {'v}\ \mathit{set} \Rightarrow {'v} \Rightarrow {'v}\ +$
fixes $\mathit{less\text{-}B} :: ('f,\ 'v)\ \mathit{term} \Rightarrow ('f,\ 'v)\ \mathit{term} \Rightarrow \mathit{bool}$ (**infix** ‹$\prec_B$› *50*)
assumes
finite-less-B: $\bigwedge\beta.\ \mathit{finite}\ \{x.\ x \prec_B \beta\}$
begin

abbreviation *lesseq-B* (**infix** ‹$\preceq_B$› *50*) **where**
$\mathit{lesseq\text{-}B} \equiv (\prec_B)^{==}$

5.1 Lemmas About $(\prec_B)$

lemma *lits-less-B-conv*: $\{L.\ \mathit{atm\text{-}of}\ L \prec_B \beta\} = (\bigcup x{\in}\{x.\ x \prec_B \beta\}.\ \{\mathit{Pos}\ x,\ \mathit{Neg}\ x\})$
⟨*proof*⟩

lemma *lits-eq-conv*: $\{L.\ \mathit{atm\text{-}of}\ L = \beta\} = \{\mathit{Pos}\ \beta,\ \mathit{Neg}\ \beta\}$
⟨*proof*⟩

lemma *lits-less-eq-B-conv*:
$\{L.\ \mathit{atm\text{-}of}\ L \prec_B \beta \lor \mathit{atm\text{-}of}\ L = \beta\} = \mathit{insert}\ (\mathit{Pos}\ \beta)\ (\mathit{insert}\ (\mathit{Neg}\ \beta)\ \{L.\ \mathit{atm\text{-}of}\ L \prec_B \beta\})$
⟨*proof*⟩

lemma *finite-lits-less-B*: $\mathit{finite}\ \{L.\ \mathit{atm\text{-}of}\ L \prec_B \beta\}$
⟨*proof*⟩

lemma *finite-lits-less-eq-B*: $\mathit{finite}\ \{L.\ \mathit{atm\text{-}of}\ L \preceq_B \beta\}$
⟨*proof*⟩

lemma *Collect-ball-eq-Pow-Collect*: $\{X.\ \forall x \in X.\ P\ x\} = \mathit{Pow}\ \{x.\ P\ x\}$
⟨*proof*⟩

lemma *finite-lit-clss-nodup-less-B*: $\mathit{finite}\ \{C.\ \forall L \in\#\ C.\ \mathit{atm\text{-}of}\ L \prec_B \beta \land \mathit{count}\ C\ L = \mathit{1}\}$
⟨*proof*⟩

5.2 Rules

inductive $\mathit{propagate} :: ('f,\ 'v)\ \mathit{term}\ \mathit{clause}\ \mathit{fset} \Rightarrow ('f,\ 'v)\ \mathit{term} \Rightarrow ('f,\ 'v)\ \mathit{state} \Rightarrow ('f,\ 'v)\ \mathit{state} \Rightarrow \mathit{bool}$ **for** $N\ \beta$ **where**
propagateI: $C\ |{\in}|\ N\ |{\cup}|\ U \Longrightarrow C = \mathit{add\text{-}mset}\ L\ C' \Longrightarrow \mathit{is\text{-}ground\text{-}cls}\ (C \cdot \gamma) \Longrightarrow$

$\forall K \in\# \ C \cdot \gamma. \ atm\text{-}of\ K \preceq_B \beta \Longrightarrow$
$C_0 = \{\#K \in\# \ C'. \ K \cdot l\ \gamma \neq L \cdot l\ \gamma\#\} \Longrightarrow C_1 = \{\#K \in\# \ C'. \ K \cdot l\ \gamma = L \cdot l\ \gamma\#\} \Longrightarrow$
$trail\text{-}false\text{-}cls\ \Gamma\ (C_0 \cdot \gamma) \Longrightarrow \neg\ trail\text{-}defined\text{-}lit\ \Gamma\ (L \cdot l\ \gamma) \Longrightarrow$
$is\text{-}imgu\ \mu\ \{atm\text{-}of\ `\ set\text{-}mset\ (add\text{-}mset\ L\ C_1)\} \Longrightarrow$
$propagate\ N\ \beta\ (\Gamma,\ U,\ None)\ (trail\text{-}propagate\ \Gamma\ (L \cdot l\ \mu)\ (C_0 \cdot \mu)\ \gamma,\ U,\ None)$

lemma $C\ |{\in}|\ N\ |{\cup}|\ U \Longrightarrow C = add\text{-}mset\ L\ C' \Longrightarrow is\text{-}ground\text{-}cls\ (C \cdot \gamma) \Longrightarrow$
$\forall K \in\# \ C. \ atm\text{-}of\ (K \cdot l\ \gamma) \preceq_B \beta \Longrightarrow$
$C_0 = \{\#K \in\# \ C'. \ K \cdot l\ \gamma \neq L \cdot l\ \gamma\#\} \Longrightarrow C_1 = \{\#K \in\# \ C'. \ K \cdot l\ \gamma = L \cdot l\ \gamma\#\} \Longrightarrow$
$trail\text{-}false\text{-}cls\ \Gamma\ (C_0 \cdot \gamma) \Longrightarrow \neg\ trail\text{-}defined\text{-}lit\ \Gamma\ (L \cdot l\ \gamma) \Longrightarrow$
$is\text{-}imgu\ \mu\ \{atm\text{-}of\ `\ set\text{-}mset\ (add\text{-}mset\ L\ C_1)\} \Longrightarrow$
$propagate\ N\ \beta\ (\Gamma,\ U,\ None)\ (trail\text{-}propagate\ \Gamma\ (L \cdot l\ \mu)\ (C_0 \cdot \mu)\ \gamma,\ U,\ None)$
$\langle proof \rangle$

inductive *decide* :: $('f,\ 'v)\ term\ clause\ fset \Rightarrow ('f,\ 'v)\ term \Rightarrow ('f,\ 'v)\ state \Rightarrow ('f,\ 'v)\ state \Rightarrow bool$ **for** $N\ \beta$ **where**
decideI: $is\text{-}ground\text{-}lit\ (L \cdot l\ \gamma) \Longrightarrow$
$\neg\ trail\text{-}defined\text{-}lit\ \Gamma\ (L \cdot l\ \gamma) \Longrightarrow atm\text{-}of\ L \cdot a\ \gamma \preceq_B \beta \Longrightarrow$
$decide\ N\ \beta\ (\Gamma,\ U,\ None)\ (trail\text{-}decide\ \Gamma\ (L \cdot l\ \gamma),\ U,\ None)$

inductive *conflict* :: $('f,\ 'v)\ term\ clause\ fset \Rightarrow ('f,\ 'v)\ term \Rightarrow ('f,\ 'v)\ state \Rightarrow ('f,\ 'v)\ state \Rightarrow bool$ **for** $N\ \beta$ **where**
conflictI: $D\ |{\in}|\ N\ |{\cup}|\ U \Longrightarrow is\text{-}ground\text{-}cls\ (D \cdot \gamma) \Longrightarrow trail\text{-}false\text{-}cls\ \Gamma\ (D \cdot \gamma) \Longrightarrow$
$conflict\ N\ \beta\ (\Gamma,\ U,\ None)\ (\Gamma,\ U,\ Some\ (D,\ \gamma))$

inductive *skip* :: $('f,\ 'v)\ term\ clause\ fset \Rightarrow ('f,\ 'v)\ term \Rightarrow ('f,\ 'v)\ state \Rightarrow ('f,\ 'v)\ state \Rightarrow bool$ **for** $N\ \beta$ **where**
skipI: $-L \notin\# \ D \cdot \sigma \Longrightarrow$
$skip\ N\ \beta\ ((L,\ n)\ \#\ \Gamma,\ U,\ Some\ (D,\ \sigma))\ (\Gamma,\ U,\ Some\ (D,\ \sigma))$

lemma $-(fst\ \mathcal{K}) \notin\# \ D \cdot \sigma \Longrightarrow skip\ N\ \beta\ (\mathcal{K}\ \#\ \Gamma,\ U,\ Some\ (D,\ \sigma))\ (\Gamma,\ U,\ Some\ (D,\ \sigma))$
$\langle proof \rangle$

inductive *factorize* :: $('f,\ 'v)\ term\ clause\ fset \Rightarrow ('f,\ 'v)\ term \Rightarrow ('f,\ 'v)\ state \Rightarrow ('f,\ 'v)\ state \Rightarrow bool$ **for** $N\ \beta$ **where**
factorizeI: $L \cdot l\ \gamma = L' \cdot l\ \gamma \Longrightarrow is\text{-}imgu\ \mu\ \{\{atm\text{-}of\ L,\ atm\text{-}of\ L'\}\} \Longrightarrow$
$factorize\ N\ \beta\ (\Gamma,\ U,\ Some\ (add\text{-}mset\ L'\ (add\text{-}mset\ L\ D),\ \gamma))\ (\Gamma,\ U,\ Some\ (add\text{-}mset\ L\ D \cdot \mu,\ \gamma))$

inductive *resolve* :: $('f,\ 'v)\ term\ clause\ fset \Rightarrow ('f,\ 'v)\ term \Rightarrow ('f,\ 'v)\ state \Rightarrow ('f,\ 'v)\ state \Rightarrow bool$ **for** $N\ \beta$ **where**
resolveI: $\Gamma = trail\text{-}propagate\ \Gamma'\ K\ D\ \gamma_D \Longrightarrow K \cdot l\ \gamma_D = -(L \cdot l\ \gamma_C) \Longrightarrow$
$is\text{-}renaming\ \varrho_C \Longrightarrow is\text{-}renaming\ \varrho_D \Longrightarrow$

vars-cls (*add-mset* L $C \cdot \varrho_C$) $\cap$ *vars-cls* (*add-mset* K $D \cdot \varrho_D$) = {} $\Longrightarrow$
is-imgu μ {{*atm-of* L $\cdot a$ ϱ_C, *atm-of* K $\cdot a$ ϱ_D}} $\Longrightarrow$
is-grounding-merge γ
(*vars-cls* (*add-mset* L $C \cdot \varrho_C$)) (*rename-subst-domain* ϱ_C γ_C)
(*vars-cls* (*add-mset* K $D \cdot \varrho_D$)) (*rename-subst-domain* ϱ_D γ_D) $\Longrightarrow$
resolve N β (Γ, U, *Some* (*add-mset* L C, γ_C)) (Γ, U, *Some* (($C \cdot \varrho_C + D \cdot \varrho_D) \cdot \mu$, γ))

inductive *backtrack* :: ('*f*, '*v*) *term clause fset* $\Rightarrow$ ('*f*, '*v*) *term* $\Rightarrow$ ('*f*, '*v*) *state* $\Rightarrow$ ('*f*, '*v*) *state* $\Rightarrow$ *bool* **for** N β **where**
backtrackI: Γ = *trail-decide* (Γ' @ Γ'') $K \Longrightarrow K = -(L \cdot l\ \sigma) \Longrightarrow$
$\nexists\gamma$. *is-ground-cls* (*add-mset* L $D \cdot \gamma$) $\wedge$ *trail-false-cls* Γ'' (*add-mset* L $D \cdot \gamma$) $\Longrightarrow$
backtrack N β (Γ, U, *Some* (*add-mset* L D, σ)) (Γ'', *finsert* (*add-mset* L D) U, *None*)

definition *scl* :: ('*f*, '*v*) *term clause fset* $\Rightarrow$ ('*f*, '*v*) *term* $\Rightarrow$ ('*f*, '*v*) *state* $\Rightarrow$ ('*f*, '*v*) *state* $\Rightarrow$ *bool* **where**
scl N β S S' $\longleftrightarrow$ *propagate* N β S S' $\vee$ *decide* N β S S' $\vee$ *conflict* N β S S' $\vee$ *skip* N β S S' $\vee$
factorize N β S S' $\vee$ *resolve* N β S S' $\vee$ *backtrack* N β S S'

Note that, in contrast to Fiori and Weidenbach (CADE 2019), the set N of initial clauses and the ground atom β are parameters of the relation instead of being repeated twice in the states. This is to highlight the fact that they are constant.

5.3 Well-Defined

lemma *propagate-well-defined*:
assumes *propagate* N β S S'
shows
$\neg$ *decide* N' β' S S'
$\neg$ *conflict* N' β' S S'
$\neg$ *skip* N' β' S S'
$\neg$ *factorize* N' β' S S'
$\neg$ *resolve* N' β' S S'
$\neg$ *backtrack* N' β' S S'
$\langle$*proof*$\rangle$

lemma *decide-well-defined*:
assumes *decide* N β S S'
shows
$\neg$ *propagate* N' β' S S'
$\neg$ *conflict* N' β' S S'
$\neg$ *skip* N' β' S S'
$\neg$ *factorize* N' β' S S'

¬ *resolve* N' β' S S'
¬ *backtrack* N' β' S S'
⟨*proof*⟩

lemma *conflict-well-defined*:
assumes *conflict* N β S S'
shows
¬ *propagate* N' β' S S'
¬ *decide* N' β' S S'
¬ *skip* N' β' S S'
¬ *factorize* N' β' S S'
¬ *resolve* N' β' S S'
¬ *backtrack* N' β' S S'
⟨*proof*⟩

lemma *skip-well-defined*:
assumes *skip* N β S S'
shows
¬ *propagate* N' β' S S'
¬ *decide* N' β' S S'
¬ *conflict* N' β' S S'
¬ *factorize* N' β' S S'
¬ *resolve* N' β' S S'
¬ *backtrack* N' β' S S'
⟨*proof*⟩

lemma *factorize-well-defined*:
assumes *factorize* N β S S'
shows
¬ *propagate* N β S S'
¬ *decide* N β S S'
¬ *conflict* N β S S'
¬ *skip* N β S S'

¬ *backtrack* N β S S'
⟨*proof*⟩

lemma *resolve-well-defined*:
assumes *resolve* N β S S'
shows
¬ *propagate* N β S S'
¬ *decide* N β S S'
¬ *conflict* N β S S'
¬ *skip* N β S S'

¬ *backtrack* N β S S'
⟨*proof*⟩

lemma *backtrack-well-defined*:

assumes *backtrack* $N\ \beta\ S\ S'$
shows
$\neg$ *propagate* $N'\ \beta'\ S\ S'$
$\neg$ *decide* $N'\ \beta'\ S\ S'$
$\neg$ *conflict* $N'\ \beta'\ S\ S'$
$\neg$ *skip* $N'\ \beta'\ S\ S'$
$\neg$ *factorize* $N'\ \beta'\ S\ S'$
$\neg$ *resolve* $N'\ \beta'\ S\ S'$
$\langle$*proof*$\rangle$

5.4 Some rules are right unique

lemma *right-unique-skip*: *right-unique* (*skip* $N\ \beta$)
$\langle$*proof*$\rangle$

5.5 Miscellaneous Lemmas

lemma *conflict-set-after-factorization*:
assumes *fact*: *factorize* $N\ \beta\ S\ S'$ **and** *conflict-S*: *state-conflict* S = *Some* (C, γ)
shows $\exists\, C'\ \gamma'$. *state-conflict* S' = *Some* (C', γ') $\wedge$ *set-mset* $(C \cdot \gamma)$ = *set-mset* $(C' \cdot \gamma')$
$\langle$*proof*$\rangle$

lemma *not-trail-false-ground-cls-if-no-conflict*:
assumes
no-conf: $\nexists S'$. *conflict* $N\ \beta\ S\ S'$ **and**
could-conf: *state-conflict* S = *None* **and**
C-in: C $|\in|$ N $|\cup|$ *state-learned* S **and**
gr-C-γ: *is-ground-cls* $(C \cdot \gamma)$
shows $\neg$ *trail-false-cls* (*state-trail* S) $(C \cdot \gamma)$
$\langle$*proof*$\rangle$

lemma *scl-mempty-not-in-sate-learned*:
scl $N\ \beta\ S\ S' \Longrightarrow \{\#\}$ $|\notin|$ *state-learned* $S \Longrightarrow \{\#\}$ $|\notin|$ *state-learned* S'
$\langle$*proof*$\rangle$

lemma *conflict-if-mempty-in-initial-clauses-and-no-conflict*:
assumes $\{\#\}$ $|\in|$ N **and** *state-conflict* S = *None*
shows *conflict* $N\ \beta\ S$ (*state-trail* S, *state-learned* S, *Some* $(\{\#\}, Var)$)
$\langle$*proof*$\rangle$

lemma *conflict-initial-state-if-mempty-in-intial-clauses*:
$\{\#\}$ $|\in|$ $N \Longrightarrow$ *conflict* $N\ \beta$ *initial-state* $([], \{||\}, Some\ (\{\#\}, Var))$
$\langle$*proof*$\rangle$

lemma *conflict-empty-trail*:
assumes *conf*: *conflict* $N\ \beta\ S\ S'$ **and** *empty-trail*: *state-trail* $S = []$
shows $\{\#\}$ $|\in|$ N $|\cup|$ *state-learned* S
$\langle$*proof*$\rangle$

lemma *conflict-empty-trail'*:
assumes $\{\#\}$ $|\in|$ N $|\cup|$ U
shows $\exists S'$. *conflict* N β ([], U, *None*) S'
⟨*proof*⟩

lemma *mempty-in-iff-ex-conflict*: $\{\#\}$ $|\in|$ N $|\cup|$ U $\longleftrightarrow$ ($\exists S'$. *conflict* N β ([], U, *None*) S')
⟨*proof*⟩

lemma *conflict-initial-state-only-with-mempty*:
assumes *conflict* N β *initial-state* S
shows $\exists \gamma$. S = ([], $\{||\}$, *Some* ($\{\#\}$, γ))
⟨*proof*⟩

lemma *no-more-step-if-conflict-mempty*:
assumes *state-trail* S = [] *state-conflict* S = *Some* ($\{\#\}$, γ)
shows $\nexists S'$. *scl* N β S S'
⟨*proof*⟩

lemma *ex-conflict-if-trail-false-cls*:
assumes *tr-false-*Γ*-D*: *trail-false-cls* Γ D **and** *D-in*: $D \in$ *grounding-of-clss* (*fset* N $\cup$ *fset* U)
shows $\exists S'$. *conflict* N β (Γ, U, *None*) S'
⟨*proof*⟩

lemma *no-conflict-tail-trail*:
assumes $\nexists S$. *conflict* N β (*Ln* # Γ, U, *None*) S
shows $\nexists S$. *conflict* N β (Γ, U, *None*) S
⟨*proof*⟩

lemma *subst-domain-rename-subst-domain-subset-vars-cls-subst-cls*:
assumes $\forall x$. *is-Var* (ϱ_C x) **and**
dom-γ_C: *subst-domain* $\gamma_C \subseteq$ *vars-cls* (*add-mset* L C)
shows *subst-domain* (*rename-subst-domain* ϱ_C γ_C) $\subseteq$ *vars-cls* (*add-mset* L $C \cdot \varrho_C$)
⟨*proof*⟩

lemma *renamed-comp-renamed-simp*:
fixes γ_C γ_D
assumes
$K \cdot l\ \gamma_D = -\ (L \cdot l\ \gamma_C)$ **and**
ground-conf: *is-ground-cls* (*add-mset* L $C \cdot \gamma_C$) **and**
ground-prop: *is-ground-cls* (*add-mset* K $D \cdot \gamma_D$) **and**
dom-γ_D: *subst-domain* $\gamma_D \subseteq$ *vars-cls* (*add-mset* K D) **and**
ren-ϱ_C: *is-renaming* ϱ_C **and**
ren-ϱ_D: *is-renaming* ϱ_D **and**
disjoint-vars: *vars-cls* (*add-mset* L $C \cdot \varrho_C$) $\cap$ *vars-cls* (*add-mset* K $D \cdot \varrho_D$) = {}
defines $\gamma \equiv$ *rename-subst-domain* ϱ_D $\gamma_D \odot$ *rename-subst-domain* ϱ_C γ_C

shows
subst-renamed-comp-renamed-simp:
$L \cdot l\ \varrho_C \cdot l\ \gamma = L \cdot l\ \gamma_C\ \ C \cdot \varrho_C \cdot \gamma = C \cdot \gamma_C$
$K \cdot l\ \varrho_D \cdot l\ \gamma = K \cdot l\ \gamma_D\ \ D \cdot \varrho_D \cdot \gamma = D \cdot \gamma_D$ **and**
imgu-comp-renamed-comp-renamed-simp:
$\textit{is-imgu}\ \mu\ \{\{\textit{atm-of}\ L \cdot a\ \varrho_C,\ \textit{atm-of}\ K \cdot a\ \varrho_D\}\} \Longrightarrow \mu \odot \gamma = \gamma$
⟨*proof*⟩

6 Invariants

6.1 Initial Literals Generalize Learned, Trail, and Conflict Literals

definition *clss-lits-generalize-clss-lits* **where**
$\textit{clss-lits-generalize-clss-lits}\ N\ U \longleftrightarrow$
$(\forall L \in \bigcup\ (\textit{set-mset}\ `\ U).\ \exists K \in \bigcup (\textit{set-mset}\ `\ N).\ \textit{generalizes-lit}\ K\ L)$

lemma *clss-lits-generalize-clss-lits-if-superset*[*simp*]:
assumes $N2 \subseteq N1$
shows *clss-lits-generalize-clss-lits N1 N2*
⟨*proof*⟩

lemma *clss-lits-generalize-clss-lits-subset*:
$\textit{clss-lits-generalize-clss-lits}\ N\ U1 \Longrightarrow U2 \subseteq U1 \Longrightarrow \textit{clss-lits-generalize-clss-lits}\ N\ U2$
⟨*proof*⟩

lemma *clss-lits-generalize-clss-lits-insert*:
$\textit{clss-lits-generalize-clss-lits}\ N\ (\textit{insert}\ C\ U) \longleftrightarrow$
$(\forall L \in\#\ C.\ \exists K \in \bigcup (\textit{set-mset}\ `\ N).\ \textit{generalizes-lit}\ K\ L) \land \textit{clss-lits-generalize-clss-lits}\ N\ U$
⟨*proof*⟩

lemma *clss-lits-generalize-clss-lits-trans*:
assumes
clss-lits-generalize-clss-lits N1 N2 **and**
clss-lits-generalize-clss-lits N2 N3
shows *clss-lits-generalize-clss-lits N1 N3*
⟨*proof*⟩

lemma *clss-lits-generalize-clss-lits-subst-clss*:
assumes *clss-lits-generalize-clss-lits N1 N2*
shows $\textit{clss-lits-generalize-clss-lits}\ N1\ ((\lambda C.\ C \cdot \sigma)\ `\ N2)$
⟨*proof*⟩

lemma *clss-lits-generalize-clss-lits-singleton-subst-cls*:
$\textit{clss-lits-generalize-clss-lits}\ N\ \{C\} \Longrightarrow \textit{clss-lits-generalize-clss-lits}\ N\ \{C \cdot \sigma\}$
⟨*proof*⟩

lemma *clss-lits-generalize-clss-lits-subst-cls*:
  assumes *clss-lits-generalize-clss-lits N* $\{$*add-mset L1* $($*add-mset L2 C*$)\}$
  shows *clss-lits-generalize-clss-lits N* $\{$*add-mset* $($*L1* $\cdot l\ \sigma)$ $(C \cdot \sigma)\}$
$\langle proof \rangle$

definition *initial-lits-generalize-learned-trail-conflict* **where**
  initial-lits-generalize-learned-trail-conflict N S $\longleftrightarrow$ *clss-lits-generalize-clss-lits* $($*fset N*$)$
    $($*fset* $($*state-learned S* $|\cup|$ *clss-of-trail* $($*state-trail S*$)$ $|\cup|$
      $($*case state-conflict S of None* $\Rightarrow \{||\}$ | *Some* $(C, \text{-}) \Rightarrow \{|C|\})))$

lemma *initial-lits-generalize-learned-trail-conflict-initial-state*[*simp*]:
  initial-lits-generalize-learned-trail-conflict N initial-state
  $\langle proof \rangle$

lemma *propagate-preserves-initial-lits-generalize-learned-trail-conflict*:
  propagate N β *S S'* $\Longrightarrow$ *initial-lits-generalize-learned-trail-conflict N S* $\Longrightarrow$
    initial-lits-generalize-learned-trail-conflict N S'
$\langle proof \rangle$

lemma *decide-preserves-initial-lits-generalize-learned-trail-conflict*:
  decide N β *S S'* $\Longrightarrow$ *initial-lits-generalize-learned-trail-conflict N S* $\Longrightarrow$
    initial-lits-generalize-learned-trail-conflict N S'
$\langle proof \rangle$

lemma *conflict-preserves-initial-lits-generalize-learned-trail-conflict*:
  assumes *conflict N* β *S S'* **and** *initial-lits-generalize-learned-trail-conflict N S*
  shows *initial-lits-generalize-learned-trail-conflict N S'*
  $\langle proof \rangle$

lemma *skip-preserves-initial-lits-generalize-learned-trail-conflict*:
  skip N β *S S'* $\Longrightarrow$ *initial-lits-generalize-learned-trail-conflict N S* $\Longrightarrow$
    initial-lits-generalize-learned-trail-conflict N S'
$\langle proof \rangle$

lemma *factorize-preserves-initial-lits-generalize-learned-trail-conflict*:
  factorize N β *S S'* $\Longrightarrow$ *initial-lits-generalize-learned-trail-conflict N S* $\Longrightarrow$
    initial-lits-generalize-learned-trail-conflict N S'
$\langle proof \rangle$

lemma *resolve-preserves-initial-lits-generalize-learned-trail-conflict*:
  resolve N β *S S'* $\Longrightarrow$ *initial-lits-generalize-learned-trail-conflict N S* $\Longrightarrow$
    initial-lits-generalize-learned-trail-conflict N S'
$\langle proof \rangle$

lemma *backtrack-preserves-initial-lits-generalize-learned-trail-conflict*:
  backtrack N β *S S'* $\Longrightarrow$ *initial-lits-generalize-learned-trail-conflict N S* $\Longrightarrow$
    initial-lits-generalize-learned-trail-conflict N S'
$\langle proof \rangle$

lemma *scl-preserves-initial-lits-generalize-learned-trail-conflict*:
assumes *scl* N β S S' **and** *initial-lits-generalize-learned-trail-conflict* N S
shows *initial-lits-generalize-learned-trail-conflict* N S'
⟨*proof*⟩

6.2 Trail Literals Are Ground

definition *trail-lits-ground* **where**
trail-lits-ground $S \longleftrightarrow (\forall L \in$ *fst* ‘ *set* (*state-trail* S). *is-ground-lit* L)

lemma *trail-lits-ground-initial-state*[*simp*]: *trail-lits-ground initial-state*
⟨*proof*⟩

lemma *propagate-preserves-trail-lits-ground*:
assumes *propagate* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *decide-preserves-trail-lits-ground*:
assumes *decide* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *conflict-preserves-trail-lits-ground*:
assumes *conflict* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *skip-preserves-trail-lits-ground*:
assumes *skip* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *factorize-preserves-trail-lits-ground*:
assumes *factorize* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *resolve-preserves-trail-lits-ground*:
assumes *resolve* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *backtrack-preserves-trail-lits-ground*:
assumes *backtrack* N β S S' **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

lemma *scl-preserves-trail-lits-ground*:
assumes *scl* $N\ \beta\ S\ S'$ **and** *trail-lits-ground* S
shows *trail-lits-ground* S'
⟨*proof*⟩

6.3 Trail Literals Are Defined Only Once

definition *trail-lits-consistent* **where**
trail-lits-consistent $S \longleftrightarrow$ *trail-consistent* (*state-trail* S)

lemma *trail-lits-consistent-initial-state*[*simp*]: *trail-lits-consistent initial-state*
⟨*proof*⟩

lemma *propagate-preserves-trail-lits-consistent*:
assumes *propagate* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *decide-preserves-trail-lits-consistent*:
assumes *decide* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *conflict-preserves-trail-lits-consistent*:
assumes *conflict* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *skip-preserves-trail-lits-consistent*:
assumes *skip* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *factorize-preserves-trail-lits-consistent*:
assumes *factorize* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *resolve-preserves-trail-lits-consistent*:
assumes *resolve* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *backtrack-preserves-trail-lits-consistent*:
assumes *backtrack* $N\ \beta\ S\ S'$ **and** *invar*: *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *scl-preserves-trail-lits-consistent*:

assumes *scl* $N\ \beta\ S\ S'$ **and** *trail-lits-consistent* S
shows *trail-lits-consistent* S'
⟨*proof*⟩

lemma *trail-consistent-iff*: *trail-consistent* $\Gamma \longleftrightarrow (\forall \Gamma'\ Ln\ \Gamma''.\ \Gamma = \Gamma''\ @\ Ln\ \#\ \Gamma' \longrightarrow \neg$ *trail-defined-lit* Γ' (*fst Ln*))
⟨*proof*⟩

6.4 Trail Closures Are False In Subtrails

definition *trail-closures-false'* **where**
trail-closures-false' $S \longleftrightarrow$ *trail-closures-false* (*state-trail* S)

lemma *trail-closures-false'-initial-state*[*simp*]: *trail-closures-false' initial-state*
⟨*proof*⟩

lemma *propagate-preserves-trail-closures-false'*:
assumes *step*: *propagate* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'
⟨*proof*⟩

lemma *decide-preserves-trail-closures-false'*:
assumes *step*: *decide* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'
⟨*proof*⟩

lemma *conflict-preserves-trail-closures-false'*:
assumes *step*: *conflict* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'
⟨*proof*⟩

lemma *skip-preserves-trail-closures-false'*:
assumes *step*: *skip* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'
⟨*proof*⟩

lemma *factorize-preserves-trail-closures-false'*:
assumes *step*: *factorize* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'
⟨*proof*⟩

lemma *resolve-preserves-trail-closures-false'*:
assumes *step*: *resolve* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'
⟨*proof*⟩

lemma *backtrack-preserves-trail-closures-false'*:
assumes *step*: *backtrack* $N\ \beta\ S\ S'$ **and** *invar*: *trail-closures-false'* S
shows *trail-closures-false'* S'

⟨*proof*⟩

lemma *scl-preserves-trail-closures-false′*:
assumes *scl* N β S S' **and** *trail-closures-false′* S
shows *trail-closures-false′* S'
⟨*proof*⟩

lemma *trail-closures-false* $\Gamma \longleftrightarrow$
$(\forall K\ D\ \gamma\ \Gamma'\ \Gamma''.\ \Gamma = \Gamma''\ @\ \textit{propagate-lit}\ K\ D\ \gamma\ \#\ \Gamma' \longrightarrow \textit{trail-false-cls}\ \Gamma'\ (D \cdot \gamma))$
⟨*proof*⟩

6.5 Trail Literals Were Propagated or Decided

inductive *trail-propagated-or-decided* **for** N β U **where**
Nil[*simp*]: *trail-propagated-or-decided* N β U [] |
Propagate:
$C\ |{\in}|\ N\ |{\cup}|\ U \Longrightarrow$
$C = \textit{add-mset}\ L\ C' \Longrightarrow$
$\textit{is-ground-cls}\ (C \cdot \gamma) \Longrightarrow$
$\forall K{\in}\#C \cdot \gamma.\ \textit{atm-of}\ K \preceq_B \beta \Longrightarrow$
$C_0 = \{\#K \in\# C'.\ K \cdot l\ \gamma \neq L \cdot l\ \gamma\#\} \Longrightarrow$
$C_1 = \{\#K \in\# C'.\ K \cdot l\ \gamma = L \cdot l\ \gamma\#\} \Longrightarrow$
$\textit{trail-false-cls}\ \Gamma\ (C_0 \cdot \gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ (L \cdot l\ \gamma) \Longrightarrow$
$\textit{is-imgu}\ \mu\ \{\textit{atm-of}\ \text{‘}\ \textit{set-mset}\ (\textit{add-mset}\ L\ C_1)\} \Longrightarrow$
$\textit{trail-propagated-or-decided}\ N\ \beta\ U\ \Gamma \Longrightarrow$
$\textit{trail-propagated-or-decided}\ N\ \beta\ U\ (\textit{trail-propagate}\ \Gamma\ (L \cdot l\ \mu)\ (C_0 \cdot \mu)\ \gamma)$ |
Decide:
$\textit{is-ground-lit}\ (L \cdot l\ \gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ (L \cdot l\ \gamma) \Longrightarrow$
$\textit{atm-of}\ L \cdot a\ \gamma \preceq_B \beta \Longrightarrow$
$\textit{trail-propagated-or-decided}\ N\ \beta\ U\ \Gamma \Longrightarrow$
$\textit{trail-propagated-or-decided}\ N\ \beta\ U\ (\textit{trail-decide}\ \Gamma\ (L \cdot l\ \gamma))$

lemma *trail-propagate-or-decide-suffixI*:
assumes *trail-propagated-or-decided* N β U *ys* **and** *suffix xs ys*
shows *trail-propagated-or-decided* N β U *xs*
⟨*proof*⟩

definition *trail-propagated-or-decided′* **where**
trail-propagated-or-decided′ N β S =
trail-propagated-or-decided N β (*state-learned* S) (*state-trail* S)

lemma *trail-propagated-or-decided-learned-finsert*:
assumes *trail-propagated-or-decided* N β U Γ
shows *trail-propagated-or-decided* N β (*finsert* C U) Γ
⟨*proof*⟩

lemma *trail-propagated-or-decided-trail-append*:
assumes *trail-propagated-or-decided* N β U $(\Gamma_1 @ \Gamma_2)$
shows *trail-propagated-or-decided* N β U Γ_2
⟨*proof*⟩

lemma *trail-propagated-or-decided-initial-state*[*simp*]:
trail-propagated-or-decided' N β *initial-state*
⟨*proof*⟩

lemma *propagate-preserves-trail-propagated-or-decided*:
assumes *propagate* N β S S' **and** *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *decide-preserves-trail-propagated-or-decided*:
assumes *decide* N β S S' **and** *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *conflict-preserves-trail-propagated-or-decided*:
assumes *conflict* N β S S' **and** *invar*: *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *skip-preserves-trail-propagated-or-decided*:
assumes *skip* N β S S' **and** *invar*: *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *factorize-preserves-trail-propagated-or-decided*:
assumes *factorize* N β S S' **and** *invar*: *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *resolve-preserves-trail-propagated-or-decided*:
assumes *resolve* N β S S' **and** *invar*: *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *backtrack-preserves-trail-propagated-or-decided*:
assumes *backtrack* N β S S' **and** *invar*: *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

lemma *scl-preserves-trail-propagated-or-decided*:
assumes *scl* N β S S' **and** *trail-propagated-or-decided'* N β S
shows *trail-propagated-or-decided'* N β S'
⟨*proof*⟩

definition *trail-propagated-wf* **where**
trail-propagated-wf $\Gamma \longleftrightarrow (\forall (L_\gamma, n) \in set\ \Gamma.$
case n of
$None \Rightarrow True$
$|\ Some\ (\text{-}, L, \gamma) \Rightarrow L_\gamma = L \cdot l\ \gamma)$

lemma *trail-propagated-wf-iff*:
trail-propagated-wf $\Gamma \longleftrightarrow (\forall Ln \in set\ \Gamma.\ \forall D\ K\ \gamma.\ snd\ Ln = Some\ (D, K, \gamma) \longrightarrow fst\ Ln = K \cdot l\ \gamma)$
(**is** $?lhs \longleftrightarrow ?rhs$)
⟨*proof*⟩

lemma *trail-propagated-wf-if-trail-propagated-or-decided*:
trail-propagated-or-decided $N\ U\ \beta\ \Gamma \Longrightarrow$ *trail-propagated-wf* Γ
⟨*proof*⟩

lemma *trail-propagated-wf-if-trail-propagated-or-decided′*:
trail-propagated-or-decided′ $N\ \beta\ S \Longrightarrow$ *trail-propagated-wf* (*state-trail* S)
⟨*proof*⟩

lemma *trail-propagated-lit-wf-initial-state*:
$\forall \mathcal{K}\in set\ (state\text{-}trail\ initial\text{-}state).\ \forall D\ K\ \gamma.\ snd\ \mathcal{K} = Some\ (D, K, \gamma) \longrightarrow fst\ \mathcal{K} = K \cdot l\ \gamma$
⟨*proof*⟩

lemma *scl-preserves-trail-propagated-lit-wf*:
assumes *step*: *scl* $N\ \beta\ S\ S'$ **and**
inv: $\forall \mathcal{K} \in set\ (state\text{-}trail\ S).\ \forall D\ K\ \gamma.\ snd\ \mathcal{K} = Some\ (D, K, \gamma) \longrightarrow fst\ \mathcal{K} = K \cdot l\ \gamma$
shows $\forall \mathcal{K} \in set\ (state\text{-}trail\ S').\ \forall D\ K\ \gamma.\ snd\ \mathcal{K} = Some\ (D, K, \gamma) \longrightarrow fst\ \mathcal{K} = K \cdot l\ \gamma$
⟨*proof*⟩

6.6 Trail Atoms Are Less Than Bound

definition *trail-atoms-lt* **where**
trail-atoms-lt $\beta\ S \longleftrightarrow (\forall A \in atm\text{-}of\ `\ fst\ `\ set\ (state\text{-}trail\ S).\ A \preceq_B \beta)$

lemma *trail-atoms-lt-initial-state*[*simp*]: *trail-atoms-lt* β *initial-state*
⟨*proof*⟩

lemma *propagate-preserves-trail-atoms-lt*:
assumes *propagate* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

lemma *decide-preserves-trail-atoms-lt*:
assumes *decide* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$

⟨*proof*⟩

lemma *conflict-preserves-trail-atoms-lt*:
assumes *conflict* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

lemma *skip-preserves-trail-atoms-lt*:
assumes *skip* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

lemma *factorize-preserves-trail-atoms-lt*:
assumes *factorize* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

lemma *resolve-preserves-trail-atoms-lt*:
assumes *resolve* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

lemma *backtrack-preserves-trail-atoms-lt*:
assumes *backtrack* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

lemma *scl-preserves-trail-atoms-lt*:
assumes *scl* $N\ \beta\ S\ S'$ **and** *trail-atoms-lt* $\beta\ S$
shows *trail-atoms-lt* $\beta\ S'$
⟨*proof*⟩

6.7 Trail Resolved Literals Have Unique Polarity

definition *trail-resolved-lits-pol* **where**
trail-resolved-lits-pol $S \longleftrightarrow$
$(\forall\, Ln \in set\ (state\text{-}trail\ S).\ \forall\, C\ L\ \gamma.\ snd\ Ln = Some\ (C,\ L,\ \gamma) \longrightarrow -(L \cdot l\ \gamma) \notin\#\ C \cdot \gamma)$

lemma *trail-resolved-lits-pol-initial-state*[*simp*]: *trail-resolved-lits-pol initial-state*
⟨*proof*⟩

lemma *propagate-preserves-trail-resolved-lits-pol*:
assumes *step*: *propagate* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *decide-preserves-trail-resolved-lits-pol*:
assumes *step*: *decide* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S

shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *conflict-preserves-trail-resolved-lits-pol*:
assumes *step*: *conflict* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *skip-preserves-trail-resolved-lits-pol*:
assumes *step*: *skip* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *factorize-preserves-trail-resolved-lits-pol*:
assumes *step*: *factorize* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *resolve-preserves-trail-resolved-lits-pol*:
assumes *step*: *resolve* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *backtrack-preserves-trail-resolved-lits-pol*:
assumes *step*: *backtrack* $N\ \beta\ S\ S'$ **and** *invar*: *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

lemma *scl-preserves-trail-resolved-lits-pol*:
assumes *scl* $N\ \beta\ S\ S'$ **and** *trail-resolved-lits-pol* S
shows *trail-resolved-lits-pol* S'
⟨*proof*⟩

6.8 Trail And Conflict Closures Are Ground

definition *ground-closures* **where**
ground-closures $S \longleftrightarrow$
$(\forall\, Ln \in set\ (state\text{-}trail\ S).\ \forall\, C\ L\ \gamma.\ snd\ Ln = Some\ (C,\ L,\ \gamma) \longrightarrow is\text{-}ground\text{-}cls$ $(add\text{-}mset\ L\ C \cdot \gamma)) \land$
$(\forall\, C\ \gamma.\ state\text{-}conflict\ S = Some\ (C,\ \gamma) \longrightarrow is\text{-}ground\text{-}cls\ (C \cdot \gamma))$

lemma *ground-closures-initial-state*[*simp*]: *ground-closures initial-state*
⟨*proof*⟩

lemma *propagate-preserves-ground-closures*:
assumes *step*: *propagate* $N\ \beta\ S\ S'$ **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *decide-preserves-ground-closures*:
assumes *step*: *decide* N β S S' **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *conflict-preserves-ground-closures*:
assumes *step*: *conflict* N β S S' **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *skip-preserves-ground-closures*:
assumes *step*: *skip* N β S S' **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *factorize-preserves-ground-closures*:
assumes *step*: *factorize* N β S S' **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *merge-of-renamed-groundings*:
assumes
ren-ϱ_C: *is-renaming* ϱ_C **and**
ren-ϱ_D: *is-renaming* ϱ_D **and**
disjoint-vars: *vars-cls* $(C \cdot \varrho_C) \cap$ *vars-cls* $(D \cdot \varrho_D) = \{\}$ **and**
ground-conf: *is-ground-cls* $(C \cdot \gamma_C)$ **and**
ground-prop: *is-ground-cls* $(D \cdot \gamma_D)$ **and**
merge-γ: *is-grounding-merge* γ
(*vars-cls* $(C \cdot \varrho_C)$) (*rename-subst-domain* ϱ_C γ_C)
(*vars-cls* $(D \cdot \varrho_D)$) (*rename-subst-domain* ϱ_D γ_D)
shows
$\forall L \in\# C.\ L \cdot l\ \varrho_C \cdot l\ \gamma = L \cdot l\ \gamma_C$
$\forall K \in\# D.\ K \cdot l\ \varrho_D \cdot l\ \gamma = K \cdot l\ \gamma_D$
⟨*proof*⟩

lemma *resolve-preserves-ground-closures*:
assumes *step*: *resolve* N β S S' **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *backtrack-preserves-ground-closures*:
assumes *step*: *backtrack* N β S S' **and** *invar*: *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

lemma *scl-preserves-ground-closures*:
assumes *scl* N β S S' **and** *ground-closures* S
shows *ground-closures* S'
⟨*proof*⟩

6.9 Trail And Conflict Closures Are Ground And False

definition *ground-false-closures* **where**
ground-false-closures $S \longleftrightarrow$ *ground-closures* $S \wedge$
trail-closures-false (*state-trail* S) $\wedge$
($\forall$ C γ. *state-conflict* S = *Some* (C, γ) $\longrightarrow$ *trail-false-cls* (*state-trail* S) (C $\cdot$ γ))

lemma *ground-false-closures-initial-state*[*simp*]: *ground-false-closures initial-state*
⟨*proof*⟩

lemma *propagate-preserves-ground-false-closures*:
assumes *step*: *propagate* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *decide-preserves-ground-false-closures*:
assumes *step*: *decide* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *conflict-preserves-ground-false-closures*:
assumes *step*: *conflict* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *skip-preserves-ground-false-closures*:
assumes *step*: *skip* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *factorize-preserves-ground-false-closures*:
assumes *step*: *factorize* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *resolve-preserves-ground-false-closures*:
assumes *step*: *resolve* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *backtrack-preserves-ground-false-closures*:
assumes *step*: *backtrack* N β S S' **and** *invar*: *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

lemma *scl-preserves-ground-false-closures*:
assumes *scl* N β S S' **and** *ground-false-closures* S
shows *ground-false-closures* S'
⟨*proof*⟩

6.10 Learned Clauses Are Non-empty

definition *learned-nonempty* **where**
learned-nonempty $S \longleftrightarrow \{\#\}$ $|\notin|$ *state-learned* S

lemma *learned-nonempty-initial-state*[*simp*]: *learned-nonempty initial-state*
⟨*proof*⟩

lemma *propagate-preserves-learned-nonempty*:
assumes *propagate* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *decide-preserves-learned-nonempty*:
assumes *decide* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *conflict-preserves-learned-nonempty*:
assumes *conflict* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *skip-preserves-learned-nonempty*:
assumes *skip* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *factorize-preserves-learned-nonempty*:
assumes *factorize* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *resolve-preserves-learned-nonempty*:
assumes *resolve* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *backtrack-preserves-learned-nonempty*:
assumes *backtrack* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

lemma *scl-preserves-learned-nonempty*:
assumes *scl* N β S S' **and** *learned-nonempty* S
shows *learned-nonempty* S'
⟨*proof*⟩

6.11 Backtrack Follows Conflict Resolution

definition *conflict-resolution* **where**
conflict-resolution N β $S \longleftrightarrow$ (*state-conflict* $S \neq$ *None* $\longrightarrow$
($\exists$ *S0 S1*. *conflict* N β *S0 S1* $\wedge$ (*skip* N β $\sqcup$ *factorize* N β $\sqcup$ *resolve* N β)** *S1*
S))

lemma *conflict-resolution-initial-state*[*simp*]: *conflict-resolution* N β *initial-state*
⟨*proof*⟩

lemma *propagate-preserves-conflict-resolution*:
assumes *step*: *propagate* N β S S'
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *decide-preserves-conflict-resolution*:
assumes *step*: *decide* N β S S'
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *conflict-preserves-conflict-resolution*:
assumes *step*: *conflict* N β S S'
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *skip-preserves-conflict-resolution*:
assumes *step*: *skip* N β S S' **and** *invar*: *conflict-resolution* N β S
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *factorize-preserves-conflict-resolution*:
assumes *step*: *factorize* N β S S' **and** *invar*: *conflict-resolution* N β S
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *resolve-preserves-conflict-resolution*:
assumes *step*: *resolve* N β S S' **and** *invar*: *conflict-resolution* N β S
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *backtrack-preserves-conflict-resolution*:
assumes *step*: *backtrack* N β S S'
shows *conflict-resolution* N β S'
⟨*proof*⟩

lemma *scl-preserves-conflict-resolution*:
assumes *scl* N β S S' **and** *conflict-resolution* N β S
shows *conflict-resolution* N β S'
⟨*proof*⟩

6.12 Miscellaneous Lemmas

lemma *before-conflict*:
assumes *conflict* N β *S1 S2* **and**
invars: *learned-nonempty S1 trail-propagated-or-decided′* N β *S1*
shows $\{\#\}$ $|\in|$ $N \lor (\exists$ *S0*. *propagate* N β *S0 S1*$) \lor (\exists$ *S0*. *decide* N β *S0 S1*$)$
⟨*proof*⟩

lemma *before-backtrack*:
assumes *backt*: *backtrack* N β *Sn Sm* **and**
invar: *conflict-resolution* N β *Sn*
shows $\exists$ *S0 S1*. *conflict* N β *S0 S1* $\land$ (*skip* N β $\sqcup$ *factorize* N β $\sqcup$ *resolve* N $\beta)^{**}$ *S1 Sn*
⟨*proof*⟩

lemma *ball-less-B-if-trail-false-and-trail-atoms-lt*:
trail-false-cls (*state-trail* S) $C \Longrightarrow$ *trail-atoms-lt* β $S \Longrightarrow \forall L \in\# C$. *atm-of* $L \preceq_B \beta$
⟨*proof*⟩

7 Soundness

7.1 Sound Trail

abbreviation *entails-*$\mathcal{G}$ (**infix** ‹$\Vdash\mathcal{G}e$› *50*) **where**
entails-$\mathcal{G}$ N $U \equiv$ *grounding-of-clss* $N \Vdash e$ *grounding-of-clss* U

definition *sound-trail* **where**
sound-trail N $\Gamma \longleftrightarrow$
$(\forall Ln \in set\ \Gamma.\ \forall D\ K\ \gamma.\ snd\ Ln = Some\ (D, K, \gamma) \longrightarrow fset\ N \Vdash\mathcal{G}e\ \{add\text{-}mset\ K\ D\})$

lemma *sound-trail-Nil*[*simp*]: *sound-trail* N []
⟨*proof*⟩

lemma *entails-*$\mathcal{G}$*-mono*: $N \Vdash\mathcal{G}e\ U \Longrightarrow N \subseteq NN \Longrightarrow NN \Vdash\mathcal{G}e\ U$
⟨*proof*⟩

lemma *sound-trail-supersetI*: *sound-trail* N $\Gamma \Longrightarrow N\ |\subseteq|\ NN \Longrightarrow$ *sound-trail* NN Γ
⟨*proof*⟩

lemma *sound-trail-ConsD*: *sound-trail* N $(Ln\ \#\ \Gamma) \Longrightarrow$ *sound-trail* N Γ
⟨*proof*⟩

lemma *sound-trail-appendD*: *sound-trail* N $(\Gamma\ @\ \Gamma') \Longrightarrow$ *sound-trail* N Γ'
⟨*proof*⟩

lemma *sound-trail-propagate*:

assumes
sound-Γ: *sound-trail* N Γ **and**
N-entails-C-L: *fset* N $\Vert\!\!\models\mathcal{G}e$ $\{C + \{\#L\#\}\}$
shows *sound-trail* N (*trail-propagate* Γ L C σ)
⟨*proof*⟩

lemma *sound-trail-decide*:
sound-trail N $\Gamma \Longrightarrow$ *sound-trail* N (*trail-decide* Γ L)
⟨*proof*⟩

7.2 Sound State

definition *sound-state* :: $('f, 'v)$ *term clause fset* $\Rightarrow$ $('f, 'v)$ *term* $\Rightarrow$ $('f, 'v)$ *state* $\Rightarrow$ *bool* **where**
sound-state N β $S \longleftrightarrow$
($\exists\Gamma$ U u. $S = (\Gamma, U, u)$ $\wedge$ *sound-trail* N Γ $\wedge$ *fset* N $\Vert\!\!\models\mathcal{G}e$ *fset* U $\wedge$
(*case* u *of None* $\Rightarrow$ *True* | *Some* (C, γ) $\Rightarrow$ *fset* N $\Vert\!\!\models\mathcal{G}e$ $\{C\}$))

7.3 Initial State Is Sound

lemma *sound-initial-state*[*simp*]: *sound-state* N β *initial-state*
⟨*proof*⟩

7.4 SCL Rules Preserve Soundness

lemma *mem-vars-cls-subst-clsD*: $x' \in$ *vars-cls* $(C \cdot \varrho) \Longrightarrow \exists x{\in}$*vars-cls* C. $x' \in$ *vars-term* $(\varrho\ x)$
⟨*proof*⟩

lemma *propagate-preserves-sound-state*:
assumes *step*: *propagate* N β S S' **and** *sound*: *sound-state* N β S
shows *sound-state* N β S'
⟨*proof*⟩

lemma *decide-preserves-sound-state*:
assumes *step*: *decide* N β S S' **and** *sound*: *sound-state* N β S
shows *sound-state* N β S'
⟨*proof*⟩

lemma *conflict-preserves-sound-state*:
assumes *step*: *conflict* N β S S' **and** *sound*: *sound-state* N β S
shows *sound-state* N β S'
⟨*proof*⟩

lemma *skip-preserves-sound-state*:
assumes *step*: *skip* N β S S' **and** *sound*: *sound-state* N β S
shows *sound-state* N β S'
⟨*proof*⟩

lemma *factorize-preserves-sound-state*:

assumes *step*: *factorize* $N\ \beta\ S\ S'$ **and** *sound*: *sound-state* $N\ \beta\ S$
shows *sound-state* $N\ \beta\ S'$
⟨*proof*⟩

lemma *resolve-preserves-sound-state*:
assumes *step*: *resolve* $N\ \beta\ S\ S'$ **and** *sound*: *sound-state* $N\ \beta\ S$
shows *sound-state* $N\ \beta\ S'$
⟨*proof*⟩

lemma *backtrack-preserves-sound-state*:
assumes *step*: *backtrack* $N\ \beta\ S\ S'$ **and** *sound*: *sound-state* $N\ \beta\ S$
shows *sound-state* $N\ \beta\ S'$
⟨*proof*⟩

theorem *scl-preserves-sound-state*:
fixes N :: $('f,\ 'v)$ *Term.term clause fset*
shows *scl* $N\ \beta\ S\ S' \Longrightarrow$ *sound-state* $N\ \beta\ S \Longrightarrow$ *sound-state* $N\ \beta\ S'$
⟨*proof*⟩

8 Strategies

definition *reasonable-scl* **where**
reasonable-scl $N\ \beta\ S\ S' \longleftrightarrow$
scl $N\ \beta\ S\ S' \wedge ($*decide* $N\ \beta\ S\ S' \longrightarrow \neg(\exists S''.$ *conflict* $N\ \beta\ S'\ S''))$

lemma *scl-if-reasonable*: *reasonable-scl* $N\ \beta\ S\ S' \Longrightarrow$ *scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

definition *regular-scl* **where**
regular-scl $N\ \beta\ S\ S' \longleftrightarrow$
conflict $N\ \beta\ S\ S' \vee \neg\ (\exists S''.$ *conflict* $N\ \beta\ S\ S'') \wedge$ *reasonable-scl* $N\ \beta\ S\ S'$

lemma *reasonable-if-regular*:
regular-scl $N\ \beta\ S\ S' \Longrightarrow$ *reasonable-scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

lemma *scl-if-regular*:
regular-scl $N\ \beta\ S\ S' \Longrightarrow$ *scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

The following specification of *regular-scl* is better for the paper as it highlights that it is a restriction of *reasonable-scl*.

lemma *regular-scl* $N\ \beta\ S\ S' \longleftrightarrow$ *reasonable-scl* $N\ \beta\ S\ S' \wedge$
$((\exists S''.$ *conflict* $N\ \beta\ S\ S'') \longrightarrow$ *conflict* $N\ \beta\ S\ S')$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

definition *ex-conflict* **where**
ex-conflict $C\ \Gamma \longleftrightarrow (\exists \gamma.$ *is-ground-cls* $(C \cdot \gamma) \wedge$ *trail-false-cls* $\Gamma\ (C \cdot \gamma))$

definition *is-shortest-backtrack* **where**
$\textit{is-shortest-backtrack}\ C\ \Gamma\ \Gamma_0 \longleftrightarrow C \neq \{\#\} \longrightarrow \textit{suffix}\ \Gamma_0\ \Gamma \wedge \neg\ \textit{ex-conflict}\ C\ \Gamma_0$
$\wedge$
$(\forall\ Kn.\ \textit{suffix}\ (Kn\ \#\ \Gamma_0)\ \Gamma \longrightarrow \textit{ex-conflict}\ C\ (Kn\ \#\ \Gamma_0))$

definition *shortest-backtrack-strategy* **where**
$\textit{shortest-backtrack-strategy}\ R\ N\ \beta\ S\ S' \longleftrightarrow R\ N\ \beta\ S\ S' \wedge (\textit{backtrack}\ N\ \beta\ S\ S'$
$\longrightarrow$
$\textit{is-shortest-backtrack}\ (\textit{fst}\ (\textit{the}\ (\textit{state-conflict}\ S)))\ (\textit{state-trail}\ S)\ (\textit{state-trail}\ S'))$

lemma *regular-scl-if-shortest-backtrack-strategy*:
$\textit{shortest-backtrack-strategy regular-scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{regular-scl}\ N\ \beta\ S\ S'$
$\langle \textit{proof} \rangle$

lemma *strategy-restrictions*:
shows
$\textit{shortest-backtrack-strategy regular-scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{regular-scl}\ N\ \beta\ S\ S'$ **and**
$\textit{regular-scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{reasonable-scl}\ N\ \beta\ S\ S'$ **and**
$\textit{reasonable-scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{scl}\ N\ \beta\ S\ S'$
$\langle \textit{proof} \rangle$

primrec *shortest-backtrack* **where**
$\textit{shortest-backtrack}\ C\ [] = []\ |$
$\textit{shortest-backtrack}\ C\ (Ln\ \#\ \Gamma) =$
$(\textit{if ex-conflict}\ C\ (Ln\ \#\ \Gamma)\ \textit{then}$
$\textit{shortest-backtrack}\ C\ \Gamma$
$\textit{else}$
$Ln\ \#\ \Gamma)$

lemma *suffix-shortest-backtrack*: $\textit{suffix}\ (\textit{shortest-backtrack}\ C\ \Gamma)\ \Gamma$
$\langle \textit{proof} \rangle$

lemma *ex-conflict-shortest-backtrack*: $\textit{ex-conflict}\ C\ (\textit{shortest-backtrack}\ C\ \Gamma) \longleftrightarrow$
$C = \{\#\}$
$\langle \textit{proof} \rangle$

lemma *is-shortest-backtrack-shortest-backtrack*:
$C \neq \{\#\} \Longrightarrow \textit{is-shortest-backtrack}\ C\ \Gamma\ (\textit{shortest-backtrack}\ C\ \Gamma)$
$\langle \textit{proof} \rangle$

9 Monotonicity w.r.t. the Bounding Atom

lemma *scl-monotone-wrt-bound*:
assumes $\bigwedge A.\ \textit{is-ground-atm}\ A \Longrightarrow A \preceq_B \beta \Longrightarrow A \preceq_B \beta'$ **and** $\textit{scl}\ N\ \beta\ S_0\ S_1$
shows $\textit{scl}\ N\ \beta'\ S_0\ S_1$
$\langle \textit{proof} \rangle$

lemma *reasonable-scl-monotone-wrt-bound*:

assumes $\bigwedge A.$ *is-ground-atm* $A \Longrightarrow A \preceq_B \beta \Longrightarrow A \preceq_B \beta'$ **and** *reasonable-scl* N β S_0 S_1
shows *reasonable-scl* N β' S_0 S_1
⟨*proof*⟩

lemma *regular-scl-monotone-wrt-bound*:
assumes $\bigwedge A.$ *is-ground-atm* $A \Longrightarrow A \preceq_B \beta \Longrightarrow A \preceq_B \beta'$ **and** *regular-scl* N β S_0 S_1
shows *regular-scl* N β' S_0 S_1
⟨*proof*⟩

lemma *min-back-regular-scl-monotone-wrt-bound*:
assumes
$\bigwedge A.$ *is-ground-atm* $A \Longrightarrow A \preceq_B \beta \Longrightarrow A \preceq_B \beta'$ **and**
shortest-backtrack-strategy regular-scl N β S_0 S_1
shows *shortest-backtrack-strategy regular-scl* N β' S_0 S_1
⟨*proof*⟩

lemma *monotonicity-wrt-bound*:
assumes $\bigwedge A.$ *is-ground-atm* $A \Longrightarrow A \preceq_B \beta \Longrightarrow A \preceq_B \beta'$
shows
scl N β S_0 $S_1 \Longrightarrow$ *scl* N β' S_0 S_1 **and**
reasonable-scl N β S_0 $S_1 \Longrightarrow$ *reasonable-scl* N β' S_0 S_1 **and**
regular-scl N β S_0 $S_1 \Longrightarrow$ *regular-scl* N β' S_0 S_1 **and**
shortest-backtrack-strategy regular-scl N β S_0 $S_1 \Longrightarrow$
shortest-backtrack-strategy regular-scl N β' S_0 S_1
⟨*proof*⟩

corollary
assumes
transp-on $\{A.$ *is-ground-atm* $A\}$ $(\prec_B)$ **and**
is-ground-atm β **and**
is-ground-atm β' **and**
$\beta \prec_B \beta'$
shows
scl N β S_0 $S_1 \Longrightarrow$ *scl* N β' S_0 S_1 **and**
reasonable-scl N β S_0 $S_1 \Longrightarrow$ *reasonable-scl* N β' S_0 S_1 **and**
regular-scl N β S_0 $S_1 \Longrightarrow$ *regular-scl* N β' S_0 S_1 **and**
shortest-backtrack-strategy regular-scl N β S_0 $S_1 \Longrightarrow$
shortest-backtrack-strategy regular-scl N β' S_0 S_1
⟨*proof*⟩

end

end
theory *Correct-Termination*
imports *SCL-FOL*
begin

context *scl-fol-calculus* **begin**

lemma *not-satisfiable-if-sound-state-conflict-bottom*:
assumes *sound-S*: *sound-state* N β S **and** *conflict-S*: *state-conflict* S = *Some* $(\{\#\}, \gamma)$
shows $\neg$ *satisfiable* (*grounding-of-clss* (*fset* N))
⟨*proof*⟩

lemma *propagate-if-conflict-follows-decide*:
assumes
trail-lt-β: *trail-atoms-lt* β S_2 **and**
no-conf: $\nexists S_1$. *conflict* N β S_0 S_1 **and** *deci*: *decide* N β S_0 S_2 **and** *conf*: *conflict* N β S_2 S_3
shows $\exists S_4$. *propagate* N β S_0 S_4
⟨*proof*⟩

theorem *correct-termination*:
fixes *gnd-N* **and** *gnd-N-lt-β*
assumes
sound-S: *sound-state* N β S **and**
invars: *trail-atoms-lt* β S *trail-propagated-wf* (*state-trail* S) *trail-lits-consistent* S
ground-false-closures S **and**
no-new-conflict: $\nexists S'$. *conflict* N β S S' **and**
no-new-propagate: $\nexists S'$. *propagate* N β S S' **and**
no-new-decide: $\nexists S'$. *decide* N β S S' $\wedge$ ($\nexists S''$. *conflict* N β S' S'') **and**
no-new-skip: $\nexists S'$. *skip* N β S S' **and**
no-new-resolve: $\nexists S'$. *resolve* N β S S' **and**
no-new-backtrack: $\nexists S'$. *backtrack* N β S S' $\wedge$
is-shortest-backtrack (*fst* (*the* (*state-conflict* S))) (*state-trail* S) (*state-trail* S')
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-β $\equiv$ $\{C \in$ *gnd-N*. $\forall L \in\# C$. *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists \gamma$. *state-conflict* S = *Some* $(\{\#\}, \gamma)$) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-β*
⟨*proof*⟩

corollary *correct-termination-strategy*:
fixes *gnd-N* **and** *gnd-N-lt-β*
assumes
run: (*strategy* N β)** *initial-state* S **and**
no-step: $\nexists S'$. *strategy* N β S S' **and**
strategy-restricted-by-min-back:
$\bigwedge S\ S'$. *shortest-backtrack-strategy regular-scl* N β S S' $\Longrightarrow$ *strategy* N β S S' **and**
strategy-preserves-invars:
$\bigwedge N\ \beta\ S\ S'$. *strategy* N β S S' $\Longrightarrow$ *sound-state* N β S $\Longrightarrow$ *sound-state* N β S'
$\bigwedge N\ \beta\ S\ S'$. *strategy* N β S S' $\Longrightarrow$ *trail-atoms-lt* β S $\Longrightarrow$ *trail-atoms-lt* β S'

$\bigwedge N\ \beta\ S\ S'.$ *strategy* $N\ \beta\ S\ S' \Longrightarrow$ *trail-propagated-or-decided'* $N\ \beta\ S \Longrightarrow$ *trail-propagated-or-decided'* $N\ \beta\ S'$
$\bigwedge N\ \beta\ S\ S'.$ *strategy* $N\ \beta\ S\ S' \Longrightarrow$ *trail-lits-consistent* $S \Longrightarrow$ *trail-lits-consistent* S'
$\bigwedge N\ \beta\ S\ S'.$ *strategy* $N\ \beta\ S\ S' \Longrightarrow$ *ground-false-closures* $S \Longrightarrow$ *ground-false-closures* S'
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-$\beta \equiv \{C \in$ *gnd-N*. $\forall L \in\# C.$ *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists\gamma.$ *state-conflict* S = *Some* ($\{\#\}, \gamma$)) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-*β
⟨*proof*⟩

corollary *correct-termination-scl-run*:
fixes *gnd-N* **and** *gnd-N-lt-*β
assumes
run: (*scl* $N\ \beta$)** *initial-state* S **and**
no-step: $\nexists S'.$ *scl* $N\ \beta\ S\ S'$
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-$\beta \equiv \{C \in$ *gnd-N*. $\forall L \in\# C.$ *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists\gamma.$ *state-conflict* S = *Some* ($\{\#\}, \gamma$)) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-*β
⟨*proof*⟩

corollary *correct-termination-reasonable-scl-run*:
fixes *gnd-N* **and** *gnd-N-lt-*β
assumes
run: (*reasonable-scl* $N\ \beta$)** *initial-state* S **and**
no-step: $\nexists S'.$ *reasonable-scl* $N\ \beta\ S\ S'$
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-$\beta \equiv \{C \in$ *gnd-N*. $\forall L \in\# C.$ *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists\gamma.$ *state-conflict* S = *Some* ($\{\#\}, \gamma$)) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-*β
⟨*proof*⟩

corollary *correct-termination-regular-scl-run*:
fixes *gnd-N* **and** *gnd-N-lt-*β
assumes
run: (*regular-scl* $N\ \beta$)** *initial-state* S **and**
no-step: $\nexists S'.$ *regular-scl* $N\ \beta\ S\ S'$
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-$\beta \equiv \{C \in$ *gnd-N*. $\forall L \in\# C.$ *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists\gamma.$ *state-conflict* S = *Some* ($\{\#\}, \gamma$)) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-*β
⟨*proof*⟩

corollary *correct-termination-shortest-backtrack-strategy-regular-scl*:
fixes *gnd-N* **and** *gnd-N-lt-*β
assumes
run: (*shortest-backtrack-strategy regular-scl* N β)** *initial-state* S **and**
no-step: $\nexists S'$. *shortest-backtrack-strategy regular-scl* N β S S'
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-β $\equiv$ $\{C \in$ *gnd-N*. $\forall L \in\# C$. *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists\gamma$. *state-conflict* S = *Some* ($\{\#\}$, γ)) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-*β
⟨*proof*⟩

corollary *correct-termination-strategies*:
fixes *gnd-N* **and** *gnd-N-lt-*β
assumes
(*scl* N β)** *initial-state* S $\wedge$ ($\nexists S'$. *scl* N β S S') $\vee$
(*reasonable-scl* N β)** *initial-state* S $\wedge$ ($\nexists S'$. *reasonable-scl* N β S S') $\vee$
(*regular-scl* N β)** *initial-state* S $\wedge$ ($\nexists S'$. *regular-scl* N β S S') $\vee$
(*shortest-backtrack-strategy regular-scl* N β)** *initial-state* S $\wedge$
($\nexists S'$. *shortest-backtrack-strategy regular-scl* N β S S')
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-β $\equiv$ $\{C \in$ *gnd-N*. $\forall L \in\# C$. *atm-of* $L \preceq_B \beta\}$
shows $\neg$ *satisfiable gnd-N* $\wedge$ ($\exists\gamma$. *state-conflict* S = *Some* ($\{\#\}$, γ)) $\vee$
satisfiable gnd-N-lt-β $\wedge$ *trail-true-clss* (*state-trail* S) *gnd-N-lt-*β
⟨*proof*⟩

end

end
theory *Trail-Induced-Ordering*
imports

Main

List−Index.List-Index
begin

lemma *wf-if-convertible-to-wf*:
fixes r :: *'a rel* **and** s :: *'b rel* **and** f :: $'a \Rightarrow 'b$
assumes *wf s* **and** *convertible*: $\bigwedge x\ y.\ (x, y) \in r \Longrightarrow (f\ x, f\ y) \in s$
shows *wf r*
⟨*proof*⟩

lemma *wfP-if-convertible-to-wfP*: *wfP* $S \Longrightarrow (\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ (f\ x)\ (f\ y)) \Longrightarrow$ *wfP* R
⟨*proof*⟩

Converting to *nat* is a very common special case that might be found more easily by Sledgehammer.

lemma *wfP-if-convertible-to-nat*:
fixes f :: - $\Rightarrow$ *nat*
shows $(\bigwedge x\ y.\ R\ x\ y \Longrightarrow f\ x < f\ y) \Longrightarrow wfP\ R$
⟨*proof*⟩

definition *trail-less-id-id* **where**
$\textit{trail-less-id-id}\ Ls\ L\ K \longleftrightarrow$
$(\exists\, i < \textit{length}\ Ls.\ \exists\, j < \textit{length}\ Ls.\ i > j \wedge L = Ls\ !\ i \wedge K = Ls\ !\ j)$

definition *trail-less-comp-id* **where**
$\textit{trail-less-comp-id}\ Ls\ L\ K \longleftrightarrow$
$(\exists\, i < \textit{length}\ Ls.\ \exists\, j < \textit{length}\ Ls.\ i > j \wedge L = -\ (Ls\ !\ i) \wedge K = Ls\ !\ j)$

definition *trail-less-id-comp* **where**
$\textit{trail-less-id-comp}\ Ls\ L\ K \longleftrightarrow$
$(\exists\, i < \textit{length}\ Ls.\ \exists\, j < \textit{length}\ Ls.\ i \geq j \wedge L = Ls\ !\ i \wedge K = -\ (Ls\ !\ j))$

definition *trail-less-comp-comp* **where**
$\textit{trail-less-comp-comp}\ Ls\ L\ K \longleftrightarrow$
$(\exists\, i < \textit{length}\ Ls.\ \exists\, j < \textit{length}\ Ls.\ i > j \wedge L = -\ (Ls\ !\ i) \wedge K = -\ (Ls\ !\ j))$

definition *trail-less* **where**
$\textit{trail-less}\ Ls\ L\ K \longleftrightarrow \textit{trail-less-id-id}\ Ls\ L\ K \vee \textit{trail-less-comp-id}\ Ls\ L\ K \vee$
$\textit{trail-less-id-comp}\ Ls\ L\ K \vee \textit{trail-less-comp-comp}\ Ls\ L\ K$

definition *trail-less′* **where**
$\textit{trail-less}'\ Ls = (\lambda L\ K.$
$(\exists\, i.\ i < \textit{length}\ Ls \wedge L = Ls\ !\ i \wedge K = -\ (Ls\ !\ i)) \vee$
$(\exists\, i.\ \textit{Suc}\ i < \textit{length}\ Ls \wedge L = -\ (Ls\ !\ \textit{Suc}\ i) \wedge K = Ls\ !\ i))^{++}$

lemma *transp-trail-less′*: $\textit{transp}\ (\textit{trail-less}'\ Ls)$
⟨*proof*⟩

lemma *trail-less′-Suc*:
assumes $\textit{Suc}\ i < \textit{length}\ Ls$
shows $\textit{trail-less}'\ Ls\ (Ls\ !\ \textit{Suc}\ i)\ (Ls\ !\ i)$
⟨*proof*⟩

lemma *trail-less′-comp-Suc-comp*:
assumes $\textit{Suc}\ i < \textit{length}\ Ls$
shows $\textit{trail-less}'\ Ls\ (-\ (Ls\ !\ \textit{Suc}\ i))\ (-\ (Ls\ !\ i))$
⟨*proof*⟩

lemma *trail-less′-id-id*: $j < i \Longrightarrow i < \textit{length}\ Ls \Longrightarrow \textit{trail-less}'\ Ls\ (Ls\ !\ i)\ (Ls\ !\ j)$

⟨*proof*⟩

lemma *trail-less′-comp-comp*:
  $j < i \Longrightarrow i < \mathit{length}\ Ls \Longrightarrow \mathit{trail\text{-}less'}\ Ls\ (-\ (Ls\ !\ i))\ (-\ (Ls\ !\ j))$
⟨*proof*⟩

lemma *trail-less′-id-comp*:
  assumes $j < i$ **and** $i < \mathit{length}\ Ls$
  shows $\mathit{trail\text{-}less'}\ Ls\ (Ls\ !\ i)\ (-\ (Ls\ !\ j))$
⟨*proof*⟩

lemma *trail-less′-comp-id*:
  assumes $j < i$ **and** $i < \mathit{length}\ Ls$
  shows $\mathit{trail\text{-}less'}\ Ls\ (-\ (Ls\ !\ i))\ (Ls\ !\ j)$
⟨*proof*⟩

lemma *trail-less-eq-trail-less′*:
  fixes $Ls :: ('a :: \mathit{uminus})\ \mathit{list}$
  assumes
    uminus-not-id: $\bigwedge x :: 'a.\ -\ x \neq x$ **and**
    uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
    pairwise-distinct:
      $\forall i < \mathit{length}\ Ls.\ \forall j < \mathit{length}\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq -\ (Ls\ !$
$j)$
  shows $\mathit{trail\text{-}less}\ Ls = \mathit{trail\text{-}less'}\ Ls$
⟨*proof*⟩

9.1 Examples

experiment
  fixes $L0\ L1\ L2 :: 'a :: \mathit{uminus}$
begin

lemma *trail-less-id-comp* $[L2,\ L1,\ L0]\ L2\ (-\ L2)$
  ⟨*proof*⟩

lemma *trail-less-comp-id* $[L2,\ L1,\ L0]\ (-\ L1)\ L2$
  ⟨*proof*⟩

lemma *trail-less-id-comp* $[L2,\ L1,\ L0]\ L1\ (-\ L1)$
  ⟨*proof*⟩

lemma *trail-less-comp-id* $[L2,\ L1,\ L0]\ (-\ L0)\ L1$
  ⟨*proof*⟩

lemma *trail-less-id-comp* $[L2,\ L1,\ L0]\ L0\ (-\ L0)$
  ⟨*proof*⟩

lemma *trail-less-id-id* $[L2,\ L1,\ L0]\ L1\ L2$

⟨*proof*⟩

lemma *trail-less-id-id* $[L2, L1, L0]$ $L0$ $L1$
⟨*proof*⟩

lemma *trail-less-comp-comp* $[L2, L1, L0]$ $(- L1)$ $(- L2)$
⟨*proof*⟩

lemma *trail-less-comp-comp* $[L2, L1, L0]$ $(- L0)$ $(- L1)$
⟨*proof*⟩

end

9.2 Miscellaneous Lemmas

lemma *not-trail-less-Nil*: $\neg$ *trail-less* [] L K
⟨*proof*⟩

lemma *defined-if-trail-less*:
assumes *trail-less* Ls L K
shows $L \in$ *set* $Ls \cup$ *uminus* ‘ *set* Ls $K \in$ *set* $Ls \cup$ *uminus* ‘ *set* Ls
⟨*proof*⟩

lemma *not-less-if-undefined*:
fixes $L :: 'a ::$ *uminus*
assumes
uminus-uminus-id: $\bigwedge x :: 'a.\ - (- x) = x$ **and**
$L \notin$ *set* Ls $- L \notin$ *set* Ls
shows $\neg$ *trail-less* Ls L K $\neg$ *trail-less* Ls K L
⟨*proof*⟩

lemma *defined-conv*:
fixes $L :: 'a ::$ *uminus*
assumes *uminus-uminus-id*: $\bigwedge x :: 'a.\ - (- x) = x$
shows $L \in$ *set* $Ls \cup$ *uminus* ‘ *set* $Ls \longleftrightarrow L \in$ *set* $Ls \vee - L \in$ *set* Ls
⟨*proof*⟩

lemma *trail-less-comp-rightI*: $L \in$ *set* $Ls \Longrightarrow$ *trail-less* Ls L $(- L)$
⟨*proof*⟩

lemma *trail-less-comp-leftI*:
fixes $Ls :: ('a ::$ *uminus*) *list*
assumes *uminus-uminus-id*: $\bigwedge x :: 'a.\ - (- x) = x$
shows $- L \in$ *set* $Ls \Longrightarrow$ *trail-less* Ls $(- L)$ L
⟨*proof*⟩

9.3 Well-Defined

lemma *trail-less-id-id-well-defined*:

assumes
pairwise-distinct: $\forall x \in set\ Ls.\ \forall y \in set\ Ls.\ x \neq -\ y$ **and**
L-le-K: *trail-less-id-id Ls L K*
shows
$\neg$ *trail-less-id-comp Ls L K*
$\neg$ *trail-less-comp-id Ls L K*
$\neg$ *trail-less-comp-comp Ls L K*
⟨*proof*⟩

lemma *trail-less-id-comp-well-defined*:
assumes
pairwise-distinct: $\forall x \in set\ Ls.\ \forall y \in set\ Ls.\ x \neq -\ y$ **and**
L-le-K: *trail-less-id-comp Ls L K*
shows
$\neg$ *trail-less-id-id Ls L K*
$\neg$ *trail-less-comp-id Ls L K*
$\neg$ *trail-less-comp-comp Ls L K*
⟨*proof*⟩

lemma *trail-less-comp-id-well-defined*:
assumes
pairwise-distinct: $\forall x \in set\ Ls.\ \forall y \in set\ Ls.\ x \neq -\ y$ **and**
L-le-K: *trail-less-comp-id Ls L K*
shows
$\neg$ *trail-less-id-id Ls L K*
$\neg$ *trail-less-id-comp Ls L K*
$\neg$ *trail-less-comp-comp Ls L K*
⟨*proof*⟩

lemma *trail-less-comp-comp-well-defined*:
assumes
pairwise-distinct: $\forall x \in set\ Ls.\ \forall y \in set\ Ls.\ x \neq -\ y$ **and**
L-le-K: *trail-less-comp-comp Ls L K*
shows
$\neg$ *trail-less-id-id Ls L K*
$\neg$ *trail-less-id-comp Ls L K*
$\neg$ *trail-less-comp-id Ls L K*
⟨*proof*⟩

9.4 Strict Partial Order

lemma *irreflp-trail-less*:
fixes $Ls :: ('a :: uminus)\ list$
assumes
uminus-not-id: $\bigwedge x :: 'a.\ -\ x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq -\ (Ls\ !\ j)$

shows *irreflp* (*trail-less Ls*)
⟨*proof*⟩

lemma *transp-trail-less*:
fixes *Ls* :: (*'a* :: *uminus*) *list*
assumes
uminus-not-id: $\bigwedge x :: 'a. - x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a. - (- x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq - (Ls\ !$
$j)$
shows *transp* (*trail-less Ls*)
⟨*proof*⟩

lemma *asymp-trail-less*:
fixes *Ls* :: (*'a* :: *uminus*) *list*
assumes
uminus-not-id: $\bigwedge x :: 'a. - x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a. - (- x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq - (Ls\ !$
$j)$
shows *asymp* (*trail-less Ls*)
⟨*proof*⟩

9.5 Strict Total (w.r.t. Elements in Trail) Order

lemma *totalp-on-trail-less*:
totalp-on (*set Ls* ∪ *uminus* ‘ *set Ls*) (*trail-less Ls*)
⟨*proof*⟩

9.6 Well-Founded

lemma *not-trail-less-Cons-id-comp*:
fixes *Ls* :: (*'a* :: *uminus*) *list*
assumes
uminus-not-id: $\bigwedge x :: 'a. - x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a. - (- x) = x$ **and**
pairwise-distinct:
$\forall i < length\ (L \# Ls).\ \forall j < length\ (L \# Ls).\ i \neq j \longrightarrow$
$(L \# Ls)\ !\ i \neq (L \# Ls)\ !\ j \wedge (L \# Ls)\ !\ i \neq - ((L \# Ls)\ !\ j)$
shows ¬ *trail-less* (*L* # *Ls*) (− *L*) *L*
⟨*proof*⟩

lemma *not-trail-less-if-undefined*:
fixes *L* :: *'a* :: *uminus*
assumes
undefined: $L \notin set\ Ls - L \notin set\ Ls$ **and**
uminus-uminus-id: $\bigwedge x :: 'a. - (- x) = x$
shows ¬ *trail-less Ls L K* ¬ *trail-less Ls K L*

$\langle proof \rangle$

lemma *trail-less-ConsD*:
fixes $L\ H\ K :: 'a :: uminus$
assumes *uminus-uminus-id*: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
L-neq-K: $L \neq K$ **and** *L-neq-minus-K*: $L \neq -\ K$ **and**
less-Cons: *trail-less* $(L \# Ls)\ H\ K$
shows *trail-less* $Ls\ H\ K$
$\langle proof \rangle$

lemma *trail-subset-empty-or-ex-smallest*:
fixes $Ls :: ('a :: uminus)\ list$
assumes
uminus-not-id: $\bigwedge x :: 'a.\ -\ x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \land Ls\ !\ i \neq -\ (Ls\ !\ j)$
shows $Q \subseteq set\ Ls \cup uminus\ `\ set\ Ls \Longrightarrow Q = \{\} \lor (\exists z \in Q.\ \forall y.\ trail\text{-}less\ Ls\ y\ z \longrightarrow y \notin Q)$
$\langle proof \rangle$

lemma *wfP-trail-less*:
fixes $Ls :: ('a :: uminus)\ list$
assumes
uminus-not-id: $\bigwedge x :: 'a.\ -\ x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \land Ls\ !\ i \neq -\ (Ls\ !\ j)$
shows *wfP* (*trail-less* Ls)
$\langle proof \rangle$

9.7 Extension on All Literals

definition *trail-less-ex* **where**
trail-less-ex $lt\ Ls\ L\ K \longleftrightarrow$
$(if\ L \in set\ Ls \lor -\ L \in set\ Ls\ then$
$if\ K \in set\ Ls \lor -\ K \in set\ Ls\ then$
trail-less $Ls\ L\ K$
else
True
else
$if\ K \in set\ Ls \lor -\ K \in set\ Ls\ then$
False
else
$lt\ L\ K)$

lemma

fixes Ls :: $('a$ :: $uminus)$ $list$
assumes
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$
shows $K \in set\ Ls \vee -\ K \in set\ Ls \Longrightarrow trail\text{-}less\text{-}ex\ lt\ Ls\ L\ K \longleftrightarrow trail\text{-}less\ Ls\ L\ K$
$\langle proof \rangle$

lemma *trail-less-ex-if-trail-less*:
fixes Ls :: $('a$ :: $uminus)$ $list$
assumes
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$
shows $trail\text{-}less\ Ls\ L\ K \Longrightarrow trail\text{-}less\text{-}ex\ lt\ Ls\ L\ K$
$\langle proof \rangle$

lemma
fixes Ls :: $('a$ :: $uminus)$ $list$
assumes
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$
shows $L \in set\ Ls \cup uminus\ `\ set\ Ls \Longrightarrow K \notin set\ Ls \cup uminus\ `\ set\ Ls \Longrightarrow trail\text{-}less\text{-}ex\ lt\ Ls\ L\ K$
$\langle proof \rangle$

lemma *irreflp-trail-ex-less*:
fixes Ls :: $('a$ :: $uminus)$ $list$ **and** $lt :: 'a \Rightarrow 'a \Rightarrow bool$
assumes
uminus-not-id: $\bigwedge x :: 'a.\ -\ x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq -\ (Ls\ !\ j)$ **and**
irreflp-lt: $irreflp\ lt$
shows $irreflp\ (trail\text{-}less\text{-}ex\ lt\ Ls)$
$\langle proof \rangle$

lemma *transp-trail-less-ex*:
fixes Ls :: $('a$ :: $uminus)$ $list$
assumes
uminus-not-id: $\bigwedge x :: 'a.\ -\ x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ -\ (-\ x) = x$ **and**
pairwise-distinct:
$\forall i < length\ Ls.\ \forall j < length\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq -\ (Ls\ !\ j)$ **and**
transp-lt: $transp\ lt$
shows $transp\ (trail\text{-}less\text{-}ex\ lt\ Ls)$
$\langle proof \rangle$

lemma *asymp-trail-less-ex*:
fixes Ls :: $('a$ :: $uminus)$ $list$

assumes
uminus-not-id: $\bigwedge x :: 'a.\ - x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ - (- x) = x$ **and**
pairwise-distinct:
$\forall i < \mathit{length}\ Ls.\ \forall j < \mathit{length}\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq - (Ls\ !\ j)$ **and**
asymp-lt: *asymp lt*
shows *asymp* (*trail-less-ex lt Ls*)
⟨*proof*⟩

lemma *totalp-on-trail-less-ex*:
fixes $Ls :: ('a :: \mathit{uminus})\ \mathit{list}$
assumes
uminus-uminus-id: $\bigwedge x :: 'a.\ - (- x) = x$ **and**
totalp-on-lt: *totalp-on A lt*
shows *totalp-on* $(A \cup \mathit{set}\ Ls \cup \mathit{uminus}\ `\ \mathit{set}\ Ls)$ (*trail-less-ex lt Ls*)
⟨*proof*⟩

9.7.1 Well-Founded

lemma *wfP-trail-less-ex*:
fixes $Ls :: ('a :: \mathit{uminus})\ \mathit{list}$
assumes
uminus-not-id: $\bigwedge x :: 'a.\ - x \neq x$ **and**
uminus-uminus-id: $\bigwedge x :: 'a.\ - (- x) = x$ **and**
pairwise-distinct:
$\forall i < \mathit{length}\ Ls.\ \forall j < \mathit{length}\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \wedge Ls\ !\ i \neq - (Ls\ !\ j)$ **and**
wfP-lt: *wfP lt*
shows *wfP* (*trail-less-ex lt Ls*)
⟨*proof*⟩

9.8 Alternative only for terms

definition *trail-term-less* **where**
trail-term-less ts t1 t2 $\longleftrightarrow (\exists i < \mathit{length}\ ts.\ \exists j < i.\ t1 = ts\ !\ i \wedge t2 = ts\ !\ j)$

lemma *transp-trail-term-less*:
assumes *distinct ts*
shows *transp* (*trail-term-less ts*)
⟨*proof*⟩

lemma *asymp-trail-term-less*:
assumes *distinct ts*
shows *asymp* (*trail-term-less ts*)
⟨*proof*⟩

lemma *irreflp-trail-term-less*:
assumes *distinct ts*
shows *irreflp* (*trail-term-less ts*)

⟨*proof*⟩

lemma *totalp-on-trail-term-less*:
shows *totalp-on* (*set ts*) (*trail-term-less ts*)
⟨*proof*⟩

lemma *wfP-trail-term-less*:
assumes *distinct ts*
shows *wfP* (*trail-term-less ts*)
⟨*proof*⟩

lemma *trail-term-less-Cons-if-mem*:
assumes $y \in$ *set xs*
shows *trail-term-less* (x # *xs*) y x
⟨*proof*⟩

end
theory *Initial-Literals-Generalize-Learned-Literals*
imports *SCL-FOL*
begin

global-interpretation *comp-finsert-commute*: *comp-fun-commute finsert*
⟨*proof*⟩

definition *fset-mset* :: *'a multiset* ⇒ *'a fset*
where *fset-mset* = *fold-mset finsert* {||}

lemma *fset-mset-mempty*[*simp*]: *fset-mset* {#} = {||}
⟨*proof*⟩

lemma *fset-mset-add-mset*[*simp*]: *fset-mset* (*add-mset x M*) = *finsert x* (*fset-mset M*)
⟨*proof*⟩

lemma *fset-fset-mset*[*simp*]: *fset* (*fset-mset M*) = *set-mset M*
⟨*proof*⟩

lemma *fmember-fset-mset-iff*[*simp*]: x |∈| *fset-mset M* ⟷ x ∈# M
⟨*proof*⟩

lemma *fBall-fset-mset-iff*[*simp*]: (∀ x |∈| *fset-mset M*. P x) ⟷ (∀ x ∈# M. P x)
⟨*proof*⟩

lemma *fBex-fset-mset-iff*[*simp*]: (∃ x |∈| *fset-mset M*. P x) ⟷ (∃ x ∈# M. P x)
⟨*proof*⟩

lemma *fmember-ffUnion-iff*: a |∈| *ffUnion* (f |\`| A) ⟷ (∃ x |∈| A. a |∈| f x)
⟨*proof*⟩

lemma *fBex-ffUnion-iff*: $(\exists\, z\ |\in|\ \textit{ffUnion}\ (f\ |`|\ A).\ P\ z) \longleftrightarrow (\exists\, x\ |\in|\ A.\ \exists\, z\ |\in|\ f\ x.\ P\ z)$
⟨*proof*⟩

lemma *fBall-ffUnion-iff*: $(\forall\, z\ |\in|\ \textit{ffUnion}\ (f\ |`|\ A).\ P\ z) \longleftrightarrow (\forall\, x\ |\in|\ A.\ \forall\, z\ |\in|\ f\ x.\ P\ z)$
⟨*proof*⟩

abbreviation *grounding-lits-of-clss* **where**
grounding-lits-of-clss $N \equiv \{L \cdot l\ \gamma \mid L\ \gamma.\ L \in \bigcup(\textit{set-mset}\ `\ N) \wedge \textit{is-ground-lit}\ (L \cdot l\ \gamma)\}$

context *scl-fol-calculus* **begin**

corollary *grounding-lits-of-learned-subset-grounding-lits-of-initial*:
assumes *initial-lits-generalize-learned-trail-conflict N S*
shows *grounding-lits-of-clss* (*fset* (*state-learned S*)) $\subseteq$ *grounding-lits-of-clss* (*fset N*)
(**is** *?lhs* $\subseteq$ *?rhs*)
⟨*proof*⟩

lemma *grounding-lits-of-clss-conv*:
grounding-lits-of-clss $N = \{L \mid L\ C.\ \textit{add-mset}\ L\ C \in \textit{grounding-of-clss}\ N\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

corollary *groundings-of-learned-subset-groundings-of-initial*:
assumes *initial-lits-generalize-learned-trail-conflict N S*
defines $U \equiv$ *state-learned S*
shows $\{L \mid L\ C.\ \textit{add-mset}\ L\ C \in \textit{grounding-of-clss}\ (\textit{fset}\ U)\} \subseteq$
$\{L \mid L\ C.\ \textit{add-mset}\ L\ C \in \textit{grounding-of-clss}\ (\textit{fset}\ N)\}$
⟨*proof*⟩

end

end
theory *Multiset-Order-Extra*
imports *HOL−Library.Multiset-Order*
begin

lemma *strict-subset-implies-multp*$_{HO}$: $A \subset\#\ B \Longrightarrow \textit{multp}_{HO}\ r\ A\ B$
⟨*proof*⟩

end
theory *Non-Redundancy*
imports
SCL-FOL
Trail-Induced-Ordering

Initial-Literals-Generalize-Learned-Literals
Multiset-Order-Extra
begin

context *scl-fol-calculus* **begin**

10 Reasonable Steps

lemma *reasonable-scl-sound-state*:
$\textit{reasonable-scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{sound-state}\ N\ \beta\ S \Longrightarrow \textit{sound-state}\ N\ \beta\ S'$
⟨*proof*⟩

lemma *reasonable-run-sound-state*:
$(\textit{reasonable-scl}\ N\ \beta)^{**}\ S\ S' \Longrightarrow \textit{sound-state}\ N\ \beta\ S \Longrightarrow \textit{sound-state}\ N\ \beta\ S'$
⟨*proof*⟩

10.1 Invariants

10.1.1 No Conflict After Decide

inductive *no-conflict-after-decide* **for** $N\ \beta\ U$ **where**
$\textit{Nil}[\textit{simp}]$: $\textit{no-conflict-after-decide}\ N\ \beta\ U\ []$ |
Cons: $(\textit{is-decision-lit}\ Ln \longrightarrow (\nexists S'.\ \textit{conflict}\ N\ \beta\ (Ln\ \#\ \Gamma,\ U,\ \textit{None})\ S')) \Longrightarrow$
$\textit{no-conflict-after-decide}\ N\ \beta\ U\ \Gamma \Longrightarrow \textit{no-conflict-after-decide}\ N\ \beta\ U\ (Ln\ \#\ \Gamma)$

definition *no-conflict-after-decide'* **where**
$\textit{no-conflict-after-decide}'\ N\ \beta\ S = \textit{no-conflict-after-decide}\ N\ \beta\ (\textit{state-learned}\ S)$ $(\textit{state-trail}\ S)$

lemma *no-conflict-after-decide'-initial-state*[*simp*]: $\textit{no-conflict-after-decide}'\ N\ \beta\ \textit{initial-state}$
⟨*proof*⟩

lemma *propagate-preserves-no-conflict-after-decide'*:
assumes $\textit{propagate}\ N\ \beta\ S\ S'$ **and** $\textit{no-conflict-after-decide}'\ N\ \beta\ S$
shows $\textit{no-conflict-after-decide}'\ N\ \beta\ S'$
⟨*proof*⟩

lemma *decide-preserves-no-conflict-after-decide'*:
assumes $\textit{decide}\ N\ \beta\ S\ S'$ **and** $\nexists S''.\ \textit{conflict}\ N\ \beta\ S'\ S''$ **and** $\textit{no-conflict-after-decide}'\ N\ \beta\ S$
shows $\textit{no-conflict-after-decide}'\ N\ \beta\ S'$
⟨*proof*⟩

lemma *conflict-preserves-no-conflict-after-decide'*:
assumes $\textit{conflict}\ N\ \beta\ S\ S'$ **and** $\textit{no-conflict-after-decide}'\ N\ \beta\ S$
shows $\textit{no-conflict-after-decide}'\ N\ \beta\ S'$
⟨*proof*⟩

lemma *skip-preserves-no-conflict-after-decide'*:
assumes *skip* $N\ \beta\ S\ S'$ **and** *no-conflict-after-decide'* $N\ \beta\ S$
shows *no-conflict-after-decide'* $N\ \beta\ S'$
⟨*proof*⟩

lemma *factorize-preserves-no-conflict-after-decide'*:
assumes *factorize* $N\ \beta\ S\ S'$ **and** *no-conflict-after-decide'* $N\ \beta\ S$
shows *no-conflict-after-decide'* $N\ \beta\ S'$
⟨*proof*⟩

lemma *resolve-preserves-no-conflict-after-decide'*:
assumes *resolve* $N\ \beta\ S\ S'$ **and** *no-conflict-after-decide'* $N\ \beta\ S$
shows *no-conflict-after-decide'* $N\ \beta\ S'$
⟨*proof*⟩

lemma *learning-clause-without-conflict-preserves-nex-conflict*:
fixes N :: $('f,\ 'v)$ *Term.term clause fset*
assumes $\nexists\gamma.$ *is-ground-cls* $(C \cdot \gamma) \land$ *trail-false-cls* $\Gamma\ (C \cdot \gamma)$
shows $\nexists S'.$ *conflict* $N\ \beta\ (\Gamma,\ U,\ None)\ S' \Longrightarrow \nexists S'.$ *conflict* $N\ \beta\ (\Gamma,$ *finsert* $C\ U,$ $None)\ S'$
⟨*proof*⟩

lemma *backtrack-preserves-no-conflict-after-decide'*:
assumes *step*: *backtrack* $N\ \beta\ S\ S'$ **and** *invar*: *no-conflict-after-decide'* $N\ \beta\ S$
shows *no-conflict-after-decide'* $N\ \beta\ S'$
⟨*proof*⟩

lemma *reasonable-scl-preserves-no-conflict-after-decide'*:
assumes *reasonable-scl* $N\ \beta\ S\ S'$ **and** *no-conflict-after-decide'* $N\ \beta\ S$
shows *no-conflict-after-decide'* $N\ \beta\ S'$
⟨*proof*⟩

10.2 Miscellaneous Lemmas

lemma *before-reasonable-conflict*:
assumes *conf*: *conflict* $N\ \beta\ S1\ S2$ **and**
invars: *learned-nonempty* $S1$ *trail-propagated-or-decided'* $N\ \beta\ S1$
no-conflict-after-decide' $N\ \beta\ S1$
shows $\{\#\}\ |\in|\ N \lor (\exists S0.$ *propagate* $N\ \beta\ S0\ S1)$
⟨*proof*⟩

11 Regular Steps

lemma *regular-scl-if-conflict*[*simp*]: *conflict* $N\ \beta\ S\ S' \Longrightarrow$ *regular-scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

lemma *regular-scl-if-skip*[*simp*]: *skip* $N\ \beta\ S\ S' \Longrightarrow$ *regular-scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

lemma *regular-scl-if-factorize*[*simp*]: *factorize* $N\ \beta\ S\ S' \Longrightarrow$ *regular-scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

lemma *regular-scl-if-resolve*[*simp*]: *resolve* $N\ \beta\ S\ S' \Longrightarrow$ *regular-scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

lemma *regular-scl-if-backtrack*[*simp*]: *backtrack* $N\ \beta\ S\ S' \Longrightarrow$ *regular-scl* $N\ \beta\ S\ S'$
⟨*proof*⟩

lemma *regular-scl-sound-state*: *regular-scl* $N\ \beta\ S\ S' \Longrightarrow$ *sound-state* $N\ \beta\ S \Longrightarrow$ *sound-state* $N\ \beta\ S'$
⟨*proof*⟩

lemma *regular-run-sound-state*:
(*regular-scl* $N\ \beta)^{**}\ S\ S' \Longrightarrow$ *sound-state* $N\ \beta\ S \Longrightarrow$ *sound-state* $N\ \beta\ S'$
⟨*proof*⟩

11.1 Invariants

11.1.1 Almost No Conflict With Trail

inductive *no-conflict-with-trail* **for** $N\ \beta\ U$ **where**
Nil: ($\nexists S'$. *conflict* $N\ \beta$ ([], U, *None*) S') $\Longrightarrow$ *no-conflict-with-trail* $N\ \beta\ U$ [] |
Cons: ($\nexists S'$. *conflict* $N\ \beta$ (*Ln* # Γ, U, *None*) S') $\Longrightarrow$
no-conflict-with-trail $N\ \beta\ U\ \Gamma \Longrightarrow$ *no-conflict-with-trail* $N\ \beta\ U$ (*Ln* # Γ)

lemma *nex-conflict-if-no-conflict-with-trail*:
assumes *no-conflict-with-trail* $N\ \beta\ U\ \Gamma$
shows $\nexists S'$. *conflict* $N\ \beta$ (Γ, U, *None*) S'
⟨*proof*⟩

lemma *nex-conflict-if-no-conflict-with-trail'*:
assumes *no-conflict-with-trail* $N\ \beta\ U\ \Gamma$
shows $\nexists S'$. *conflict* $N\ \beta$ ([], U, *None*) S'
⟨*proof*⟩

lemma *no-conflict-after-decide-if-no-conflict-with-trail*:
no-conflict-with-trail $N\ \beta\ U\ \Gamma \Longrightarrow$ *no-conflict-after-decide* $N\ \beta\ U\ \Gamma$
⟨*proof*⟩

lemma *not-trail-false-cls-if-no-conflict-with-trail*:
no-conflict-with-trail $N\ \beta\ U\ \Gamma \Longrightarrow D \mid\in\mid N \mid\cup\mid U \Longrightarrow D \neq \{\#\} \Longrightarrow$ *is-ground-cls* $(D \cdot \gamma) \Longrightarrow$
$\neg$ *trail-false-cls* Γ $(D \cdot \gamma)$
⟨*proof*⟩

definition *almost-no-conflict-with-trail* **where**
almost-no-conflict-with-trail $N\ \beta\ S \longleftrightarrow$
$\{\#\} \mid\in\mid N \wedge$ *state-trail* S = [] $\vee$
no-conflict-with-trail $N\ \beta$ (*state-learned* S)

(*case state-trail S of* [] ⇒ [] | *Ln* # Γ ⇒ *if is-decision-lit Ln then Ln* # Γ *else* Γ)

lemma *nex-conflict-if-no-conflict-with-trail''*:
assumes *no-conf*: *state-conflict S = None* **and** {#} |∉| *N* **and** *learned-nonempty S*
no-conflict-with-trail N β (*state-learned S*) (*state-trail S*)
shows $\nexists S'$. *conflict N β S S'*
⟨*proof*⟩

lemma *no-conflict-with-trail-if-nex-conflict*:
assumes *no-conf*: $\nexists S'$. *conflict N β S S' state-conflict S = None*
shows *no-conflict-with-trail N β* (*state-learned S*) (*state-trail S*)
⟨*proof*⟩

lemma *almost-no-conflict-with-trail-if-no-conflict-with-trail*:
no-conflict-with-trail N β U Γ ⟹ *almost-no-conflict-with-trail N β* (Γ, *U*, *Cl*)
⟨*proof*⟩

lemma *almost-no-conflict-with-trail-initial-state*[*simp*]:
almost-no-conflict-with-trail N β initial-state
⟨*proof*⟩

lemma *propagate-preserves-almost-no-conflict-with-trail*:
assumes *step*: *propagate N β S S'* **and** *reg-step*: *regular-scl N β S S'*
shows *almost-no-conflict-with-trail N β S'*
⟨*proof*⟩

lemma *decide-preserves-almost-no-conflict-with-trail*:
assumes *step*: *decide N β S S'* **and** *reg-step*: *regular-scl N β S S'*
shows *almost-no-conflict-with-trail N β S'*
⟨*proof*⟩

lemma *almost-no-conflict-with-trail-conflict-not-relevant*:
almost-no-conflict-with-trail N β (Γ, *U*, *Cl1*) ⟷
almost-no-conflict-with-trail N β (Γ, *U*, *Cl2*)
⟨*proof*⟩

lemma *conflict-preserves-almost-no-conflict-with-trail*:
assumes *step*: *conflict N β S S'* **and** *invar*: *almost-no-conflict-with-trail N β S*
shows *almost-no-conflict-with-trail N β S'*
⟨*proof*⟩

lemma *skip-preserves-almost-no-conflict-with-trail*:
assumes *step*: *skip N β S S'* **and** *invar*: *almost-no-conflict-with-trail N β S*
shows *almost-no-conflict-with-trail N β S'*
⟨*proof*⟩

lemma *factorize-preserves-almost-no-conflict-with-trail*:

assumes *step*: $factorize\ N\ \beta\ S\ S'$ **and** *invar*: $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S$
shows $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S'$
⟨*proof*⟩

lemma *resolve-preserves-almost-no-conflict-with-trail*:
assumes *step*: $resolve\ N\ \beta\ S\ S'$ **and** *invar*: $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S$
shows $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S'$
⟨*proof*⟩

lemma *backtrack-preserves-almost-no-conflict-with-trail*:
assumes *step*: $backtrack\ N\ \beta\ S\ S'$ **and** *invar*: $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta$
S
shows $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S'$
⟨*proof*⟩

lemma *regular-scl-preserves-almost-no-conflict-with-trail*:
assumes $regular\text{-}scl\ N\ \beta\ S\ S'$ **and** $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S$
shows $almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta\ S'$
⟨*proof*⟩

11.1.2 Backtrack Follows Regular Conflict Resolution

lemma *before-conflict-in-regular-run*:
assumes
reg-run: $(regular\text{-}scl\ N\ \beta)^{**}\ initial\text{-}state\ S1$ **and**
conf: $conflict\ N\ \beta\ S1\ S2$ **and**
$\{\#\}\ |\notin|\ N$
shows $\exists\, S0.\ (regular\text{-}scl\ N\ \beta)^{**}\ initial\text{-}state\ S0 \land regular\text{-}scl\ N\ \beta\ S0\ S1 \land$
$(propagate\ N\ \beta\ S0\ S1)$
⟨*proof*⟩

definition *regular-conflict-resolution* **where**
$regular\text{-}conflict\text{-}resolution\ N\ \beta\ S \longleftrightarrow \{\#\}\ |\notin|\ N \longrightarrow$
$(case\ state\text{-}conflict\ S\ of$
$None \Rightarrow (regular\text{-}scl\ N\ \beta)^{**}\ initial\text{-}state\ S\ |$
$Some\ \text{-} \Rightarrow (\exists\, S0\ S1\ S2\ S3.\ (regular\text{-}scl\ N\ \beta)^{**}\ initial\text{-}state\ S0 \land$
$propagate\ N\ \beta\ S0\ S1 \land regular\text{-}scl\ N\ \beta\ S0\ S1 \land$
$conflict\ N\ \beta\ S1\ S2 \land regular\text{-}scl\ N\ \beta\ S1\ S2 \land$
$(factorize\ N\ \beta)^{**}\ S2\ S3 \land (regular\text{-}scl\ N\ \beta)^{**}\ S2\ S3 \land$
$(S3 = S \lor (\exists\, S4.\ resolve\ N\ \beta\ S3\ S4 \land (skip\ N\ \beta \sqcup factorize\ N\ \beta \sqcup resolve$
$N\ \beta)^{**}\ S4\ S))))$

lemma *regular-conflict-resolution-initial-state*[*simp*]:
$regular\text{-}conflict\text{-}resolution\ N\ \beta\ initial\text{-}state$
⟨*proof*⟩

lemma *propagate-preserves-regular-conflict-resolution*:
assumes *step*: $propagate\ N\ \beta\ S\ S'$ **and** *reg-step*: $regular\text{-}scl\ N\ \beta\ S\ S'$ **and**
invar: $regular\text{-}conflict\text{-}resolution\ N\ \beta\ S$

shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma *decide-preserves-regular-conflict-resolution*:
assumes *step*: *decide* $N\ \beta\ S\ S'$ **and** *reg-step*: *regular-scl* $N\ \beta\ S\ S'$ **and**
invar: *regular-conflict-resolution* $N\ \beta\ S$
shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma *conflict-preserves-regular-conflict-resolution*:
assumes *step*: *conflict* $N\ \beta\ S\ S'$ **and** *reg-step*: *regular-scl* $N\ \beta\ S\ S'$ **and**
invar: *regular-conflict-resolution* $N\ \beta\ S$
shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma
assumes *almost-no-conflict-with-trail* $N\ \beta\ S$ **and** $\{\#\}\ |\notin|\ N$
shows *no-conflict-after-decide'* $N\ \beta\ S$
⟨*proof*⟩

lemma *mempty-not-in-learned-if-almost-no-conflict-with-trail*:
almost-no-conflict-with-trail $N\ \beta\ S \Longrightarrow \{\#\}\ |\notin|\ N \Longrightarrow \{\#\}\ |\notin|$ *state-learned* S
⟨*proof*⟩

lemma *skip-preserves-regular-conflict-resolution*:
assumes *step*: *skip* $N\ \beta\ S\ S'$ **and** *reg-step*: *regular-scl* $N\ \beta\ S\ S'$ **and**
invar: *regular-conflict-resolution* $N\ \beta\ S$
shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma *factorize-preserves-regular-conflict-resolution*:
assumes *step*: *factorize* $N\ \beta\ S\ S'$ **and** *reg-step*: *regular-scl* $N\ \beta\ S\ S'$ **and**
invar: *regular-conflict-resolution* $N\ \beta\ S$
shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma *resolve-preserves-regular-conflict-resolution*:
assumes *step*: *resolve* $N\ \beta\ S\ S'$ **and** *reg-step*: *regular-scl* $N\ \beta\ S\ S'$ **and**
invar: *regular-conflict-resolution* $N\ \beta\ S$
shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma *backtrack-preserves-regular-conflict-resolution*:
assumes *step*: *backtrack* $N\ \beta\ S\ S'$ **and** *reg-step*: *regular-scl* $N\ \beta\ S\ S'$ **and**
invar: *regular-conflict-resolution* $N\ \beta\ S$
shows *regular-conflict-resolution* $N\ \beta\ S'$
⟨*proof*⟩

lemma *regular-scl-preserves-regular-conflict-resolution*:

assumes *reg-step*: *regular-scl* N β S S' **and**
invars: *regular-conflict-resolution* N β S
shows *regular-conflict-resolution* N β S'
⟨*proof*⟩

11.2 Miscellaneous Lemmas

lemma *mempty-not-in-initial-clauses-if-non-empty-regular-conflict*:
assumes *state-conflict* S = *Some* (C, γ) **and** $C \neq \{\#\}$ **and**
invars: *almost-no-conflict-with-trail* N β S *sound-state* N β S *ground-false-closures* S
shows $\{\#\}$ $|\notin|$ N
⟨*proof*⟩

lemma *mempty-not-in-initial-clauses-if-regular-run-reaches-non-empty-conflict*:
assumes $($*regular-scl* N $\beta)^{**}$ *initial-state* S **and** *state-conflict* S = *Some* (C, γ) **and** $C \neq \{\#\}$
shows $\{\#\}$ $|\notin|$ N
⟨*proof*⟩

lemma *before-regular-backtrack*:
assumes
backt: *backtrack* N β S S' **and**
invars: *sound-state* N β S *almost-no-conflict-with-trail* N β S
regular-conflict-resolution N β S *ground-false-closures* S
shows $\exists$ *S0 S1 S2 S3 S4*. $($*regular-scl* N $\beta)^{**}$ *initial-state S0* $\wedge$
propagate N β *S0 S1* $\wedge$ *regular-scl* N β *S0 S1* $\wedge$
conflict N β *S1 S2* $\wedge$ $($*factorize* N $\beta)^{**}$ *S2 S3* $\wedge$ *resolve* N β *S3 S4* $\wedge$
$($*skip* N β $\sqcup$ *factorize* N β $\sqcup$ *resolve* N $\beta)^{**}$ *S4* S
⟨*proof*⟩

12 Resolve in Regular Runs

lemma *resolve-if-conflict-follows-propagate*:
assumes
no-conf: $\nexists S_1$. *conflict* N β S_0 S_1 **and**
propa: *propagate* N β S_0 S_1 **and**
conf: *conflict* N β S_1 S_2
shows $\exists S_3$. *resolve* N β S_2 S_3
⟨*proof*⟩

lemma *factorize-preserves-resolvability*:
assumes *reso*: *resolve* N β S_1 S_2 **and** *fact*: *factorize* N β S_1 S_3 **and**
invar: *ground-closures* S_1
shows $\exists S_4$. *resolve* N β S_3 S_4
⟨*proof*⟩

The following lemma corresponds to Lemma 7 in the paper.

lemma *no-backtrack-after-conflict-if*:

assumes *conf*: *conflict N β S1 S2* **and** *trail-S2*: *state-trail S1* = *trail-propagate* Γ *L C* γ
shows ∄*S4*. *backtrack N β S2 S4*
⟨*proof*⟩

lemma *skip-state-trail*: *skip N β S S'* ⟹ *suffix* (*state-trail S'*) (*state-trail S*)
⟨*proof*⟩

lemma *factorize-state-trail*: *factorize N β S S'* ⟹ *state-trail S'* = *state-trail S*
⟨*proof*⟩

lemma *resolve-state-trail*: *resolve N β S S'* ⟹ *state-trail S'* = *state-trail S*
⟨*proof*⟩

lemma *mempty-not-in-initial-clauses-if-run-leads-to-trail*:
assumes
reg-run: (*regular-scl N β*)** *initial-state S1* **and**
trail-lit: *state-trail S1* = *Lc* # Γ
shows {#} |∉| *N*
⟨*proof*⟩

lemma *conflict-with-literal-gets-resolved*:
assumes
trail-lit: *state-trail S1* = *Lc* # Γ **and**
conf: *conflict N β S1 S2* **and**
resolution: (*skip N β* ⊔ *factorize N β* ⊔ *resolve N β*)** *S2 Sn* **and**
backtrack: ∃*Sn'*. *backtrack N β Sn Sn'* **and**
mempty-not-in-init-clss: {#} |∉| *N* **and**
invars: *learned-nonempty S1 trail-propagated-or-decided' N β S1 no-conflict-after-decide' N β S1*
shows ¬ *is-decision-lit Lc* ∧ *strict-suffix* (*state-trail Sn*) (*state-trail S1*)
⟨*proof*⟩

13 Clause Redundancy

definition *ground-redundant* **where**
ground-redundant lt N C ⟷ {*D* ∈ *N*. *lt D C*} ⊫*e* {*C*}

definition *redundant* **where**
redundant lt N C ⟷
(∀ *C'* ∈ *grounding-of-cls C*. *ground-redundant lt* (*grounding-of-clss N*) *C'*)

lemma *redundant lt N C* ⟷ (∀ *C'*∈ *grounding-of-cls C*. {*D'* ∈ *grounding-of-clss N*. *lt D' C'*} ⊫*e* {*C'*})
⟨*proof*⟩

lemma *ground-redundant-iff*:

ground-redundant lt N C $\longleftrightarrow$ $(\exists\, M \subseteq N.\ M \models e\ \{C\} \wedge (\forall\, D \in M.\ lt\ D\ C))$
$\langle proof \rangle$

lemma *ground-redundant-is-ground-standard-redundancy*:
fixes *lt*
defines $Red\text{-}F_{\mathcal{G}} \equiv \lambda N.\ \{C.\ ground\text{-}redundant\ lt\ N\ C\}$
shows $Red\text{-}F_{\mathcal{G}}\ N = \{C.\ \exists\, M \subseteq N.\ M \models e\ \{C\} \wedge (\forall\, D \in M.\ lt\ D\ C)\}$
$\langle proof \rangle$

lemma *redundant-is-standard-redundancy*:
fixes $lt\ \mathcal{G}_F\ \mathcal{G}_{F\,s}\ Red\text{-}F_{\mathcal{G}}\ Red\text{-}F$
defines
$\mathcal{G}_F \equiv grounding\text{-}of\text{-}cls$ **and**
$\mathcal{G}_{F\,s} \equiv grounding\text{-}of\text{-}clss$ **and**
$Red\text{-}F_{\mathcal{G}} \equiv \lambda N.\ \{C.\ ground\text{-}redundant\ lt\ N\ C\}$ **and**
$Red\text{-}F \equiv \lambda N.\ \{C.\ redundant\ lt\ N\ C\}$
shows $Red\text{-}F\ N = \{C.\ \forall\, D \in \mathcal{G}_F\ C.\ D \in Red\text{-}F_{\mathcal{G}}\ (\mathcal{G}_{F\,s}\ N)\}$
$\langle proof \rangle$

lemma *ground-redundant-if-strict-subset*:
assumes $D \in N$ **and** $D \subset\#\ C$
shows *ground-redundant* $(multp_{HO}\ R)\ N\ C$
$\langle proof \rangle$

lemma *redundant-if-strict-subset*:
assumes $D \in N$ **and** $D \subset\#\ C$
shows *redundant* $(multp_{HO}\ R)\ N\ C$
$\langle proof \rangle$

lemma *redundant-if-strict-subsumes*:
assumes $D \cdot \sigma \subset\#\ C$ **and** $D \in N$
shows *redundant* $(multp_{HO}\ R)\ N\ C$
$\langle proof \rangle$

lemma *ground-redundant-mono-strong*:
ground-redundant $R\ N\ C \Longrightarrow (\bigwedge x.\ x \in N \Longrightarrow R\ x\ C \Longrightarrow S\ x\ C) \Longrightarrow$ *ground-redundant*
$S\ N\ C$
$\langle proof \rangle$

lemma *redundant-mono-strong*:
redundant $R\ N\ C \Longrightarrow$
$(\bigwedge x\ y.\ x \in grounding\text{-}of\text{-}clss\ N \Longrightarrow y \in grounding\text{-}of\text{-}cls\ C \Longrightarrow R\ x\ y \Longrightarrow S\ x$
$y) \Longrightarrow$
redundant $S\ N\ C$
$\langle proof \rangle$

lemma *redundant-multp-if-redundant-strict-subset*:
redundant $(\subset\#)\ N\ C \Longrightarrow$ *redundant* $(multp_{HO}\ R)\ N\ C$
$\langle proof \rangle$

lemma *redundant-multp-if-redundant-subset*:
redundant $(\subset\#)$ *N C* $\Longrightarrow$ *redundant* (*multp* (*trail-less-ex lt Ls*)) *N C*
⟨*proof*⟩

lemma *not-bex-subset-mset-if-not-ground-redundant*:
assumes *is-ground-cls C* **and** *is-ground-clss N*
shows $\neg$ *ground-redundant* $(\subset\#)$ *N C* $\Longrightarrow \neg (\exists D \in N.\ D \subset\# C)$
⟨*proof*⟩

14 Trail-Induced Ordering

14.1 Miscellaneous Lemmas

lemma *pairwise-distinct-if-trail-consistent*:
fixes Γ
defines $Ls \equiv$ (*map fst* Γ)
shows *trail-consistent* $\Gamma \Longrightarrow$
$\forall i < \mathit{length}\ Ls.\ \forall j < \mathit{length}\ Ls.\ i \neq j \longrightarrow Ls\ !\ i \neq Ls\ !\ j \land Ls\ !\ i \neq -\ (Ls\ !\ j)$
⟨*proof*⟩

14.2 Strict Partial Order

lemma *irreflp-trail-less-if-trail-consistant*:
trail-consistent $\Gamma \Longrightarrow$ *irreflp* (*trail-less* (*map fst* Γ))
⟨*proof*⟩

lemma *transp-trail-less-if-trail-consistant*:
trail-consistent $\Gamma \Longrightarrow$ *transp* (*trail-less* (*map fst* Γ))
⟨*proof*⟩

lemma *asymp-trail-less-if-trail-consistant*:
trail-consistent $\Gamma \Longrightarrow$ *asymp* (*trail-less* (*map fst* Γ))
⟨*proof*⟩

14.3 Properties

lemma *trail-defined-lit-if-trail-term-less*:
assumes *trail-term-less* (*map* (*atm-of o fst*) Γ) (*atm-of L*) (*atm-of K*)
shows *trail-defined-lit* Γ *L trail-defined-lit* Γ *K*
⟨*proof*⟩

lemma *trail-defined-cls-if-lt-defined*:
assumes *consistent*-Γ: *trail-consistent* Γ **and**
C-lt-D: multp_{HO} (*lit-less* (*trail-term-less* (*map* (*atm-of o fst*) Γ))) *C D* **and**
tr-def-D: *trail-defined-cls* Γ *D* **and**
lit-less-preserves-term-order: $\bigwedge R\ L1\ L2.\ \mathit{lit\text{-}less}\ R\ L1\ L2 \Longrightarrow R^{==}\ (\mathit{atm\text{-}of}\ L1)$ (*atm-of L2*)
shows *trail-defined-cls* Γ *C*

⟨*proof*⟩

15 Dynamic Non-Redundancy

lemma *regular-run-if-skip-factorize-resolve-run*:
  assumes $(skip\ N\ \beta \sqcup factorize\ N\ \beta \sqcup resolve\ N\ \beta)^{**}\ S\ S'$
  shows $(regular\text{-}scl\ N\ \beta)^{**}\ S\ S'$
  ⟨*proof*⟩

lemma *not-trail-true-and-false-lit*:
  $trail\text{-}consistent\ \Gamma \Longrightarrow \neg\ (trail\text{-}true\text{-}lit\ \Gamma\ L \wedge trail\text{-}false\text{-}lit\ \Gamma\ L)$
  ⟨*proof*⟩

lemma *not-trail-true-and-false-cls*:
  $trail\text{-}consistent\ \Gamma \Longrightarrow \neg\ (trail\text{-}true\text{-}cls\ \Gamma\ C \wedge trail\text{-}false\text{-}cls\ \Gamma\ C)$
  ⟨*proof*⟩

fun *standard-lit-less* **where**
  $standard\text{-}lit\text{-}less\ R\ (Pos\ t1)\ (Pos\ t2) = R\ t1\ t2\ |$
  $standard\text{-}lit\text{-}less\ R\ (Pos\ t1)\ (Neg\ t2) = R^{==}\ t1\ t2\ |$
  $standard\text{-}lit\text{-}less\ R\ (Neg\ t1)\ (Pos\ t2) = R\ t1\ t2\ |$
  $standard\text{-}lit\text{-}less\ R\ (Neg\ t1)\ (Neg\ t2) = R\ t1\ t2$

lemma *standard-lit-less-preserves-term-less*:
  shows $standard\text{-}lit\text{-}less\ R\ L1\ L2 \Longrightarrow R^{==}\ (atm\text{-}of\ L1)\ (atm\text{-}of\ L2)$
  ⟨*proof*⟩

theorem *learned-clauses-in-regular-runs-invars*:
  fixes Γ *lit-less*
  assumes
    sound-S0: *sound-state* $N\ \beta$ *S0* **and**
    invars: *learned-nonempty S0 trail-propagated-or-decided′* $N\ \beta$ *S0*
      no-conflict-after-decide′ $N\ \beta$ *S0 almost-no-conflict-with-trail* $N\ \beta$ *S0*
      trail-lits-consistent S0 trail-closures-false′ S0 ground-false-closures S0 **and**
    conflict: *conflict* $N\ \beta$ *S0 S1* **and**
    resolution: $(skip\ N\ \beta \sqcup factorize\ N\ \beta \sqcup resolve\ N\ \beta)^{++}$ *S1 Sn* **and**
    backtrack: *backtrack* $N\ \beta$ *Sn Sn′* **and**
   lit-less-preserves-term-order: $\bigwedge R\ L1\ L2.\ lit\text{-}less\ R\ L1\ L2 \Longrightarrow R^{==}\ (atm\text{-}of\ L1)$
$(atm\text{-}of\ L2)$
  defines
    $\Gamma \equiv$ *state-trail S1* **and**
    $U \equiv$ *state-learned S1* **and**
    $trail\text{-}ord \equiv multp_{HO}\ (lit\text{-}less\ (trail\text{-}term\text{-}less\ (map\ (atm\text{-}of\ o\ fst)\ \Gamma)))$
  shows $(\exists\ C\ \gamma.\ state\text{-}conflict\ Sn = Some\ (C, \gamma) \wedge$
    $C \cdot \gamma \notin grounding\text{-}of\text{-}clss\ (fset\ N \cup fset\ U) \wedge$
    $set\text{-}mset\ (C \cdot \gamma) \notin set\text{-}mset$ ‘ $grounding\text{-}of\text{-}clss\ (fset\ N \cup fset\ U) \wedge$
    $C \notin (fset\ N \cup fset\ U) \wedge$
    $\neg\ (\exists\ D \in fset\ N \cup fset\ U.\ \exists \sigma.\ D \cdot \sigma = C) \wedge$
    $\neg\ redundant\ trail\text{-}ord\ (fset\ N \cup fset\ U)\ C)$

⟨*proof*⟩

theorem *dynamic-non-redundancy-regular-scl*:
fixes Γ
assumes
regular-run: $(\textit{regular-scl } N\ \beta)^{**}$ *initial-state S0* **and**
conflict: *conflict N β S0 S1* **and**
resolution: $(\textit{skip } N\ \beta \sqcup \textit{factorize } N\ \beta \sqcup \textit{resolve } N\ \beta)^{++}$ *S1 Sn* **and**
backtrack: *backtrack N β Sn Sn′* **and**
lit-less-preserves-term-order: $\bigwedge R\ L1\ L2.\ \textit{lit-less } R\ L1\ L2 \Longrightarrow R^{==}\ (\textit{atm-of } L1)$ $(\textit{atm-of } L2)$
defines
Γ ≡ *state-trail S1* **and**
U ≡ *state-learned S1* **and**
trail-ord ≡ $\textit{multp}_{HO}$ (*lit-less* (*trail-term-less* (*map* (*atm-of o fst*) Γ)))
shows $(\textit{regular-scl } N\ \beta)^{**}$ *initial-state Sn′* ∧
(∃ *C γ*. *state-conflict Sn* = *Some* (*C*, *γ*) ∧
C · *γ* ∉ *grounding-of-clss* (*fset N* ∪ *fset U*) ∧
set-mset (*C* · *γ*) ∉ *set-mset* ‘ *grounding-of-clss* (*fset N* ∪ *fset U*) ∧
C ∉ *fset N* ∪ *fset U* ∧
¬ (∃ *D* ∈ *fset N* ∪ *fset U*. ∃ *σ*. *D* · *σ* = *C*) ∧
¬ *redundant trail-ord* (*fset N* ∪ *fset U*) *C*)
⟨*proof*⟩

theorem *dynamic-non-redundancy-projectable-strategy*:
fixes
S1 :: (′*f*, ′*v*) *state* **and**
lit-less :: ((′*f*, ′*v*) *term* ⇒ (′*f*, ′*v*) *term* ⇒ *bool*) ⇒
(′*f*, ′*v*) *term literal* ⇒ (′*f*, ′*v*) *term literal* ⇒ *bool* **and**
strategy **and** *strategy-init* **and** *proj*
defines
Γ ≡ *state-trail S1* **and**
U ≡ *state-learned S1*
defines
trail-ord ≡ $\textit{multp}_{HO}$ (*lit-less* (*trail-term-less* (*map* (*atm-of o fst*) Γ)))
assumes
run: $\textit{strategy}^{**}$ *strategy-init S0* **and**
conflict: *conflict N β* (*proj S0*) *S1* **and**
resolution: $(\textit{skip } N\ \beta \sqcup \textit{factorize } N\ \beta \sqcup \textit{resolve } N\ \beta)^{++}$ *S1 Sn* **and**
backtrack: *backtrack N β Sn Sn′* **and**
strategy-restricts-regular-scl:
$\bigwedge S\ S'.\ \textit{strategy}^{**}\ \textit{strategy-init } S \Longrightarrow \textit{strategy } S\ S' \Longrightarrow \textit{regular-scl } N\ \beta\ (\textit{proj}$ $S)\ (\textit{proj } S')$ **and**
initial-state: *proj strategy-init* = *initial-state* **and**
lit-less-preserves-term-order: $\bigwedge R\ L1\ L2.\ \textit{lit-less } R\ L1\ L2 \Longrightarrow R^{==}\ (\textit{atm-of } L1)$ $(\textit{atm-of } L2)$
shows (∃ *C γ*. *state-conflict Sn* = *Some* (*C*, *γ*) ∧
C · *γ* ∉ *grounding-of-clss* (*fset N* ∪ *fset U*) ∧
set-mset (*C* · *γ*) ∉ *set-mset* ‘ *grounding-of-clss* (*fset N* ∪ *fset U*) ∧

$C \notin \mathit{fset}\ N \cup \mathit{fset}\ U\ \wedge$
$\neg\ (\exists D \in \mathit{fset}\ N \cup \mathit{fset}\ U.\ \exists\sigma.\ D \cdot \sigma = C)\ \wedge$
$\neg\ \mathit{redundant}\ \mathit{trail\text{-}ord}\ (\mathit{fset}\ N \cup \mathit{fset}\ U)\ C)$
⟨*proof*⟩

corollary *dynamic-non-redundancy-strategy*:
fixes Γ
assumes
run: $\mathit{strategy}^{**}$ *initial-state S0* **and**
conflict: *conflict N* β *S0 S1* **and**
resolution: $(\mathit{skip}\ N\ \beta \sqcup \mathit{factorize}\ N\ \beta \sqcup \mathit{resolve}\ N\ \beta)^{++}$ *S1 Sn* **and**
backtrack: *backtrack N* β *Sn Sn′* **and**
strategy-imp-regular-scl: $\bigwedge S\ S'.\ \mathit{strategy}\ S\ S' \Longrightarrow \mathit{regular\text{-}scl}\ N\ \beta\ S\ S'$ **and**
lit-less-preserves-term-order: $\bigwedge R\ L1\ L2.\ \mathit{lit\text{-}less}\ R\ L1\ L2 \Longrightarrow R^{==}\ (\mathit{atm\text{-}of}\ L1)$
$(\mathit{atm\text{-}of}\ L2)$
defines
$\Gamma \equiv$ *state-trail S1* **and**
$U \equiv$ *state-learned S1* **and**
$\mathit{trail\text{-}ord} \equiv \mathit{multp}_{HO}\ (\mathit{lit\text{-}less}\ (\mathit{trail\text{-}term\text{-}less}\ (\mathit{map}\ (\mathit{atm\text{-}of}\ o\ \mathit{fst})\ \Gamma)))$
shows $(\exists C\ \gamma.\ \mathit{state\text{-}conflict}\ Sn = \mathit{Some}\ (C, \gamma)\ \wedge$
$C \cdot \gamma \notin \mathit{grounding\text{-}of\text{-}clss}\ (\mathit{fset}\ N \cup \mathit{fset}\ U)\ \wedge$
$\mathit{set\text{-}mset}\ (C \cdot \gamma) \notin \mathit{set\text{-}mset}\ `\ \mathit{grounding\text{-}of\text{-}clss}\ (\mathit{fset}\ N \cup \mathit{fset}\ U)\ \wedge$
$C \notin \mathit{fset}\ N \cup \mathit{fset}\ U\ \wedge$
$\neg\ (\exists D \in \mathit{fset}\ N \cup \mathit{fset}\ U.\ \exists\sigma.\ D \cdot \sigma = C)\ \wedge$
$\neg\ \mathit{redundant}\ \mathit{trail\text{-}ord}\ (\mathit{fset}\ N \cup \mathit{fset}\ U)\ C)$
⟨*proof*⟩

16 Static Non-Redundancy

lemma *before-regular-backtrack′*:
assumes
run: $(\mathit{regular\text{-}scl}\ N\ \beta)^{**}$ *initial-state S* **and**
step: *backtrack N* β *S S′*
shows $\exists S0\ S1\ S2\ S3\ S4.\ (\mathit{regular\text{-}scl}\ N\ \beta)^{**}\ \mathit{initial\text{-}state}\ S0\ \wedge$
$\mathit{propagate}\ N\ \beta\ S0\ S1\ \wedge\ \mathit{regular\text{-}scl}\ N\ \beta\ S0\ S1\ \wedge$
$\mathit{conflict}\ N\ \beta\ S1\ S2\ \wedge\ (\mathit{factorize}\ N\ \beta)^{**}\ S2\ S3\ \wedge\ \mathit{resolve}\ N\ \beta\ S3\ S4\ \wedge$
$(\mathit{skip}\ N\ \beta \sqcup \mathit{factorize}\ N\ \beta \sqcup \mathit{resolve}\ N\ \beta)^{**}\ S4\ S$
⟨*proof*⟩

theorem *static-non-subsumption-regular-scl*:
assumes
run: $(\mathit{regular\text{-}scl}\ N\ \beta)^{**}$ *initial-state S* **and**
step: *backtrack N* β *S S′*
defines
$U \equiv$ *state-learned S*
shows $\exists C\ \gamma.\ \mathit{state\text{-}conflict}\ S = \mathit{Some}\ (C, \gamma)\ \wedge\ \neg\ (\exists D\ |{\in}|\ N\ |{\cup}|\ U.\ \mathit{subsumes}$
$D\ C)$
⟨*proof*⟩

corollary *static-non-subsumption-projectable-strategy*:
fixes *strategy* **and** *strategy-init* **and** *proj*
assumes
run: $strategy^{**}$ *strategy-init* S **and**
step: *backtrack* N β (*proj* S) S' **and**
strategy-restricts-regular-scl:
$\bigwedge S\ S'.$ $strategy^{**}$ *strategy-init* $S \Longrightarrow$ *strategy* $S\ S' \Longrightarrow$ *regular-scl* N β (*proj* S) (*proj* S') **and**
initial-state: *proj strategy-init* = *initial-state*
defines
$U \equiv$ *state-learned* (*proj* S)
shows $\exists\, C\ \gamma.$ *state-conflict* (*proj* S) = *Some* $(C, \gamma) \wedge \neg$ ($\exists\, D$ $|\in|$ N $|\cup|$ U. *subsumes* D C)
⟨*proof*⟩

corollary *static-non-subsumption-strategy*:
assumes
run: $strategy^{**}$ *initial-state* S **and**
step: *backtrack* N β S S' **and**
strategy-imp-regular-scl: $\bigwedge S\ S'.$ *strategy* $S\ S' \Longrightarrow$ *regular-scl* N β S S'
defines
$U \equiv$ *state-learned* S
shows $\exists\, C\ \gamma.$ *state-conflict* S = *Some* $(C, \gamma) \wedge \neg$ ($\exists\, D$ $|\in|$ N $|\cup|$ U. *subsumes* D C)
⟨*proof*⟩

end

end
theory *Wellfounded-Extra*
imports
Main
Ordered-Resolution-Prover.Lazy-List-Chain
begin

lemma *wf-onI*:
$(\bigwedge P\ x.\ (\bigwedge y.\ y \in A \Longrightarrow (\bigwedge z.\ z \in A \Longrightarrow (z, y) \in r \Longrightarrow P\ z) \Longrightarrow P\ y) \Longrightarrow x \in A \Longrightarrow P\ x) \Longrightarrow$ *wf-on* $A\ r$
⟨*proof*⟩

lemma *wfI*: $(\bigwedge P\ x.\ (\bigwedge y.\ (\bigwedge z.\ (z, y) \in r \Longrightarrow P\ z) \Longrightarrow P\ y) \Longrightarrow P\ x) \Longrightarrow$ *wf* r
⟨*proof*⟩

lemma *wf-on-induct*[*consumes 1*, *case-names less in-dom*]:
assumes
wf-on $A\ r$ **and**
$\bigwedge x.\ x \in A \Longrightarrow (\bigwedge y.\ y \in A \Longrightarrow (y, x) \in r \Longrightarrow P\ y) \Longrightarrow P\ x$ **and**
$x \in A$
shows $P\ x$

⟨*proof*⟩

16.1 Basic Results

16.1.1 Minimal-element characterization of well-foundedness

lemma *minimal-if-wf-on*:
assumes *wf*: *wf-on A R* **and** $B \subseteq A$ **and** $B \neq \{\}$
shows $\exists\, z \in B.\ \forall\, y.\ (y,\ z) \in R \longrightarrow y \notin B$
⟨*proof*⟩

lemma *wfE-min*:
assumes *wf*: *wf R* **and** *Q*: $x \in Q$
obtains *z* **where** $z \in Q$ $\bigwedge y.\ (y,\ z) \in R \Longrightarrow y \notin Q$
⟨*proof*⟩

lemma *wfE-min′*:
wf R $\Longrightarrow Q \neq \{\} \Longrightarrow (\bigwedge z.\ z \in Q \Longrightarrow (\bigwedge y.\ (y,\ z) \in R \Longrightarrow y \notin Q) \Longrightarrow \textit{thesis}) \Longrightarrow \textit{thesis}$
⟨*proof*⟩

lemma *wf-on-if-minimal*:
assumes $\bigwedge B.\ B \subseteq A \Longrightarrow B \neq \{\} \Longrightarrow \exists\, z \in B.\ \forall\, y.\ (y,\ z) \in R \longrightarrow y \notin B$
shows *wf-on A R*
⟨*proof*⟩

lemma *ex-trans-min-element-if-wf-on*:
assumes *wf*: *wf-on A r* **and** *x-in*: $x \in A$
shows $\exists\, y \in A.\ (y,\ x) \in r^* \wedge \neg(\exists\, z \in A.\ (z,\ y) \in r)$
⟨*proof*⟩

lemma *ex-trans-min-element-if-wfp-on*: *wfp-on A R* $\Longrightarrow x \in A \Longrightarrow \exists\, y{\in}A.\ R^{**}\ y\ x \wedge \neg\ (\exists\, z{\in}A.\ R\ z\ y)$
⟨*proof*⟩

Well-foundedness of the empty relation

definition *inv-imagep-on* :: $'a\ set \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$ **where**
inv-imagep-on $A\ R\ f = (\lambda x\ y.\ x \in A \wedge y \in A \wedge R\ (f\ x)\ (f\ y))$

lemma *wfp-on-inv-imagep*:
assumes *wf*: *wfp-on* $(f\ `\ A)\ R$
shows *wfp-on A* (*inv-imagep R f*)
⟨*proof*⟩

lemma *wfp-on-if-convertible-to-wfp*:
assumes
wf: *wfp-on* $(f\ `\ A)\ Q$ **and**
convertible: $(\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow R\ x\ y \Longrightarrow Q\ (f\ x)\ (f\ y))$
shows *wfp-on A R*

⟨*proof*⟩

definition *lex-prodp* **where**
$lex\text{-}prodp\ R_A\ R_B\ x\ y \longleftrightarrow R_A\ (fst\ x)\ (fst\ y) \lor fst\ x = fst\ y \land R_B\ (snd\ x)\ (snd\ y)$

lemma *lex-prodp-lex-prod-iff* [*pred-set-conv*]:
$lex\text{-}prodp\ R_A\ R_B\ x\ y \longleftrightarrow (x,\ y) \in lex\text{-}prod\ \{(x,\ y).\ R_A\ x\ y\}\ \{(x,\ y).\ R_B\ x\ y\}$
⟨*proof*⟩

lemma *lex-prod-lex-prodp-iff*:
$lex\text{-}prod\ \{(x,\ y).\ R_A\ x\ y\}\ \{(x,\ y).\ R_B\ x\ y\} = \{(x,\ y).\ lex\text{-}prodp\ R_A\ R_B\ x\ y\}$
⟨*proof*⟩

lemma *wf-on-lex-prod*:
assumes *wfA*: $wf\text{-}on\ A\ r_A$ **and** *wfB*: $wf\text{-}on\ B\ r_B$ **and** *AB-subset*: $AB \subseteq A \times B$
shows $wf\text{-}on\ AB\ (r_A <\!*lex*\!> r_B)$
⟨*proof*⟩

lemma *wfp-on-lex-prodp*: $wfp\text{-}on\ A\ R_A \Longrightarrow wfp\text{-}on\ B\ R_B \Longrightarrow AB \subseteq A \times B \Longrightarrow wfp\text{-}on\ AB\ (lex\text{-}prodp\ R_A\ R_B)$
⟨*proof*⟩

corollary *wfp-lex-prodp*: $wfp\ R_A \Longrightarrow wfp\ R_B \Longrightarrow wfp\ (lex\text{-}prodp\ R_A\ R_B)$
⟨*proof*⟩

lemma *wfp-on-sup-if-convertible-to-wfp*:
includes *lattice-syntax*
assumes
wf-S: $wfp\text{-}on\ A\ S$ **and**
wf-Q: $wfp\text{-}on\ (f\ `\ A)\ Q$ **and**
convertible-R: $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow R\ x\ y \Longrightarrow Q\ (f\ x)\ (f\ y)$ **and**
convertible-S: $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow S\ x\ y \Longrightarrow Q\ (f\ x)\ (f\ y) \lor f\ x = f\ y$
shows $wfp\text{-}on\ A\ (R \sqcup S)$
⟨*proof*⟩

lemma *wfp-on-iff-wfp*: $wfp\text{-}on\ A\ R \longleftrightarrow wfp\ (\lambda x\ y.\ R\ x\ y \land\ x \in A \land y \in A)$
⟨*proof*⟩

lemma *chain-lnth-rtranclp*:
assumes
chain: $Lazy\text{-}List\text{-}Chain.chain\ R\ xs$ **and**
len: $enat\ j < llength\ xs$
shows $R^{**}\ (lhd\ xs)\ (lnth\ xs\ j)$
⟨*proof*⟩

lemma *chain-conj-rtranclpI*:
fixes $xs :: 'a\ llist$
assumes $Lazy\text{-}List\text{-}Chain.chain\ (\lambda x\ y.\ R\ x\ y)\ (LCons\ init\ xs)$
shows $Lazy\text{-}List\text{-}Chain.chain\ (\lambda x\ y.\ R\ x\ y \land R^{**}\ init\ x)\ (LCons\ init\ xs)$

⟨*proof*⟩

lemma *rtranclp-implies-ex-lfinite-chain*:
assumes *run*: R^{**} x_0 x
shows $\exists$ *xs*. *lfinite xs* $\wedge$ *chain* (λy z. R y z $\wedge$ R^{**} x_0 y) (*LCons* x_0 *xs*) $\wedge$ *llast* (*LCons* x_0 *xs*) = x
⟨*proof*⟩

lemma *chain-conj-rtranclpD*:
fixes *xs* :: *'a llist*
assumes *inf*: $\neg$ *lfinite xs* **and** *chain*: *chain* (λy z. R y z $\wedge$ R^{**} x_0 y) *xs*
shows $\exists$ *ys*. *lfinite ys* $\wedge$ *chain* (λy z. R y z $\wedge$ R^{**} x_0 y) (*lappend ys xs*) $\wedge$ *lhd* (*lappend ys xs*) = x_0
⟨*proof*⟩

lemma *wfp-on-rtranclp-conversep-iff-no-infinite-down-chain-llist*:
fixes R x_0
shows *wfp-on* $\{x.\ R^{**}\ x_0\ x\}$ R^{-1-1} $\longleftrightarrow$ ($\nexists$ *xs*. $\neg$ *lfinite xs* $\wedge$ *Lazy-List-Chain.chain* R (*LCons* x_0 *xs*))
⟨*proof*⟩

end
theory *Termination*
imports
SCL-FOL
Non-Redundancy
Wellfounded-Extra
HOL−Library.Monad-Syntax
begin

17 Extra Lemmas

17.1 Set Extra

lemma *minus-psubset-minusI*:
assumes $C \subset B$ **and** $B \subseteq A$
shows ($A - B \subset A - C$)
⟨*proof*⟩

17.2 Prod Extra

lemma *lex-prod-lex-prodp-eq*:
lex-prod $\{(x, y).\ RA\ x\ y\}$ $\{(x, y).\ RB\ x\ y\}$ = $\{(x, y).$ *lex-prodp* $RA\ RB\ x\ y\}$
⟨*proof*⟩

lemma *reflp-on-lex-prodp*:
assumes *reflp-on* A RA
shows *reflp-on* ($A \times B$) (*lex-prodp* RA RB)
⟨*proof*⟩

lemma *transp-lex-prodp*:
  assumes *transp RA* **and** *transp RB*
  shows *transp* (*lex-prodp RA RB*)
⟨*proof*⟩

lemma *asymp-lex-prodp*:
  assumes *asymp RA* **and** *asymp RB*
  shows *asymp* (*lex-prodp RA RB*)
⟨*proof*⟩

lemma *totalp-on-lex-prodp*:
  assumes *totalp-on A RA* **and** *totalp-on B RB*
  shows *totalp-on* ($A \times B$) (*lex-prodp RA RB*)
⟨*proof*⟩

17.3 FSet Extra

lemma *finsert-Abs-fset*: *finite A* $\Longrightarrow$ *finsert a* (*Abs-fset A*) = *Abs-fset* (*insert a A*)
  ⟨*proof*⟩

lemma *minus-pfsubset-minusI*:
  assumes C |⊂| B **and** B |⊆| A
  shows (A |−| B |⊂| A |−| C)
⟨*proof*⟩

lemma *Abs-fset-minus*: *finite A* $\Longrightarrow$ *finite B* $\Longrightarrow$ *Abs-fset* ($A - B$) = *Abs-fset A* |−| *Abs-fset B*
  ⟨*proof*⟩

lemma *fminus-conv*: A |⊂| B $\longleftrightarrow$ *fset A* ⊂ *fset B* ∧ *finite* (*fset A*) ∧ *finite* (*fset B*)
  ⟨*proof*⟩

18 Termination

context *scl-fol-calculus* **begin**

18.1 SCL without backtracking terminates

definition $\mathcal{M}$*-prop-deci* :: - ⇒ - ⇒ (-, -) *Term.term literal fset* **where**
  $\mathcal{M}$*-prop-deci* β Γ = *Abs-fset* {*L. atm-of L* $\preceq_B$ β} |−| (*fst* |‘| *fset-of-list* Γ)

primrec $\mathcal{M}$*-skip-fact-reso* **where**
  $\mathcal{M}$*-skip-fact-reso* [] *C* = [] |
  $\mathcal{M}$*-skip-fact-reso* (*Ln* # Γ) *C* =
    (*let n* = *count C* (− (*fst Ln*)) *in*
    (*case snd Ln of None* ⇒ *0* | *Some* - ⇒ *n*) #

$\mathcal{M}$*-skip-fact-reso* Γ (C + (*case snd Ln of None* $\Rightarrow$ {#} | *Some* (D, -, γ) $\Rightarrow$ *repeat-mset n* ($D \cdot \gamma$))))

fun $\mathcal{M}$*-skip-fact-reso′* **where**
$\mathcal{M}$*-skip-fact-reso′* C [] = [] |
$\mathcal{M}$*-skip-fact-reso′* C ((-, *None*) # Γ) = *0* # $\mathcal{M}$*-skip-fact-reso′* C Γ |
$\mathcal{M}$*-skip-fact-reso′* C ((K, *Some* (D, -, γ)) # Γ) =
(*let n* = *count* C ($-$ K) *in n* # $\mathcal{M}$*-skip-fact-reso′* (C + *repeat-mset n* ($D \cdot \gamma$)) Γ)

lemma $\mathcal{M}$*-skip-fact-reso* Γ C = $\mathcal{M}$*-skip-fact-reso′* C Γ
⟨*proof*⟩

lemma $\mathcal{M}$*-skip-fact-reso′* C (*decide-lit* K # Γ) = *0* # $\mathcal{M}$*-skip-fact-reso′* C Γ
⟨*proof*⟩

lemma $\mathcal{M}$*-skip-fact-reso′* C (*propagate-lit* K D γ # Γ) =
(*let n* = *count* C ($-$ (K $\cdot l$ γ)) *in n* # $\mathcal{M}$*-skip-fact-reso′* (C + *repeat-mset n* ($D \cdot \gamma$)) Γ)
⟨*proof*⟩

fun $\mathcal{M}$ :: - $\Rightarrow$ (′*f*, ′*v*) *state* $\Rightarrow$
bool $\times$ (′*f*, ′*v*) *Term.term literal fset* $\times$ *nat list* $\times$ *nat* **where**
$\mathcal{M}$ β (Γ, U, *None*) = (*True*, $\mathcal{M}$*-prop-deci* β Γ, [], *0*) |
$\mathcal{M}$ β (Γ, U, *Some* (C, γ)) = (*False*, {||}, $\mathcal{M}$*-skip-fact-reso* Γ ($C \cdot \gamma$), *size* C)

lemma *length-*$\mathcal{M}$*-skip-fact-reso*[*simp*]: *length* ($\mathcal{M}$*-skip-fact-reso* Γ C) = *length* Γ
⟨*proof*⟩

lemma $\mathcal{M}$*-skip-fact-reso-add-mset*:
($\mathcal{M}$*-skip-fact-reso* Γ C, $\mathcal{M}$*-skip-fact-reso* Γ (*add-mset* L C)) $\in$ (*List.lenlex* {(x, y). $x < y$})$^=$
⟨*proof*⟩

lemma *termination-scl-without-back-invars*:
fixes N β
defines
scl-without-backtrack $\equiv$ *propagate* N β $\sqcup$ *decide* N β $\sqcup$ *conflict* N β $\sqcup$ *skip* N β $\sqcup$
factorize N β $\sqcup$ *resolve* N β **and**
invars $\equiv$ *trail-atoms-lt* β $\sqcap$ *trail-resolved-lits-pol* $\sqcap$ *trail-lits-ground* $\sqcap$
initial-lits-generalize-learned-trail-conflict N $\sqcap$ *ground-closures*
shows *wfp-on* {S. *invars* S} *scl-without-backtrack*$^{-1-1}$
⟨*proof*⟩

corollary *termination-scl-without-back*:
fixes
N :: (′*f*, ′*v*) *Term.term clause fset* **and**
β :: (′*f*, ′*v*) *Term.term*

defines
scl-without-backtrack $\equiv$ *propagate N* β $\sqcup$ *decide N* β $\sqcup$ *conflict N* β $\sqcup$ *skip N* β $\sqcup$
factorize N β $\sqcup$ *resolve N* β **and**
invars $\equiv$ *trail-atoms-lt* β $\sqcap$ *trail-resolved-lits-pol* $\sqcap$ *trail-lits-ground* $\sqcap$
initial-lits-generalize-learned-trail-conflict N $\sqcap$ *ground-closures*
shows *wfp-on* $\{S.$ *scl-without-backtrack*** *initial-state S*$\}$ *scl-without-backtrack*$^{-1-1}$
$\langle proof \rangle$

corollary *termination-stragegy-without-back*:
fixes
N :: $('f,\ 'v)$ *Term.term clause fset* **and**
β :: $('f,\ 'v)$ *Term.term*
defines
scl-without-backtrack $\equiv$ *propagate N* β $\sqcup$ *decide N* β $\sqcup$ *conflict N* β $\sqcup$ *skip N* β $\sqcup$
factorize N β $\sqcup$ *resolve N* β
assumes *strategy-stronger*: $\bigwedge S\ S'.$ *strategy S S'* $\Longrightarrow$ *scl-without-backtrack S S'*
shows *wfp-on* $\{S.$ *strategy*** *initial-state S*$\}$ *strategy*$^{-1-1}$
$\langle proof \rangle$

18.2 Backtracking can only be done finitely often

definition *fclss-no-dup* :: $('f,\ 'v)$ *Term.term* $\Rightarrow$ $('f,\ 'v)$ *Term.term literal fset fset*
where
fclss-no-dup β = *fPow* (*Abs-fset* $\{L.$ *atm-of L* $\preceq_B$ $\beta\}$)

lemma *image-fset-fset-fPow-eq*: *fset* ‘ *fset* (*fPow A*) = *Pow* (*fset A*)
$\langle proof \rangle$

lemma
assumes $\forall L \in\#\ C.$ *count C L* = *1*
shows $\exists C'.\ C$ = *mset-set C'*
$\langle proof \rangle$

lemma *fmember-fclss-no-dup-if*:
assumes $\forall L\ |\in|\ C.$ *atm-of L* $\preceq_B$ β
shows $C\ |\in|$ *fclss-no-dup* β
$\langle proof \rangle$

definition $\mathcal{M}$*-back* :: - $\Rightarrow$ $('f,\ 'v)$ *state* $\Rightarrow$ $('f,\ 'v)$ *Term.term literal fset fset*
where
$\mathcal{M}$*-back* β *S* = *Abs-fset* (*fset* (*fclss-no-dup* β) −
Abs-fset ‘ *set-mset* ‘ *grounding-of-clss* (*fset* (*state-learned S*)))

lemma $\mathcal{M}$*-back-after-regular-backtrack*:
assumes
regular-run: (*regular-scl N* β)** *initial-state S0* **and**
conflict: *conflict N* β *S0 S1* **and**

resolution: $(skip\ N\ \beta \sqcup factorize\ N\ \beta \sqcup resolve\ N\ \beta)^{++}\ S1\ Sn$ **and**
backtrack: $backtrack\ N\ \beta\ Sn\ Sn'$
defines $U \equiv state\text{-}learned\ Sn$
shows
$\exists\, C\ \gamma.\ state\text{-}conflict\ Sn = Some\ (C, \gamma) \wedge$
$set\text{-}mset\ (C \cdot \gamma) \notin set\text{-}mset\ `\ grounding\text{-}of\text{-}clss\ (fset\ N \cup fset\ U)$ **and**
$\mathcal{M}\text{-}back\ \beta\ Sn'\ |{\subset}|\ \mathcal{M}\text{-}back\ \beta\ Sn$
⟨*proof*⟩

18.3 Regular SCL terminates

theorem *termination-regular-scl-invars*:
fixes
$N :: ('f, 'v)\ Term.term\ clause\ fset$ **and**
$\beta :: ('f, 'v)\ Term.term$
defines
$invars \equiv trail\text{-}atoms\text{-}lt\ \beta \sqcap trail\text{-}resolved\text{-}lits\text{-}pol \sqcap trail\text{-}lits\text{-}ground \sqcap$
$initial\text{-}lits\text{-}generalize\text{-}learned\text{-}trail\text{-}conflict\ N \sqcap ground\text{-}closures \sqcap ground\text{-}false\text{-}closures$
$\sqcap$
$sound\text{-}state\ N\ \beta \sqcap almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta \sqcap regular\text{-}conflict\text{-}resolution$
$N\ \beta$
shows
$wfp\text{-}on\ \{S.\ invars\ S\}\ (regular\text{-}scl\ N\ \beta)^{-1-1}$
⟨*proof*⟩

corollary *termination-regular-scl*:
fixes
$N :: ('f, 'v)\ Term.term\ clause\ fset$ **and**
$\beta :: ('f, 'v)\ Term.term$
defines
$invars \equiv trail\text{-}atoms\text{-}lt\ \beta \sqcap trail\text{-}resolved\text{-}lits\text{-}pol \sqcap trail\text{-}lits\text{-}ground \sqcap$
$initial\text{-}lits\text{-}generalize\text{-}learned\text{-}trail\text{-}conflict\ N \sqcap ground\text{-}closures \sqcap ground\text{-}false\text{-}closures$
$\sqcap$
$sound\text{-}state\ N\ \beta \sqcap almost\text{-}no\text{-}conflict\text{-}with\text{-}trail\ N\ \beta \sqcap regular\text{-}conflict\text{-}resolution$
$N\ \beta$
shows $wfp\text{-}on\ \{S.\ (regular\text{-}scl\ N\ \beta)^{**}\ initial\text{-}state\ S\}\ (regular\text{-}scl\ N\ \beta)^{-1-1}$
⟨*proof*⟩

corollary *termination-projectable-strategy*:
fixes
$N :: ('f, 'v)\ Term.term\ clause\ fset$ **and**
$\beta :: ('f, 'v)\ Term.term$ **and**
strategy **and** *strategy-init* **and** *proj*
assumes *strategy-restricts-regular-scl*:
$\bigwedge S\ S'.\ strategy^{**}\ strategy\text{-}init\ S \Longrightarrow strategy\ S\ S' \Longrightarrow regular\text{-}scl\ N\ \beta\ (proj\ S)$
$(proj\ S')$ **and**
initial-state: $proj\ strategy\text{-}init = initial\text{-}state$
shows $wfp\text{-}on\ \{S.\ strategy^{**}\ strategy\text{-}init\ S\}\ strategy^{-1-1}$
⟨*proof*⟩

corollary *termination-strategy*:
fixes
N :: ($'f$, $'v$) *Term.term clause fset* **and**
β :: ($'f$, $'v$) *Term.term*
assumes *strategy-restricts-regular-scl*: $\bigwedge S\ S'$. *strategy* $S\ S' \Longrightarrow$ *regular-scl* $N\ \beta$ $S\ S'$
shows *wfp-on* $\{S$. *strategy*** *initial-state* $S\}$ *strategy*$^{-1-1}$
⟨*proof*⟩

end

end
theory *Completeness*
imports
Correct-Termination
Termination
Functional-Ordered-Resolution-Prover.IsaFoR-Term
begin

lemma (**in** *scl-fol-calculus*) *regular-scl-run-derives-contradiction-if-unsat*:
fixes $N\ \beta$ *gnd-N*
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-$\beta \equiv \{C \in$ *gnd-N*. $\forall L \in\#\ C$. *atm-of* $L \preceq_B \beta\}$
assumes
unsat: $\neg$ *satisfiable gnd-N-lt-*β **and**
run: (*regular-scl* $N\ \beta$)** *initial-state* S **and**
no-more-step: $\nexists S'$. *regular-scl* $N\ \beta\ S\ S'$
shows $\exists\gamma$. *state-conflict* S = *Some* ($\{\#\}$, γ)
⟨*proof*⟩

theorem (**in** *scl-fol-calculus*)
fixes $N\ \beta$ *gnd-N*
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-$\beta \equiv \{C \in$ *gnd-N*. $\forall L \in\#\ C$. *atm-of* $L \preceq_B \beta\}$
assumes *unsat*: $\neg$ *satisfiable gnd-N-lt-*β
shows $\exists S$. (*regular-scl* $N\ \beta$)** *initial-state* $S \wedge$
($\nexists S'$. *regular-scl* $N\ \beta\ S\ S'$) $\wedge$
($\exists\gamma$. *state-conflict* S = *Some* ($\{\#\}$, γ))
⟨*proof*⟩

lemma (**in** *scl-fol-calculus*) *no-infinite-down-chain*:
$\nexists Ss$. $\neg$ *lfinite Ss* $\wedge$ *Lazy-List-Chain.chain* ($\lambda S\ S'$. *regular-scl* $N\ \beta\ S\ S'$) (*LCons initial-state Ss*)
⟨*proof*⟩

theorem (**in** *scl-fol-calculus*) *completeness-wrt-bound*:

fixes N β *gnd-N*
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N) **and**
gnd-N-lt-β $\equiv$ $\{C \in$ *gnd-N*. $\forall L \in\# C$. *atm-of* $L \preceq_B \beta\}$
assumes *unsat*: $\neg$ *satisfiable gnd-N-lt-*β
shows
$\nexists$ *Ss*. $\neg$ *lfinite Ss* $\wedge$ *Lazy-List-Chain.chain* ($\lambda S\ S'$. *regular-scl* N β S S')
(*LCons initial-state Ss*) **and**
$\forall S$. (*regular-scl* N β)** *initial-state* $S \longrightarrow$ ($\nexists S'$. *regular-scl* N β S S') $\longrightarrow$
($\exists \gamma$. *state-conflict* S = *Some* ($\{\#\}$, γ))
⟨*proof*⟩

locale *compact-scl* =
scl-fol-calculus renaming-vars (<) :: ('*f* :: *weighted*, '*v*) *term* $\Rightarrow$ ('*f*, '*v*) *term* $\Rightarrow$ *bool*
for *renaming-vars* :: '*v set* $\Rightarrow$ '*v* $\Rightarrow$ '*v*
begin

theorem *ex-bound-if-unsat*:
fixes N :: ('*f*, '*v*) *term clause fset*
defines
gnd-N $\equiv$ *grounding-of-clss* (*fset* N)
assumes *unsat*: $\neg$ *satisfiable gnd-N*
shows $\exists \beta$. $\neg$ *satisfiable* $\{C \in$ *gnd-N*. $\forall L \in\# C$. *atm-of* $L \leq \beta\}$
⟨*proof*⟩

end

end
theory *Invariants*
imports *SCL-FOL*
begin

The following lemma restate existing invariants in a compact, paper-friendly way.

lemma (**in** *scl-fol-calculus*) *scl-state-invariants*:
shows
inv-trail-lits-ground:
trail-lits-ground initial-state
scl N β S S' $\Longrightarrow$ *trail-lits-ground* S $\Longrightarrow$ *trail-lits-ground* S' **and**
inv-trail-atoms-lt:
trail-atoms-lt β *initial-state*
scl N β S S' $\Longrightarrow$ *trail-atoms-lt* β S $\Longrightarrow$ *trail-atoms-lt* β S' **and**
inv-undefined-trail-lits:
$\forall \Gamma'$ *Ln* Γ''. *state-trail initial-state* = Γ'' @ *Ln* # Γ' $\longrightarrow$ $\neg$ *trail-defined-lit* Γ' (*fst Ln*)
scl N β S S' $\Longrightarrow$
($\forall \Gamma'$ *Ln* Γ''. *state-trail* S = Γ'' @ *Ln* # Γ' $\longrightarrow$ $\neg$ *trail-defined-lit* Γ' (*fst Ln*))

$\Longrightarrow$
$(\forall \Gamma'\ Ln\ \Gamma''.\ \textit{state-trail}\ S' = \Gamma''\ @\ Ln\ \#\ \Gamma' \longrightarrow \neg\ \textit{trail-defined-lit}\ \Gamma'\ (\textit{fst}\ Ln))$ **and**
inv-ground-closures:
$\textit{ground-closures initial-state}$
$\textit{scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{ground-closures}\ S \Longrightarrow \textit{ground-closures}\ S'$ **and**
inv-ground-false-closures:
$\textit{ground-false-closures initial-state}$
$\textit{scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{ground-false-closures}\ S \Longrightarrow \textit{ground-false-closures}\ S'$ **and**
inv-trail-propagated-lits-wf:
$\forall \mathcal{K} \in \textit{set}\ (\textit{state-trail initial-state}).\ \forall D\ K\ \gamma.\ \textit{snd}\ \mathcal{K} = \textit{Some}\ (D, K, \gamma) \longrightarrow \textit{fst}\ \mathcal{K} = K \cdot l\ \gamma$
$\textit{scl}\ N\ \beta\ S\ S' \Longrightarrow$
$(\forall \mathcal{K} \in \textit{set}\ (\textit{state-trail}\ S).\ \forall D\ K\ \gamma.\ \textit{snd}\ \mathcal{K} = \textit{Some}\ (D, K, \gamma) \longrightarrow \textit{fst}\ \mathcal{K} = K \cdot l\ \gamma) \Longrightarrow$
$(\forall \mathcal{K} \in \textit{set}\ (\textit{state-trail}\ S').\ \forall D\ K\ \gamma.\ \textit{snd}\ \mathcal{K} = \textit{Some}\ (D, K, \gamma) \longrightarrow \textit{fst}\ \mathcal{K} = K \cdot l\ \gamma)$ **and**
inv-trail-resolved-lits-pol:
$\textit{trail-resolved-lits-pol initial-state}$
$\textit{scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{trail-resolved-lits-pol}\ S \Longrightarrow \textit{trail-resolved-lits-pol}\ S'$ **and**
inv-initial-lits-generalize-learned-trail-conflict:
$\textit{initial-lits-generalize-learned-trail-conflict}\ N\ \textit{initial-state}$
$\textit{scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{initial-lits-generalize-learned-trail-conflict}\ N\ S \Longrightarrow \textit{initial-lits-generalize-learned-trail-conflict}\ N\ S'$ **and**
inv-sound-state:
$\textit{sound-state}\ N\ \beta\ \textit{initial-state}$
$\textit{scl}\ N\ \beta\ S\ S' \Longrightarrow \textit{sound-state}\ N\ \beta\ S \Longrightarrow \textit{sound-state}\ N\ \beta\ S'$
$\langle \textit{proof} \rangle$

end

References

[1] M. Bromberger, S. Schwarz, and C. Weidenbach. SCL(FOL) revisited, 2023.

[2] A. Fiori and C. Weidenbach. SCL clause learning from simple models. In P. Fontaine, editor, *Automated Deduction – CADE 27*, volume 11716 of *Lecture Notes in Artificial Intelligence*, pages 233–249, Natal, Brazil, 2019. Springer.