SCL(FOL) Can Simulate Ground Nonredundant Ordered Resolution

Martin Desharnais

November 4, 2024

Abstract

SCL(FOL) (i.e., Simple Clause Learning for First-Order Logic without equality) is known to be able to simulate the derivation of nonredundant clauses by the ground ordered resolution calculus [1]. Due to the space constraints of a 16-pages paper, the published proof is monolithic and hard to comprehend. In this work, we reuse the existing strategy for ground ordered resolution and present a new, simpler strategy for SCL(FOL). We prove a stronger bisimulation theorem between these two strategies (i.e., they both simulate each other). Our proof is modular: it consists of ten refinement steps focusing on different aspects of the two strategies.

Contents

theory *Isabelle-2024-Compatibility*
  imports
    Main
    HOL−Library.Multiset
begin

lemmas *wfp-def* = *wfP-def*
lemmas *wfp-if-convertible-to-wfp* = *wfP-if-convertible-to-wfP*
lemmas *wfp-imp-asymp* = *wfP-imp-asymp*
lemmas *wfp-induct-rule* = *wfP-induct-rule*
lemmas *wfp-multp* = *wfP-multp*
lemmas *wfp-subset-mset* = *wfP-subset-mset*

end
theory *Ground-Ordered-Resolution*
  imports
    Saturation-Framework.Calculus
    Saturation-Framework-Extensions.Clausal-Calculus
    Isabelle-2024-Compatibility
    Superposition-Calculus.Ground-Ctxt-Extra
    Superposition-Calculus.HOL-Extra
    Superposition-Calculus.Transitive-Closure-Extra
    Min-Max-Least-Greatest.Min-Max-Least-Greatest-FSet
    Min-Max-Least-Greatest.Min-Max-Least-Greatest-Multiset
    Superposition-Calculus.Multiset-Extra
    Superposition-Calculus.Relation-Extra
begin

hide-type *Inference-System.inference*
hide-const
  Inference-System.Infer
  Inference-System.prems-of
  Inference-System.concl-of
  Inference-System.main-prem-of

primrec *mset-lit* :: *'a literal* ⇒ *'a multiset* **where**
mset-lit (*Pos A*) = {#*A*#} |
mset-lit (*Neg A*) = {#*A*, *A*#}

type-synonym *'t atom* = *'t*

1 Ground Resolution Calculus

locale *ground-ordered-resolution-calculus* =
fixes
less-trm :: *'f gterm* ⇒ *'f gterm* ⇒ *bool* (**infix** $\prec_t$ *50*) **and**
select :: *'f gterm atom clause* ⇒ *'f gterm atom clause*
assumes
transp-less-trm[*simp*]: *transp* ($\prec_t$) **and**
asymp-less-trm[*intro*]: *asymp* ($\prec_t$) **and**
wfP-less-trm[*intro*]: *wfP* ($\prec_t$) **and**
totalp-less-trm[*intro*]: *totalp* ($\prec_t$) **and**
less-trm-compatible-with-gctxt[*simp*]: $\bigwedge ctxt\ t\ t'.\ t \prec_t t' \Longrightarrow ctxt\langle t\rangle_G \prec_t ctxt\langle t'\rangle_G$
and
less-trm-if-subterm[*simp*]: $\bigwedge t\ ctxt.\ ctxt \neq \Box_G \Longrightarrow t \prec_t ctxt\langle t\rangle_G$ **and**
select-subset: $\bigwedge C.$ *select C* ⊆# *C* **and**
select-negative-lits: $\bigwedge C\ L.$ *L* ∈# *select C* ⟹ *is-neg L*
begin

lemma *irreflp-on-less-trm*[*simp*]: *irreflp-on A* ($\prec_t$)
⟨*proof*⟩

abbreviation *lesseq-trm* (**infix** $\preceq_t$ *50*) **where**
lesseq-trm ≡ $(\prec_t)^{==}$

lemma *lesseq-trm-if-subtermeq*: $t \preceq_t ctxt\langle t\rangle_G$
⟨*proof*⟩

definition *less-lit* ::
'f gterm atom literal ⇒ *'f gterm atom literal* ⇒ *bool* (**infix** $\prec_l$ *50*) **where**
less-lit L1 L2 ≡ *multp* ($\prec_t$) (*mset-lit L1*) (*mset-lit L2*)

abbreviation *lesseq-lit* (**infix** $\preceq_l$ *50*) **where**
lesseq-lit ≡ $(\prec_l)^{==}$

abbreviation *less-cls* ::
'f gterm atom clause ⇒ *'f gterm atom clause* ⇒ *bool* (**infix** $\prec_c$ *50*) **where**
less-cls ≡ *multp* ($\prec_l$)

abbreviation *lesseq-cls* (**infix** $\preceq_c$ *50*) **where**
lesseq-cls ≡ $(\prec_c)^{==}$

lemma *transp-on-less-lit*[*simp*]: *transp-on A* ($\prec_l$)

⟨*proof*⟩

corollary *transp-less-lit*: *transp* $(\prec_l)$
⟨*proof*⟩

lemma *transp-less-cls*[*simp*]: *transp* $(\prec_c)$
⟨*proof*⟩

lemma *asymp-on-less-lit*[*simp*]: *asymp-on* A $(\prec_l)$
⟨*proof*⟩

corollary *asymp-less-lit*[*simp*]: *asymp* $(\prec_l)$
⟨*proof*⟩

lemma *asymp-less-cls*[*simp*]: *asymp* $(\prec_c)$
⟨*proof*⟩

lemma *irreflp-on-less-lit*[*simp*]: *irreflp-on* A $(\prec_l)$
⟨*proof*⟩

lemma *wfP-less-lit*[*simp*]: *wfP* $(\prec_l)$
⟨*proof*⟩

lemma *wfP-less-cls*[*simp*]: *wfP* $(\prec_c)$
⟨*proof*⟩

lemma *totalp-on-less-lit*[*simp*]: *totalp-on* A $(\prec_l)$
⟨*proof*⟩

corollary *totalp-less-lit*: *totalp* $(\prec_l)$
⟨*proof*⟩

lemma *totalp-less-cls*[*simp*]: *totalp* $(\prec_c)$
⟨*proof*⟩

interpretation *term-order*: *linorder lesseq-trm less-trm*
⟨*proof*⟩

interpretation *literal-order*: *linorder lesseq-lit less-lit*
⟨*proof*⟩

interpretation *clause-order*: *linorder lesseq-cls less-cls*
⟨*proof*⟩

lemma *less-lit-simps*[*simp*]:
$Pos\ A_1 \prec_l Pos\ A_2 \longleftrightarrow A_1 \prec_t A_2$
$Pos\ A_1 \prec_l Neg\ A_2 \longleftrightarrow A_1 \preceq_t A_2$
$Neg\ A_1 \prec_l Neg\ A_2 \longleftrightarrow A_1 \prec_t A_2$
$Neg\ A_1 \prec_l Pos\ A_2 \longleftrightarrow A_1 \prec_t A_2$

$\langle proof \rangle$

1.1 Ground Rules

abbreviation *is-maximal-lit* :: *'f gterm literal* ⇒ *'f gterm clause* ⇒ *bool* **where**
is-maximal-lit L M ≡ *is-maximal-in-mset-wrt* $(\prec_l)$ *M L*

abbreviation *is-strictly-maximal-lit* :: *'f gterm literal* ⇒ *'f gterm clause* ⇒ *bool* **where**
is-strictly-maximal-lit L M ≡ *is-greatest-in-mset-wrt* $(\prec_l)$ *M L*

inductive *ground-resolution* ::
'f gterm atom clause ⇒ *'f gterm atom clause* ⇒ *'f gterm atom clause* ⇒ *bool*
where
ground-resolutionI:
P_1 = *add-mset* (*Neg t*) $P_1{'}$ ⟹
P_2 = *add-mset* (*Pos t*) $P_2{'}$ ⟹
$P_2 \prec_c P_1$ ⟹
select P_1 = {#} ∧ *is-maximal-lit* (*Neg t*) P_1 ∨ *Neg t* ∈# *select* P_1 ⟹
select P_2 = {#} ⟹
is-strictly-maximal-lit (*Pos t*) P_2 ⟹
$C = P_1{'} + P_2{'}$ ⟹
ground-resolution P_1 P_2 *C*

inductive *ground-factoring* :: *'f gterm atom clause* ⇒ *'f gterm atom clause* ⇒ *bool* **where**
ground-factoringI:
P = *add-mset* (*Pos t*) (*add-mset* (*Pos t*) *P'*) ⟹
select P = {#} ⟹
is-maximal-lit (*Pos t*) *P* ⟹
C = *add-mset* (*Pos t*) *P'* ⟹
ground-factoring P C

1.2 Ground Layer

definition *G-Inf* :: *'f gterm atom clause inference set* **where**
G-Inf =
{*Infer* [P_2, P_1] *C* | P_2 P_1 *C*. *ground-resolution* P_1 P_2 *C*} ∪
{*Infer* [*P*] *C* | *P C*. *ground-factoring P C*}

abbreviation *G-Bot* :: *'f gterm atom clause set* **where**
G-Bot ≡ {{#}}

definition *G-entails* :: *'f gterm atom clause set* ⇒ *'f gterm atom clause set* ⇒ *bool* **where**
G-entails N_1 N_2 ⟷ (∀(*I* :: *'f gterm set*). *I* ⊫*s* N_1 ⟶ *I* ⊫*s* N_2)

1.3 Correctness

lemma *soundness-ground-resolution*:

assumes
step: *ground-resolution P1 P2 C*
shows *G-entails* $\{P1, P2\}$ $\{C\}$
⟨*proof*⟩

lemma *soundness-ground-factoring*:
assumes *step*: *ground-factoring P C*
shows *G-entails* $\{P\}$ $\{C\}$
⟨*proof*⟩

interpretation *G*: *sound-inference-system G-Inf G-Bot G-entails*
⟨*proof*⟩

1.4 Redundancy Criterion

lemma *ground-resolution-smaller-conclusion*:
assumes
step: *ground-resolution P1 P2 C*
shows $C \prec_c P1$
⟨*proof*⟩

lemma *ground-factoring-smaller-conclusion*:
assumes *step*: *ground-factoring P C*
shows $C \prec_c P$
⟨*proof*⟩

interpretation *G*: *calculus-with-finitary-standard-redundancy G-Inf G-Bot G-entails* $(\prec_c)$
⟨*proof*⟩

1.5 Refutational Completeness

context
fixes N :: *'f gterm atom clause set*
begin

function *production* :: *'f gterm atom clause* ⇒ *'f gterm set* **where**
production $C = \{A \mid A\ C'.$
$C \in N \wedge$
$C =$ *add-mset* $(Pos\ A)\ C' \wedge$
select $C = \{\#\} \wedge$
is-strictly-maximal-lit $(Pos\ A)\ C \wedge$
$\neg\ (\bigcup D \in \{D \in N.\ D \prec_c C\}.$ *production* $D) \Vdash C\}$
⟨*proof*⟩

termination *production*
⟨*proof*⟩

declare *production.simps*[*simp del*]

end

lemma *Uniq-striclty-maximal-lit-in-ground-cls*:
$\exists_{\leq 1} L.$ *is-strictly-maximal-lit* L C
⟨*proof*⟩

lemma *production-eq-empty-or-singleton*:
production N C = {} ∨ (∃ A. *production* N C = {A})
⟨*proof*⟩

lemma *production-eq-singleton-if-atom-in-production*:
assumes A ∈ *production* N C
shows *production* N C = {A}
⟨*proof*⟩

definition *interp* **where**
interp N C ≡ (⋃ D ∈ {D ∈ N. $D \prec_c C$}. *production* N D)

lemma *interp-mempty*[*simp*]: *interp* N {#} = {}
⟨*proof*⟩

lemma *production-unfold*: *production* N C = {A | A C'.
$C \in N$ ∧
C = *add-mset* (*Pos* A) C' ∧
select C = {#} ∧
is-strictly-maximal-lit (*Pos* A) C ∧
¬ *interp* N C ⊫ C}
⟨*proof*⟩

lemma *production-unfold'*: *production* N C = {A | A.
$C \in N$ ∧
select C = {#} ∧
is-strictly-maximal-lit (*Pos* A) C ∧
¬ *interp* N C ⊫ C}
⟨*proof*⟩

lemma *mem-productionE*:
assumes *C-prod*: A ∈ *production* N C
obtains C' **where**
$C \in N$ **and**
C = *add-mset* (*Pos* A) C' **and**
select C = {#} **and**
is-strictly-maximal-lit (*Pos* A) C **and**
¬ *interp* N C ⊫ C
⟨*proof*⟩

lemma *production-subset-if-less-cls*: $C \prec_c D \Longrightarrow$ *production* N C ⊆ *interp* N D
⟨*proof*⟩

lemma *Uniq-production-eq-singleton*: $\exists_{\le 1}\ C.\ production\ N\ C = \{A\}$
⟨*proof*⟩

lemma *singleton-eq-CollectD*: $\{x\} = \{y.\ P\ y\} \Longrightarrow P\ x$
⟨*proof*⟩

lemma *subset-Union-mem-CollectI*: $P\ x \Longrightarrow f\ x \subseteq (\bigcup y \in \{z.\ P\ z\}.\ f\ y)$
⟨*proof*⟩

lemma *interp-subset-if-less-cls*: $C \prec_c D \Longrightarrow interp\ N\ C \subseteq interp\ N\ D$
⟨*proof*⟩

lemma *interp-subset-if-less-cls'*: $C \prec_c D \Longrightarrow interp\ N\ C \subseteq interp\ N\ D \cup production\ N\ D$
⟨*proof*⟩

lemma *split-Union-production*:
assumes *D-in*: $D \in N$
shows $(\bigcup C \in N.\ production\ N\ C) =$
$interp\ N\ D \cup production\ N\ D \cup (\bigcup C \in \{C \in N.\ D \prec_c C\}.\ production\ N\ C)$
⟨*proof*⟩

lemma *split-Union-production'*:
assumes *D-in*: $D \in N$
shows $(\bigcup C \in N.\ production\ N\ C) = interp\ N\ D \cup (\bigcup C \in \{C \in N.\ D \preceq_c C\}.\ production\ N\ C)$
⟨*proof*⟩

lemma *split-interp*:
assumes $C \in N$ **and** *D-in*: $D \in N$ **and** $D \prec_c C$
shows $interp\ N\ C = interp\ N\ D \cup (\bigcup C' \in \{C' \in N.\ D \preceq_c C' \wedge C' \prec_c C\}.\ production\ N\ C')$
⟨*proof*⟩

lemma *less-imp-Interp-subseteq-interp*: $C \prec_c D \Longrightarrow interp\ N\ C \cup production\ N\ C \subseteq interp\ N\ D$
⟨*proof*⟩

lemma *not-interp-to-Interp-imp-le*: $A \notin interp\ N\ C \Longrightarrow A \in interp\ N\ D \cup production\ N\ D \Longrightarrow C \preceq_c D$
⟨*proof*⟩

lemma *produces-imp-in-interp*:
assumes $Neg\ A \in\#\ C$ **and** *D-prod*: $A \in production\ N\ D$
shows $A \in interp\ N\ C$
⟨*proof*⟩

lemma *neg-notin-Interp-not-produce*:
$Neg\ A \in\#\ C \Longrightarrow A \notin interp\ N\ D \cup production\ N\ D \Longrightarrow C \preceq_c D \Longrightarrow A \notin$

production N D''
⟨*proof*⟩

lemma *lift-interp-entails*:
assumes
D-in: $D \in N$ **and**
D-entailed: *interp N D* $\Vdash$ *D* **and**
C-in: $C \in N$ **and**
D-lt-C: $D \prec_c C$
shows *interp N C* $\Vdash$ *D*
⟨*proof*⟩

lemma *lift-interp-entails-to-interp-production-entails*:
assumes
C-in: $C \in N$ **and**
D-in: $D \in N$ **and**
C-lt-D: $D \prec_c C$ **and**
D-entailed: *interp N C* $\Vdash$ *D*
shows *interp N C* $\cup$ *production N C* $\Vdash$ *D*
⟨*proof*⟩

lemma *lift-entailment-to-Union*:
fixes *N D*
assumes
D-in: $D \in N$ **and**
R_D*-entails-D*: *interp N D* $\Vdash$ *D*
shows
$(\bigcup C \in N.$ *production N C*$)$ $\Vdash$ *D*
⟨*proof*⟩

lemma
assumes
$D \preceq_c C$ **and**
C-prod: $A \in$ *production N C* **and**
L-in: $L \in\# D$
shows
lesseq-trm-if-pos: *is-pos L* $\Longrightarrow$ *atm-of L* $\preceq_t A$ **and**
less-trm-if-neg: *is-neg L* $\Longrightarrow$ *atm-of L* $\prec_t A$
⟨*proof*⟩

lemma *less-trm-iff-less-cls-if-mem-production*:
assumes *C-prod*: $A_C \in$ *production N C* **and** *D-prod*: $A_D \in$ *production N D*
shows $A_C \prec_t A_D \longleftrightarrow C \prec_c D$
⟨*proof*⟩

lemma *false-cls-if-productive-production*:
assumes *C-prod*: $A \in$ *production N C* **and** $D \in N$ **and** $C \prec_c D$
shows $\neg$ *interp N D* $\Vdash$ $C - \{\#Pos\ A\#\}$

⟨*proof*⟩

lemma *production-subset-Union-production*:
$\bigwedge C\ N.\ C \in N \Longrightarrow \mathit{production}\ N\ C \subseteq (\bigcup D \in N.\ \mathit{production}\ N\ D)$
⟨*proof*⟩

lemma *interp-subset-Union-production*:
$\bigwedge C\ N.\ C \in N \Longrightarrow \mathit{interp}\ N\ C \subseteq (\bigcup D \in N.\ \mathit{production}\ N\ D)$
⟨*proof*⟩

lemma *model-construction*:
fixes
N :: *'f gterm atom clause set* **and**
C :: *'f gterm atom clause*
assumes *G.saturated* N **and** $\{\#\} \notin N$ **and** *C-in*: $C \in N$
shows
$\mathit{production}\ N\ C = \{\} \longleftrightarrow \mathit{interp}\ N\ C \Vdash C$
$(\bigcup D \in N.\ \mathit{production}\ N\ D) \Vdash C$
$D \in N \Longrightarrow C \prec_c D \Longrightarrow \mathit{interp}\ N\ D \Vdash C$
⟨*proof*⟩

lemma
assumes *clause-order.is-least-in-fset* $N\ C$ **and** $A \in \mathit{production}\ (\mathit{fset}\ N)\ C$
shows $\bigwedge A'.\ A' \prec_t A \Longrightarrow A' \notin (\bigcup D \in \mathit{fset}\ N.\ \mathit{production}\ (\mathit{fset}\ N)\ D)$
⟨*proof*⟩

lemma *lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-productive*:
assumes $A \in \mathit{production}\ (\mathit{fset}\ N)\ C$
shows $\bigwedge A'.\ A' \prec_t A \Longrightarrow A' \notin \mathit{interp}\ (\mathit{fset}\ N)\ C \Longrightarrow A' \notin (\bigcup D \in \mathit{fset}\ N.\ \mathit{production}\ (\mathit{fset}\ N)\ D)$
⟨*proof*⟩

lemma *lesser-atoms-not-in-previous-interp-are-not-in-final-interp-if-not-productive*:
assumes *literal-order.is-maximal-in-mset* $C\ L$ **and** $\mathit{production}\ (\mathit{fset}\ N)\ C = \{\}$
shows $\bigwedge A'.\ A' \prec_t \mathit{atm\text{-}of}\ L \Longrightarrow A' \notin \mathit{interp}\ (\mathit{fset}\ N)\ C \Longrightarrow A' \notin (\bigcup D \in \mathit{fset}\ N.\ \mathit{production}\ (\mathit{fset}\ N)\ D)$
⟨*proof*⟩

lemma *lesser-atoms-not-in-previous-interp-are-not-in-final-interp*:
fixes A
assumes
L-max: *literal-order.is-maximal-in-mset* $C\ L$ **and**
A-less: $A \prec_t \mathit{atm\text{-}of}\ L$ **and**
A-no-in: $A \notin \mathit{interp}\ (\mathit{fset}\ N)\ C$
shows $A \notin (\bigcup D \in \mathit{fset}\ N.\ \mathit{production}\ (\mathit{fset}\ N)\ D)$
⟨*proof*⟩

lemma *lesser-atoms-in-previous-interp-are-in-final-interp*:
fixes A

assumes
L-max: *literal-order.is-maximal-in-mset* C L **and**
A-less: $A \prec_t$ *atm-of* L **and**
A-in: $A \in$ *interp* N C
shows $A \in (\bigcup D \in N.$ *production* N $D)$
⟨*proof*⟩

lemma *interp-fixed-for-smaller-literals*:
fixes A
assumes
L-max: *literal-order.is-maximal-in-mset* C L **and**
A-less: $A \prec_t$ *atm-of* L **and**
$C \prec_c D$
shows $A \in$ *interp* N $C \longleftrightarrow A \in$ *interp* N D
⟨*proof*⟩

lemma *neg-lits-not-in-model-stay-out-of-model*:
assumes
L-in: $L \in\# C$ **and**
L-neg: *is-neg* L **and**
atm-L-not-in: *atm-of* $L \notin$ *interp* N C
shows *atm-of* $L \notin (\bigcup D \in N.$ *production* N $D)$
⟨*proof*⟩

lemma *neg-lits-already-in-model-stay-in-model*:
assumes
L-in: $L \in\# C$ **and**
L-neg: *is-neg* L **and**
atm-L-not-in: *atm-of* $L \in$ *interp* N C
shows *atm-of* $L \in (\bigcup D \in N.$ *production* N $D)$
⟨*proof*⟩

lemma *image-eq-imageI*:
assumes $\bigwedge x.\ x \in X \Longrightarrow f\ x = g\ x$
shows f ‘ $X = g$ ‘ X
⟨*proof*⟩

lemma *production-swap-clause-set*:
assumes
agree: $\{D \in N1.\ D \preceq_c C\} = \{D \in N2.\ D \preceq_c C\}$
shows *production* $N1$ C = *production* $N2$ C
⟨*proof*⟩

lemma *interp-swap-clause-set*:
assumes *agree*: $\{D \in N1.\ D \prec_c C\} = \{D \in N2.\ D \prec_c C\}$
shows *interp* $N1$ C = *interp* $N2$ C
⟨*proof*⟩

definition *interp′* **where**

$interp'\ N \equiv (\bigcup C \in N.\ production\ N\ C)$

lemma *interp-eq-interp'*: $interp\ N\ D = interp'\ \{C \in N.\ C \prec_c D\}$
⟨*proof*⟩

lemma *production-unfold''*: $production\ N\ C = \{A \mid A.$
$C \in N \wedge select\ C = \{\#\} \wedge$
$is\text{-}strictly\text{-}maximal\text{-}lit\ (Pos\ A)\ C \wedge$
$\neg\ interp'\ \{B \in N.\ B \prec_c C\} \models C\}$
⟨*proof*⟩

lemma *Interp-swap-clause-set*:
assumes *agree*: $\{D \in N1.\ D \preceq_c C\} = \{D \in N2.\ D \preceq_c C\}$
shows $interp\ N1\ C \cup production\ N1\ C = interp\ N2\ C \cup production\ N2\ C$
⟨*proof*⟩

lemma *production-insert-greater-clause*:
assumes $C \prec_c D$
shows $production\ (insert\ D\ N)\ C = production\ N\ C$
⟨*proof*⟩

lemma *interp-insert-greater-clause-strong*:
assumes $C \preceq_c D$
shows $interp\ (insert\ D\ N)\ C = interp\ N\ C$
⟨*proof*⟩

lemma *interp-insert-greater-clause*:
assumes $C \prec_c D$
shows $interp\ (insert\ D\ N)\ C = interp\ N\ C$
⟨*proof*⟩

lemma *Interp-insert-greater-clause*:
assumes $C \prec_c D$
shows $interp\ (insert\ D\ N)\ C \cup production\ (insert\ D\ N)\ C = interp\ N\ C \cup production\ N\ C$
⟨*proof*⟩

lemma *production-add-irrelevant-clause-to-set0*:
assumes
fin: *finite* N **and**
D-irrelevant: $E \in N\ E \subset\#\ D\ set\text{-}mset\ D = set\text{-}mset\ E$ **and**
no-select: $select\ E = \{\#\}$
shows $production\ (insert\ D\ N)\ D = \{\}$
⟨*proof*⟩

lemma *production-add-irrelevant-clause-to-set*:
assumes
fin: *finite* N **and**
C-in: $C \in N$ **and**

D-irrelevant: $\exists E \in N.\ E \subset\# D \wedge$ *set-mset* $D =$ *set-mset* E **and**
no-select: $\bigwedge C.$ *select* $C = \{\#\}$
shows *production* (*insert* D N) $C =$ *production* N C
⟨*proof*⟩

lemma *production-add-irrelevant-clauses-to-set0*:
assumes
fin: *finite* N *finite* N' **and**
D-in: $D \in N'$ **and**
irrelevant: $\forall D \in N'.\ \exists E \in N.\ E \subset\# D \wedge$ *set-mset* $D =$ *set-mset* E **and**
no-select: $\bigwedge C.$ *select* $C = \{\#\}$
shows *production* $(N \cup N')$ $D = \{\}$
⟨*proof*⟩

lemma *production-add-irrelevant-clauses-to-set*:
assumes
fin: *finite* N *finite* N' **and**
C-in: $C \in N$ **and**
irrelevant: $\forall D \in N'.\ \exists E \in N.\ E \subset\# D \wedge$ *set-mset* $D =$ *set-mset* E **and**
no-select: $\bigwedge C.$ *select* $C = \{\#\}$
shows *production* $(N \cup N')$ $C =$ *production* N C
⟨*proof*⟩

lemma *interp-add-irrelevant-clauses-to-set*:
assumes
fin: *finite* N *finite* N' **and**
C-in: $C \in N$ **and**
irrelevant: $\forall D \in N'.\ \exists E \in N.\ E \subset\# D \wedge$ *set-mset* $D =$ *set-mset* E **and**
no-select: $\bigwedge C.$ *select* $C = \{\#\}$
shows *interp* $(N \cup N')$ $C =$ *interp* N C
⟨*proof*⟩

lemma *interp-add-irrelevant-clauses-to-set'*:
assumes
fin: *finite* N *finite* N' **and**
C-in: $C \in N$ **and**
irrelevant: $\forall D \in N'.\ \exists E \in N.\ E \subseteq\# D \wedge$ *set-mset* $D =$ *set-mset* E **and**
no-select: $\bigwedge C.$ *select* $C = \{\#\}$
shows *interp* $(N \cup N')$ $C =$ *interp* N C
⟨*proof*⟩

lemma *lesser-entailed-clause-stays-entailed'*:
assumes $C \preceq_c D$ **and** *D-lt*: $D \prec_c E$ **and** *C-entailed*: *interp* N $D \cup$ *production* N $D \models C$
shows *interp* N $E \models C$
⟨*proof*⟩

lemma *lesser-entailed-clause-stays-entailed*:
assumes *C-le*: $C \preceq_c D$ **and** *D-lt*: $D \prec_c E$ **and** *C-entailed*: *interp* N $D \cup$ *pro-*

duction N D $\models$ *C*
  shows *interp N E* $\cup$ *production N E* $\models$ *C*
⟨*proof*⟩

lemma *entailed-clause-stays-entailed'*:
  assumes *C-lt*: $C \prec_c D$ **and** *C-entailed*: *interp N C* $\cup$ *production N C* $\models$ *C*
  shows *interp N D* $\models$ *C*
  ⟨*proof*⟩

lemma *entailed-clause-stays-entailed*:
  assumes *C-lt*: $C \prec_c D$ **and** *C-entailed*: *interp N C* $\cup$ *production N C* $\models$ *C*
  shows *interp N D* $\cup$ *production N D* $\models$ *C*
  ⟨*proof*⟩

lemma *multp-if-all-left-smaller*: $M2 \neq \{\#\} \Longrightarrow \forall k \in \# M1.\ \exists j \in \# M2.\ R\ k\ j \Longrightarrow$ *multp R M1 M2*
  ⟨*proof*⟩

lemma
  fixes
    P1 :: *'f gterm* **and**
    C1 :: *'f gterm clause* **and**
    N :: *'f gterm clause set*
  defines
    $C1 \equiv \{\#Neg\ P1\#\}$ **and**
    $N \equiv \{C1\}$
  assumes
    no-select: $\bigwedge C.\ select\ C = \{\#\}$
  shows
    False
⟨*proof*⟩

lemma
  fixes
    P1 P2 :: *'f gterm* **and**
    C1 :: *'f gterm clause* **and**
    N :: *'f gterm clause set*
  defines
    $C1 \equiv \{\#Pos\ P1,\ Neg\ P2\#\}$ **and**
    $N \equiv \{C1\}$
  assumes
    term-order: $P1 \prec_t P2$ **and**
    no-select: $\bigwedge C.\ select\ C = \{\#\}$
  shows *False*
⟨*proof*⟩

lemma
fixes
P1 P2 P3 P4 :: *'f gterm* **and**
C1 C2 C3 C4 C5 :: *'f gterm clause* **and**
N :: *'f gterm clause set*
defines
C1 ≡ {#*Neg P1*, *Neg P2*#} **and**
C2 ≡ {#*Pos P2*, *Neg P3*#} **and**
C3 ≡ {#*Pos P1*, *Pos P2*, *Pos P4*#} **and**
C4 ≡ {#*Pos P2*, *Pos P3*, *Pos P4*#} **and**
C5 ≡ {#*Pos P2*, *Neg P4*#} **and**
N ≡ {*C1*, *C2*, *C3*, *C4*, *C5*}
assumes
term-order: $P1 \prec_t P2$ $P2 \prec_t P3$ $P3 \prec_t P4$ **and**
no-select: ⋀*C*. *select C* = {#}
shows
$C1 \prec_c C2$ $C2 \prec_c C3$ $C3 \prec_c C4$ $C4 \prec_c C5$
⟨*proof*⟩

interpretation *G*: *statically-complete-calculus G-Bot G-Inf G-entails G.Red-I G.Red-F*
⟨*proof*⟩

end

end
theory *Lower-Set*
imports *Main*
begin

definition *is-lower-set-wrt* :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *'a set* ⇒ *'a set* ⇒ *bool* **where**
transp-on X R ⟹ *asymp-on X R* ⟹
is-lower-set-wrt R L X ⟷ *L* ⊆ *X* ∧ (∀ *l* ∈ *L*. ∀ *x* ∈ *X*. *R x l* ⟶ *x* ∈ *L*)

definition *is-strict-lower-set-wrt* :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *'a set* ⇒ *'a set* ⇒ *bool*
where
transp-on X R ⟹ *asymp-on X R* ⟹
is-strict-lower-set-wrt R L X ⟷ *L* ⊂ *X* ∧ (∀ *l* ∈ *L*. ∀ *x* ∈ *X*. *R x l* ⟶ *x* ∈ *L*)

lemma *is-lower-set-wrt-empty*:
fixes *X* :: *'a set* **and** *R* :: *'a* ⇒ *'a* ⇒ *bool*
assumes *transp-on X R* **and** *asymp-on X R*
shows *is-lower-set-wrt R* {} *X*
⟨*proof*⟩

lemma *is-lower-set-wrt-refl*:
fixes *X* :: *'a set* **and** *R* :: *'a* ⇒ *'a* ⇒ *bool*
assumes *transp-on X R* **and** *asymp-on X R*
shows *is-lower-set-wrt R X X*

⟨proof⟩

lemma *is-lower-set-wrt-trans*:
fixes X Y Z :: *'a set* **and** R :: *'a ⇒ 'a ⇒ bool*
assumes
transp-on Z R **and** *asymp-on* Z R **and**
is-lower-set-wrt R X Y **and** *is-lower-set-wrt* R Y Z
shows *is-lower-set-wrt* R X Z
⟨proof⟩

lemma *is-lower-set-wrt-antisym*:
fixes X Y :: *'a set* **and** R :: *'a ⇒ 'a ⇒ bool*
assumes
transp-on Y R **and** *asymp-on* Y R **and**
is-lower-set-wrt R X Y **and** *is-lower-set-wrt* R Y X
shows $X = Y$
⟨proof⟩

lemma *order-is-lower-set-wrt*:
fixes R :: *'a ⇒ 'a ⇒ bool*
assumes *transp* R **and** *asymp* R
shows *class.order* (*is-lower-set-wrt* R) (*is-strict-lower-set-wrt* R)
⟨proof⟩

lemma *is-lower-set-wrt-insertI*:
assumes *transp-on* (*insert* x X) R **and** *asymp-on* (*insert* x X) R **and**
$x \in X$ **and** $\forall w \in X.\ R\ w\ x \longrightarrow w \in L$ **and** *is-lower-set-wrt* R L X
shows *is-lower-set-wrt* R (*insert* x L) X
⟨proof⟩

lemma *lower-set-wrt-appendI*:
assumes
trans: *transp-on* (*set* (*xs* @ *ys*)) R **and**
asym: *asymp-on* (*set* (*xs* @ *ys*)) R **and**
sorted: *sorted-wrt* R (*xs* @ *ys*)
shows *is-lower-set-wrt* R (*set xs*) (*set* (*xs* @ *ys*))
⟨proof⟩

lemma *sorted-and-lower-set-wrt-appendD-left*:
assumes *transp-on* A R **and** *asymp-on* A R **and**
sorted-wrt R (*xs* @ *ys*) **and** *is-lower-set-wrt* R (*set* (*xs* @ *ys*)) A
shows *sorted-wrt* R *xs* **and** *is-lower-set-wrt* R (*set xs*) A
⟨proof⟩

lemma *sorted-and-lower-set-wrt-appendD-right*:
assumes *transp-on* A R **and** *asymp-on* A R **and**
sorted-wrt ($\lambda x\ y.\ R\ y\ x$) (*xs* @ *ys*) **and** *is-lower-set-wrt* R (*set* (*xs* @ *ys*)) A
shows *sorted-wrt* ($\lambda x\ y.\ R\ y\ x$) *ys* **and** *is-lower-set-wrt* R (*set ys*) A
⟨proof⟩

lemma *not-in-lower-set-wrtI*:
fixes R :: $'a \Rightarrow 'a \Rightarrow$ *bool*
assumes *trans*: *transp-on* Y R **and** *asym*: *asymp-on* Y R
shows *is-lower-set-wrt* R X $Y \Longrightarrow y \notin X \Longrightarrow y \in Y \Longrightarrow R\ y\ z \Longrightarrow z \notin X$
⟨*proof*⟩

abbreviation (**in** *preorder*) *is-lower-set* **where**
is-lower-set ≡ *is-lower-set-wrt* (<)

lemmas (**in** *preorder*) *is-lower-set-iff* =
is-lower-set-wrt-def[*OF transp-on-less asymp-on-less*]

context *linorder* **begin**

sublocale *is-lower-set*: *order is-lower-set-wrt* (<) *is-strict-lower-set-wrt* (<)
⟨*proof*⟩

end

lemmas (**in** *preorder*) *is-lower-set-empty*[*simp*] =
is-lower-set-wrt-empty[*OF transp-on-less asymp-on-less*]

lemmas (**in** *preorder*) *is-lower-set-insertI* =
is-lower-set-wrt-insertI[*OF transp-on-less asymp-on-less*]

lemmas (**in** *preorder*) *lower-set-appendI* =
lower-set-wrt-appendI[*OF transp-on-less asymp-on-less*]

lemmas (**in** *preorder*) *sorted-and-lower-set-appendD-left* =
sorted-and-lower-set-wrt-appendD-left[*OF transp-on-less asymp-on-less*]

lemmas (**in** *preorder*) *sorted-and-lower-set-appendD-right* =
sorted-and-lower-set-wrt-appendD-right[*OF transp-on-less asymp-on-less*]

lemmas (**in** *preorder*) *not-in-lower-setI* =
not-in-lower-set-wrtI[*OF transp-on-less asymp-on-less*]

end
theory *HOL-Extra-Extra*
imports *Superposition-Calculus.HOL-Extra*
begin

no-notation *restrict-map* (**infixl** |‘ *110*)

lemma
assumes $\exists_{\leq 1} x.\ P\ x$

shows *finite* $\{x.\ P\ x\}$
⟨*proof*⟩

lemma *finite-if-Uniq-Uniq*:
assumes
$\exists_{\leq 1} x.\ P\ x$
$\forall x.\ \exists_{\leq 1} y.\ Q\ x\ y$
shows *finite* $\{y.\ \exists x.\ P\ x \wedge Q\ x\ y\}$
⟨*proof*⟩

lemma *finite-if-finite-finite*:
assumes
finite $\{x.\ P\ x\}$
$\forall x.$ *finite* $\{y.\ Q\ x\ y\}$
shows *finite* $\{y.\ \exists x.\ P\ x \wedge Q\ x\ y\}$
⟨*proof*⟩

lemma *strict-partial-order-wfp-on-finite-set*:
assumes *transp-on* $\mathcal{X}$ *R* **and** *asymp-on* $\mathcal{X}$ *R*
shows *finite* $\mathcal{X} \Longrightarrow$ *Wellfounded.wfp-on* $\mathcal{X}$ *R*
⟨*proof*⟩

lemma (**in** *order*) *greater-wfp-on-finite-set*: *finite* $\mathcal{X} \Longrightarrow$ *Wellfounded.wfp-on* $\mathcal{X}$ $(>)$
⟨*proof*⟩

lemma (**in** *order*) *less-wfp-on-finite-set*: *finite* $\mathcal{X} \Longrightarrow$ *Wellfounded.wfp-on* $\mathcal{X}$ $(<)$
⟨*proof*⟩

lemma *sorted-wrt-dropWhile*: *sorted-wrt R xs* $\Longrightarrow$ *sorted-wrt R* (*dropWhile P xs*)
⟨*proof*⟩

lemma *sorted-wrt-takeWhile*: *sorted-wrt R xs* $\Longrightarrow$ *sorted-wrt R* (*takeWhile P xs*)
⟨*proof*⟩

lemma *distinct-if-sorted-wrt-asymp*:
assumes *asymp-on* (*set xs*) *R* **and** *sorted-wrt R xs*
shows *distinct xs*
⟨*proof*⟩

lemma *dropWhile-append-eq-rhs*:
fixes *xs ys* :: *'a list* **and** *P* :: *'a* ⇒ *bool*
assumes
$\bigwedge x.\ x \in$ *set xs* $\Longrightarrow P\ x$ **and**
$\bigwedge y.\ y \in$ *set ys* $\Longrightarrow \neg\ P\ y$
shows *dropWhile P* (*xs* @ *ys*) = *ys*
⟨*proof*⟩

lemma *mem-set-dropWhile-conv-if-list-sorted-and-pred-monotone*:
fixes $R :: 'a \Rightarrow 'a \Rightarrow bool$ **and** $xs :: 'a\ list$ **and** $P :: 'a \Rightarrow bool$
assumes *sorted-wrt R xs* **and** *monotone-on (set xs) R* $(\geq)$ P
shows $x \in set\ (dropWhile\ P\ xs) \longleftrightarrow \neg\ P\ x \wedge x \in set\ xs$
⟨*proof*⟩

lemma *ball-set-dropWhile-if-sorted-wrt-and-monotone-on*:
fixes $R :: 'a \Rightarrow 'a \Rightarrow bool$ **and** $xs :: 'a\ list$ **and** $P :: 'a \Rightarrow bool$
assumes *sorted-wrt R xs* **and** *monotone-on (set xs) R* $(\geq)$ P
shows $\forall x \in set\ (dropWhile\ P\ xs).\ \neg\ P\ x$
⟨*proof*⟩

lemma *filter-set-eq-filter-set-minus-singleton*:
assumes $\neg\ P\ y$
shows $\{x \in X.\ P\ x\} = \{x \in X - \{y\}.\ P\ x\}$
⟨*proof*⟩

lemma *ex1-subset-eq-image-if-bij-betw*:
fixes $f :: 'a \Rightarrow 'b$ **and** $X :: 'a\ set$ **and** $Y :: 'b\ set$
assumes *bij-betw f X Y* **and** $Y' \subseteq Y$
shows $\exists!X'.\ X' \subseteq X \wedge Y' = f\ `\ X'$
⟨*proof*⟩

lemma *Collect-eq-image-filter-Collect-if-bij-betw*:
fixes $f :: 'a \Rightarrow 'b$ **and** $X :: 'a\ set$ **and** $Y :: 'b\ set$
assumes *bij*: *bij-betw f X Y* **and** *sub*: $\{y.\ P\ y\} \subseteq Y$
shows $\{y.\ P\ y\} = f\ `\ \{x.\ x \in X \wedge P\ (f\ x)\}$
⟨*proof*⟩

lemma (**in** *linorder*) *ex1-sorted-list-for-set-if-finite*:
$finite\ X \Longrightarrow \exists!xs.\ sorted\text{-}wrt\ (<)\ xs \wedge set\ xs = X$
⟨*proof*⟩

lemma *restrict-map-ident-if-dom-subset*: $dom\ \mathcal{M} \subseteq A \Longrightarrow restrict\text{-}map\ \mathcal{M}\ A = \mathcal{M}$
⟨*proof*⟩

lemma *dropWhile-ident-if-pred-always-false*:
assumes $\bigwedge x.\ x \in set\ xs \Longrightarrow \neg\ P\ x$
shows $dropWhile\ P\ xs = xs$
⟨*proof*⟩

1.6 Move to *HOL.Transitive-Closure*

lemma *relpowp-right-unique*:
fixes $R :: 'a \Rightarrow 'a \Rightarrow bool$ **and** $n :: nat$ **and** $x\ y\ z :: 'a$
assumes *runique*: $\bigwedge x\ y\ z.\ R\ x\ y \Longrightarrow R\ x\ z \Longrightarrow y = z$
shows $(R\ \widehat{\ }\widehat{\ }\ n)\ x\ y \Longrightarrow (R\ \widehat{\ }\widehat{\ }\ n)\ x\ z \Longrightarrow y = z$
⟨*proof*⟩

lemma *Uniq-relpowp*:
 fixes $n :: nat$ **and** $R :: 'a \Rightarrow 'a \Rightarrow bool$
 assumes *runiq*: $\forall x.\ \exists_{\leq 1} y.\ R\ x\ y$
 shows $\exists_{\leq 1} y.\ (R\ \widehat{}\widehat{}\ n)\ x\ y$
⟨*proof*⟩

lemma *relpowp-plus-of-right-unique*:
 assumes
  right-unique R
  $(R\ \widehat{}\widehat{}\ m)\ x\ y$ **and**
  $(R\ \widehat{}\widehat{}\ (m + n))\ x\ z$
 shows $(R\ \widehat{}\widehat{}\ n)\ y\ z$
 ⟨*proof*⟩

lemma *relpowp-plusD*:
 assumes $(R\ \widehat{}\widehat{}\ (m + n))\ x\ z$
 shows $\exists\, y.\ (R\ \widehat{}\widehat{}\ m)\ x\ y \wedge (R\ \widehat{}\widehat{}\ n)\ y\ z$
 ⟨*proof*⟩

lemma *relpowp-Suc-of-right-unique*:
 assumes
  right-unique R
  $R\ x\ y$ **and**
  $(R\ \widehat{}\widehat{}\ Suc\ n)\ x\ z$
 shows $(R\ \widehat{}\widehat{}\ n)\ y\ z$
 ⟨*proof*⟩

lemma *relpowp-trans*[*trans*]:
 $(R\ \widehat{}\widehat{}\ i)\ x\ y \Longrightarrow (R\ \widehat{}\widehat{}\ j)\ y\ z \Longrightarrow (R\ \widehat{}\widehat{}\ (i + j))\ x\ z$
⟨*proof*⟩

lemma *tranclp-if-relpowp*: $n \neq 0 \Longrightarrow (R\ \widehat{}\widehat{}\ n)\ x\ y \Longrightarrow R^{++}\ x\ y$
 ⟨*proof*⟩

lemma *ex-terminating-rtranclp-strong*:
 assumes *wf*: *Wellfounded.wfp-on* $\{x'.\ R^{**}\ x\ x'\}\ R^{-1-1}$
 shows $\exists\, y.\ R^{**}\ x\ y \wedge (\nexists\, z.\ R\ y\ z)$
⟨*proof*⟩

lemma *transp-on-singleton*[*simp*]: *transp-on* $\{x\}\ R$
 ⟨*proof*⟩

lemma *rtranclp-rtranclp-compose-if-right-unique*:
 assumes *runique*: *right-unique* R **and** $R^{**}\ a\ b$ **and** $R^{**}\ a\ c$
 shows $R^{**}\ a\ b \wedge R^{**}\ b\ c \vee R^{**}\ a\ c \wedge R^{**}\ c\ b$
 ⟨*proof*⟩

lemma *right-unique-terminating-rtranclp*:

assumes *right-unique R*
shows *right-unique* $(\lambda x\ y.\ R^{**}\ x\ y \wedge (\nexists z.\ R\ y\ z))$
⟨*proof*⟩

lemma *ex-terminating-rtranclp*:
assumes *wf*: *wfp* R^{-1-1}
shows $\exists y.\ R^{**}\ x\ y \wedge (\nexists z.\ R\ y\ z)$
⟨*proof*⟩

end
theory *The-Optional*
imports *Main*
begin

definition *The-optional* :: $('a \Rightarrow bool) \Rightarrow 'a\ option$ **where**
The-optional P = (*if* $\exists!x.\ P\ x$ *then Some* (*THE x. P x*) *else None*)

lemma *The-optional-eq-SomeD*: *The-optional P* = *Some x* $\Longrightarrow$ *P x*
⟨*proof*⟩

lemma *Some-eq-The-optionalD*: *Some x* = *The-optional P* $\Longrightarrow$ *P x*
⟨*proof*⟩

lemma *The-optional-eq-NoneD*: *The-optional P* = *None* $\Longrightarrow \nexists!x.\ P\ x$
⟨*proof*⟩

lemma *None-eq-The-optionalD*: *None* = *The-optional P* $\Longrightarrow \nexists!x.\ P\ x$
⟨*proof*⟩

lemma *The-optional-eq-SomeI*:
assumes $\exists_{\leq 1}x.\ P\ x$ **and** *P x*
shows *The-optional P* = *Some x*
⟨*proof*⟩

end
theory *Full-Run*
imports
VeriComp.Transfer-Extras
HOL-Extra-Extra
begin

definition *full-run* **where**
full-run $\mathcal{R}\ x\ y \longleftrightarrow \mathcal{R}^{**}\ x\ y \wedge (\nexists z.\ \mathcal{R}\ y\ z)$

lemma *Uniq-full-run*:
assumes *Uniq-R*: $\bigwedge x.\ \exists_{\leq 1}y.\ R\ x\ y$
shows $\exists_{\leq 1}y.$ *full-run R x y*
⟨*proof*⟩

lemma *ex1-full-run*:
  assumes *Uniq-R*: $\bigwedge x.\ \exists_{\leq 1} y.\ R\ x\ y$ **and** *wfP-R*: *wfP* R^{-1-1}
  shows $\exists!y.$ *full-run* $R\ x\ y$
⟨*proof*⟩

lemma *full-run-preserves-invariant*:
  assumes
    run: *full-run* $R\ x\ y$ **and**
    P-init: $P\ x$ **and**
    R-preserves-P: $\bigwedge x\ y.\ R\ x\ y \Longrightarrow P\ x \Longrightarrow P\ y$
  shows $P\ y$
⟨*proof*⟩

end
theory *Background*
  imports
    Simple-Clause-Learning.SCL-FOL
    Simple-Clause-Learning.Correct-Termination
    Simple-Clause-Learning.Initial-Literals-Generalize-Learned-Literals
    Simple-Clause-Learning.Termination
    Ground-Ordered-Resolution
    Min-Max-Least-Greatest.Min-Max-Least-Greatest-FSet
    Superposition-Calculus.Multiset-Extra
    VeriComp.Compiler
    HOL−ex.Sketch-and-Explore
    HOL−Library.FuncSet
    Lower-Set
    HOL-Extra-Extra
    The-Optional
    Full-Run
begin

lemma $I \models l\ L \longleftrightarrow$ (*is-pos* $L \longleftrightarrow$ *atm-of* $L \in I$)
  ⟨*proof*⟩

2 Move to *HOL−Library.Multiset*

lemmas *strict-subset-implies-multp* = *subset-implies-multp*
hide-fact *subset-implies-multp*

lemma *subset-implies-reflclp-multp*: $A \subseteq\#\ B \Longrightarrow$ (*multp* $R)^{==}\ A\ B$
  ⟨*proof*⟩

lemma *member-mset-repeat-msetD*: $L \in\#$ *repeat-mset* $n\ M \Longrightarrow L \in\#\ M$
  ⟨*proof*⟩

lemma *member-mset-repeat-mset-Suc*[*simp*]: $L \in\#$ *repeat-mset* (*Suc* n) $M \longleftrightarrow L$
$\in\#\ M$
  ⟨*proof*⟩

lemma *image-msetI*: $x \in\# M \Longrightarrow f\, x \in\#$ *image-mset f M*
⟨*proof*⟩

lemma *inj-image-mset-mem-iff*: *inj f* $\Longrightarrow f\, x \in\#$ *image-mset f M* $\longleftrightarrow x \in\# M$
⟨*proof*⟩

3 Move to *HOL–Library.FSet*

declare *wfP-pfsubset*[*intro*]

syntax
-FFilter :: *pttrn* ⇒ *'a fset* ⇒ *bool* ⇒ *'a fset* ((*1*{|- |∈| -./ -|}))

translations
{|*x* |∈| *X*. *P*|} == *CONST ffilter* (λ*x*. *P*) *X*

lemma *fimage-ffUnion*: *f* |'| *ffUnion SS* = *ffUnion* ((|'|) *f* |'| *SS*)
⟨*proof*⟩

lemma *ffilter-eq-ffilter-minus-singleton*:
assumes ¬ *P y*
shows {|*x* |∈| *X*. *P x*|} = {|*x* |∈| *X* − {|*y*|}. *P x*|}
⟨*proof*⟩

lemma *fun-upd-fimage*: *f*(*x* := *y*) |'| *A* = (*if x* |∈| *A then finsert y* (*f* |'| (*A* − {|*x*|})) *else f* |'| *A*)
⟨*proof*⟩

lemma *ffilter-fempty*[*simp*]: *ffilter P* {||} = {||}
⟨*proof*⟩

lemma *fstrict-subset-iff-fset-strict-subset-fset*:
fixes $\mathcal{X}$ $\mathcal{Y}$ :: *- fset*
shows $\mathcal{X}$ |⊂| $\mathcal{Y}$ ⟷ *fset* $\mathcal{X}$ ⊂ *fset* $\mathcal{Y}$
⟨*proof*⟩

lemma (**in** *linorder*) *ex1-sorted-list-for-fset*:
∃!*xs*. *sorted-wrt* (<) *xs* ∧ *fset-of-list xs* = *X*
⟨*proof*⟩

lemma (**in** *linorder*) *is-least-in-fset-ffilterD*:
assumes *is-least-in-fset-wrt* (<) (*ffilter P X*) *x*
shows *x* |∈| *X P x*
⟨*proof*⟩

4 Move to *VeriComp.Simulation*

locale *forward-simulation-with-measuring-function* =
L1: *semantics step1 final1* +
L2: *semantics step2 final2*
for
step1 :: *'state1* ⇒ *'state1* ⇒ *bool* **and**
step2 :: *'state2* ⇒ *'state2* ⇒ *bool* **and**
final1 :: *'state1* ⇒ *bool* **and**
final2 :: *'state2* ⇒ *bool* +
fixes
match :: *'state1* ⇒ *'state2* ⇒ *bool* **and**
measure :: *'state1* ⇒ *'index* **and**
order :: *'index* ⇒ *'index* ⇒ *bool* (**infix** ⊏ *70*)
assumes
wfp-order:
wfp (⊏) **and**
match-final:
match s1 s2 ⟹ *final1 s1* ⟹ *final2 s2* **and**
simulation:
match s1 s2 ⟹ *step1 s1 s1'* ⟹
(∃ *s2'*. *step2*$^{++}$ *s2 s2'* ∧ *match s1' s2'*) ∨ (*match s1' s2* ∧ *measure s1'* ⊏
measure s1)
begin

sublocale *forward-simulation* **where**
step1 = *step1* **and** *step2* = *step2* **and** *final1* = *final1* **and** *final2* = *final2* **and**
order = *order* **and**
match = λ*i x y*. *i* = *measure x* ∧ *match x y*
⟨*proof*⟩

end

locale *backward-simulation-with-measuring-function* =
L1: *semantics step1 final1* +
L2: *semantics step2 final2*
for
step1 :: *'state1* ⇒ *'state1* ⇒ *bool* **and**
step2 :: *'state2* ⇒ *'state2* ⇒ *bool* **and**
final1 :: *'state1* ⇒ *bool* **and**
final2 :: *'state2* ⇒ *bool* +
fixes
match :: *'state1* ⇒ *'state2* ⇒ *bool* **and**
measure :: *'state2* ⇒ *'index* **and**
order :: *'index* ⇒ *'index* ⇒ *bool* (**infix** ⊏ *70*)
assumes
wfp-order:
wfp (⊏) **and**
match-final:

$match\ s1\ s2 \Longrightarrow final2\ s2 \Longrightarrow final1\ s1$ **and**
simulation:
$match\ s1\ s2 \Longrightarrow step2\ s2\ s2' \Longrightarrow$
$(\exists\, s1'.\ step1^{++}\ s1\ s1' \wedge match\ s1'\ s2') \vee (match\ s1\ s2' \wedge measure\ s2' \sqsubset measure\ s2)$
begin

sublocale *backward-simulation* **where**
$step1 = step1$ **and** $step2 = step2$ **and** $final1 = final1$ **and** $final2 = final2$ **and** $order = order$ **and**
$match = \lambda i\ x\ y.\ i = measure\ y \wedge match\ x\ y$
⟨*proof*⟩

end

5 Move to *Simple-Clause-Learning.SCL-FOL*

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

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

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

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

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

definition *trail-defined-lit* :: $('\text{-}\ literal \times '\text{-}\ option)\ list \Rightarrow '\text{-}\ literal \Rightarrow bool$ **where**
$trail\text{-}defined\text{-}lit\ \Gamma\ L \longleftrightarrow (L \in fst\ `\ set\ \Gamma \vee -\ L \in fst\ `\ set\ \Gamma)$

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

definition *trail-defined-cls* :: $('\text{-}\ literal \times '\text{-}\ option)\ list \Rightarrow '\text{-}\ clause \Rightarrow bool$ **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*:
$trail\text{-}defined\text{-}lit\ \Gamma\ L \longleftrightarrow trail\text{-}true\text{-}lit\ \Gamma\ L \vee trail\text{-}false\text{-}lit\ \Gamma\ L$
⟨*proof*⟩

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

inductive *trail-consistent* :: $('a\ literal \times 'b\ option)\ list \Rightarrow bool$ **where**
Nil[*simp*]: *trail-consistent* [] |
Cons: $\neg$ *trail-defined-lit* Γ $L \Longrightarrow$ *trail-consistent* $\Gamma \Longrightarrow$ *trail-consistent* $((L,\ u)\ \#\ \Gamma)$

lemma *distinct-lits-if-trail-consistent*:
trail-consistent $\Gamma \Longrightarrow$ *distinct* (*map fst* Γ)
⟨*proof*⟩

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

lemma *trail-true-cls-if-trail-true-suffix*:
suffix Γ' $\Gamma \Longrightarrow$ *trail-true-cls* Γ' $C \Longrightarrow$ *trail-true-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-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 *not-trail-true-lit-and-trail-false-lit*:
fixes Γ :: $('a\ literal \times 'b\ option)\ list$ **and** L :: $'a\ literal$
shows *trail-consistent* $\Gamma \Longrightarrow \neg$ (*trail-true-lit* Γ L $\wedge$ *trail-false-lit* Γ L)
⟨*proof*⟩

lemma *not-trail-true-cls-and-trail-false-cls*:
fixes Γ :: $('a\ literal \times 'b\ option)\ list$ **and** C :: $'a\ clause$
shows *trail-consistent* $\Gamma \Longrightarrow \neg$ (*trail-true-cls* Γ C $\wedge$ *trail-false-cls* Γ C)
⟨*proof*⟩

lemma *not-lit-and-comp-lit-false-if-trail-consistent*:
assumes *trail-consistent* Γ
shows ¬ (*trail-false-lit* Γ *L* ∧ *trail-false-lit* Γ (− *L*))
⟨*proof*⟩

lemma *not-both-lit-and-comp-lit-in-false-clause-if-trail-consistent*:
assumes Γ-*consistent*: *trail-consistent* Γ **and** *C-false*: *trail-false-cls* Γ *C*
shows ¬ (*L* ∈# *C* ∧ − *L* ∈# *C*)
⟨*proof*⟩

6 Move to ground ordered resolution

lemma (**in** *ground-ordered-resolution-calculus*) *unique-ground-resolution*:
shows $\exists_{\leq 1}C$. *ground-resolution P1 P2 C*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *unique-ground-factoring*:
shows $\exists_{\leq 1}C$. *ground-factoring P C*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *termination-ground-factoring*:
shows *wfP ground-factoring*$^{-1-1}$
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *atms-of-concl-subset-if-ground-resolution*:
assumes *ground-resolution* P_1 P_2 *C*
shows *atms-of C* ⊆ *atms-of* P_1 ∪ *atms-of* P_2
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *strict-subset-mset-if-ground-factoring*:
assumes *ground-factoring P C*
shows *C* ⊂# *P*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *set-mset-eq-set-mset-if-ground-factoring*:
assumes *ground-factoring P C*
shows *set-mset P* = *set-mset C*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *atms-of-concl-eq-if-ground-factoring*:
assumes *ground-factoring P C*
shows *atms-of C* = *atms-of P*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *ground-factoring-preserves-maximal-literal*:
assumes *ground-factoring P C*
shows *is-maximal-lit L P* = *is-maximal-lit L C*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *ground-factorings-preserves-maximal-literal*:
assumes *ground-factoring*** *P C*
shows *is-maximal-lit L P* = *is-maximal-lit L C*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *ground-factoring-reduces-maximal-pos-lit*:
assumes *ground-factoring P C* **and** *is-pos L* **and**
is-maximal-lit L P **and** *count P L* = *Suc* (*Suc n*)
shows *is-maximal-lit L C* **and** *count C L* = *Suc n*
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *ground-factorings-reduces-maximal-pos-lit*:
assumes (*ground-factoring* ^^ *m*) *P C* **and** $m \leq$ *Suc n* **and** *is-pos L* **and**
is-maximal-lit L P **and** *count P L* = *Suc* (*Suc n*)
shows *is-maximal-lit L C* **and** *count C L* = *Suc* (*Suc n* − *m*)
⟨*proof*⟩

lemma (**in** *ground-ordered-resolution-calculus*) *full-ground-factorings-reduces-maximal-pos-lit*:
assumes *steps*: (*ground-factoring* ^^ *Suc n*) *P C* **and** *L-pos*: *is-pos L* **and**
L-max: *is-maximal-lit L P* **and** *L-count*: *count P L* = *Suc* (*Suc n*)
shows *is-maximal-lit L C* **and** *count C L* = *Suc 0*
⟨*proof*⟩

7 Move somewhere?

lemma *true-cls-imp-neq-mempty*: $\mathcal{I} \Vdash C \Longrightarrow C \neq \{\#\}$
⟨*proof*⟩

lemma *lift-tranclp-to-pairs-with-constant-fst*:
$(R\ x)^{++}\ y\ z \Longrightarrow (\lambda(x,\ y)\ (x',\ z).\ x = x' \wedge R\ x\ y\ z)^{++}\ (x,\ y)\ (x,\ z)$
⟨*proof*⟩

abbreviation (**in** *preorder*) *is-lower-fset* **where**
is-lower-fset X Y ≡ *is-lower-set-wrt* (<) (*fset X*) (*fset Y*)

lemma *lower-set-wrt-prefixI*:
assumes
trans: *transp-on* (*set zs*) *R* **and**
asym: *asymp-on* (*set zs*) *R* **and**
sorted: *sorted-wrt R zs* **and**
prefix: *prefix xs zs*
shows *is-lower-set-wrt R* (*set xs*) (*set zs*)
⟨*proof*⟩

lemmas (**in** *preorder*) *lower-set-prefixI* =
lower-set-wrt-prefixI[*OF transp-on-less asymp-on-less*]

lemma *lower-set-wrt-suffixI*:
assumes

trans: *transp-on* (*set zs*) *R* **and**
asym: *asymp-on* (*set zs*) *R* **and**
sorted: *sorted-wrt* R^{-1-1} *zs* **and**
suffix: *suffix ys zs*
shows *is-lower-set-wrt R* (*set ys*) (*set zs*)
⟨*proof*⟩

lemmas (**in** *preorder*) *lower-set-suffixI* =
lower-set-wrt-suffixI[*OF transp-on-less asymp-on-less*]

lemma *true-cls-repeat-mset-Suc*[*simp*]: $I \models$ *repeat-mset* (*Suc n*) $C \longleftrightarrow I \models C$
⟨*proof*⟩

lemma (**in** *backward-simulation*)
assumes *match i S1 S2* **and** ¬ *L1.inf-step S1*
shows ¬ *L2.inf-step S2*
⟨*proof*⟩

lemma (**in** *scl-fol-calculus*) *grounding-of-clss-ground*:
assumes *is-ground-clss N*
shows *grounding-of-clss N* = *N*
⟨*proof*⟩

lemma (**in** *scl-fol-calculus*) *propagateI'*:
C |∈| N |∪| $U \Longrightarrow C =$ *add-mset* $L\ C' \Longrightarrow$ *is-ground-cls* $(C \cdot \gamma) \Longrightarrow$
$\forall K \in\# C \cdot \gamma.$ *atm-of* $K \preceq_B \beta \Longrightarrow$
$C_0 = \{\#K \in\# C'.\ K \cdot l\ \gamma \neq L \cdot l\ \gamma\#\} \Longrightarrow C_1 = \{\#K \in\# C'.\ K \cdot l\ \gamma = L \cdot l\ \gamma\#\} \Longrightarrow$
SCL-FOL.trail-false-cls Γ $(C_0 \cdot \gamma) \Longrightarrow \neg$ *SCL-FOL.trail-defined-lit* Γ $(L \cdot l\ \gamma) \Longrightarrow$
is-imgu μ {*atm-of* ‘ *set-mset* (*add-mset* $L\ C_1$)} $\Longrightarrow$
Γ' = *trail-propagate* Γ $(L \cdot l\ \mu)$ $(C_0 \cdot \mu)$ $\gamma \Longrightarrow$
propagate $N\ \beta$ (Γ, U, *None*) (Γ', U, *None*)
⟨*proof*⟩

lemma (**in** *scl-fol-calculus*) *decideI'*:
is-ground-lit $(L \cdot l\ \gamma) \Longrightarrow \neg$ *SCL-FOL.trail-defined-lit* Γ $(L \cdot l\ \gamma) \Longrightarrow$ *atm-of* $L \cdot a\ \gamma \preceq_B \beta \Longrightarrow$
Γ' = *trail-decide* Γ $(L \cdot l\ \gamma) \Longrightarrow$
decide $N\ \beta$ (Γ, U, *None*) (Γ', U, *None*)
⟨*proof*⟩

lemma *ground-iff-vars-term-empty*: *ground t* $\longleftrightarrow$ *vars-term t* = {}
⟨*proof*⟩

lemma *is-ground-atm-eq-ground*[*iff*]: *is-ground-atm* = *ground*
⟨*proof*⟩

definition *lit-of-glit* :: *'f gterm literal* ⇒ (*'f*, *'v*) *term literal* **where**
lit-of-glit = *map-literal term-of-gterm*

definition *glit-of-lit* **where**
glit-of-lit = *map-literal gterm-of-term*

definition *cls-of-gcls* **where**
cls-of-gcls = *image-mset lit-of-glit*

definition *gcls-of-cls* **where**
gcls-of-cls = *image-mset glit-of-lit*

lemma *inj-lit-of-glit*: *inj lit-of-glit*
⟨*proof*⟩

lemma *atm-of-lit-of-glit-conv*: *atm-of* (*lit-of-glit L*) = *term-of-gterm* (*atm-of L*)
⟨*proof*⟩

lemma *ground-atm-of-lit-of-glit*[*simp*]: *Term-Context.ground* (*atm-of* (*lit-of-glit L*))
⟨*proof*⟩

lemma *is-ground-lit-lit-of-glit*[*simp*]: *is-ground-lit* (*lit-of-glit L*)
⟨*proof*⟩

lemma *is-ground-cls-cls-of-gcls*[*simp*]: *is-ground-cls* (*cls-of-gcls C*)
⟨*proof*⟩

lemma *glit-of-lit-lit-of-glit*[*simp*]: *glit-of-lit* (*lit-of-glit L*) = *L*
⟨*proof*⟩

lemma *gcls-of-cls-cls-of-gcls*[*simp*]: *gcls-of-cls* (*cls-of-gcls L*) = *L*
⟨*proof*⟩

lemma *lit-of-glit-glit-of-lit-ident*[*simp*]: *is-ground-lit L* ⟹ *lit-of-glit* (*glit-of-lit L*) = *L*
⟨*proof*⟩

lemma *cls-of-gcls-gcls-of-cls-ident*[*simp*]: *is-ground-cls D* ⟹ *cls-of-gcls* (*gcls-of-cls D*) = *D*
⟨*proof*⟩

lemma *vars-lit-lit-of-glit*[*simp*]: *vars-lit* (*lit-of-glit L*) = {}
⟨*proof*⟩

lemma *vars-cls-cls-of-gcls*[*simp*]: *vars-cls* (*cls-of-gcls C*) = {}
⟨*proof*⟩

definition *atms-of-cls* :: *'a clause* ⇒ *'a fset* **where**
atms-of-cls C = *atm-of* |'| *fset-mset C*

definition *atms-of-clss* :: *'a clause fset* ⇒ *'a fset* **where**
atms-of-clss N = *ffUnion* (*atms-of-cls* |'| *N*)

lemma *atms-of-clss-fempty*[*simp*]: *atms-of-clss* {||} = {||}
⟨*proof*⟩

lemma *atms-of-clss-finsert*[*simp*]:
atms-of-clss (*finsert C N*) = *atms-of-cls C* |∪| *atms-of-clss N*
⟨*proof*⟩

definition *lits-of-clss* :: *'a clause fset* ⇒ *'a literal fset* **where**
lits-of-clss N = *ffUnion* (*fset-mset* |'| *N*)

definition *lit-occures-in-clss* **where**
lit-occures-in-clss L N ⟷ *fBex N* (λ*C*. *L* ∈# *C*)

inductive *constant-context* **for** *R* **where**
$R\ \mathcal{C}\ \mathcal{D}\ \mathcal{D}' \Longrightarrow$ *constant-context R* $(\mathcal{C}, \mathcal{D})\ (\mathcal{C}, \mathcal{D}')$

lemma *rtranclp-constant-context*: $(R\ \mathcal{C})^{**}\ \mathcal{D}\ \mathcal{D}' \Longrightarrow$ (*constant-context R*)** $(\mathcal{C}, \mathcal{D})$ $(\mathcal{C}, \mathcal{D}')$
⟨*proof*⟩

lemma *tranclp-constant-context*: $(R\ \mathcal{C})^{++}\ \mathcal{D}\ \mathcal{D}' \Longrightarrow$ (*constant-context R*)$^{++}$ $(\mathcal{C}, \mathcal{D})$ $(\mathcal{C}, \mathcal{D}')$
⟨*proof*⟩

lemma *right-unique-constant-context*:
assumes *R-ru*: $\bigwedge\mathcal{C}$. *right-unique* $(R\ \mathcal{C})$
shows *right-unique* (*constant-context R*)
⟨*proof*⟩

lemma *safe-state-constant-context-if-invars*:
fixes *N s*
assumes
$\mathcal{R}$*-preserves-*$\mathcal{I}$:
$\bigwedge N\ s\ s'.\ \mathcal{R}\ N\ s\ s' \Longrightarrow \mathcal{I}\ N\ s \Longrightarrow \mathcal{I}\ N\ s'$ **and**
ex-$\mathcal{R}$*-if-not-final*:
$\bigwedge N\ s.\ \neg\ \mathcal{F}\ (N, s) \Longrightarrow \mathcal{I}\ N\ s \Longrightarrow \exists s'.\ \mathcal{R}\ N\ s\ s'$
assumes *invars*: $\mathcal{I}\ N\ s$
shows *safe-state* (*constant-context* $\mathcal{R}$) $\mathcal{F}$ (N, s)
⟨*proof*⟩

primrec *trail-atms* :: (- *literal* × -) *list* ⇒ - *fset* **where**
trail-atms [] = {||} |
trail-atms (*Ln* # Γ) = *finsert* (*atm-of* (*fst Ln*)) (*trail-atms* Γ)

lemma *fset-trail-atms*: *fset* (*trail-atms* Γ) = *atm-of* ' *fst* ' *set* Γ

⟨*proof*⟩

lemma *trail-defined-lit-iff-trail-defined-atm*:
  trail-defined-lit Γ $L \longleftrightarrow$ *atm-of* L $|\in|$ *trail-atms* Γ
⟨*proof*⟩

lemma *trail-atms-subset-if-suffix*:
  assumes *suffix* Γ' Γ
  shows *trail-atms* Γ' $|\subseteq|$ *trail-atms* Γ
⟨*proof*⟩

lemma *dom-model-eq-trail-interp*:
  assumes
    $\forall A\ C.\ \mathcal{M}\ A = Some\ C \longleftrightarrow map\text{-}of\ \Gamma\ (Pos\ A) = Some\ (Some\ C)$ **and**
    $\forall Ln \in set\ \Gamma.\ \forall L.\ Ln = (L, None) \longrightarrow is\text{-}neg\ L$
  shows *dom* $\mathcal{M}$ = *trail-interp* Γ
⟨*proof*⟩

type-synonym *'f gliteral* = *'f gterm literal*
type-synonym *'f gclause* = *'f gterm clause*

locale *simulation-SCLFOL-ground-ordered-resolution* =
  renaming-apart renaming-vars
  for *renaming-vars* :: *'v set* ⇒ *'v* ⇒ *'v* +
  fixes
    less-trm :: *'f gterm* ⇒ *'f gterm* ⇒ *bool* (**infix** $\prec_t$ *50*)
  assumes
    transp-less-trm[*simp*]: *transp* $(\prec_t)$ **and**
    asymp-less-trm[*intro*]: *asymp* $(\prec_t)$ **and**
    wfP-less-trm[*intro*]: *wfP* $(\prec_t)$ **and**
    totalp-less-trm[*intro*]: *totalp* $(\prec_t)$ **and**
    finite-less-trm: $\bigwedge\beta.\ finite\ \{x.\ x \prec_t \beta\}$ **and**
    less-trm-compatible-with-gctxt[*simp*]: $\bigwedge ctxt\ t\ t'.\ t \prec_t t' \Longrightarrow ctxt\langle t\rangle_G \prec_t ctxt\langle t'\rangle_G$
and
    less-trm-if-subterm[*simp*]: $\bigwedge t\ ctxt.\ ctxt \neq \Box_G \Longrightarrow t \prec_t ctxt\langle t\rangle_G$

8 Ground ordered resolution for ground terms

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

sublocale *ord-res*: *ground-ordered-resolution-calculus* $(\prec_t)$ λ-. {#}
  ⟨*proof*⟩

sublocale *linorder-trm*: *linorder* $(\preceq_t)$ $(\prec_t)$
⟨*proof*⟩

sublocale *linorder-lit*: *linorder* $(\preceq_l)$ $(\prec_l)$
⟨*proof*⟩

sublocale *linorder-cls*: *linorder* $(\preceq_c)$ $(\prec_c)$
⟨*proof*⟩

declare *linorder-trm.is-least-in-fset-ffilterD*[*no-atp*]
declare *linorder-lit.is-least-in-fset-ffilterD*[*no-atp*]
declare *linorder-cls.is-least-in-fset-ffilterD*[*no-atp*]

end

9 Common definitions and lemmas

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

abbreviation *ord-res-Interp* **where**
ord-res-Interp N C ≡ *ord-res.interp N C* ∪ *ord-res.production N C*

definition *is-least-false-clause* **where**
is-least-false-clause N C ⟷
linorder-cls.is-least-in-fset {|*C* |∈| *N*. ¬ *ord-res-Interp* (*fset N*) *C* ⊫ *C*|} *C*

lemma *is-least-false-clause-finsert-smaller-false-clause*:
assumes
D-least: *is-least-false-clause N D* **and**
$C \prec_c D$ **and**
C-false: ¬ *ord-res-Interp* (*fset* (*finsert C N*)) *C* ⊫ *C*
shows *is-least-false-clause* (*finsert C N*) *C*
⟨*proof*⟩

lemma *is-least-false-clause-swap-swap-compatible-fsets*:
assumes {|*x* |∈| *N1*. $x \preceq_c C$|} = {|*x* |∈| *N2*. $x \preceq_c C$|}
shows *is-least-false-clause N1 C* ⟷ *is-least-false-clause N2 C*
⟨*proof*⟩

lemma *Uniq-is-least-false-clause*: $\exists_{\leq 1}$ *C*. *is-least-false-clause N C*
⟨*proof*⟩

lemma *mempty-lesseq-cls*[*simp*]: {#} $\preceq_c$ *C* **for** *C*
⟨*proof*⟩

lemma *is-least-false-clause-mempty*: {#} |∈| *N* ⟹ *is-least-false-clause N* {#}
⟨*proof*⟩

lemma *production-union-unproductive-strong*:
assumes
fin: *finite N1 finite N2* **and**

N2-unproductive: $\forall x \in N2 - N1.$ *ord-res.production* $(N1 \cup N2)$ $x = \{\}$ **and**
C-in: $C \in N1$
shows *ord-res.production* $(N1 \cup N2)$ C = *ord-res.production* $N1$ C
⟨*proof*⟩

lemma *production-union-unproductive*:
assumes
fin: *finite N1 finite N2* **and**
N2-unproductive: $\forall x \in N2.$ *ord-res.production* $(N1 \cup N2)$ $x = \{\}$ **and**
C-in: $C \in N1$
shows *ord-res.production* $(N1 \cup N2)$ C = *ord-res.production* $N1$ C
⟨*proof*⟩

lemma *interp-union-unproductive*:
assumes
fin: *finite N1 finite N2* **and**
N2-unproductive: $\forall x \in N2.$ *ord-res.production* $(N1 \cup N2)$ $x = \{\}$
shows *ord-res.interp* $(N1 \cup N2)$ = *ord-res.interp* $N1$
⟨*proof*⟩

lemma *Interp-union-unproductive*:
assumes
fin: *finite N1 finite N2* **and**
N2-unproductive: $\forall x \in N2.$ *ord-res.production* $(N1 \cup N2)$ $x = \{\}$
shows *ord-res-Interp* $(N1 \cup N2)$ C = *ord-res-Interp* $N1$ C
⟨*proof*⟩

lemma *Interp-insert-unproductive*:
assumes
fin: *finite N1* **and**
x-unproductive: *ord-res.production* (*insert x N1*) $x = \{\}$
shows *ord-res-Interp* (*insert x N1*) C = *ord-res-Interp* $N1$ C
⟨*proof*⟩

lemma *extended-partial-model-entails-iff-partial-model-entails*:
assumes
fin: *finite N finite N'* **and**
irrelevant: $\forall D \in N'. \exists E \in N. E \subset\# D \land$ *set-mset* D = *set-mset* E **and**
C-in: $C \in N$
shows *ord-res-Interp* $(N \cup N')$ $C \Vdash C \longleftrightarrow$ *ord-res-Interp* N $C \Vdash C$
⟨*proof*⟩

lemma *nex-strictly-maximal-pos-lit-if-factorizable*:
assumes *ord-res.ground-factoring* C C'
shows $\nexists L.$ *is-pos* $L \land$ *ord-res.is-strictly-maximal-lit* L C
⟨*proof*⟩

lemma *unproductive-if-nex-strictly-maximal-pos-lit*:
assumes $\nexists L.$ *is-pos* $L \land$ *ord-res.is-strictly-maximal-lit* L C

shows *ord-res.production N C* = {}
⟨*proof*⟩

lemma *ball-unproductive-if-nex-strictly-maximal-pos-lit*:
assumes $\forall$ *C* $\in$ *N′*. $\nexists$ *L*. *is-pos L* $\wedge$ *ord-res.is-strictly-maximal-lit L C*
shows $\forall$ *C* $\in$ *N′*. *ord-res.production* (*N* $\cup$ *N′*) *C* = {}
⟨*proof*⟩

lemma *is-least-false-clause-finsert-cancel*:
assumes
C-unproductive: *ord-res.production* (*fset* (*finsert C N*)) *C* = {} **and**
C-entailed-by-smaller: $\exists$ *D* |$\in$| *N*. *D* $\prec_c$ *C* $\wedge$ {*D*} $\Vdash$*e* {*C*}
shows *is-least-false-clause* (*finsert C N*) = *is-least-false-clause N*
⟨*proof*⟩

lemma *is-least-false-clause-funion-cancel-right-strong*:
assumes
$\forall$ *C* |$\in$| *N2* − *N1*. $\forall$ *U*. *ord-res.production U C* = {} **and**
$\forall$ *C* |$\in$| *N2* − *N1*. $\exists$ *D* |$\in$| *N1*. *D* $\prec_c$ *C* $\wedge$ {*D*} $\Vdash$*e* {*C*}
shows *is-least-false-clause* (*N1* |$\cup$| *N2*) = *is-least-false-clause N1*
⟨*proof*⟩

lemma *is-least-false-clause-funion-cancel-right*:
assumes
$\forall$ *C* |$\in$| *N2*. $\forall$ *U*. *ord-res.production U C* = {} **and**
$\forall$ *C* |$\in$| *N2*. $\exists$ *D* |$\in$| *N1*. *D* $\prec_c$ *C* $\wedge$ {*D*} $\Vdash$*e* {*C*}
shows *is-least-false-clause* (*N1* |$\cup$| *N2*) = *is-least-false-clause N1*
⟨*proof*⟩

definition *ex-false-clause* **where**
ex-false-clause N = ($\exists$ *C* $\in$ *N*. $\neg$ *ord-res.interp N C* $\cup$ *ord-res.production N C* $\Vdash$ *C*)

lemma *obtains-least-false-clause-if-ex-false-clause*:
assumes *ex-false-clause* (*fset N*)
obtains *C* **where** *is-least-false-clause N C*
⟨*proof*⟩

lemma *ex-false-clause-if-least-false-clause*:
assumes *is-least-false-clause N C*
shows *ex-false-clause* (*fset N*)
⟨*proof*⟩

lemma *ex-false-clause-iff*: *ex-false-clause* (*fset N*) $\longleftrightarrow$ ($\exists$ *C*. *is-least-false-clause N C*)
⟨*proof*⟩

definition *ord-res-model* **where**
ord-res-model N = ($\bigcup$ *D* $\in$ *N*. *ord-res.production N D*)

lemma *ord-res-model-eq-interp-union-production-of-greatest-clause*:
  assumes *C-greatest*: *linorder-cls.is-greatest-in-set N C*
  shows *ord-res-model N* = *ord-res.interp N C* ∪ *ord-res.production N C*
⟨*proof*⟩

lemma *ord-res-model-entails-clauses-if-nex-false-clause*:
  assumes *finite N* **and** $N \neq \{\}$ **and** ¬ *ex-false-clause N*
  shows *ord-res-model N* ⊫*s N*
  ⟨*proof*⟩

lemma *pos-lit-not-greatest-if-maximal-in-least-false-clause*:
  assumes
    C-least: *linorder-cls.is-least-in-fset* {|*C* |∈| *N*. ¬ *ord-res-Interp* (*fset N*) *C* ⊫ *C*|} *C* **and**
    C-max-lit: *ord-res.is-maximal-lit* (*Pos A*) *C*
  shows ¬ *ord-res.is-strictly-maximal-lit* (*Pos A*) *C*
⟨*proof*⟩

lemma *ex-ground-factoringI*:
  assumes
    C-max-lit: *ord-res.is-maximal-lit* (*Pos A*) *C* **and**
    C-not-max-lit: ¬ *ord-res.is-strictly-maximal-lit* (*Pos A*) *C*
  shows ∃ *C'*. *ord-res.ground-factoring C C'*
⟨*proof*⟩

lemma *true-cls-if-true-lit-in*: *L* ∈# *C* ⟹ *I* ⊫*l L* ⟹ *I* ⊫ *C*
  ⟨*proof*⟩

lemma *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit*:
  assumes
    C-least-false: *is-least-false-clause N C* **and**
    Neg-A-max: *ord-res.is-maximal-lit* (*Neg A*) *C*
  shows *fBex N* (λ*D*. $D \prec_c C$ ∧ *ord-res.is-strictly-maximal-lit* (*Pos A*) *D* ∧
    ord-res.production (*fset N*) *D* = {*A*})
⟨*proof*⟩

lemma *bex-smaller-productive-clause-if-least-false-clause-has-negative-max-lit'*:
  assumes
    C-least-false: *is-least-false-clause N C* **and**
    L-max: *ord-res.is-maximal-lit L C* **and**
    L-neg: *is-neg L*
  shows *fBex N* (λ*D*. $D \prec_c C$ ∧ *ord-res.is-strictly-maximal-lit* (− *L*) *D* ∧
    ord-res.production (*fset N*) *D* = {*atm-of L*})
  ⟨*proof*⟩

lemma *ex-ground-resolutionI*:
  assumes
    C-max-lit: *ord-res.is-maximal-lit* (*Neg A*) *C* **and**

D-lt: $D \prec_c C$ **and**
D-max-lit: *ord-res.is-strictly-maximal-lit* (*Pos A*) *D*
shows $\exists$ *CD*. *ord-res.ground-resolution C D CD*
⟨*proof*⟩

lemma
fixes $N\ N'$
assumes
fin: *finite N finite* N' **and**
irrelevant: $\forall D \in N'. \exists E \in N. E \subset\# D \wedge$ *set-mset D* = *set-mset E* **and**
C-in: $C \in N$ **and**
C-not-entailed: $\neg$ *ord-res.interp N C* $\cup$ *ord-res.production N C* $\Vdash$ *C*
shows $\neg$ *ord-res.interp* $(N \cup N')$ *C* $\cup$ *ord-res.production* $(N \cup N')$ *C* $\Vdash$ *C*
⟨*proof*⟩

lemma *trail-consistent-if-sorted-wrt-atoms*:
assumes *sorted-wrt* ($\lambda x\ y.$ *atm-of* (*fst y*) $\prec_t$ *atm-of* (*fst x*)) Γ
shows *trail-consistent* Γ
⟨*proof*⟩

lemma *mono-atms-lt*: *monotone-on* (*set* Γ) ($\lambda x\ y.$ *atm-of* (*fst y*) $\prec_t$ *atm-of* (*fst x*)) ($\lambda x\ y.\ y \leq x$)
($\lambda x.$ *atm-of K* $\preceq_t$ *atm-of* (*fst x*)) **for** *K*
⟨*proof*⟩

lemma *in-trail-atms-dropWhileI*:
assumes
sorted-wrt R Γ **and**
monotone-on (*set* Γ) *R* ($\geq$) ($\lambda x.$ *P* (*atm-of* (*fst x*))) **and**
$\neg$ *P A* **and**
A $|\in|$ *trail-atms* Γ
shows *A* $|\in|$ *trail-atms* (*dropWhile* ($\lambda Ln.$ *P* (*atm-of* (*fst Ln*))) Γ)
⟨*proof*⟩

lemma *trail-defined-lit-dropWhileI*:
assumes
sorted-wrt R Γ **and**
monotone-on (*set* Γ) *R* ($\geq$) ($\lambda x.$ *P* (*fst x*)) **and**
$\neg$ *P L* $\wedge$ $\neg$ *P* ($-$ *L*) **and**
trail-defined-lit Γ *L*
shows *trail-defined-lit* (*dropWhile* ($\lambda Ln.$ *P* (*fst Ln*)) Γ) *L*
⟨*proof*⟩

lemma *trail-defined-cls-dropWhileI*:
assumes
sorted-wrt R Γ **and**
monotone-on (*set* Γ) *R* ($\geq$) ($\lambda x.$ *P* (*fst x*)) **and**
$\forall L \in\# C.$ $\neg$ *P L* $\wedge$ $\neg$ *P* ($-$ *L*) **and**

trail-defined-cls Γ C
shows *trail-defined-cls* (*dropWhile* ($\lambda Ln.$ P (*fst* Ln)) Γ) C
⟨*proof*⟩

lemma *nbex-less-than-least-in-fset*: $\neg$ ($\exists\, w$ |∈| $X.$ $w \prec_c x$)
if *linorder-cls.is-least-in-fset* X x **for** X x
⟨*proof*⟩

lemma *clause-le-if-lt-least-greater*:
fixes N U_{er} $\mathcal{F}$ C D
defines
$\mathcal{C} \equiv$ *The-optional* (*linorder-cls.is-least-in-fset* (*ffilter* (($\prec_c$) D) N))
assumes
C-lt: $\bigwedge E.$ $\mathcal{C}$ = *Some* $E \Longrightarrow C \prec_c E$ **and**
C-in: C |∈| N
shows $C \preceq_c D$
⟨*proof*⟩

end

10 Lemmas about going between ground and first-order terms

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

lemma *ex1-gterm-of-term*:
fixes t :: *'f gterm*
shows $\exists!(t'$ :: (*'f*, *'v*) *term*). *ground* $t' \wedge t$ = *gterm-of-term* t'
⟨*proof*⟩

lemma *binj-betw-gterm-of-term*: *bij-betw gterm-of-term* {$t.$ *ground* t} *UNIV*
⟨*proof*⟩

end

11 SCL(FOL) for first-order terms

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

definition *less-B* **where**
less-B x y $\longleftrightarrow$ *ground* $x \wedge$ *ground* $y \wedge$ *gterm-of-term* $x \prec_t$ *gterm-of-term* y

sublocale *order-less-B*: *order less-B*$^{==}$ *less-B*
⟨*proof*⟩

sublocale *scl-fol*: *scl-fol-calculus renaming-vars less-B*
⟨*proof*⟩

end

lemma *trail-atms-eq-trail-atms-if-same-lits*:
assumes *list-all2* ($\lambda x\ y.\ fst\ x = fst\ y$) Γ_9 Γ_{10}
shows *trail-atms* Γ_9 = *trail-atms* Γ_{10}
⟨*proof*⟩

lemma *trail-false-lit-eq-trail-false-lit-if-same-lits*:
assumes *list-all2* ($\lambda x\ y.\ fst\ x = fst\ y$) Γ_9 Γ_{10}
shows *trail-false-lit* Γ_9 = *trail-false-lit* Γ_{10}
⟨*proof*⟩

lemma *trail-false-cls-eq-trail-false-cls-if-same-lits*:
assumes *list-all2* ($\lambda x\ y.\ fst\ x = fst\ y$) Γ_9 Γ_{10}
shows *trail-false-cls* Γ_9 = *trail-false-cls* Γ_{10}
⟨*proof*⟩

lemma *trail-defined-lit-eq-trail-defined-lit-if-same-lits*:
assumes *list-all2* ($\lambda x\ y.\ fst\ x = fst\ y$) Γ_9 Γ_{10}
shows *trail-defined-lit* Γ_9 = *trail-defined-lit* Γ_{10}
⟨*proof*⟩

lemma *trail-defined-cls-eq-trail-defined-cls-if-same-lits*:
assumes *list-all2* ($\lambda x\ y.\ fst\ x = fst\ y$) Γ_9 Γ_{10}
shows *trail-defined-cls* Γ_9 = *trail-defined-cls* Γ_{10}
⟨*proof*⟩

end
theory *ORD-RES*
imports *Background*
begin

12 ORD-RES

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

lemma *ex-false-clause* $N \longleftrightarrow \neg$ ($\forall C \in N.$ *ord-res-Interp* $N\ C \models C$)
⟨*proof*⟩

lemma $\neg$ *ex-false-clause* $N \longleftrightarrow$ ($\forall C \in N.$ *ord-res-Interp* $N\ C \models C$)
⟨*proof*⟩

definition *ord-res-final* **where**
ord-res-final $N \longleftrightarrow \{\#\}\ |\in|\ N \lor \neg$ *ex-false-clause* (*fset* N)

inductive *ord-res* **where**
factoring: $\{\#\}\ |\notin|\ N \Longrightarrow$ *ex-false-clause* (*fset* N) $\Longrightarrow$
— Maybe write $\neg$ *ord-res-final* N instead?

P |∈| $N \Longrightarrow$ *ord-res.ground-factoring* P $C \Longrightarrow$
$N' =$ *finsert* C $N \Longrightarrow$
ord-res N N' |

resolution: {#} |∉| $N \Longrightarrow$ *ex-false-clause* (*fset* N) $\Longrightarrow$
$P1$ |∈| $N \Longrightarrow$ $P2$ |∈| $N \Longrightarrow$ *ord-res.ground-resolution* $P1$ $P2$ $C \Longrightarrow$
$N' =$ *finsert* C $N \Longrightarrow$
ord-res N N'

inductive *ord-res-load* **where**
$N \neq$ {||} $\Longrightarrow$ *ord-res-load* N N

sublocale *ord-res-semantics*: *semantics* **where**
step = *ord-res* **and**
final = *ord-res-final*
⟨*proof*⟩

sublocale *ord-res-language*: *language* **where**
step = *ord-res* **and**
final = *ord-res-final* **and**
load = *ord-res-load*
⟨*proof*⟩

end

end
theory *ORD-RES-1*
imports *ORD-RES*
begin

13 ORD-RES-1 (deterministic)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-1* **where**
factoring:
is-least-false-clause N $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-pos $L \Longrightarrow$
ord-res.ground-factoring C $C' \Longrightarrow$
$N' =$ *finsert* C' $N \Longrightarrow$
ord-res-1 N N' |

resolution:
is-least-false-clause N $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-neg $L \Longrightarrow$
D |∈| $N \Longrightarrow$
$D \prec_c C \Longrightarrow$

ord-res.production (*fset N*) *D* = {*atm-of L*} $\Longrightarrow$
ord-res.ground-resolution C D CD $\Longrightarrow$
N' = *finsert CD N* $\Longrightarrow$
ord-res-1 N N'

lemma
assumes *ord-res.ground-resolution C D CD*
shows $D \prec_c C$
⟨*proof*⟩

lemma *right-unique-ord-res-1*: *right-unique ord-res-1*
⟨*proof*⟩

definition *ord-res-1-final* **where**
ord-res-1-final N $\longleftrightarrow$ *ord-res-final N*

inductive *ord-res-1-load* **where**
$N \neq \{||\}$ $\Longrightarrow$ *ord-res-1-load N N*

sublocale *ord-res-1-semantics*: *semantics* **where**
step = *ord-res-1* **and**
final = *ord-res-1-final*
⟨*proof*⟩

sublocale *ord-res-1-language*: *language* **where**
step = *ord-res-1* **and**
final = *ord-res-1-final* **and**
load = *ord-res-1-load*
⟨*proof*⟩

lemma *ex-ord-res-1-if-not-final*:
assumes ¬ *ord-res-1-final N*
shows ∃ *N'*. *ord-res-1 N N'*
⟨*proof*⟩

corollary *ord-res-1-safe*: *ord-res-1-final N* ∨ (∃ *N'*. *ord-res-1 N N'*)
⟨*proof*⟩

end

end
theory *Exhaustive-Factorization*
imports *Background*
begin

14 Function for full factorization

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

definition *efac* :: *'f gterm clause* ⇒ *'f gterm clause* **where**
efac C = (*THE* C'. *ord-res.ground-factoring*** C C' ∧ (∄ C''. *ord-res.ground-factoring* C' C''))

The function *efac* performs exhaustive factorization of its input clause.

lemma *ex1-efac-eq-factoring-chain*:
∃!C'. *efac* C = C' ∧ *ord-res.ground-factoring*** C C' ∧ (∄ C''. *ord-res.ground-factoring* C' C'')
⟨*proof*⟩

lemma *efac-eq-disj*:
efac C = C ∨ (∃!C'. *efac* C = C' ∧ *ord-res.ground-factoring*** C C' ∧ (∄ C''. *ord-res.ground-factoring* C' C''))
⟨*proof*⟩

lemma *member-mset-if-count-eq-Suc*: *count* X x = *Suc* n ⟹ x ∈# X
⟨*proof*⟩

lemmas *member-fsetE* = *mset-add*

lemma *ord-res-ground-factoring-iff*: *ord-res.ground-factoring* C C' ⟷
(∃ A. *ord-res.is-maximal-lit* (*Pos* A) C ∧ (∃ n. *count* C (*Pos* A) = *Suc* (*Suc* n) ∧ C' = C − {#*Pos* A#}))
⟨*proof*⟩

lemma *tranclp-ord-res-ground-factoring-iff*:
ord-res.ground-factoring^{++} C C' ∧ (∄ C''. *ord-res.ground-factoring* C' C'') ⟷
(∃ A. *ord-res.is-maximal-lit* (*Pos* A) C ∧ (∃ n. *count* C (*Pos* A) = *Suc* (*Suc* n) ∧
C' = C − *replicate-mset* (*Suc* n) (*Pos* A)))
⟨*proof*⟩

lemma *minus-mset-replicate-mset-eq-add-mset-filter-mset*:
assumes *count* X x = *Suc* n
shows X − *replicate-mset* n x = *add-mset* x {#y ∈# X. y ≠ x#}
⟨*proof*⟩

lemma *minus-mset-replicate-mset-eq-add-mset-add-mset-filter-mset*:
assumes *count* X x = *Suc* (*Suc* n)
shows X − *replicate-mset* n x = *add-mset* x (*add-mset* x {#y ∈# X. y ≠ x#})
⟨*proof*⟩

lemma *rtrancl-ground-factoring-iff*:
shows *ord-res.ground-factoring*** C C' ∧ (∄ C''. *ord-res.ground-factoring* C' C'') ⟷
((∄ A. *ord-res.is-maximal-lit* (*Pos* A) C ∧ *count* C (*Pos* A) ≥ *2*) ∧ C = C' ∨
(∃ A. *ord-res.is-maximal-lit* (*Pos* A) C ∧ C' = *add-mset* (*Pos* A) {#L ∈# C. L ≠ *Pos* A#}))
⟨*proof*⟩

lemma *efac-spec*: *efac* $C = C \vee$
$(\exists A.\ \textit{ord-res.is-maximal-lit}\ (\textit{Pos}\ A)\ C \wedge \textit{efac}\ C = \textit{add-mset}\ (\textit{Pos}\ A)\ \{\#L \in\#\ C.\ L \neq \textit{Pos}\ A\#\})$
⟨*proof*⟩

lemma *efac-spec-if-pos-lit-is-maximal*:
assumes *L-pos*: *is-pos* L **and** *L-max*: *ord-res.is-maximal-lit* L C
shows *efac* C = *add-mset* L $\{\#K \in\#\ C.\ K \neq L\#\}$
⟨*proof*⟩

lemma *efac-mempty*[*simp*]: *efac* $\{\#\} = \{\#\}$
⟨*proof*⟩

lemma *set-mset-efac*[*simp*]: *set-mset* (*efac* C) = *set-mset* C
⟨*proof*⟩

lemma *efac-subset*: *efac* $C \subseteq\#\ C$
⟨*proof*⟩

lemma *true-cls-efac-iff*[*simp*]:
fixes $\mathcal{I}$ :: *'f gterm set* **and** C :: *'f gclause*
shows $\mathcal{I} \models$ *efac* $C \longleftrightarrow \mathcal{I} \models C$
⟨*proof*⟩

lemma *obtains-positive-greatest-lit-if-efac-not-ident*:
assumes *efac* $C \neq C$
obtains L **where** *is-pos* L **and** *linorder-lit.is-greatest-in-mset* (*efac* C) L
⟨*proof*⟩

lemma *mempty-in-image-efac-iff*[*simp*]: $\{\#\} \in$ *efac* ‘ $N \longleftrightarrow \{\#\} \in N$
⟨*proof*⟩

lemma *greatest-literal-in-efacI*:
assumes *is-pos* L **and** *C-max-lit*: *linorder-lit.is-maximal-in-mset* C L
shows *linorder-lit.is-greatest-in-mset* (*efac* C) L
⟨*proof*⟩

lemma *linorder-lit.is-maximal-in-mset* (*efac* C) $L \longleftrightarrow$ *linorder-lit.is-maximal-in-mset* C L
⟨*proof*⟩

lemma
assumes *is-pos* L
shows *linorder-lit.is-greatest-in-mset* (*efac* C) $L \longleftrightarrow$ *linorder-lit.is-maximal-in-mset* C L
⟨*proof*⟩

lemma *factorizable-if-neq-efac*:
assumes $C \neq$ *efac* C

shows $\exists\, C'$. *ord-res.ground-factoring* C C'
⟨*proof*⟩

lemma *nex-strictly-maximal-pos-lit-if-neq-efac*:
assumes $C \neq$ *efac* C
shows $\nexists\, L$. *is-pos* $L \wedge$ *ord-res.is-strictly-maximal-lit* L C
⟨*proof*⟩

lemma *efac-properties-if-not-ident*:
assumes *efac* $C \neq C$
shows *efac* $C \prec_c C$ **and** $\{$*efac* $C\} \Vdash e\ \{C\}$
⟨*proof*⟩

end

end
theory *ORD-RES-2*
imports
ORD-RES
Exhaustive-Factorization
begin

15 ORD-RES-2 (full factorization)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-2* **where**
factoring:
is-least-false-clause $(N\ |\cup|\ U_r\ |\cup|\ U_{ef})$ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-pos $L \Longrightarrow$
$U_{ef}{}' =$ *finsert* (*efac* C) $U_{ef} \Longrightarrow$
ord-res-2 N $(U_r,\ U_{ef})$ $(U_r,\ U_{ef}{}')$ |

resolution:
is-least-false-clause $(N\ |\cup|\ U_r\ |\cup|\ U_{ef})$ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-neg $L \Longrightarrow$
$D\ |\in|\ N\ |\cup|\ U_r\ |\cup|\ U_{ef} \Longrightarrow$
$D \prec_c C \Longrightarrow$
ord-res.production (*fset* $(N\ |\cup|\ U_r\ |\cup|\ U_{ef})$) $D = \{$*atm-of* $L\} \Longrightarrow$
ord-res.ground-resolution C D $CD \Longrightarrow$
$U_r{}' =$ *finsert* CD $U_r \Longrightarrow$
ord-res-2 N $(U_r,\ U_{ef})$ $(U_r{}',\ U_{ef})$

inductive *ord-res-2-step* **where**
ord-res-2 N S $S' \Longrightarrow$ *ord-res-2-step* $(N,\ S)$ $(N,\ S')$

inductive *ord-res-2-final* **where**

ord-res-final $(N\ |\cup|\ U_r\ |\cup|\ U_{ef}) \Longrightarrow$ *ord-res-2-final* $(N, (U_r, U_{ef}))$

inductive *ord-res-2-load* **where**
$N \neq \{||\} \Longrightarrow$ *ord-res-2-load* N $(N, (\{||\}, \{||\}))$

sublocale *ord-res-2-semantics*: *semantics* **where**
step = *ord-res-2-step* **and**
final = *ord-res-2-final*
⟨*proof*⟩

sublocale *ord-res-2-language*: *language* **where**
step = *ord-res-2-step* **and**
final = *ord-res-2-final* **and**
load = *ord-res-2-load*
⟨*proof*⟩

lemma *is-least-in-fset-with-irrelevant-clauses-if-is-least-in-fset*:
assumes
irrelevant: $\forall D\ |\in|\ N'.\ \exists E\ |\in|\ N.\ E \subset\#\ D \wedge$ *set-mset* D = *set-mset* E **and**
C-least: *linorder-cls.is-least-in-fset* $\{|C\ |\in|\ N.\ \neg$ *ord-res-Interp* (*fset* N) $C \Vdash C|\}$ C
shows *linorder-cls.is-least-in-fset* $\{|C\ |\in|\ N\ |\cup|\ N'.\ \neg$ *ord-res-Interp* (*fset* $(N\ |\cup|\ N')$) $C \Vdash C|\}$ C
⟨*proof*⟩

primrec *fset-upto* :: *nat* ⇒ *nat* ⇒ *nat fset* **where**
fset-upto i 0 = (*if i = 0 then* {|*0*|} *else* {||}) |
fset-upto i (*Suc n*) = (*if i* ≤ *Suc n then finsert* (*Suc n*) (*fset-upto i n*) *else* {||})

lemma
fset-upto 0 0 = {|*0*|}
fset-upto 0 1 = {|*0*, *1*|}
fset-upto 0 2 = {|*0*, *1*, *2*|}
fset-upto 0 3 = {|*0*, *1*, *2*, *3*|}
fset-upto 1 3 = {|*1*, *2*, *3*|}
fset-upto 2 3 = {|*2*, *3*|}
fset-upto 3 3 = {|*3*|}
fset-upto 4 3 = {||}
⟨*proof*⟩

lemma $i \leq 1 + j \Longrightarrow$ *List.upto* i $(1 + j)$ = *List.upto* i j @ $[1 + j]$
⟨*proof*⟩

lemma *fset-of-append-singleton*: *fset-of-list* (*xs* @ [*x*]) = *finsert x* (*fset-of-list xs*)
⟨*proof*⟩

lemma *fset-of-list* (*List.upto* (*int i*) (*int j*)) = *int* |'| *fset-upto i j*
⟨*proof*⟩

lemma *fset-fset-upto*[*simp*]: *fset* (*fset-upto* *m* *n*) = {*m*..*n*}
⟨*proof*⟩

lemma *minus-mset-replicate-strict-subset-minus-msetI*:
assumes $m < n$ **and** $n <$ *count* *C* *L*
shows *C* − *replicate-mset* *n* *L* ⊂# *C* − *replicate-mset* *m* *L*
⟨*proof*⟩

lemma *factoring-all-is-between-efac-and-original-clause*:
fixes *z*
assumes
is-pos *L* **and** *ord-res.is-maximal-lit* *L* *C* **and** *count* *C* *L* = *Suc* (*Suc* *n*)
$m' \leq n$ **and**
z-in: *z* |∈| (λ*i*. *C* − *replicate-mset* *i* *L*) |'| *fset-upto* *0* *m'*
shows *efac* *C* ⊂# *z* **and** *z* ⊆# *C*
⟨*proof*⟩

lemma
assumes
linorder-cls.is-least-in-fset {|*x* |∈| *N1*. *P* *N1* *x*|} *x* **and**
linorder-cls.is-least-in-fset *N2* *y* **and**
∀ *z* |∈| *N2*. $z \preceq_c x$ **and**
P (*N1* |∪| *N2*) *y* **and**
∀ *z* |∈| *N1*. $z \prec_c x$ ⟶ ¬ *P* (*N1* |∪| *N2*) *z*
shows *linorder-cls.is-least-in-fset* {|*x* |∈| *N1* |∪| *N2*. *P* (*N1* |∪| *N2*) *x*|} *y*
⟨*proof*⟩

lemma *ground-factoring-preserves-efac*:
assumes *ord-res.ground-factoring* *P* *C*
shows *efac* *P* = *efac* *C*
⟨*proof*⟩

lemma *ground-factorings-preserves-efac*:
assumes *ord-res.ground-factoring*** *P* *C*
shows *efac* *P* = *efac* *C*
⟨*proof*⟩

lemma *ex-ord-res-2-if-not-final*:
assumes ¬ *ord-res-2-final* *S*
shows ∃ *S'*. *ord-res-2-step* *S* *S'*
⟨*proof*⟩

corollary *ord-res-2-step-safe*: *ord-res-2-final* *S* ∨ (∃ *S'*. *ord-res-2-step* *S* *S'*)
⟨*proof*⟩

lemma *is-least-false-clause-if-is-least-false-clause-in-union-unproductive*:
assumes
N2-unproductive: ∀ *C* |∈| *N2*. *ord-res.production* (*fset* (*N1* |∪| *N2*)) *C* = {}

and
  C-in: *C* |∈| *N1* **and**
  C-least-false: *is-least-false-clause* (*N1* |∪| *N2*) *C*
 shows *is-least-false-clause N1 C*
 ⟨*proof*⟩

lemma *ground-factoring-replicate-max-pos-lit*:
 ord-res.ground-factoring
  (C_0 + *replicate-mset* (*Suc* (*Suc n*)) (*Pos A*))
  (C_0 + *replicate-mset* (*Suc n*) (*Pos A*))
 if *ord-res.is-maximal-lit* (*Pos A*) (C_0 + *replicate-mset* (*Suc* (*Suc n*)) (*Pos A*))
 for *A* C_0 *n*
⟨*proof*⟩

lemma *ground-factorings-replicate-max-pos-lit*:
 assumes
  ord-res.is-maximal-lit (*Pos A*) (C_0 + *replicate-mset* (*Suc* (*Suc n*)) (*Pos A*))
 shows $m \leq$ *Suc n* $\Longrightarrow$ (*ord-res.ground-factoring* ^^ *m*)
  (C_0 + *replicate-mset* (*Suc* (*Suc n*)) (*Pos A*))
  (C_0 + *replicate-mset* (*Suc* (*Suc n* − *m*)) (*Pos A*))
⟨*proof*⟩

lemma *ord-res-Interp-entails-if-greatest-lit-is-pos*:
 assumes *C-in*: *C* ∈ *N* **and** *L-greatest*: *linorder-lit.is-greatest-in-mset C L* **and**
L-pos: *is-pos L*
 shows *ord-res-Interp N C* ⊫ *C*
⟨*proof*⟩

lemma *right-unique-ord-res-2*: *right-unique* (*ord-res-2 N*)
⟨*proof*⟩

lemma *right-unique-ord-res-2-step*: *right-unique ord-res-2-step*
⟨*proof*⟩

end

end
theory *Exhaustive-Resolution*
 imports *Background*
begin

16 Function for full resolution

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

definition *ground-resolution* **where**
 ground-resolution D C CD = *ord-res.ground-resolution C D CD*

lemma *Uniq-ground-resolution*: $\exists_{\leq 1} DC.$ *ground-resolution D C DC*

⟨*proof*⟩

lemma *ground-resolution-terminates*: $wfP\ (ground\text{-}resolution\ D)^{-1-1}$
⟨*proof*⟩

lemma *not-ground-resolution-mempty-left*: $\neg\ ground\text{-}resolution\ \{\#\}\ C\ x$
⟨*proof*⟩

lemma *not-ground-resolution-mempty-right*: $\neg\ ground\text{-}resolution\ C\ \{\#\}\ x$
⟨*proof*⟩

lemma *not-tranclp-ground-resolution-mempty-left*: $\neg\ (ground\text{-}resolution\ \{\#\})^{++}\ C\ x$
⟨*proof*⟩

lemma *not-tranclp-ground-resolution-mempty-right*: $\neg\ (ground\text{-}resolution\ C)^{++}\ \{\#\}\ x$
⟨*proof*⟩

lemma *left-premise-lt-right-premise-if-ground-resolution*:
$ground\text{-}resolution\ D\ C\ DC \Longrightarrow D \prec_c C$
⟨*proof*⟩

lemma *left-premise-lt-right-premise-if-tranclp-ground-resolution*:
$(ground\text{-}resolution\ D)^{++}\ C\ DC \Longrightarrow D \prec_c C$
⟨*proof*⟩

lemma *resolvent-lt-right-premise-if-ground-resolution*:
$ground\text{-}resolution\ D\ C\ DC \Longrightarrow DC \prec_c C$
⟨*proof*⟩

lemma *resolvent-lt-right-premise-if-tranclp-ground-resolution*:
$(ground\text{-}resolution\ D)^{++}\ C\ DC \Longrightarrow DC \prec_c C$
⟨*proof*⟩

Exhaustive resolution

definition *eres* **where**
$eres\ D\ C = (THE\ DC.\ full\text{-}run\ (ground\text{-}resolution\ D)\ C\ DC)$

The function *eres* performs exhaustive resolution between its two input clauses. The first clause is repeatedly used, while the second clause is only use to start the resolution chain.

lemma *eres-ident-iff*: $eres\ D\ C = C \longleftrightarrow (\nexists DC.\ ground\text{-}resolution\ D\ C\ DC)$
⟨*proof*⟩

lemma
assumes
step1: $ground\text{-}resolution\ D\ C\ DC$ **and**
stuck: $\nexists DDC.\ ground\text{-}resolution\ D\ DC\ DDC$

shows *eres D C* = *DC*
⟨*proof*⟩

lemma
assumes
step1: *ground-resolution D C DC* **and**
step2: *ground-resolution D DC DDC* **and**
stuck: $\nexists$ *DDDC*. *ground-resolution D DDC DDDC*
shows *eres D C* = *DDC*
⟨*proof*⟩

lemma
assumes
step1: *ground-resolution D C DC* **and**
step2: *ground-resolution D DC DDC* **and**
step3: *ground-resolution D DDC DDDC* **and**
stuck: $\nexists$ *DDDDC*. *ground-resolution D DDDC DDDDC*
shows *eres D C* = *DDDC*
⟨*proof*⟩

lemma *eres-mempty-left*[*simp*]: *eres* {#} *C* = *C*
⟨*proof*⟩

lemma *eres-mempty-right*[*simp*]: *eres C* {#} = {#}
⟨*proof*⟩

lemma *ex1-eres-eq-full-run-ground-resolution*: ∃!*DC*. *eres D C* = *DC* ∧ *full-run* (*ground-resolution D*) *C DC*
⟨*proof*⟩

lemma *eres-le*: *eres D C* $\preceq_c$ *C*
⟨*proof*⟩

lemma *clause-lt-clause-if-max-lit-comp*:
assumes *E-max-lit*: *linorder-lit.is-maximal-in-mset E L* **and** *L-neg*: *is-neg L* **and**
D-max-lit: *linorder-lit.is-maximal-in-mset D* (− *L*)
shows *D* $\prec_c$ *E*
⟨*proof*⟩

lemma *eres-lt-if*:
assumes *E-max-lit*: *ord-res.is-maximal-lit L E* **and** *L-neg*: *is-neg L* **and**
D-max-lit: *linorder-lit.is-greatest-in-mset D* (− *L*)
shows *eres D E* $\prec_c$ *E*
⟨*proof*⟩

lemma *eres-eq-after-ground-resolution*:
assumes *ground-resolution D C DC*
shows *eres D C* = *eres D DC*
⟨*proof*⟩

lemma *eres-eq-after-rtranclp-ground-resolution*:
assumes (*ground-resolution* D)** C DC
shows *eres* D C = *eres* D DC
⟨*proof*⟩

lemma *eres-eq-after-tranclp-ground-resolution*:
assumes (*ground-resolution* D)$^{++}$ C DC
shows *eres* D C = *eres* D DC
⟨*proof*⟩

lemma *resolvable-if-neq-eres*:
assumes $C \neq$ *eres* D C
shows $\exists!DC$. *ground-resolution* D C DC
⟨*proof*⟩

lemma *nex-maximal-pos-lit-if-resolvable*:
assumes *ground-resolution* D C DC
shows $\nexists L$. *is-pos* L ∧ *ord-res.is-maximal-lit* L C
⟨*proof*⟩

corollary *nex-strictly-maximal-pos-lit-if-resolvable*:
assumes *ground-resolution* D C DC
shows $\nexists L$. *is-pos* L ∧ *ord-res.is-strictly-maximal-lit* L C
⟨*proof*⟩

corollary *nex-maximal-pos-lit-if-neq-eres*:
assumes $C \neq$ *eres* D C
shows $\nexists L$. *is-pos* L ∧ *ord-res.is-maximal-lit* L C
⟨*proof*⟩

corollary *nex-strictly-maximal-pos-lit-if-neq-eres*:
assumes $C \neq$ *eres* D C
shows $\nexists L$. *is-pos* L ∧ *ord-res.is-strictly-maximal-lit* L C
⟨*proof*⟩

lemma *ground-resolutionD*:
assumes *ground-resolution* D C DC
shows $\exists\, m\ A\ D'\ C'$.
linorder-lit.is-greatest-in-mset D (*Pos* A) ∧
linorder-lit.is-maximal-in-mset C (*Neg* A) ∧
D = *add-mset* (*Pos* A) D' ∧
C = *replicate-mset* (*Suc* m) (*Neg* A) + C' ∧ *Neg* A ∉# C' ∧
DC = D' + *replicate-mset* m (*Neg* A) + C'
⟨*proof*⟩

lemma *relpowp-ground-resolutionD*:
assumes $n \neq 0$ **and** (*ground-resolution* D ^^ n) C DnC
shows $\exists\, m\ A\ D'\ C'$. *Suc* $m \geq n$ ∧

linorder-lit.is-greatest-in-mset D (*Pos* A) $\wedge$
linorder-lit.is-maximal-in-mset C (*Neg* A) $\wedge$
D = *add-mset* (*Pos* A) D' $\wedge$
C = *replicate-mset* (*Suc* m) (*Neg* A) + C' $\wedge$ *Neg* $A \notin\#$ C' $\wedge$
DnC = *repeat-mset* n D' + *replicate-mset* (*Suc* $m - n$) (*Neg* A) + C'
⟨*proof*⟩

lemma *tranclp-ground-resolutionD*:
assumes (*ground-resolution* D)$^{++}$ C *DnC*
shows $\exists\, n\ m\ A\ D'\ C'$. *Suc* $m \geq$ *Suc* n $\wedge$
linorder-lit.is-greatest-in-mset D (*Pos* A) $\wedge$
linorder-lit.is-maximal-in-mset C (*Neg* A) $\wedge$
D = *add-mset* (*Pos* A) D' $\wedge$
C = *replicate-mset* (*Suc* m) (*Neg* A) + C' $\wedge$ *Neg* $A \notin\#$ C' $\wedge$
DnC = *repeat-mset* (*Suc* n) D' + *replicate-mset* (*Suc* m − *Suc* n) (*Neg* A) +
C'
⟨*proof*⟩

lemma *eres-not-identD*:
assumes *eres* D $C \neq C$
shows $\exists\, m\ A\ D'\ C'$.
linorder-lit.is-greatest-in-mset D (*Pos* A) $\wedge$
linorder-lit.is-maximal-in-mset C (*Neg* A) $\wedge$
D = *add-mset* (*Pos* A) D' $\wedge$
C = *replicate-mset* (*Suc* m) (*Neg* A) + C' $\wedge$ *Neg* $A \notin\#$ C' $\wedge$
eres D C = *repeat-mset* (*Suc* m) D' + C'
⟨*proof*⟩

lemma *lit-in-one-of-resolvents-if-in-eres*:
fixes L :: *'f gterm literal* **and** C D :: *'f gclause*
assumes $L \in\#$ *eres* C D
shows $L \in\# C \vee L \in\# D$
⟨*proof*⟩

lemma *strong-lit-in-one-of-resolvents-if-in-eres*:
fixes L :: *'f gterm literal* **and** C D :: *'f gclause*
assumes
D-max-lit: *linorder-lit.is-maximal-in-mset* D L **and**
K-in: $K \in\#$ *eres* C D
shows $K \in\# C \wedge K \neq -L \vee K \in\# D$
⟨*proof*⟩

lemma *stronger-lit-in-one-of-resolvents-if-in-eres*:
fixes K L :: *'f gterm literal* **and** C D :: *'f gclause*
assumes *eres* C $D \neq D$ **and**
D-max-lit: *linorder-lit.is-maximal-in-mset* D L **and**
K-in-eres: $K \in\#$ *eres* C D
shows $K \in\# C \wedge K \neq -\ L \vee K \in\# D \wedge K \neq L$

⟨*proof*⟩

lemma *lit-in-eres-lt-greatest-lit-in-grestest-resolvant*:
  fixes $K\ L$ :: *'f gterm literal* **and** $C\ D$ :: *'f gclause*
  assumes *eres* $C\ D \neq D$ **and**
    D-max-lit: *linorder-lit.is-maximal-in-mset* $D\ L$ **and**
    $-\ L \notin\#\ D$ **and**
    K-in-eres: $K \in\#$ *eres* $C\ D$
  shows *atm-of* $K \prec_t$ *atm-of* L
⟨*proof*⟩

lemma *eres-entails-resolvent*:
  fixes $C\ D$ :: *'f gterm clause*
  assumes (*ground-resolution* $C)^{++}\ D_0\ D$
  shows $\{$*eres* $C\ D_0\} \models e\ \{D\}$
  ⟨*proof*⟩

lemma *clause-true-if-resolved-true*:
  assumes
    (*ground-resolution* $D)^{++}$ C *DC* **and**
    D-productive: *ord-res.production* $N\ D \neq \{\}$ **and**
    C-true: *ord-res-Interp* N *DC* $\models$ *DC*
  shows *ord-res-Interp* $N\ C \models C$
⟨*proof*⟩

lemma *clause-true-if-eres-true*:
  assumes
    (*ground-resolution* *D1*$)^{++}$ *D2* C **and**
    $C \neq$ *eres D1* C **and**
    eres-C-true: *ord-res-Interp* N (*eres D1* C) $\models$ *eres D1* C
  shows *ord-res-Interp* $N\ C \models C$
⟨*proof*⟩

end

end
theory *ORD-RES-3*
  imports
    ORD-RES
    Exhaustive-Factorization
    Exhaustive-Resolution
begin

17 ORD-RES-3 (full resolve)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-3* **where**
factoring:
$\textit{is-least-false-clause}\ (N\ |\cup|\ U_{er}\ |\cup|\ U_{ef})\ C \Longrightarrow$
$\textit{linorder-lit.is-maximal-in-mset}\ C\ L \Longrightarrow$
$\textit{is-pos}\ L \Longrightarrow$
$U_{ef}{}' = \textit{finsert}\ (\textit{efac}\ C)\ U_{ef} \Longrightarrow$
$\textit{ord-res-3}\ N\ (U_{er},\ U_{ef})\ (U_{er},\ U_{ef}{}')\ |$

resolution:
$\textit{is-least-false-clause}\ (N\ |\cup|\ U_{er}\ |\cup|\ U_{ef})\ C \Longrightarrow$
$\textit{linorder-lit.is-maximal-in-mset}\ C\ L \Longrightarrow$
$\textit{is-neg}\ L \Longrightarrow$
$D\ |\in|\ N\ |\cup|\ U_{er}\ |\cup|\ U_{ef} \Longrightarrow$
$D \prec_c C \Longrightarrow$
$\textit{ord-res.production}\ (\textit{fset}\ (N\ |\cup|\ U_{er}\ |\cup|\ U_{ef}))\ D = \{\textit{atm-of}\ L\} \Longrightarrow$
$U_{er}{}' = \textit{finsert}\ (\textit{eres}\ D\ C)\ U_{er} \Longrightarrow$
$\textit{ord-res-3}\ N\ (U_{er},\ U_{ef})\ (U_{er}{}',\ U_{ef})$

inductive *ord-res-3-step* **where**
$\textit{ord-res-3}\ N\ s\ s' \Longrightarrow \textit{ord-res-3-step}\ (N,\ s)\ (N,\ s')$

inductive *ord-res-3-final* **where**
$\textit{ord-res-final}\ (N\ |\cup|\ U_{rr}\ |\cup|\ U_{ef}) \Longrightarrow \textit{ord-res-3-final}\ (N, (U_{rr},\ U_{ef}))$

inductive *ord-res-3-load* **where**
$N \neq \{||\} \Longrightarrow \textit{ord-res-3-load}\ N\ (N, (\{||\}, \{||\}))$

sublocale *ord-res-3-semantics*: *semantics* **where**
step = *ord-res-3-step* **and**
final = *ord-res-3-final*
⟨*proof*⟩

sublocale *ord-res-3-language*: *language* **where**
step = *ord-res-3-step* **and**
final = *ord-res-3-final* **and**
load = *ord-res-3-load*
⟨*proof*⟩

lemma *is-least-false-clause-conv-if-partial-resolution-invariant*:
assumes $\forall\, C\ |\in|\ U_{pr}.\ \exists\, D1\ |\in|\ N\ |\cup|\ U_{er}\ |\cup|\ U_{ef}.\ \exists\, D2\ |\in|\ N\ |\cup|\ U_{er}\ |\cup|\ U_{ef}.$
$(\textit{ground-resolution}\ D1)^{++}\ D2\ C \land C \neq \textit{eres}\ D1\ D2 \land \textit{eres}\ D1\ D2\ |\in|\ U_{er}$
shows $\textit{is-least-false-clause}\ (N\ |\cup|\ U_{pr}\ |\cup|\ U_{er}\ |\cup|\ U_{ef}) = \textit{is-least-false-clause}$
$(N\ |\cup|\ U_{er}\ |\cup|\ U_{ef})$
⟨*proof*⟩

lemma *right-unique-ord-res-3*: *right-unique* (*ord-res-3 N*)
⟨*proof*⟩

lemma *right-unique-ord-res-3-step*: *right-unique ord-res-3-step*

⟨*proof*⟩

lemma *ex-ord-res-3-if-not-final*:
  assumes $\neg$ *ord-res-3-final S*
  shows $\exists S'$. *ord-res-3-step S S'*
⟨*proof*⟩

corollary *ord-res-3-step-safe*: *ord-res-3-final S* $\vee$ ($\exists S'$. *ord-res-3-step S S'*)
  ⟨*proof*⟩

end

end
theory *Implicit-Exhaustive-Factorization*
  imports
    Exhaustive-Factorization
    Exhaustive-Resolution
begin

18 Function for implicit full factorization

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

definition *iefac* **where**
  iefac $\mathcal{F}$ C = (*if* C |∈| $\mathcal{F}$ *then efac* C *else* C)

lemma *iefac-mempty*[*simp*]:
  fixes $\mathcal{F}$ :: *'f gclause fset*
  shows *iefac* $\mathcal{F}$ {#} = {#}
  ⟨*proof*⟩

lemma *fset-mset-iefac*[*simp*]:
  fixes $\mathcal{F}$ :: *'f gclause fset* **and** C :: *'f gclause*
  shows *fset-mset* (*iefac* $\mathcal{F}$ C) = *fset-mset* C
⟨*proof*⟩

lemma *atms-of-cls-iefac*[*simp*]:
  fixes $\mathcal{F}$ :: *'f gclause fset* **and** C :: *'f gclause*
  shows *atms-of-cls* (*iefac* $\mathcal{F}$ C) = *atms-of-cls* C
  ⟨*proof*⟩

lemma *iefac-le*:
  fixes $\mathcal{F}$ :: *'f gclause fset* **and** C :: *'f gclause*
  shows *iefac* $\mathcal{F}$ $C \preceq_c C$
  ⟨*proof*⟩

lemma *true-cls-iefac-iff*[*simp*]:
  fixes $\mathcal{I}$ :: *'f gterm set* **and** $\mathcal{F}$ :: *'f gclause fset* **and** C :: *'f gclause*
  shows $\mathcal{I} \models$ *iefac* $\mathcal{F}$ $C \longleftrightarrow \mathcal{I} \models C$

⟨*proof*⟩

lemma *funion-funion-eq-funion-funion-fimage-iefac-if*:
assumes U_{ef}-*eq*: $U_{ef} = iefac\ \mathcal{F}\ |`|\ \{|C\ |\in|\ N\ |\cup|\ U_{er}.\ iefac\ \mathcal{F}\ C \neq C|\}$
shows $N\ |\cup|\ U_{er}\ |\cup|\ U_{ef} = N\ |\cup|\ U_{er}\ |\cup|\ (iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}))$
⟨*proof*⟩

lemma *clauses-for-iefac-are-unproductive*:
$\forall\, C\ |\in|\ N\ |-|\ iefac\ \mathcal{F}\ |`|\ N.\ \forall\, U.\ ord\text{-}res.production\ U\ C = \{\}$
⟨*proof*⟩

lemma *clauses-for-iefac-have-smaller-entailing-clause*:
$\forall\, C\ |\in|\ N\ |-|\ iefac\ \mathcal{F}\ |`|\ N.\ \exists\, D\ |\in|\ iefac\ \mathcal{F}\ |`|\ N.\ D \prec_c C \wedge \{D\} \models e\ \{C\}$
⟨*proof*⟩

lemma *is-least-false-clause-with-iefac-conv*:
is-least-false-clause $(N\ |\cup|\ U_{er}\ |\cup|\ iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})) =$
is-least-false-clause $(iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}))$
⟨*proof*⟩

lemma *MAGIC4*:
fixes $N\ \mathcal{F}\ \mathcal{F}'\ U_{er}\ U_{er}'$
defines
$N1 \equiv iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})$ **and**
$N2 \equiv iefac\ \mathcal{F}'\ |`|\ (N\ |\cup|\ U_{er}')$
assumes
subsets-agree: $\{|x\ |\in|\ N1.\ x \prec_c C|\} = \{|x\ |\in|\ N2.\ x \prec_c C|\}$ **and**
is-least-false-clause $N1\ D$ **and**
is-least-false-clause $N2\ E$ **and**
$C \prec_c D$
shows $C \preceq_c E$
⟨*proof*⟩

lemma *atms-of-clss-fimage-iefac*[*simp*]:
atms-of-clss $(iefac\ \mathcal{F}\ |`|\ N) =$ *atms-of-clss* N
⟨*proof*⟩

lemma *atm-of-in-atms-of-clssI*:
assumes *L-in*: $L \in\#\ C$ **and** *C-in*: $C\ |\in|\ iefac\ \mathcal{F}\ |`|\ N$
shows *atm-of* $L\ |\in|$ *atms-of-clss* N
⟨*proof*⟩

lemma *clause-almost-almost-definedI*:
fixes $\Gamma\ D\ K$
assumes
D-in: $D\ |\in|\ iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})$ **and**
D-max-lit: *ord-res.is-maximal-lit* $K\ D$ **and**

no-undef-atm: $\neg$ ($\exists$ $A|\in|$*atms-of-clss* (N $|\cup|$ U_{er}). $A \prec_t$ *atm-of* K $\wedge$ A $|\notin|$ *trail-atms* Γ)
shows *trail-defined-cls* Γ {#$L \in$# D. $L \neq K \wedge L \neq - K$#}
⟨*proof*⟩

lemma *clause-almost-definedI*:
fixes Γ D K
assumes
D-in: D $|\in|$ *iefac* $\mathcal{F}$ |'| (N $|\cup|$ U_{er}) **and**
D-max-lit: *ord-res.is-maximal-lit* K D **and**
no-undef-atm: $\neg$ ($\exists$ $A|\in|$*atms-of-clss* (N $|\cup|$ U_{er}). $A \prec_t$ *atm-of* K $\wedge$ A $|\notin|$ *trail-atms* Γ) **and**
K-defined: *trail-defined-lit* Γ K
shows *trail-defined-cls* Γ {#$Ka \in$# D. $Ka \neq K$#}
⟨*proof*⟩

lemma *eres-not-in-known-clauses-if-trail-false-cls*:
fixes
$\mathcal{F}$ :: *'f gclause fset* **and**
Γ :: (*'f gliteral* $\times$ *'f gclause option*) *list*
assumes
Γ*-consistent*: *trail-consistent* Γ **and**
clauses-lt-E-true: $\forall$ C $|\in|$ *iefac* $\mathcal{F}$ |'| (N $|\cup|$ U_{er}). $C \prec_c E \longrightarrow$ *trail-true-cls* Γ C **and**
eres D $E \prec_c E$ **and**
trail-false-cls Γ (*eres* D E)
shows *eres* D E $|\notin|$ N $|\cup|$ U_{er}
⟨*proof*⟩

lemma *no-undefined-atom-le-max-lit-of-false-clause*:
assumes
Γ*-lower-set*: *linorder-trm.is-lower-fset* (*trail-atms* Γ) (*atms-of-clss* (N $|\cup|$ U_{er})) **and**
D-in: D $|\in|$ *iefac* $\mathcal{F}$ |'| (N $|\cup|$ U_{er}) **and**
D-false: *trail-false-cls* Γ D **and**
D-max-lit: *linorder-lit.is-maximal-in-mset* D L
shows $\neg$ ($\exists$ $A|\in|$*atms-of-clss* (N $|\cup|$ U_{er}). $A \preceq_t$ *atm-of* L $\wedge$ A $|\notin|$ *trail-atms* Γ)
⟨*proof*⟩

lemma *trail-defined-if-no-undef-atom-le-max-lit*:
assumes
C-in: C $|\in|$ *iefac* $\mathcal{F}$ |'| (N $|\cup|$ U_{er}) **and**
C-max-lit: *linorder-lit.is-maximal-in-mset* C K **and**
no-undef-atom-le-K:
$\neg$ ($\exists$ $A|\in|$*atms-of-clss* (N $|\cup|$ U_{er}). $A \preceq_t$ *atm-of* K $\wedge$ A $|\notin|$ *trail-atms* Γ)
shows *trail-defined-cls* Γ C
⟨*proof*⟩

lemma *no-undef-atom-le-max-lit-if-lt-false-clause*:

assumes
Γ-*lower-set*: *linorder-trm.is-lower-fset* (*trail-atms* Γ) (*atms-of-clss* (*N* |∪| U_{er}))
and
D-in: *D* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}) **and**
D-false: *trail-false-cls* Γ *D* **and**
D-max-lit: *linorder-lit.is-maximal-in-mset* *D* *L* **and**
C-in: *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}) **and**
C-max-lit: *linorder-lit.is-maximal-in-mset* *C* *K* **and**
C-lt: $C \prec_c D$
shows ¬ (∃ *A*|∈|*atms-of-clss* (*N* |∪| U_{er}). $A \preceq_t$ *atm-of* *K* ∧ *A* |∉| *trail-atms* Γ)
⟨*proof*⟩

lemma *bex-trail-false-cls-simp*:
fixes $\mathcal{F}$ *N* Γ
shows *fBex* (*iefac* $\mathcal{F}$ |'| *N*) (*trail-false-cls* Γ) ⟷ *fBex* *N* (*trail-false-cls* Γ)
⟨*proof*⟩

end

end
theory *ORD-RES-4*
imports
ORD-RES
Implicit-Exhaustive-Factorization
Exhaustive-Resolution
begin

19 ORD-RES-4 (implicit factorization)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-4* **where**
factoring:
NN = *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}) ⟹
is-least-false-clause *NN* *C* ⟹
linorder-lit.is-maximal-in-mset *C* *L* ⟹
is-pos *L* ⟹
$\mathcal{F}'$ = *finsert* *C* $\mathcal{F}$ ⟹
ord-res-4 *N* (U_{er}, $\mathcal{F}$) (U_{er}, $\mathcal{F}'$) |

resolution:
NN = *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}) ⟹
is-least-false-clause *NN* *C* ⟹
linorder-lit.is-maximal-in-mset *C* *L* ⟹
is-neg *L* ⟹
D |∈| *NN* ⟹
$D \prec_c C$ ⟹
ord-res.production (*fset* *NN*) *D* = {*atm-of* *L*} ⟹
U_{er}' = *finsert* (*eres* *D* *C*) U_{er} ⟹

ord-res-4 N $(U_{er}, \mathcal{F})$ $(U_{er}', \mathcal{F})$

inductive *ord-res-4-step* **where**
ord-res-4 $N\ s\ s' \Longrightarrow$ *ord-res-4-step* (N, s) (N, s')

inductive *ord-res-4-final* **where**
ord-res-final (*iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er})) \Longrightarrow$ *ord-res-4-final* $(N, U_{er}, \mathcal{F})$

sublocale *ord-res-4-semantics*: *semantics* **where**
step = *ord-res-4-step* **and**
final = *ord-res-4-final*
⟨*proof*⟩

lemma *right-unique-ord-res-4*: *right-unique* (*ord-res-4* N)
⟨*proof*⟩

lemma *right-unique-ord-res-4-step*: *right-unique ord-res-4-step*
⟨*proof*⟩

lemma *ex-ord-res-4-if-not-final*:
assumes ¬ *ord-res-4-final* S
shows $\exists S'$. *ord-res-4-step* $S\ S'$
⟨*proof*⟩

corollary *ord-res-4-step-safe*: *ord-res-4-final* S ∨ ($\exists S'$. *ord-res-4-step* $S\ S'$)
⟨*proof*⟩

end

end
theory *ORD-RES-5*
imports
Background
Implicit-Exhaustive-Factorization
Exhaustive-Resolution
begin

20 ORD-RES-5 (explicit model construction)

type-synonym *'f ord-res-5* = *'f gclause fset* × *'f gclause fset* × *'f gclause fset* ×
(*'f gterm* ⇒ *'f gclause option*) × *'f gclause option*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-5* **where**
skip:
(*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$
$\mathcal{C}'$ = *The-optional* (*linorder-cls.is-least-in-fset* {|D |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er})$.
$C \prec_c D$|}) $\Longrightarrow$

ord-res-5 N $(U_{er}, \mathcal{F}, \mathcal{M}, \textit{Some } C)$ $(U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}')$ |

production:
$\neg$ (*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-pos $L \Longrightarrow$
linorder-lit.is-greatest-in-mset C $L \Longrightarrow$
$\mathcal{M}' = \mathcal{M}(\textit{atm-of } L := \textit{Some } C) \Longrightarrow$
$\mathcal{C}'$ = *The-optional* (*linorder-cls.is-least-in-fset* $\{|D$ $|\in|$ *iefac* $\mathcal{F}$ |'| $(N$ $|\cup|$ $U_{er})$. $C \prec_c D|\})$ $\Longrightarrow$
ord-res-5 N $(U_{er}, \mathcal{F}, \mathcal{M}, \textit{Some } C)$ $(U_{er}, \mathcal{F}, \mathcal{M}', \mathcal{C}')$ |

factoring:
$\neg$ (*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-pos $L \Longrightarrow$
$\neg$ *linorder-lit.is-greatest-in-mset* C $L \Longrightarrow$
$\mathcal{F}'$ = *finsert* C $\mathcal{F} \Longrightarrow$
$\mathcal{M}' = (\lambda\text{-. } \textit{None}) \Longrightarrow$
$\mathcal{C}'$ = *The-optional* (*linorder-cls.is-least-in-fset* (*iefac* $\mathcal{F}'$ |'| $(N$ $|\cup|$ $U_{er})))$ $\Longrightarrow$
ord-res-5 N $(U_{er}, \mathcal{F}, \mathcal{M}, \textit{Some } C)$ $(U_{er}, \mathcal{F}', \mathcal{M}', \mathcal{C}')$ |

resolution:
$\neg$ (*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-neg $L \Longrightarrow$
$\mathcal{M}$ (*atm-of* L) = *Some* $D \Longrightarrow$
U_{er}' = *finsert* (*eres* D C) $U_{er} \Longrightarrow$
$\mathcal{M}' = (\lambda\text{-. } \textit{None}) \Longrightarrow$
$\mathcal{C}'$ = *The-optional* (*linorder-cls.is-least-in-fset* (*iefac* $\mathcal{F}$ |'| $(N$ $|\cup|$ $U_{er}')))$ $\Longrightarrow$
ord-res-5 N $(U_{er}, \mathcal{F}, \mathcal{M}, \textit{Some } C)$ $(U_{er}', \mathcal{F}, \mathcal{M}', \mathcal{C}')$

inductive *ord-res-5-step* :: *'f ord-res-5* $\Rightarrow$ *'f ord-res-5* $\Rightarrow$ *bool* **where**
ord-res-5 N s $s' \Longrightarrow$ *ord-res-5-step* (N, s) (N, s')

lemma *tranclp-ord-res-5-step-if-tranclp-ord-res-5*:
(*ord-res-5* $N)^{++}$ s $s' \Longrightarrow$ *ord-res-5-step*$^{++}$ (N, s) (N, s')
⟨*proof*⟩

inductive *ord-res-5-final* :: *'f ord-res-5* $\Rightarrow$ *bool* **where**
model-found:
ord-res-5-final $(N, U_{er}, \mathcal{F}, \mathcal{M}, \textit{None})$ |

contradiction-found:
ord-res-5-final $(N, U_{er}, \mathcal{F}, \mathcal{M}, \textit{Some } \{\#\})$

sublocale *ord-res-5-semantics*: *semantics* **where**
step = *ord-res-5-step* **and**
final = *ord-res-5-final*

⟨*proof*⟩

lemma *right-unique-ord-res-5*: *right-unique* (*ord-res-5 N*)
⟨*proof*⟩

lemma *right-unique-ord-res-5-step*: *right-unique ord-res-5-step*
⟨*proof*⟩

definition *next-clause-in-factorized-clause* **where**
next-clause-in-factorized-clause N s $\longleftrightarrow$
($\forall$ U_{er} $\mathcal{F}$ $\mathcal{M}$ C. s = (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some C*) $\longrightarrow$ C |∈| *iefac* $\mathcal{F}$ |‘| (N |∪| U_{er}))

lemma *next-clause-in-factorized-clause*:
assumes *step*: *ord-res-5 N s s′*
shows *next-clause-in-factorized-clause N s′*
⟨*proof*⟩

definition *implicitly-factorized-clauses-subset* **where**
implicitly-factorized-clauses-subset N s $\longleftrightarrow$
($\forall$ U_{er} $\mathcal{F}$ $\mathcal{M}$ $\mathcal{C}$. s = (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) $\longrightarrow$ $\mathcal{F}$ |⊆| N |∪| U_{er})

lemma *ord-res-5-preserves-implicitly-factorized-clauses-subset*:
assumes
step: *ord-res-5 N s s′* **and**
invars:
implicitly-factorized-clauses-subset N s **and**
next-clause-in-factorized-clause N s
shows *implicitly-factorized-clauses-subset N s′*
⟨*proof*⟩

lemma *interp-eq-Interp-if-least-greater*:
assumes
C-in: C |∈| *NN* **and**
D-least-gt-C: *linorder-cls.is-least-in-fset* (*ffilter* (($\prec_c$) C) *NN*) D
shows *ord-res.interp* (*fset NN*) D = *ord-res.interp* (*fset NN*) C ∪ *ord-res.production* (*fset NN*) C
⟨*proof*⟩

lemma *interp-eq-empty-if-least-in-set*:
assumes *linorder-cls.is-least-in-set N C*
shows *ord-res.interp N C* = {}
⟨*proof*⟩

definition *model-eq-interp-upto-next-clause* **where**
model-eq-interp-upto-next-clause N s $\longleftrightarrow$
($\forall$ U_{er} $\mathcal{F}$ $\mathcal{M}$ C. s = (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some C*) $\longrightarrow$
dom $\mathcal{M}$ = *ord-res.interp* (*fset* (*iefac* $\mathcal{F}$ |‘| (N |∪| U_{er}))) C)

lemma *model-eq-interp-upto-next-clause*:

assumes *step*: *ord-res-5 N s s'* **and**
invars:
model-eq-interp-upto-next-clause N s
next-clause-in-factorized-clause N s
shows *model-eq-interp-upto-next-clause N s'*
⟨*proof*⟩

definition *all-smaller-clauses-true-wrt-respective-Interp* **where**
all-smaller-clauses-true-wrt-respective-Interp N s $\longleftrightarrow$
$(\forall U_{er}\ \mathcal{F}\ \mathcal{M}\ \mathcal{C}.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \longrightarrow$
$(\forall C\ |\in|\ \mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ (\forall D.\ \mathcal{C} = \mathit{Some}\ D \longrightarrow C \prec_c D) \longrightarrow$
$\mathit{ord\text{-}res\text{-}Interp}\ (\mathit{fset}\ (\mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})))\ C \Vdash C))$

lemma *all-smaller-clauses-true-wrt-respective-Interp*:
assumes *step*: *ord-res-5 N s s'* **and**
invars:
all-smaller-clauses-true-wrt-respective-Interp N s
model-eq-interp-upto-next-clause N s
next-clause-in-factorized-clause N s
shows *all-smaller-clauses-true-wrt-respective-Interp N s'*
⟨*proof*⟩

lemma *all-smaller-clauses-true-wrt-model*:
assumes
invars:
all-smaller-clauses-true-wrt-respective-Interp N s
model-eq-interp-upto-next-clause N s
shows $\forall U_{er}\ \mathcal{F}\ \mathcal{M}\ D.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathit{Some}\ D) \longrightarrow$
$(\forall C\ |\in|\ \mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ C \prec_c D \longrightarrow \mathit{dom}\ \mathcal{M} \Vdash C)$
⟨*proof*⟩

definition *model-eq-sublocale* **where**
model-eq-sublocale N s $\longleftrightarrow$
$(\forall U_{er}\ \mathcal{F}\ \mathcal{M}.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathit{None}) \longrightarrow$
$(\mathit{let}\ \mathit{NN} = \mathit{fset}\ (\mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}))\ \mathit{in}\ \mathit{dom}\ \mathcal{M} = \bigcup\ (\mathit{ord\text{-}res.production}$
$\mathit{NN}\ `\ \mathit{NN})))$

lemma *all-smaller-clauses-true-wrt-model-strong*:
assumes
invars:
all-smaller-clauses-true-wrt-respective-Interp N s
model-eq-interp-upto-next-clause N s
model-eq-sublocale N s
shows $\forall U_{er}\ \mathcal{F}\ \mathcal{M}\ \mathcal{C}.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \longrightarrow$
$(\forall C\ |\in|\ \mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ (\forall D.\ \mathcal{C} = \mathit{Some}\ D \longrightarrow C \prec_c D) \longrightarrow \mathit{dom}\ \mathcal{M}$
$\Vdash C)$
⟨*proof*⟩

lemma *next-clause-lt-least-false-clause*:

assumes
invars:
all-smaller-clauses-true-wrt-respective-Interp N s
model-eq-interp-upto-next-clause N s
shows $\forall\, U_{er}\ \mathcal{F}\ \mathcal{M}\ C.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathit{Some}\ C) \longrightarrow$
$(\forall\, D.\ \mathit{is\text{-}least\text{-}false\text{-}clause}\ (\mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}))\ D \longrightarrow C \preceq_c D)$
⟨*proof*⟩

definition *atoms-in-model-were-produced* **where**
atoms-in-model-were-produced N s $\longleftrightarrow$
$(\forall\, U_{er}\ \mathcal{F}\ \mathcal{M}\ \mathcal{C}.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \longrightarrow (\forall\, A\ C.\ \mathcal{M}\ A = \mathit{Some}\ C \longrightarrow$
$A \in \mathit{ord\text{-}res.production}\ (\mathit{fset}\ (\mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})))\ C))$

lemma *atoms-in-model-were-produced*:
assumes *step*: *ord-res-5* N s s' **and**
invars:
atoms-in-model-were-produced N s
model-eq-interp-upto-next-clause N s
next-clause-in-factorized-clause N s
shows *atoms-in-model-were-produced* N s'
⟨*proof*⟩

definition *all-produced-atoms-in-model* **where**
all-produced-atoms-in-model N s $\longleftrightarrow$
$(\forall\, U_{er}\ \mathcal{F}\ \mathcal{M}\ D.\ s = (U_{er}, \mathcal{F}, \mathcal{M}, \mathit{Some}\ D) \longrightarrow (\forall\, C\ A.\ C \prec_c D \longrightarrow$
$A \in \mathit{ord\text{-}res.production}\ (\mathit{fset}\ (\mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})))\ C \longrightarrow \mathcal{M}\ A = \mathit{Some}$
$C))$

lemma *all-produced-atoms-in-model*:
assumes *step*: *ord-res-5* N s s' **and**
invars:
all-produced-atoms-in-model N s
model-eq-interp-upto-next-clause N s
next-clause-in-factorized-clause N s
shows *all-produced-atoms-in-model* N s'
⟨*proof*⟩

definition *ord-res-5-invars* **where**
ord-res-5-invars N s $\longleftrightarrow$
next-clause-in-factorized-clause N s $\wedge$
implicitly-factorized-clauses-subset N s $\wedge$
model-eq-interp-upto-next-clause N s $\wedge$
all-smaller-clauses-true-wrt-respective-Interp N s $\wedge$
atoms-in-model-were-produced N s $\wedge$
all-produced-atoms-in-model N s

lemma *ord-res-5-invars-initial-state*:
assumes

$\mathcal{F}$-*subset*: $\mathcal{F}$ |⊆| N |∪| U_{er} **and**
C-least: *linorder-cls.is-least-in-fset* (*iefac* $\mathcal{F}$ |‘| (N |∪| U_{er})) C
shows *ord-res-5-invars* N (U_{er}, $\mathcal{F}$, *Map.empty*, *Some* C)
⟨*proof*⟩

lemma *ord-res-5-preserves-invars*:
assumes *step*: *ord-res-5* N s s' **and** *invars*: *ord-res-5-invars* N s
shows *ord-res-5-invars* N s'
⟨*proof*⟩

lemma *rtranclp-ord-res-5-preserves-invars*:
assumes *steps*: (*ord-res-5* N)** s s' **and** *invars*: *ord-res-5-invars* N s
shows *ord-res-5-invars* N s'
⟨*proof*⟩

lemma *tranclp-ord-res-5-preserves-invars*:
assumes *steps*: (*ord-res-5* N)$^{++}$ s s' **and** *invars*: *ord-res-5-invars* N s
shows *ord-res-5-invars* N s'
⟨*proof*⟩

lemma *le-least-false-clause*:
fixes N s U_{er} $\mathcal{F}$ $\mathcal{M}$ C D
assumes
invars: *ord-res-5-invars* N s **and**
s-def: s = (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some* C) **and**
D-least-false: *is-least-false-clause* (*iefac* $\mathcal{F}$ |‘| (N |∪| U_{er})) D
shows $C \preceq_c D$
⟨*proof*⟩

lemma *ex-ord-res-5-if-not-final*:
assumes
not-final: ¬ *ord-res-5-final* S **and**
invars: $\forall N\ s.\ S = (N,\ s) \longrightarrow$ *ord-res-5-invars* N s
shows $\exists S'$. *ord-res-5-step* S S'
⟨*proof*⟩

lemma *ord-res-5-safe-state-if-invars*:
fixes N s
assumes *invars*: *ord-res-5-invars* N s
shows *safe-state ord-res-5-step ord-res-5-final* (N, s)
⟨*proof*⟩

lemma *MAGIC1*:
assumes *invars*: *ord-res-5-invars* N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$)
shows $\exists \mathcal{M}'\ \mathcal{C}'$. (*ord-res-5* N)** (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) (U_{er}, $\mathcal{F}$, $\mathcal{M}'$, $\mathcal{C}'$) ∧
($\nexists \mathcal{M}''\ \mathcal{C}''$. *ord-res-5* N (U_{er}, $\mathcal{F}$, $\mathcal{M}'$, $\mathcal{C}'$) (U_{er}, $\mathcal{F}$, $\mathcal{M}''$, $\mathcal{C}''$))
⟨*proof*⟩

lemma *MAGIC2*:

assumes *invars*: *ord-res-5-invars* N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some* C)
assumes $C \neq \{\#\}$
shows $\exists s'$. *ord-res-5* N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some* C) s'
⟨*proof*⟩

lemma *MAGIC3*:
assumes *invars*: *ord-res-5-invars* N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) **and**
steps: (*ord-res-5* N)** (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) (U_{er}, $\mathcal{F}$, $\mathcal{M}'$, $\mathcal{C}'$) **and**
no-more-steps: ($\nexists \mathcal{M}'' \mathcal{C}''$. *ord-res-5* N (U_{er}, $\mathcal{F}$, $\mathcal{M}'$, $\mathcal{C}'$) (U_{er}, $\mathcal{F}$, $\mathcal{M}''$, $\mathcal{C}''$))
shows ($\forall C$. $\mathcal{C}'$ = *Some* $C \longleftrightarrow$ *is-least-false-clause* (*iefac* $\mathcal{F}$ |‘| (N |∪| U_{er})) C)
⟨*proof*⟩

lemma *ord-res-5-construct-model-upto-least-false-clause*:
assumes *invars*: *ord-res-5-invars* N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$)
shows $\exists \mathcal{M}' \mathcal{C}'$. (*ord-res-5* N)** (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) (U_{er}, $\mathcal{F}$, $\mathcal{M}'$, $\mathcal{C}'$) ∧
($\forall C$. $\mathcal{C}'$ = *Some* $C \longleftrightarrow$ *is-least-false-clause* (*iefac* $\mathcal{F}$ |‘| (N |∪| U_{er})) C)
⟨*proof*⟩

end

end
theory *ORD-RES-6*
imports
ORD-RES-5
begin

21 ORD-RES-6 (model backjump)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-6* **where**
skip:
(*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$
$\mathcal{C}'$ = *The-optional* (*linorder-cls.is-least-in-fset* {|D |∈| *iefac* $\mathcal{F}$ |‘| (N |∪| U_{er}). $C \prec_c D$|}) $\Longrightarrow$
ord-res-6 N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some* C) (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}'$) |

production:
¬ (*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
is-pos $L \Longrightarrow$
linorder-lit.is-greatest-in-mset C $L \Longrightarrow$
$\mathcal{M}' = \mathcal{M}$(*atm-of* L := *Some* C) $\Longrightarrow$
$\mathcal{C}'$ = *The-optional* (*linorder-cls.is-least-in-fset* {|D |∈| *iefac* $\mathcal{F}$ |‘| (N |∪| U_{er}). $C \prec_c D$|}) $\Longrightarrow$
ord-res-6 N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, *Some* C) (U_{er}, $\mathcal{F}$, $\mathcal{M}'$, $\mathcal{C}'$) |

factoring:
¬ (*dom* $\mathcal{M}$) $\Vdash$ $C \Longrightarrow$

$linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ C\ L \Longrightarrow$
$is\text{-}pos\ L \Longrightarrow$
$\neg\ linorder\text{-}lit.is\text{-}greatest\text{-}in\text{-}mset\ C\ L \Longrightarrow$
$\mathcal{F}' = finsert\ C\ \mathcal{F} \Longrightarrow$
$ord\text{-}res\text{-}6\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C)\ (U_{er}, \mathcal{F}', \mathcal{M}, Some\ (efac\ C))\ |$

resolution-bot:
$\neg\ (dom\ \mathcal{M}) \models C \Longrightarrow$
$linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ C\ L \Longrightarrow$
$is\text{-}neg\ L \Longrightarrow$
$\mathcal{M}\ (atm\text{-}of\ L) = Some\ D \Longrightarrow$
$U_{er}' = finsert\ (eres\ D\ C)\ U_{er} \Longrightarrow$
$eres\ D\ C = \{\#\} \Longrightarrow$
$\mathcal{M}' = (\lambda\text{-}.\ None) \Longrightarrow$
$ord\text{-}res\text{-}6\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C)\ (U_{er}', \mathcal{F}, \mathcal{M}', Some\ \{\#\})\ |$

resolution-pos:
$\neg\ (dom\ \mathcal{M}) \models C \Longrightarrow$
$linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ C\ L \Longrightarrow$
$is\text{-}neg\ L \Longrightarrow$
$\mathcal{M}\ (atm\text{-}of\ L) = Some\ D \Longrightarrow$
$U_{er}' = finsert\ (eres\ D\ C)\ U_{er} \Longrightarrow$
$eres\ D\ C \neq \{\#\} \Longrightarrow$
$\mathcal{M}' = restrict\text{-}map\ \mathcal{M}\ \{A.\ A \prec_t atm\text{-}of\ K\} \Longrightarrow$
$linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ (eres\ D\ C)\ K \Longrightarrow$
$is\text{-}pos\ K \Longrightarrow$
$ord\text{-}res\text{-}6\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C)\ (U_{er}', \mathcal{F}, \mathcal{M}', Some\ (eres\ D\ C))\ |$

resolution-neg:
$\neg\ (dom\ \mathcal{M}) \models C \Longrightarrow$
$linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ C\ L \Longrightarrow$
$is\text{-}neg\ L \Longrightarrow$
$\mathcal{M}\ (atm\text{-}of\ L) = Some\ D \Longrightarrow$
$U_{er}' = finsert\ (eres\ D\ C)\ U_{er} \Longrightarrow$
$eres\ D\ C \neq \{\#\} \Longrightarrow$
$\mathcal{M}' = restrict\text{-}map\ \mathcal{M}\ \{A.\ A \prec_t atm\text{-}of\ K\} \Longrightarrow$
$linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ (eres\ D\ C)\ K \Longrightarrow$
$is\text{-}neg\ K \Longrightarrow$
$\mathcal{M}\ (atm\text{-}of\ K) = Some\ E \Longrightarrow$
$ord\text{-}res\text{-}6\ N\ (U_{er}, \mathcal{F}, \mathcal{M}, Some\ C)\ (U_{er}', \mathcal{F}, \mathcal{M}', Some\ E)$

inductive *ord-res-6-step* **where**
$ord\text{-}res\text{-}6\ N\ s\ s' \Longrightarrow ord\text{-}res\text{-}6\text{-}step\ (N, s)\ (N, s')$

lemma *tranclp-ord-res-6-step-if-tranclp-ord-res-6*:
$(ord\text{-}res\text{-}6\ N)^{++}\ s\ s' \Longrightarrow ord\text{-}res\text{-}6\text{-}step^{++}\ (N, s)\ (N, s')$
⟨*proof*⟩

lemma *right-unique-ord-res-6*: $right\text{-}unique\ (ord\text{-}res\text{-}6\ N)$

⟨*proof*⟩

lemma *right-unique-ord-res-6-step*: *right-unique ord-res-6-step*
⟨*proof*⟩

inductive *ord-res-6-final* **where**
model-found:
ord-res-6-final $(N,\ U_{er},\ \mathcal{F},\ \mathcal{M},\ None)$ |

contradiction-found:
ord-res-6-final $(N,\ U_{er},\ \mathcal{F},\ \mathcal{M},\ Some\ \{\#\})$

sublocale *ord-res-6-semantics*: *semantics* **where**
step = *ord-res-6-step* **and**
final = *ord-res-6-final*
⟨*proof*⟩

lemma *ord-res-6-preserves-invars*:
assumes *step*: *ord-res-6 N s s′* **and** *invars*: *ord-res-5-invars N s*
shows *ord-res-5-invars N s′*
⟨*proof*⟩

lemma *rtranclp-ord-res-6-preserves-invars*:
assumes *steps*: $(ord\text{-}res\text{-}6\ N)^{**}\ s\ s'$ **and** *invars*: *ord-res-5-invars N s*
shows *ord-res-5-invars N s′*
⟨*proof*⟩

lemma *ex-ord-res-6-if-not-final*:
assumes
not-final: ¬ *ord-res-6-final S* **and**
invars: $\forall N\ s.\ S = (N,\ s) \longrightarrow$ *ord-res-5-invars N s*
shows $\exists S'.$ *ord-res-6-step S S′*
⟨*proof*⟩

lemma *ord-res-6-safe-state-if-invars*:
safe-state ord-res-6-step ord-res-6-final $(N,\ s)$ **if** *invars*: *ord-res-5-invars N s* **for** *N s*
⟨*proof*⟩

lemma *ex-model-build-from-least-clause-to-any-less-than-least-false*:
assumes
$\mathcal{F}$*-subset*: $\mathcal{F}\ |\subseteq|\ N\ |\cup|\ U_{er}$ **and**
C-least: *linorder-cls.is-least-in-fset* $(iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}))\ C$ **and**
D-in: $D\ |\in|\ iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})$ **and**
D-lt-least-false: $\forall E.$ *is-least-false-clause* $(iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}))\ E \longrightarrow D \preceq_c E$ **and**
$C \preceq_c D$
shows $\exists \mathcal{M}.\ (ord\text{-}res\text{-}5\ N)^{**}\ (U_{er},\ \mathcal{F},\ Map.empty,\ Some\ C)\ (U_{er},\ \mathcal{F},\ \mathcal{M},\ Some\ D)$

⟨*proof*⟩

lemma *full-rtranclp-ord-res-5-run-upto*:
assumes
ord-res-6 N $(U_{er}, \mathcal{F}, \mathcal{M}, Some\ E)$ $(U_{er}', \mathcal{F}', \mathcal{M}', Some\ D)$ **and**
invars: *ord-res-5-invars N* $(U_{er}', \mathcal{F}', \mathcal{M}', Some\ D)$ **and**
$\mathcal{M}'$*-def*: $\mathcal{M}' = restrict\text{-}map\ \mathcal{M}\ \{A.\ \exists K.\ linorder\text{-}lit.is\text{-}maximal\text{-}in\text{-}mset\ D\ K \wedge A \prec_t atm\text{-}of\ K\}$ **and**
C-least: *linorder-cls.is-least-in-fset* $(iefac\ \mathcal{F}'\ |`|\ (N\ |\cup|\ U_{er}'))\ C$
shows $(ord\text{-}res\text{-}5\ N)^{**}$ $(U_{er}', \mathcal{F}', Map.empty, Some\ C)$ $(U_{er}', \mathcal{F}', \mathcal{M}', Some\ D)$
⟨*proof*⟩

end

end
theory *ORD-RES-7*
imports
Background
Implicit-Exhaustive-Factorization
Exhaustive-Resolution
begin

22 ORD-RES-7 (clause-guided literal trail construction)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-7* **where**
decide-neg:
$\neg$ *trail-false-cls* $\Gamma\ C \Longrightarrow$
linorder-lit.is-maximal-in-mset $C\ L \Longrightarrow$
linorder-trm.is-least-in-fset $\{|A\ |\in|\ atms\text{-}of\text{-}clss\ (N\ |\cup|\ U_{er}).$
$A \prec_t atm\text{-}of\ L \wedge A\ |\notin|\ trail\text{-}atms\ \Gamma|\}\ A \Longrightarrow$
$\Gamma' = (Neg\ A,\ None)\ \#\ \Gamma \Longrightarrow$
ord-res-7 N $(U_{er}, \mathcal{F}, \Gamma, Some\ C)$ $(U_{er}, \mathcal{F}, \Gamma', Some\ C)$ |

skip-defined:
$\neg$ *trail-false-cls* $\Gamma\ C \Longrightarrow$
linorder-lit.is-maximal-in-mset $C\ L \Longrightarrow$
$\neg(\exists A\ |\in|\ atms\text{-}of\text{-}clss\ (N\ |\cup|\ U_{er}).\ A \prec_t atm\text{-}of\ L \wedge A\ |\notin|\ trail\text{-}atms\ \Gamma) \Longrightarrow$
trail-defined-lit $\Gamma\ L \Longrightarrow$
$\mathcal{C}' = The\text{-}optional\ (linorder\text{-}cls.is\text{-}least\text{-}in\text{-}fset\ \{|D\ |\in|\ iefac\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ C \prec_c D|\}) \Longrightarrow$
ord-res-7 N $(U_{er}, \mathcal{F}, \Gamma, Some\ C)$ $(U_{er}, \mathcal{F}, \Gamma, \mathcal{C}')$ |

skip-undefined-neg:
$\neg$ *trail-false-cls* $\Gamma\ C \Longrightarrow$
linorder-lit.is-maximal-in-mset $C\ L \Longrightarrow$

$\neg(\exists A\ |\in|\ \textit{atms-of-clss}\ (N\ |\cup|\ U_{er}).\ A \prec_t \textit{atm-of}\ L \wedge A\ |\notin|\ \textit{trail-atms}\ \Gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ L \Longrightarrow$
$\textit{is-neg}\ L \Longrightarrow$
$\Gamma' = (L,\ \textit{None})\ \#\ \Gamma \Longrightarrow$
$\mathcal{C}' = \textit{The-optional}\ (\textit{linorder-cls.is-least-in-fset}\ \{|D\ |\in|\ \textit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).$
$C \prec_c D|\}) \Longrightarrow$
$\textit{ord-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ \textit{Some}\ C)\ (U_{er},\ \mathcal{F},\ \Gamma',\ \mathcal{C}')\ |$

skip-undefined-pos:
$\neg\ \textit{trail-false-cls}\ \Gamma\ C \Longrightarrow$
$\textit{linorder-lit.is-maximal-in-mset}\ C\ L \Longrightarrow$
$\neg(\exists A\ |\in|\ \textit{atms-of-clss}\ (N\ |\cup|\ U_{er}).\ A \prec_t \textit{atm-of}\ L \wedge A\ |\notin|\ \textit{trail-atms}\ \Gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ L \Longrightarrow$
$\textit{is-pos}\ L \Longrightarrow$
$\neg\ \textit{trail-false-cls}\ \Gamma\ \{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$
$\textit{linorder-cls.is-least-in-fset}\ \{|D\ |\in|\ \textit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ C \prec_c D|\}\ D \Longrightarrow$
$\textit{ord-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ \textit{Some}\ C)\ (U_{er},\ \mathcal{F},\ \Gamma,\ \textit{Some}\ D)\ |$

skip-undefined-pos-ultimate:
$\neg\ \textit{trail-false-cls}\ \Gamma\ C \Longrightarrow$
$\textit{linorder-lit.is-maximal-in-mset}\ C\ L \Longrightarrow$
$\neg(\exists A\ |\in|\ \textit{atms-of-clss}\ (N\ |\cup|\ U_{er}).\ A \prec_t \textit{atm-of}\ L \wedge A\ |\notin|\ \textit{trail-atms}\ \Gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ L \Longrightarrow$
$\textit{is-pos}\ L \Longrightarrow$
$\neg\ \textit{trail-false-cls}\ \Gamma\ \{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$
$\Gamma' = (-\ L,\ \textit{None})\ \#\ \Gamma \Longrightarrow$
$\neg(\exists D\ |\in|\ \textit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ C \prec_c D) \Longrightarrow$
$\textit{ord-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ \textit{Some}\ C)\ (U_{er},\ \mathcal{F},\ \Gamma',\ \textit{None})\ |$

production:
$\neg\ \textit{trail-false-cls}\ \Gamma\ C \Longrightarrow$
$\textit{linorder-lit.is-maximal-in-mset}\ C\ L \Longrightarrow$
$\neg\ (\exists A\ |\in|\ \textit{atms-of-clss}\ (N\ |\cup|\ U_{er}).\ A \prec_t \textit{atm-of}\ L \wedge A\ |\notin|\ \textit{trail-atms}\ \Gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ L \Longrightarrow$
$\textit{is-pos}\ L \Longrightarrow$
$\textit{trail-false-cls}\ \Gamma\ \{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$
$\textit{linorder-lit.is-greatest-in-mset}\ C\ L \Longrightarrow$
$\Gamma' = (L,\ \textit{Some}\ C)\ \#\ \Gamma \Longrightarrow$
$\mathcal{C}' = \textit{The-optional}\ (\textit{linorder-cls.is-least-in-fset}\ \{|D\ |\in|\ \textit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).$
$C \prec_c D|\}) \Longrightarrow$
$\textit{ord-res-7}\ N\ (U_{er},\ \mathcal{F},\ \Gamma,\ \textit{Some}\ C)\ (U_{er},\ \mathcal{F},\ \Gamma',\ \mathcal{C}')\ |$

factoring:
$\neg\ \textit{trail-false-cls}\ \Gamma\ C \Longrightarrow$
$\textit{linorder-lit.is-maximal-in-mset}\ C\ L \Longrightarrow$
$\neg\ (\exists A\ |\in|\ \textit{atms-of-clss}\ (N\ |\cup|\ U_{er}).\ A \prec_t \textit{atm-of}\ L \wedge A\ |\notin|\ \textit{trail-atms}\ \Gamma) \Longrightarrow$
$\neg\ \textit{trail-defined-lit}\ \Gamma\ L \Longrightarrow$
$\textit{is-pos}\ L \Longrightarrow$
$\textit{trail-false-cls}\ \Gamma\ \{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$

$\neg$ *linorder-lit.is-greatest-in-mset* C $L \Longrightarrow$
$\mathcal{F}' =$ *finsert* C $\mathcal{F} \Longrightarrow$
ord-res-7 N $(U_{er}, \mathcal{F}, \Gamma,$ *Some* $C)$ $(U_{er}, \mathcal{F}', \Gamma,$ *Some* (*efac* C)) |

resolution-bot:
trail-false-cls Γ $E \Longrightarrow$
linorder-lit.is-maximal-in-mset E $L \Longrightarrow$
is-neg $L \Longrightarrow$
map-of Γ $(-\ L) =$ *Some* (*Some* D) $\Longrightarrow$
$U_{er}{}' =$ *finsert* (*eres* D E) $U_{er} \Longrightarrow$
eres D $E = \{\#\} \Longrightarrow$
$\Gamma' = [] \Longrightarrow$
ord-res-7 N $(U_{er}, \mathcal{F}, \Gamma,$ *Some* $E)$ $(U_{er}{}', \mathcal{F}, \Gamma',$ *Some* $\{\#\})$ |

resolution-pos:
trail-false-cls Γ $E \Longrightarrow$
linorder-lit.is-maximal-in-mset E $L \Longrightarrow$
is-neg $L \Longrightarrow$
map-of Γ $(-\ L) =$ *Some* (*Some* D) $\Longrightarrow$
$U_{er}{}' =$ *finsert* (*eres* D E) $U_{er} \Longrightarrow$
eres D $E \neq \{\#\} \Longrightarrow$
$\Gamma' =$ *dropWhile* ($\lambda Ln.$ *atm-of* $K \preceq_t$ *atm-of* (*fst* Ln)) $\Gamma \Longrightarrow$
linorder-lit.is-maximal-in-mset (*eres* D E) $K \Longrightarrow$
is-pos $K \Longrightarrow$
ord-res-7 N $(U_{er}, \mathcal{F}, \Gamma,$ *Some* $E)$ $(U_{er}{}', \mathcal{F}, \Gamma',$ *Some* (*eres* D E)) |

resolution-neg:
trail-false-cls Γ $E \Longrightarrow$
linorder-lit.is-maximal-in-mset E $L \Longrightarrow$
is-neg $L \Longrightarrow$
map-of Γ $(-\ L) =$ *Some* (*Some* D) $\Longrightarrow$
$U_{er}{}' =$ *finsert* (*eres* D E) $U_{er} \Longrightarrow$
eres D $E \neq \{\#\} \Longrightarrow$
$\Gamma' =$ *dropWhile* ($\lambda Ln.$ *atm-of* $K \preceq_t$ *atm-of* (*fst* Ln)) $\Gamma \Longrightarrow$
linorder-lit.is-maximal-in-mset (*eres* D E) $K \Longrightarrow$
is-neg $K \Longrightarrow$
map-of Γ $(-\ K) =$ *Some* (*Some* C) $\Longrightarrow$
ord-res-7 N $(U_{er}, \mathcal{F}, \Gamma,$ *Some* $E)$ $(U_{er}{}', \mathcal{F}, \Gamma',$ *Some* $C)$

lemma *right-unique-ord-res-7*:
fixes N :: *'f gclause fset*
shows *right-unique* (*ord-res-7* N)
$\langle proof \rangle$

inductive *ord-res-7-final* **where**
model-found:
ord-res-7-final $(N, U_{er}, \mathcal{F}, \Gamma,$ *None*) |

contradiction-found:
ord-res-7-final $(N,\ U_{er},\ \mathcal{F},\ \Gamma,\ \mathit{Some}\ \{\#\})$

sublocale *ord-res-7-semantics*: *semantics* **where**
step = *constant-context* *ord-res-7* **and**
final = *ord-res-7-final*
⟨*proof*⟩

inductive *ord-res-7-invars* **for** N **where**
ord-res-7-invars N $(U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$ **if**
$\mathcal{F}$ |⊆| N |∪| U_{er} **and**
$(\forall C.\ \mathcal{C} = \mathit{Some}\ C \longrightarrow C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}))$ **and**
$(\forall D.\ \mathcal{C} = \mathit{Some}\ D \longrightarrow$
$(\forall C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}).\ C \prec_c D \longrightarrow$
$(\forall L_C.$ *linorder-lit.is-maximal-in-mset* $C\ L_C \longrightarrow$
$\neg$ *trail-defined-lit* $\Gamma\ L_C \longrightarrow$ (*trail-true-cls* Γ $\{\#K \in\# C.\ K \neq L_C\#\}$ $\wedge$ *is-pos* L_C)))) **and**
$(\forall C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}).\ (\forall D.\ \mathcal{C} = \mathit{Some}\ D \longrightarrow C \prec_c D) \longrightarrow$ *trail-true-cls* $\Gamma\ C$) **and**
$(\forall C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}).\ (\forall D.\ \mathcal{C} = \mathit{Some}\ D \longrightarrow C \prec_c D) \longrightarrow$
$(\forall K.$ *linorder-lit.is-maximal-in-mset* $C\ K \longrightarrow$
$\neg\ (\exists A$ |∈| *atms-of-clss* $(N$ |∪| $U_{er}).\ A \prec_t$ *atm-of* $K \wedge A$ |∉| *trail-atms* Γ))) **and**
sorted-wrt $(\lambda x\ y.$ *atm-of* (*fst* y) $\prec_t$ *atm-of* (*fst* x)) Γ **and**
linorder-trm.is-lower-fset (*trail-atms* Γ) (*atms-of-clss* $(N$ |∪| $U_{er})$) **and**
$(\mathcal{C} = \mathit{None} \longrightarrow$ *trail-atms* Γ = *atms-of-clss* $(N$ |∪| $U_{er}))$ **and**
$(\forall C.\ \mathcal{C} = \mathit{Some}\ C \longrightarrow$
$(\forall A$ |∈| *trail-atms* $\Gamma.\ \exists L.$ *linorder-lit.is-maximal-in-mset* $C\ L \wedge A \preceq_t$ *atm-of* L)) **and**
$(\forall Ln \in \mathit{set}\ \Gamma.$ *is-neg* (*fst* Ln) $\longleftrightarrow$ *snd* $Ln = \mathit{None}$) **and**
$(\forall Ln \in \mathit{set}\ \Gamma.$ *snd* $Ln = \mathit{None} \longrightarrow$
$(\forall C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}).\ C \prec_c \{\#\mathit{fst}\ Ln\#\} \longrightarrow$ *trail-true-cls* $\Gamma\ C$)) **and**
$(\forall Ln \in \mathit{set}\ \Gamma.\ \forall C.$ *snd* $Ln = \mathit{Some}\ C \longrightarrow$ *linorder-lit.is-greatest-in-mset* C (*fst* Ln)) **and**
$(\forall Ln \in \mathit{set}\ \Gamma.\ \forall C.$ *snd* $Ln = \mathit{Some}\ C \longrightarrow C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}))$ **and**
$(\forall \Gamma_1\ L\ C\ \Gamma_0.\ \Gamma = \Gamma_1$ @ $(L,\ \mathit{Some}\ C)\ \#\ \Gamma_0 \longrightarrow$ *trail-false-cls* Γ_0 $\{\#K \in\# C.\ K \neq L\#\})$ **and**
$(\forall \Gamma_1\ L\ D\ \Gamma_0.\ \Gamma = \Gamma_1$ @ $(L,\ \mathit{Some}\ D)\ \#\ \Gamma_0 \longrightarrow$
$(\forall C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}).\ C \prec_c D \longrightarrow$ *trail-true-cls* $\Gamma_0\ C$))

lemma *clause-almost-defined-if-lt-next-clause*:
assumes *ord-res-7-invars* N $(U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$
shows $\forall C$ |∈| *iefac* $\mathcal{F}$ |'| $(N$ |∪| $U_{er}).\ (\forall D.\ \mathcal{C} = \mathit{Some}\ D \longrightarrow C \prec_c D) \longrightarrow$
$(\forall K.$ *linorder-lit.is-maximal-in-mset* $C\ K \longrightarrow$ *trail-defined-cls* Γ $\{\#L \in\# C.\ L \neq K\#\})$
⟨*proof*⟩

lemma *ord-res-7-invars-def*:

ord-res-7-invars N s $\longleftrightarrow$
($\forall$ U_{er} $\mathcal{F}$ Γ $\mathcal{C}$. s = (U_{er}, $\mathcal{F}$, Γ, $\mathcal{C}$) $\longrightarrow$
$\mathcal{F}$ |⊆| N |∪| U_{er} ∧
(∀ C. $\mathcal{C}$ = *Some* C ⟶ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er})) ∧
(∀ D. $\mathcal{C}$ = *Some* D ⟶
(∀ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}). $C \prec_c D$ ⟶
(∀ L_C. *linorder-lit.is-maximal-in-mset* C L_C ⟶
¬ *trail-defined-lit* Γ L_C ⟶ (*trail-true-cls* Γ {#K ∈# C. $K \neq L_C$#} ∧ *is-pos* L_C)))) ∧
(∀ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}). (∀ D. $\mathcal{C}$ = *Some* D ⟶ $C \prec_c D$) ⟶ *trail-true-cls* Γ C) ∧
(∀ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}). (∀ D. $\mathcal{C}$ = *Some* D ⟶ $C \prec_c D$) ⟶
(∀ K. *linorder-lit.is-maximal-in-mset* C K ⟶
¬ (∃ A |∈| *atms-of-clss* (N |∪| U_{er}). $A \prec_t$ *atm-of* K ∧ A |∉| *trail-atms* Γ))) ∧
sorted-wrt (λx y. *atm-of* (*fst* y) $\prec_t$ *atm-of* (*fst* x)) Γ ∧
linorder-trm.is-lower-fset (*trail-atms* Γ) (*atms-of-clss* (N |∪| U_{er})) ∧
($\mathcal{C}$ = *None* ⟶ *trail-atms* Γ = *atms-of-clss* (N |∪| U_{er})) ∧
(∀ C. $\mathcal{C}$ = *Some* C ⟶
(∀ A |∈| *trail-atms* Γ. ∃ L. *linorder-lit.is-maximal-in-mset* C L ∧ $A \preceq_t$ *atm-of* L)) ∧
(∀ *Ln* ∈ *set* Γ. *is-neg* (*fst Ln*) ⟷ *snd Ln* = *None*) ∧
(∀ *Ln* ∈ *set* Γ. *snd Ln* = *None* ⟶
(∀ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}). $C \prec_c$ {#*fst Ln*#} ⟶ *trail-true-cls* Γ C)) ∧
(∀ *Ln* ∈ *set* Γ. ∀ C. *snd Ln* = *Some* C ⟶ *linorder-lit.is-greatest-in-mset* C (*fst Ln*)) ∧
(∀ *Ln* ∈ *set* Γ. ∀ C. *snd Ln* = *Some* C ⟶ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er})) ∧
(∀ Γ_1 L C Γ_0. Γ = Γ_1 @ (L, *Some* C) # Γ_0 ⟶ *trail-false-cls* Γ_0 {#K ∈# C. $K \neq L$#}) ∧
(∀ Γ_1 L D Γ_0. Γ = Γ_1 @ (L, *Some* D) # Γ_0 ⟶
(∀ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}). $C \prec_c D$ ⟶ *trail-true-cls* Γ_0 C)))
(**is** *?NICE N s* ⟷ *?UGLY N s*)
⟨*proof*⟩

lemma *ord-res-7-invars-implies-trail-consistent*:
assumes *ord-res-7-invars* N (U_{er}, $\mathcal{F}$, Γ, $\mathcal{C}$)
shows *trail-consistent* Γ
⟨*proof*⟩

lemma *ord-res-7-invars-implies-propagated-clause-almost-false*:
assumes *invars*: *ord-res-7-invars* N (U_{er}, $\mathcal{F}$, Γ, $\mathcal{C}$) **and** (L, *Some* C) ∈ *set* Γ
shows *trail-false-cls* Γ {#K ∈# C. $K \neq L$#}
⟨*proof*⟩

lemma *ord-res-7-preserves-invars*:
assumes *step*: *ord-res-7* N s s' **and** *invar*: *ord-res-7-invars* N s
shows *ord-res-7-invars* N s'
⟨*proof*⟩

lemma *rtranclp-ord-res-7-preserves-ord-res-7-invars*:
  assumes
    step: $(ord\text{-}res\text{-}7\ N)^{**}\ s\ s'$ **and**
    invars: *ord-res-7-invars N s*
  shows *ord-res-7-invars N s′*
  ⟨*proof*⟩

lemma *tranclp-ord-res-7-preserves-ord-res-7-invars*:
  assumes
    step: $(ord\text{-}res\text{-}7\ N)^{++}\ s\ s'$ **and**
    invars: *ord-res-7-invars N s*
  shows *ord-res-7-invars N s′*
  ⟨*proof*⟩

lemma *propagating-clause-almost-false*:
  assumes *invars*: *ord-res-7-invars N* $(U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$ **and** $(L, Some\ C) \in set\ \Gamma$
  shows *trail-false-cls* $\Gamma\ \{\#K \in\#\ C.\ K \neq L\#\}$
⟨*proof*⟩

lemma *ex-ord-res-7-if-not-final*:
  assumes
    not-final: ¬ *ord-res-7-final* (*N*, *s*) **and**
    invars: *ord-res-7-invars N s*
  shows ∃ *s′*. *ord-res-7 N s s′*
⟨*proof*⟩

lemma *ord-res-7-safe-state-if-invars*:
  fixes *N* :: *′f gclause fset* **and** *s*
  assumes *invars*: *ord-res-7-invars N s*
  shows *safe-state* (*constant-context ord-res-7*) *ord-res-7-final* (*N*, *s*)
  ⟨*proof*⟩

end

end
theory *Clause-Could-Propagate*
  imports
    Background
    Implicit-Exhaustive-Factorization
begin

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

definition *clause-could-propagate* **where**
  clause-could-propagate Γ *C L* ⟷ ¬ *trail-defined-lit* Γ *L* ∧
    linorder-lit.is-maximal-in-mset C L ∧ *trail-false-cls* $\Gamma\ \{\#K \in\#\ C.\ K \neq L\#\}$

lemma *trail-false-if-could-have-propagatated*:

clause-could-propagate Γ *C L* ⟹ *trail-false-cls* ((− *L*, *n*) # Γ) *C*
⟨*proof*⟩

lemma *atoms-of-trail-lt-atom-of-propagatable-literal*:
assumes
Γ-*lower*: *linorder-trm.is-lower-set* (*fset* (*trail-atms* Γ)) $\mathcal{A}$ **and**
C-prop: *clause-could-propagate* Γ *C L* **and**
atm-of L ∈ $\mathcal{A}$
shows ∀ *A* |∈| *trail-atms* Γ. *A* $\prec_t$ *atm-of L*
⟨*proof*⟩

lemma *trail-false-cls-filter-mset-iff*:
trail-false-cls Γ {#*Ka* ∈# *C*. *Ka* ≠ *K*#} ⟷ (∀ *L*∈#*C*. *L* ≠ *K* ⟶ *trail-false-lit* Γ *L*)
⟨*proof*⟩

lemma *clause-could-propagate-iff*: *clause-could-propagate* Γ *C K* ⟷
¬ *trail-defined-lit* Γ *K* ∧ *ord-res.is-maximal-lit K C* ∧ (∀ *L*∈#*C*. *L* ≠ *K* ⟶ *trail-false-lit* Γ *L*)
⟨*proof*⟩

lemma *clause-could-propagate-efac*: *clause-could-propagate* Γ (*efac C*) = *clause-could-propagate* Γ *C*
⟨*proof*⟩

lemma *bex-clause-could-propagate-simp*:
fixes $\mathcal{F}$ *N* Γ *L*
shows *fBex* (*iefac* $\mathcal{F}$ |'| *N*) (λ*C*. *clause-could-propagate* Γ *C L*) ⟷
fBex N (λ*C*. *clause-could-propagate* Γ *C L*)
sketch (*rule iffI*; *elim bexE*)
⟨*proof*⟩

end

end
theory *ORD-RES-8*
imports
Background
Implicit-Exhaustive-Factorization
Exhaustive-Resolution
Clause-Could-Propagate
begin

23 ORD-RES-8 (atom-guided literal trail construction)

type-synonym *'f ord-res-8-state* =
'f gclause fset ×*'f gclause fset* × *'f gclause fset* × (*'f gliteral* × *'f gclause option*)

list

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-8* **where**
decide-neg:
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ C) $\Longrightarrow$
linorder-trm.is-least-in-fset {|A_2 |$\in$| *atms-of-clss* (N |$\cup$| U_{er}).
$\forall$ A_1 |$\in$| *trail-atms* Γ. $A_1 \prec_t A_2$|} A $\Longrightarrow$
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *clause-could-propagate* Γ C (*Pos* A)) $\Longrightarrow$
Γ' = (*Neg* A, *None*) # Γ $\Longrightarrow$
ord-res-8 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}$, Γ') |

propagate:
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ C) $\Longrightarrow$
linorder-trm.is-least-in-fset {|A_2 |$\in$| *atms-of-clss* (N |$\cup$| U_{er}).
$\forall$ A_1 |$\in$| *trail-atms* Γ. $A_1 \prec_t A_2$|} A $\Longrightarrow$
linorder-cls.is-least-in-fset {|C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}).
clause-could-propagate Γ C (*Pos* A)|} C $\Longrightarrow$
linorder-lit.is-greatest-in-mset C (*Pos* A) $\Longrightarrow$
Γ' = (*Pos* A, *Some* C) # Γ $\Longrightarrow$
ord-res-8 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}$, Γ') |

factorize:
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ C) $\Longrightarrow$
linorder-trm.is-least-in-fset {|A_2 |$\in$| *atms-of-clss* (N |$\cup$| U_{er}).
$\forall$ A_1 |$\in$| *trail-atms* Γ. $A_1 \prec_t A_2$|} A $\Longrightarrow$
linorder-cls.is-least-in-fset {|C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}).
clause-could-propagate Γ C (*Pos* A)|} C $\Longrightarrow$
$\neg$ *linorder-lit.is-greatest-in-mset* C (*Pos* A) $\Longrightarrow$
$\mathcal{F}'$ = *finsert* C $\mathcal{F}$ $\Longrightarrow$
ord-res-8 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}'$, Γ) |

resolution:
linorder-cls.is-least-in-fset {|D |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ D|}
D $\Longrightarrow$
linorder-lit.is-maximal-in-mset D (*Neg* A) $\Longrightarrow$
map-of Γ (*Pos* A) = *Some* (*Some* C) $\Longrightarrow$
$U_{er}{}'$ = *finsert* (*eres* C D) U_{er} $\Longrightarrow$
Γ' = *dropWhile* ($\lambda Ln.$ $\forall$ K.
linorder-lit.is-maximal-in-mset (*eres* C D) K $\longrightarrow$ *atm-of* $K \preceq_t$ *atm-of* (*fst*
Ln)) Γ $\Longrightarrow$
ord-res-8 N (U_{er}, $\mathcal{F}$, Γ) ($U_{er}{}'$, $\mathcal{F}$, Γ')

lemma *right-unique-ord-res-8*:
fixes N :: *'f gclause fset*
shows *right-unique* (*ord-res-8* N)
⟨*proof*⟩

inductive *ord-res-8-final* :: *'f ord-res-8-state* ⇒ *bool* **where**
model-found:
$\neg$ ($\exists$ *A* |∈| *atms-of-clss* (*N* |∪| U_{er}). *A* |∉| *trail-atms* Γ) ⟹
$\neg$ ($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |\`| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*) ⟹
ord-res-8-final (*N*, U_{er}, $\mathcal{F}$, Γ) |

contradiction-found:
{#} |∈| *iefac* $\mathcal{F}$ |\`| (*N* |∪| U_{er}) ⟹
ord-res-8-final (*N*, U_{er}, $\mathcal{F}$, Γ)

sublocale *ord-res-8-semantics*: *semantics* **where**
step = *constant-context ord-res-8* **and**
final = *ord-res-8-final*
⟨*proof*⟩

definition *trail-is-sorted* **where**
trail-is-sorted N s ⟷
(∀ U_{er} $\mathcal{F}$ Γ. *s* = (U_{er}, $\mathcal{F}$, Γ) ⟶
sorted-wrt (λ*x y*. *atm-of* (*fst y*) $\prec_t$ *atm-of* (*fst x*)) Γ)

lemma *ord-res-8-preserves-trail-is-sorted*:
assumes
step: *ord-res-8 N s s'* **and**
invar: *trail-is-sorted N s*
shows *trail-is-sorted N s'*
⟨*proof*⟩

inductive *trail-annotations-invars*
for *N* :: *'f gterm literal multiset fset*
where
Nil:
trail-annotations-invars N (U_{er}, $\mathcal{F}$, []) |
Cons-None:
trail-annotations-invars N (U_{er}, $\mathcal{F}$, (*L*, *None*) # Γ)
if *trail-annotations-invars N* (U_{er}, $\mathcal{F}$, Γ) |
Cons-Some:
trail-annotations-invars N (U_{er}, $\mathcal{F}$, (*L*, *Some D*) # Γ)
if *linorder-lit.is-greatest-in-mset D L* **and**
D |∈| *iefac* $\mathcal{F}$ |\`| (*N* |∪| U_{er}) **and**
trail-false-cls Γ {#*K* ∈# *D*. *K* ≠ *L*#} **and**
linorder-cls.is-least-in-fset
{|*D* |∈| *iefac* $\mathcal{F}$ |\`| (*N* |∪| U_{er}). *clause-could-propagate* Γ *D L*|} *D* **and**
trail-annotations-invars N (U_{er}, $\mathcal{F}$, Γ)

lemma
assumes
linorder-lit.is-greatest-in-mset C L **and**
trail-false-cls Γ {#*K* ∈# *C*. *K* ≠ *L*#} **and**
$\neg$ *trail-defined-cls* Γ *C*

shows *clause-could-propagate* Γ *C L*
⟨*proof*⟩

lemma *propagating-clause-in-clauses*:
assumes *trail-annotations-invars N* $(U_{er}, \mathcal{F}, \Gamma)$ **and** *map-of* Γ *L* = *Some* (*Some* *C*)
shows C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er})
⟨*proof*⟩

lemma *trail-annotations-invars-mono-wrt-trail-suffix*:
assumes *suffix* Γ' Γ *trail-annotations-invars N* $(U_{er}, \mathcal{F}, \Gamma)$
shows *trail-annotations-invars N* $(U_{er}, \mathcal{F}, \Gamma')$
⟨*proof*⟩

lemma *ord-res-8-preserves-trail-annotations-invars*:
assumes
step: *ord-res-8 N s s'* **and**
invars:
trail-annotations-invars N s
trail-is-sorted N s
shows *trail-annotations-invars N s'*
⟨*proof*⟩

definition *trail-is-lower-set* **where**
trail-is-lower-set N s ⟷
(∀ U_{er} $\mathcal{F}$ Γ. $s = (U_{er}, \mathcal{F}, \Gamma)$ ⟶
linorder-trm.is-lower-fset (*trail-atms* Γ) (*atms-of-clss* (N |∪| U_{er})))

lemma *atoms-not-in-clause-set-undefined-if-trail-is-sorted-lower-set*:
assumes *invar*: *trail-is-lower-set N* $(U_{er}, \mathcal{F}, \Gamma)$
shows ∀ *A*. *A* |∉| *atms-of-clss* (N |∪| U_{er}) ⟶ *A* |∉| *trail-atms* Γ
⟨*proof*⟩

lemma *ord-res-8-preserves-atoms-in-trail-lower-set*:
assumes
step: *ord-res-8 N s s'* **and**
invars:
trail-is-lower-set N s
trail-annotations-invars N s
trail-is-sorted N s
shows *trail-is-lower-set N s'*
⟨*proof*⟩

definition *false-cls-is-mempty-or-has-neg-max-lit* **where**
false-cls-is-mempty-or-has-neg-max-lit N s ⟷
(∀ U_{er} $\mathcal{F}$ Γ. $s = (U_{er}, \mathcal{F}, \Gamma)$ ⟶ (∀ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}).
trail-false-cls Γ *C* ⟶ *C* = {#} ∨ (∃ *A*. *linorder-lit.is-maximal-in-mset* *C* (*Neg A*))))

lemma *ord-res-8-preserves-false-cls-is-mempty-or-has-neg-max-lit*:
assumes
step: *ord-res-8 N s s'* **and**
invars:
false-cls-is-mempty-or-has-neg-max-lit N s
trail-is-lower-set N s
trail-is-sorted N s
shows *false-cls-is-mempty-or-has-neg-max-lit N s'*
⟨*proof*⟩

definition *decided-literals-all-neg* **where**
decided-literals-all-neg N s ⟷
$(\forall\, U_{er}\ \mathcal{F}\ \Gamma.\ s = (U_{er}, \mathcal{F}, \Gamma) \longrightarrow$
$(\forall\, Ln \in set\ \Gamma.\ \forall\, L.\ Ln = (L, None) \longrightarrow is\text{-}neg\ L))$

lemma *ord-res-8-preserves-decided-literals-all-neg*:
assumes
step: *ord-res-8 N s s'* **and**
invar: *decided-literals-all-neg N s*
shows *decided-literals-all-neg N s'*
⟨*proof*⟩

definition *ord-res-8-invars* **where**
ord-res-8-invars N s ⟷
trail-is-sorted N s ∧
trail-is-lower-set N s ∧
false-cls-is-mempty-or-has-neg-max-lit N s ∧
trail-annotations-invars N s ∧
decided-literals-all-neg N s

lemma *ord-res-8-preserves-invars*:
assumes
step: *ord-res-8 N s s'* **and**
invars: *ord-res-8-invars N s*
shows *ord-res-8-invars N s'*
⟨*proof*⟩

lemma *rtranclp-ord-res-8-preserves-invars*:
assumes
step: $(ord\text{-}res\text{-}8\ N)^{**}\ s\ s'$ **and**
invars: *ord-res-8-invars N s*
shows *ord-res-8-invars N s'*
⟨*proof*⟩

lemma *tranclp-ord-res-8-preserves-invars*:
assumes
step: $(ord\text{-}res\text{-}8\ N)^{++}\ s\ s'$ **and**
invars: *ord-res-8-invars N s*

shows *ord-res-8-invars N s'*
⟨*proof*⟩

lemma *ex-ord-res-8-if-not-final*:
assumes
not-final: ¬ *ord-res-8-final* (*N*, *s*) **and**
invars: *ord-res-8-invars N s*
shows ∃ *s'*. *ord-res-8 N s s'*
⟨*proof*⟩

lemma *ord-res-8-safe-state-if-invars*:
fixes *N s*
assumes *invars*: *ord-res-8-invars N s*
shows *safe-state* (*constant-context ord-res-8*) *ord-res-8-final* (*N*, *s*)
⟨*proof*⟩

end

end
theory *ORD-RES-9*
imports
ORD-RES-8
begin

24 ORD-RES-9 (factorize when propagating)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-9* **where**
decide-neg:
¬ (∃ *C* |∈| *iefac* $\mathcal{F}$ |`| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*) ⟹
linorder-trm.is-least-in-fset {|A_2 |∈| *atms-of-clss* (*N* |∪| U_{er}).
∀ A_1 |∈| *trail-atms* Γ. $A_1 \prec_t A_2$|} *A* ⟹
¬ (∃ *C* |∈| *iefac* $\mathcal{F}$ |`| (*N* |∪| U_{er}). *clause-could-propagate* Γ *C* (*Pos A*)) ⟹
Γ' = (*Neg A*, *None*) # Γ ⟹
ord-res-9 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}$, Γ') |

propagate:
¬ (∃ *C* |∈| *iefac* $\mathcal{F}$ |`| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*) ⟹
linorder-trm.is-least-in-fset {|A_2 |∈| *atms-of-clss* (*N* |∪| U_{er}).
∀ A_1 |∈| *trail-atms* Γ. $A_1 \prec_t A_2$|} *A* ⟹
linorder-cls.is-least-in-fset {|*C* |∈| *iefac* $\mathcal{F}$ |`| (*N* |∪| U_{er}).
clause-could-propagate Γ *C* (*Pos A*)|} *C* ⟹
Γ' = (*Pos A*, *Some* (*efac C*)) # Γ ⟹
$\mathcal{F}'$ = (*if linorder-lit.is-greatest-in-mset C* (*Pos A*) *then* $\mathcal{F}$ *else finsert C* $\mathcal{F}$) ⟹
ord-res-9 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}'$, Γ') |

resolution:

linorder-cls.is-least-in-fset $\{|D\ |\in|\ \mathit{iefac}\ \mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).\ \mathit{trail\text{-}false\text{-}cls}\ \Gamma\ D|\}$ $D \Longrightarrow$
linorder-lit.is-maximal-in-mset D $(Neg\ A) \Longrightarrow$
map-of Γ $(Pos\ A) = Some\ (Some\ C) \Longrightarrow$
$U_{er}{}' = \mathit{finsert}\ (\mathit{eres}\ C\ D)\ U_{er} \Longrightarrow$
$\Gamma' = \mathit{dropWhile}\ (\lambda Ln.\ \forall K.$
linorder-lit.is-maximal-in-mset $(\mathit{eres}\ C\ D)\ K \longrightarrow \mathit{atm\text{-}of}\ K \preceq_t \mathit{atm\text{-}of}\ (\mathit{fst}\ Ln))\ \Gamma \Longrightarrow$
ord-res-9 $N\ (U_{er}, \mathcal{F}, \Gamma)\ (U_{er}{}', \mathcal{F}, \Gamma')$

lemma *right-unique-ord-res-9*:
fixes N :: *'f gclause fset*
shows *right-unique* (*ord-res-9* N)
⟨*proof*⟩

lemma *ord-res-9-is-one-or-two-ord-res-9-steps*:
fixes $N\ s\ s'$
assumes *step*: *ord-res-9* $N\ s\ s'$
shows *ord-res-8* $N\ s\ s' \lor$ (*ord-res-8* N *OO* *ord-res-8* N) $s\ s'$
⟨*proof*⟩

lemma *ord-res-9-preserves-invars*:
assumes
step: *ord-res-9* $N\ s\ s'$ **and**
invars: *ord-res-8-invars* $N\ s$
shows *ord-res-8-invars* $N\ s'$
⟨*proof*⟩

lemma *rtranclp-ord-res-9-preserves-ord-res-8-invars*:
assumes
step: (*ord-res-9* N)** $s\ s'$ **and**
invars: *ord-res-8-invars* $N\ s$
shows *ord-res-8-invars* $N\ s'$
⟨*proof*⟩

lemma *ex-ord-res-9-if-not-final*:
assumes
not-final: $\neg$ *ord-res-8-final* (N, s) **and**
invars: *ord-res-8-invars* $N\ s$
shows $\exists s'$. *ord-res-9* $N\ s\ s'$
⟨*proof*⟩

lemma *ord-res-9-safe-state-if-invars*:
fixes $N\ s$
assumes *invars*: *ord-res-8-invars* $N\ s$
shows *safe-state* (*constant-context* *ord-res-9*) *ord-res-8-final* (N, s)
⟨*proof*⟩

sublocale *ord-res-9-semantics*: *semantics* **where**

step = *constant-context ord-res-9* **and**
final = *ord-res-8-final*
⟨*proof*⟩

end

end
theory *ORD-RES-10*
imports *ORD-RES-8*
begin

25 ORD-RES-10 (propagate iff a conflict is produced)

type-synonym *'f ord-res-10-state* =
'f gclause fset ×*'f gclause fset* × *'f gclause fset* × (*'f gliteral* × *'f gclause option*) *list*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-10* **where**
decide-neg:
$\neg$ ($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*) ⟹
linorder-trm.is-least-in-fset {|A_2 |∈| *atms-of-clss* (*N* |∪| U_{er}).
$\forall A_1$ |∈| *trail-atms* Γ. $A_1 \prec_t A_2$|} *A* ⟹
$\neg$ ($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}). *clause-could-propagate* Γ *C* (*Pos A*)) ⟹
Γ' = (*Neg A*, *None*) # Γ ⟹
ord-res-10 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}$, Γ') |

decide-pos:
$\neg$ ($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*) ⟹
linorder-trm.is-least-in-fset {|A_2 |∈| *atms-of-clss* (*N* |∪| U_{er}).
$\forall A_1$ |∈| *trail-atms* Γ. $A_1 \prec_t A_2$|} *A* ⟹
linorder-cls.is-least-in-fset {|*C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}).
clause-could-propagate Γ *C* (*Pos A*)|} *C* ⟹
Γ' = (*Pos A*, *None*) # Γ ⟹
$\neg$ ($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}). *trail-false-cls* Γ' *C*) ⟹
$\mathcal{F}'$ = (*if linorder-lit.is-greatest-in-mset C* (*Pos A*) *then* $\mathcal{F}$ *else finsert C* $\mathcal{F}$) ⟹
ord-res-10 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}'$, Γ') |

propagate:
$\neg$ ($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*) ⟹
linorder-trm.is-least-in-fset {|A_2 |∈| *atms-of-clss* (*N* |∪| U_{er}).
$\forall A_1$ |∈| *trail-atms* Γ. $A_1 \prec_t A_2$|} *A* ⟹
linorder-cls.is-least-in-fset {|*C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}).
clause-could-propagate Γ *C* (*Pos A*)|} *C* ⟹
Γ' = (*Pos A*, *Some* (*efac C*)) # Γ ⟹
($\exists$ *C* |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er}). *trail-false-cls* Γ' *C*) ⟹

$\mathcal{F}' = $ (*if linorder-lit.is-greatest-in-mset C* (*Pos A*) *then* $\mathcal{F}$ *else finsert C* $\mathcal{F}$) $\Longrightarrow$
ord-res-10 N (U_{er}, $\mathcal{F}$, Γ) (U_{er}, $\mathcal{F}'$, Γ') |

resolution:
linorder-cls.is-least-in-fset {|*D* |∈| *iefac* $\mathcal{F}$ |‘| (*N* |∪| U_{er}). *trail-false-cls* Γ *D*|}
D $\Longrightarrow$
linorder-lit.is-maximal-in-mset D (*Neg A*) $\Longrightarrow$
map-of Γ (*Pos A*) = *Some* (*Some C*) $\Longrightarrow$
$U_{er}{}'$ = *finsert* (*eres C D*) U_{er} $\Longrightarrow$
Γ' = *dropWhile* (λ*Ln*. ∀ *K*.
linorder-lit.is-maximal-in-mset (*eres C D*) *K* ⟶ *atm-of K* $\preceq_t$ *atm-of* (*fst Ln*)) Γ $\Longrightarrow$
ord-res-10 N (U_{er}, $\mathcal{F}$, Γ) ($U_{er}{}'$, $\mathcal{F}$, Γ')

lemma *right-unique-ord-res-10*:
fixes *N* :: *'f gclause fset*
shows *right-unique* (*ord-res-10 N*)
⟨*proof*⟩

sublocale *ord-res-10-semantics*: *semantics* **where**
step = *constant-context ord-res-10* **and**
final = *ord-res-8-final*
⟨*proof*⟩

inductive *ord-res-10-invars* **for** *N* **where**
ord-res-10-invars N (U_{er}, $\mathcal{F}$, Γ) **if**
sorted-wrt (λ*x y*. *atm-of* (*fst y*) $\prec_t$ *atm-of* (*fst x*)) Γ **and**
∀ *Ln* ∈ *set* Γ. ∀ *C*. *snd Ln* = *Some C* ⟶ *linorder-lit.is-greatest-in-mset C* (*fst Ln*) **and**
linorder-trm.is-lower-fset (*trail-atms* Γ) (*atms-of-clss* (*N* |∪| U_{er})) **and**
∀ *Ln* Γ'. Γ = *Ln* # Γ' ⟶
(*snd Ln* ≠ *None* ⟷ (∃ *C* |∈| *iefac* $\mathcal{F}$ |‘| (*N* |∪| U_{er}). *trail-false-cls* Γ *C*)) ∧
(*snd Ln* ≠ *None* ⟶ *is-pos* (*fst Ln*)) ∧
(∀ *C*. *snd Ln* = *Some C* ⟶ *C* |∈| *iefac* $\mathcal{F}$ |‘| (*N* |∪| U_{er})) ∧
(∀ *C*. *snd Ln* = *Some C* ⟶ *clause-could-propagate* Γ' *C* (*fst Ln*)) ∧
(∀ *x* ∈ *set* Γ'. *snd x* = *None*) **and**
∀ Γ_1 *Ln* Γ_0. Γ = Γ_1 @ *Ln* # Γ_0 ⟶
snd Ln = *None* ⟶ ¬(∃ *C* |∈| *iefac* $\mathcal{F}$ |‘| (*N* |∪| U_{er}). *trail-false-cls* (*Ln* # Γ_0) *C*)

lemma *ord-res-10-preserves-invars*:
assumes
step: *ord-res-10 N s s'* **and**
invars: *ord-res-10-invars N s*
shows *ord-res-10-invars N s'*
⟨*proof*⟩

lemma *rtranclp-ord-res-10-preserves-invars*:

assumes
step: $(ord\text{-}res\text{-}10\ N)^{**}\ s\ s'$ **and**
invars: *ord-res-10-invars N s*
shows *ord-res-10-invars N s'*
⟨*proof*⟩

lemma *ex-ord-res-10-if-not-final*:
assumes
not-final: ¬ *ord-res-8-final* (*N*, *s*) **and**
invars: *ord-res-10-invars N s*
shows ∃ *s'*. *ord-res-10 N s s'*
⟨*proof*⟩

lemma *ord-res-10-safe-state-if-invars*:
fixes *N s*
assumes *invars*: *ord-res-10-invars N s*
shows *safe-state* (*constant-context ord-res-10*) *ord-res-8-final* (*N*, *s*)
⟨*proof*⟩

end

end
theory *ORD-RES-11*
imports *ORD-RES-10*
begin

26 ORD-RES-11 (SCL strategy)

type-synonym *'f ord-res-11-state* =
'f gclause fset × *'f gclause fset* × *'f gclause fset* × (*'f gliteral* × *'f gclause option*) *list* ×
'f gclause option

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

lemma
fixes $N\ U_{er}\ \mathcal{F}\ \Gamma\ A$
assumes
no-false-cls: ¬ (∃ C |∈| *iefac* $\mathcal{F}$ |\`| (N |∪| U_{er}). *trail-false-cls* Γ C) **and**
A-least: *linorder-trm.is-least-in-fset* {| A_2 |∈| *atms-of-clss* (N |∪| U_{er}).
∀ A_1 |∈| *trail-atms* Γ. $A_1 \prec_t A_2$ |} A **and**
C-least: *linorder-cls.is-least-in-fset* {| C |∈| *iefac* $\mathcal{F}$ |\`| (N |∪| U_{er}).
clause-could-propagate Γ C (*Pos* A) |} C
defines
Γ' ≡ (*Pos* A, *None*) # Γ **and**
$\mathcal{F}'$ ≡ (*if linorder-lit.is-greatest-in-mset* C (*Pos* A) *then* $\mathcal{F}$ *else finsert* C $\mathcal{F}$)
shows
(∃ C |∈| *iefac* $\mathcal{F}$ |\`| (N |∪| U_{er}). *trail-false-cls* Γ' C) ⟷
(∃ C |∈| *iefac* $\mathcal{F}'$ |\`| (N |∪| U_{er}). *trail-false-cls* Γ' C)

⟨*proof*⟩

inductive *ord-res-11* **where**

decide-neg:
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ C) $\Longrightarrow$
linorder-trm.is-least-in-fset {|A_2 |$\in$| *atms-of-clss* (N |$\cup$| U_{er}).
$\forall$ A_1 |$\in$| *trail-atms* Γ. $A_1 \prec_t A_2$|} A $\Longrightarrow$
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *clause-could-propagate* Γ C (*Pos* A)) $\Longrightarrow$
Γ' = (*Neg* A, *None*) # Γ $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, Γ, *None*) (U_{er}, $\mathcal{F}$, Γ', *None*) |

decide-pos:
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ C) $\Longrightarrow$
linorder-trm.is-least-in-fset {|A_2 |$\in$| *atms-of-clss* (N |$\cup$| U_{er}).
$\forall$ A_1 |$\in$| *trail-atms* Γ. $A_1 \prec_t A_2$|} A $\Longrightarrow$
linorder-cls.is-least-in-fset {|C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}).
clause-could-propagate Γ C (*Pos* A)|} C $\Longrightarrow$
Γ' = (*Pos* A, *None*) # Γ $\Longrightarrow$
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ' C) $\Longrightarrow$
$\mathcal{F}'$ = (*if linorder-lit.is-greatest-in-mset* C (*Pos* A) *then* $\mathcal{F}$ *else finsert* C $\mathcal{F}$) $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, Γ, *None*) (U_{er}, $\mathcal{F}'$, Γ', *None*) |

propagate:
$\neg$ ($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ C) $\Longrightarrow$
linorder-trm.is-least-in-fset {|A_2 |$\in$| *atms-of-clss* (N |$\cup$| U_{er}).
$\forall$ A_1 |$\in$| *trail-atms* Γ. $A_1 \prec_t A_2$|} A $\Longrightarrow$
linorder-cls.is-least-in-fset {|C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}).
clause-could-propagate Γ C (*Pos* A)|} C $\Longrightarrow$
Γ' = (*Pos* A, *Some* (*efac* C)) # Γ $\Longrightarrow$
($\exists$ C |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ' C) $\Longrightarrow$
$\mathcal{F}'$ = (*if linorder-lit.is-greatest-in-mset* C (*Pos* A) *then* $\mathcal{F}$ *else finsert* C $\mathcal{F}$) $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, Γ, *None*) (U_{er}, $\mathcal{F}'$, Γ', *None*) |

conflict:
linorder-cls.is-least-in-fset {|D |$\in$| *iefac* $\mathcal{F}$ |'| (N |$\cup$| U_{er}). *trail-false-cls* Γ D|}
D $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, Γ, *None*) (U_{er}, $\mathcal{F}$, Γ, *Some* D) |

skip: $-$ L $\notin$# C $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, (L, *n*) # Γ, *Some* C) (U_{er}, $\mathcal{F}$, Γ, *Some* C) |

resolution:
Γ = (L, *Some* D) # Γ' $\Longrightarrow$ $-$ L $\in$# C $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, Γ, *Some* C) (U_{er}, $\mathcal{F}$, Γ, *Some* ((C $-$ {#$-$ L#}) + (D
$-$ {#L#}))) |

backtrack:
Γ = (L, *None*) # Γ' $\Longrightarrow$ $-$ L $\in$# C $\Longrightarrow$
ord-res-11 N (U_{er}, $\mathcal{F}$, Γ, *Some* C) (*finsert* C U_{er}, $\mathcal{F}$, Γ', *None*)

lemma *right-unique-ord-res-11*:
fixes N :: *'f gclause fset*
shows *right-unique* (*ord-res-11 N*)
⟨*proof*⟩

inductive *ord-res-11-final* :: *'f ord-res-11-state* ⇒ *bool* **where**
model-found:
¬ (∃ A |∈| *atms-of-clss* (N |∪| U_{er}). A |∉| *trail-atms* Γ) ⟹
¬ (∃ C |∈| *iefac* $\mathcal{F}$ |'| (N |∪| U_{er}). *trail-false-cls* Γ C) ⟹
ord-res-11-final (N, U_{er}, $\mathcal{F}$, Γ, *None*) |

contradiction-found:
ord-res-11-final (N, U_{er}, $\mathcal{F}$, [], *Some* {#})

sublocale *ord-res-11-semantics*: *semantics* **where**
step = *constant-context ord-res-11* **and**
final = *ord-res-11-final*
⟨*proof*⟩

inductive *ord-res-11-invars* **where**
ord-res-11-invars N (U_{er}, $\mathcal{F}$, Γ, $\mathcal{C}$) **if**
ord-res-10-invars N (U_{er}, $\mathcal{F}$, Γ) **and**
{#} |∈| N |∪| U_{er} ⟶ Γ = [] **and**
∀ C. $\mathcal{C}$ = *Some* C ⟶ *atms-of-cls* C |⊆| *atms-of-clss* (N |∪| U_{er}) **and**
∀ C. $\mathcal{C}$ = *Some* C ⟶ *trail-false-cls* Γ C **and**
atms-of-clss U_{er} |⊆| *atms-of-clss* N **and**
∀ C |∈| $\mathcal{F}$. ∃ L. *is-pos* L ∧ *linorder-lit.is-maximal-in-mset* C L

lemma *ord-res-11-invars-initial-state*: *ord-res-11-invars* N ({||}, {||}, [], *None*)
⟨*proof*⟩

lemma *mempty-in-fimage-iefac*[*simp*]: {#} |∈| *iefac* $\mathcal{F}$ |'| N ⟷ {#} |∈| N
⟨*proof*⟩

lemma *ord-res-11-preserves-invars*:
assumes
step: *ord-res-11* N s s' **and**
invars: *ord-res-11-invars* N s
shows *ord-res-11-invars* N s'
⟨*proof*⟩

lemma *rtranclp-ord-res-11-preserves-invars*:
assumes
step: (*ord-res-11* N)** s s' **and**
invars: *ord-res-11-invars* N s

shows *ord-res-11-invars N s'*
⟨*proof*⟩

lemma *tranclp-ord-res-11-preserves-invars*:
assumes
step: $(ord\text{-}res\text{-}11\ N)^{++}\ s\ s'$ **and**
invars: *ord-res-11-invars N s*
shows *ord-res-11-invars N s'*
⟨*proof*⟩

lemma *ex-ord-res-11-if-not-final*:
assumes
not-final: ¬ *ord-res-11-final* (*N*, *s*) **and**
invars: *ord-res-11-invars N s*
shows ∃ *s'*. *ord-res-11 N s s'*
⟨*proof*⟩

lemma *ord-res-11-safe-state-if-invars*:
fixes *N s*
assumes *invars*: *ord-res-11-invars N s*
shows *safe-state* (*constant-context ord-res-11*) *ord-res-11-final* (*N*, *s*)
⟨*proof*⟩

lemma *rtrancl-ord-res-11-all-resolution-steps*:
assumes *C-max-lit*: *ord-res.is-strictly-maximal-lit K C*
shows $(ord\text{-}res\text{-}11\ N)^{**}$ (*U*, $\mathcal{F}$, (*K*, *Some C*) # Γ, *Some D*) (*U*, $\mathcal{F}$, (*K*, *Some C*) # Γ, *Some* (*eres C D*))
⟨*proof*⟩

lemma *rtrancl-ord-res-11-all-skip-steps*:
$(ord\text{-}res\text{-}11\ N)^{**}$ (*U*, $\mathcal{F}$, Γ, *Some C*) (*U*, $\mathcal{F}$, *dropWhile* (λ*Ln*. − *fst Ln* ∉# *C*) Γ, *Some C*)
⟨*proof*⟩

end

end
theory *Simulation-SCLFOL-ORDRES*
imports
Background
ORD-RES
ORD-RES-1
ORD-RES-2
ORD-RES-3
ORD-RES-4
ORD-RES-5
ORD-RES-6
ORD-RES-7
ORD-RES-8

ORD-RES-9
ORD-RES-10
ORD-RES-11
Clause-Could-Propagate
begin

27 ORD-RES-1 (deterministic)

type-synonym *'f ord-res-1-state* = *'f gclause fset*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

sublocale *backward-simulation-with-measuring-function* **where**
step1 = *ord-res* **and**
step2 = *ord-res-1* **and**
final1 = *ord-res-final* **and**
final2 = *ord-res-1-final* **and**
order = λ- -. *False* **and**
match = (=) **and**
measure = λ-. ()
⟨*proof*⟩

end

28 ORD-RES-2 (full factorization)

type-synonym *'f ord-res-2-state* = *'f gclause fset* × *'f gclause fset* × *'f gclause fset*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

fun *ord-res-1-matches-ord-res-2*
:: *'f ord-res-1-state* ⇒ - ⇒ *bool* **where**
ord-res-1-matches-ord-res-2 S1 $(N, (U_r, U_{ef})) \longleftrightarrow (\exists U_f.$
S1 $= N$ |∪| U_r |∪| U_{ef} |∪| $U_f \wedge$
$(\forall C_f$ |∈| $U_f. \exists C$ |∈| N |∪| U_r |∪| U_{ef}. *ord-res.ground-factoring*$^{++}$ $C\ C_f \wedge$
$C_f \neq$ *efac* $C_f \wedge$
(*efac* C_f |∈| $U_{ef} \vee$ *is-least-false-clause* (N |∪| U_r |∪| U_{ef}) C)))

lemma *ord-res-1-matches-ord-res-2-simps'*:
ord-res-1-matches-ord-res-2 S1 $(N, (U_r, U_{ef})) \longleftrightarrow$
$(\exists U_f.$ *S1* $= N$ |∪| U_r |∪| U_{ef} |∪| $U_f \wedge$
$(\forall C_f$ |∈| $U_f. C_f \neq$ *efac* $C_f \wedge (\exists C$ |∈| N |∪| U_r |∪| U_{ef}. *ord-res.ground-factoring*$^{++}$
$C\ C_f \wedge$
(*efac* C_f |∈| $U_{ef} \vee$ *is-least-false-clause* (N |∪| U_r |∪| U_{ef}) C))))
⟨*proof*⟩

lemma *ord-res-1-matches-ord-res-2-simps''*:

ord-res-1-matches-ord-res-2 S1 $(N, (U_r, U_{ef})) \longleftrightarrow$
$(\exists\, U_f.\ S1 = N \mid\cup\mid U_r \mid\cup\mid U_{ef} \mid\cup\mid U_f \wedge$
$(\forall\, C_f \mid\in\mid U_f.\ C_f \neq efac\ C_f \wedge (\exists\, C \mid\in\mid N \mid\cup\mid U_r \mid\cup\mid U_{ef}.$ *ord-res.ground-factoring*$^{++}$ $C\ C_f \wedge$
$(efac\ C \mid\in\mid U_{ef} \vee$ *is-least-false-clause* $(N \mid\cup\mid U_r \mid\cup\mid U_{ef})\ C))))$
⟨*proof*⟩

lemma *ord-res-1-final-iff-ord-res-2-final*:
assumes *match*: *ord-res-1-matches-ord-res-2* S_1 S_2
shows *ord-res-1-final* $S_1 \longleftrightarrow$ *ord-res-2-final* S_2
⟨*proof*⟩

lemma *safe-states-if-ord-res-1-matches-ord-res-2*:
assumes *match*: *ord-res-1-matches-ord-res-2* S_1 S_2
shows *safe-state ord-res-1 ord-res-1-final* $S_1 \wedge$ *safe-state ord-res-2-step ord-res-2-final* S_2
⟨*proof*⟩

definition *ord-res-1-measure* **where**
ord-res-1-measure s1 =
(*if* $\exists\, C.$ *is-least-false-clause s1 C then*
The (*is-least-false-clause s1*)
else
{#})

lemma *forward-simulation*:
assumes *match*: *ord-res-1-matches-ord-res-2 s1 s2* **and**
step1: *ord-res-1 s1 s1'*
shows ($\exists$ *s2'*. *ord-res-2-step*$^{++}$ *s2 s2'* $\wedge$ *ord-res-1-matches-ord-res-2 s1' s2'*) $\vee$
ord-res-1-matches-ord-res-2 s1' s2 $\wedge$ *ord-res-1-measure s1'* ⊂# *ord-res-1-measure s1*
⟨*proof*⟩

theorem *bisimulation-ord-res-1-ord-res-2*:
defines *match* $\equiv \lambda i$ *s1 s2*. $i =$ *ord-res-1-measure s1* $\wedge$ *ord-res-1-matches-ord-res-2 s1 s2*
shows $\exists$(*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-1-state* ⇒ *'f ord-res-2-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation ord-res-1 ord-res-2-step ord-res-1-final ord-res-2-final $\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

29 ORD-RES-3 (full resolve)

type-synonym *'f ord-res-3-state* = *'f gclause fset* × *'f gclause fset* × *'f gclause fset*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-2-matches-ord-res-3* :: - ⇒ *'f ord-res-3-state* ⇒ *bool* **where**
  $(\forall C \mathrel{|\in|} U_{pr}.\ \exists D1 \mathrel{|\in|} N \mathrel{|\cup|} U_{er} \mathrel{|\cup|} U_{ef}.\ \exists D2 \mathrel{|\in|} N \mathrel{|\cup|} U_{er} \mathrel{|\cup|} U_{ef}.$
    $(\textit{ground-resolution } D1)^{++}\ D2\ C \wedge C \neq \textit{eres } D1\ D2 \wedge \textit{eres } D1\ D2 \mathrel{|\in|} U_{er})$
$\Longrightarrow$
  $\textit{ord-res-2-matches-ord-res-3 } (N, (U_{pr} \mathrel{|\cup|} U_{er}, U_{ef}))\ (N, (U_{er}, U_{ef}))$

lemma *ord-res-2-final-iff-ord-res-3-final*:
  assumes *match*: *ord-res-2-matches-ord-res-3* S_2 S_3
  shows *ord-res-2-final* S_2 ⟷ *ord-res-3-final* S_3
  ⟨*proof*⟩

definition *ord-res-2-measure* **where**
  ord-res-2-measure S1 =
    $(\textit{let } (N, (U_r, U_{ef})) = S1 \textit{ in}$
    $(\textit{if } \exists C.\ \textit{is-least-false-clause } (N \mathrel{|\cup|} U_r \mathrel{|\cup|} U_{ef})\ C \textit{ then}$
      $\textit{The } (\textit{is-least-false-clause } (N \mathrel{|\cup|} U_r \mathrel{|\cup|} U_{ef}))$
    else
      {#}))

definition *resolvent-at* **where**
  resolvent-at C D i = (*THE CD*. (*ground-resolution C* ^^ *i*) *D CD*)

lemma *resolvent-at-0*[*simp*]: *resolvent-at C D 0* = *D*
  ⟨*proof*⟩

lemma *resolvent-at-less-cls-resolvent-at*:
  assumes *reso-at*: (*ground-resolution C* ^^ *n*) *D CD*
  assumes $i < j$ **and** $j \leq n$
  shows *resolvent-at C D j* $\prec_c$ *resolvent-at C D i*
⟨*proof*⟩

lemma
  assumes *reso-at*: (*ground-resolution C* ^^ *n*) *D CD* **and** $i < n$
  shows
    left-premisse-lt-resolvent-at: $C \prec_c$ *resolvent-at C D i* **and**
    max-lit-resolvent-at:
      ord-res.is-maximal-lit L D ⟹ *ord-res.is-maximal-lit L* (*resolvent-at C D i*)
and
    nex-pos-strictly-max-lit-in-resolvent-at:
      ∄*L*. *is-pos L* ∧ *ord-res.is-strictly-maximal-lit L* (*resolvent-at C D i*) **and**
    ground-resolution-resolvent-at-resolvent-at-Suc:
      ground-resolution C (*resolvent-at C D i*) (*resolvent-at C D* (*Suc i*)) **and**
    relpowp-to-resolvent-at: (*ground-resolution C* ^^ *i*) *D* (*resolvent-at C D i*)
⟨*proof*⟩

definition *resolvents-upto* **where**
  resolvents-upto C D n = *resolvent-at C D* |'| *fset-upto* (*Suc 0*) *n*

lemma *resolvents-upto-0*[*simp*]:
resolvents-upto C D 0 = {||}
⟨*proof*⟩

lemma *resolvents-upto-Suc*[*simp*]:
resolvents-upto C D (*Suc n*) = *finsert* (*resolvent-at C D* (*Suc n*)) (*resolvents-upto C D n*)
⟨*proof*⟩

lemma *resolvent-at-fmember-resolvents-upto*:
assumes $k \neq 0$
shows *resolvent-at C D k* |∈| *resolvents-upto C D k*
⟨*proof*⟩

lemma *backward-simulation-2-to-3*:
fixes *match measure less*
defines *match* ≡ *ord-res-2-matches-ord-res-3*
assumes
match: *match S2 S3* **and**
step2: *ord-res-3-step S3 S3'*
shows (∃ *S2'*. *ord-res-2-step*$^{++}$ *S2 S2'* ∧ *match S2' S3'*)
⟨*proof*⟩

lemma *safe-states-if-ord-res-2-matches-ord-res-3*:
assumes *match*: *ord-res-2-matches-ord-res-3* S_2 S_3
shows
safe-state ord-res-2-step ord-res-2-final S_2
safe-state ord-res-3-step ord-res-3-final S_3
⟨*proof*⟩

theorem *bisimulation-ord-res-2-ord-res-3*:
defines *match* ≡ λ- *S2 S3*. *ord-res-2-matches-ord-res-3 S2 S3*
shows ∃(*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-2-state* ⇒ *'f ord-res-3-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation ord-res-2-step ord-res-3-step ord-res-2-final ord-res-3-final $\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

30 ORD-RES-4 (implicit factorization)

type-synonym *'f ord-res-4-state* = *'f gclause fset* × *'f gclause fset* × *'f gclause fset*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-3-matches-ord-res-4* :: *'f ord-res-3-state* ⇒ *'f ord-res-4-state* ⇒

bool **where**
$\mathcal{F}$ |⊆| N |∪| U_{er} ⟹ U_{ef} = *iefac* $\mathcal{F}$ |‘| {|C |∈| N |∪| U_{er}. *iefac* $\mathcal{F}$ $C \neq C$|} ⟹
ord-res-3-matches-ord-res-4 (N, (U_{er}, U_{ef})) (N, U_{er}, $\mathcal{F}$)

lemma *ord-res-3-final-iff-ord-res-4-final*:
assumes *match*: *ord-res-3-matches-ord-res-4 S3 S4*
shows *ord-res-3-final S3* ⟷ *ord-res-4-final S4*
⟨*proof*⟩

lemma *forward-simulation-between-3-and-4*:
assumes
match: *ord-res-3-matches-ord-res-4 S3 S4* **and**
step: *ord-res-3-step S3 S3'*
shows (∃ *S4'*. *ord-res-4-step*$^{++}$ *S4 S4'* ∧ *ord-res-3-matches-ord-res-4 S3' S4'*)
⟨*proof*⟩

theorem *bisimulation-ord-res-3-ord-res-4*:
defines *match* ≡ λ- *S3 S4*. *ord-res-3-matches-ord-res-4 S3 S4*
shows ∃ (*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-3-state* ⇒ *'f ord-res-4-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation ord-res-3-step ord-res-4-step ord-res-3-final ord-res-4-final $\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

31 ORD-RES-5 (explicit model construction)

type-synonym *'f ord-res-5-state* = *'f gclause fset* × *'f gclause fset* × *'f gclause fset* ×
(*'f gterm* ⇒ *'f gclause option*) × *'f gclause option*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-4-matches-ord-res-5* :: *'f ord-res-4-state* ⇒ *'f ord-res-5-state* ⇒ *bool* **where**
ord-res-5-invars N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) ⟹
(∀ C. $\mathcal{C}$ = *Some* C ⟷ *is-least-false-clause* (*iefac* $\mathcal{F}$ |‘| (N |∪| U_{er})) C) ⟹
ord-res-4-matches-ord-res-5 (N, U_{er}, $\mathcal{F}$) (N, U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$)

lemma *ord-res-4-final-iff-ord-res-5-final*:
assumes *match*: *ord-res-4-matches-ord-res-5 S4 S5*
shows *ord-res-4-final S4* ⟷ *ord-res-5-final S5*
⟨*proof*⟩

lemma *forward-simulation-between-4-and-5*:
fixes *S4 S4' S5*
assumes *match*: *ord-res-4-matches-ord-res-5 S4 S5* **and** *step*: *ord-res-4-step S4 S4'*

shows $\exists$ *S5'*. *ord-res-5-step*$^{++}$ *S5 S5'* $\wedge$ *ord-res-4-matches-ord-res-5 S4' S5'*
⟨*proof*⟩

theorem *bisimulation-ord-res-4-ord-res-5*:
defines *match* $\equiv$ λ-. *ord-res-4-matches-ord-res-5*
shows $\exists$(*MATCH* :: *nat* $\times$ *nat* $\Rightarrow$ *'f ord-res-4-state* $\Rightarrow$ *'f ord-res-5-state* $\Rightarrow$ *bool*) $\mathcal{R}$.
bisimulation ord-res-4-step ord-res-5-step ord-res-4-final ord-res-5-final $\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

32 ORD-RES-6 (model backjump)

type-synonym *'f ord-res-6-state* = *'f gclause fset* $\times$ *'f gclause fset* $\times$ *'f gclause fset* $\times$
(*'f gterm* $\Rightarrow$ *'f gclause option*) $\times$ *'f gclause option*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-5-matches-ord-res-6* :: *'f ord-res-5-state* $\Rightarrow$ *'f ord-res-6-state* $\Rightarrow$ *bool* **where**
ord-res-5-invars N $(U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C}) \Longrightarrow$
ord-res-5-matches-ord-res-6 $(N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$ $(N, U_{er}, \mathcal{F}, \mathcal{M}, \mathcal{C})$

lemma *ord-res-5-final-iff-ord-res-6-final*:
fixes *i S5 S6*
assumes *match*: *ord-res-5-matches-ord-res-6 S5 S6*
shows *ord-res-5-final S5* $\longleftrightarrow$ *ord-res-6-final S6*
⟨*proof*⟩

lemma *backward-simulation-between-5-and-6*:
fixes *S5 S6 S6'*
assumes *match*: *ord-res-5-matches-ord-res-6 S5 S6* **and** *step*: *ord-res-6-step S6 S6'*
shows $\exists$ *S5'*. *ord-res-5-step*$^{++}$ *S5 S5'* $\wedge$ *ord-res-5-matches-ord-res-6 S5' S6'*
⟨*proof*⟩

theorem *bisimulation-ord-res-5-ord-res-6*:
defines *match* $\equiv$ λ-. *ord-res-5-matches-ord-res-6*
shows $\exists$(*MATCH* :: *nat* $\times$ *nat* $\Rightarrow$ *'f ord-res-5-state* $\Rightarrow$ *'f ord-res-6-state* $\Rightarrow$ *bool*) $\mathcal{R}$.
bisimulation ord-res-5-step ord-res-6-step ord-res-5-final ord-res-6-final $\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

33 ORD-RES-7 (clause-guided literal trail construction)

type-synonym *'f ord-res-7-state* =
'f gclause fset × *'f gclause fset* × *'f gclause fset* × (*'f gliteral* × *'f gclause option*) *list* ×
'f gclause option

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-6-matches-ord-res-7* ::
'f gterm fset ⇒ *'f ord-res-6-state* ⇒ *'f ord-res-7-state* ⇒ *bool* **where**
ord-res-5-invars N (U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) ⟹
ord-res-7-invars N (U_{er}, $\mathcal{F}$, Γ, $\mathcal{C}$) ⟹
(∀ *A C*. $\mathcal{M}$ *A* = *Some C* ⟷ *map-of* Γ (*Pos A*) = *Some* (*Some C*)) ⟹
(∀ *A*. $\mathcal{M}$ *A* = *None* ⟷ *map-of* Γ (*Neg A*) ≠ *None* ∨ *A* |∉| *trail-atms* Γ) ⟹
i = *atms-of-clss* (*N* |∪| U_{er}) − *trail-atms* Γ ⟹
ord-res-6-matches-ord-res-7 i (*N*, U_{er}, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{C}$) (*N*, U_{er}, $\mathcal{F}$, Γ, $\mathcal{C}$)

lemma *ord-res-6-final-iff-ord-res-7-final*:
fixes *i S6 S7*
assumes *match*: *ord-res-6-matches-ord-res-7 i S6 S7*
shows *ord-res-6-final S6* ⟷ *ord-res-7-final S7*
⟨*proof*⟩

lemma *backward-simulation-between-6-and-7*:
fixes *i S6 S7 S7'*
assumes *match*: *ord-res-6-matches-ord-res-7 i S6 S7* **and** *step*: *constant-context ord-res-7 S7 S7'*
shows
(∃ *i' S6'*. $\textit{ord-res-6-step}^{++}$ *S6 S6'* ∧ *ord-res-6-matches-ord-res-7 i' S6' S7'*) ∨
(∃ *i'*. *ord-res-6-matches-ord-res-7 i' S6 S7'* ∧ *i'* |⊂| *i*)
⟨*proof*⟩

theorem *bisimulation-ord-res-6-ord-res-7*:
defines *match* ≡ *ord-res-6-matches-ord-res-7*
shows ∃ (*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-6-state* ⇒ *'f ord-res-7-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation ord-res-6-step (*constant-context ord-res-7*) *ord-res-6-final ord-res-7-final*
$\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

34 ORD-RES-8 (atom-guided literal trail construction)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-8-can-decide-neg* **where**
$\neg$ *trail-false-cls* Γ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
linorder-trm.is-least-in-fset $\{|A\ |\in|$ *atms-of-clss* $(N\ |\cup|\ U_{er})$.
$A \prec_t$ *atm-of* $L \wedge A\ |\notin|$ *trail-atms* $\Gamma|\}$ $A \Longrightarrow$
ord-res-8-can-decide-neg N U_{er} $\mathcal{F}$ Γ C

inductive *ord-res-8-can-skip-undefined-neg* **where**
$\neg$ *trail-false-cls* Γ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
$\neg(\exists A\ |\in|$ *atms-of-clss* $(N\ |\cup|\ U_{er})$. $A \prec_t$ *atm-of* $L \wedge A\ |\notin|$ *trail-atms* $\Gamma) \Longrightarrow$
$\neg$ *trail-defined-lit* Γ $L \Longrightarrow$
is-neg $L \Longrightarrow$
ord-res-8-can-skip-undefined-neg N U_{er} $\mathcal{F}$ Γ C

inductive *ord-res-8-can-skip-undefined-pos-ultimate* **where**
$\neg$ *trail-false-cls* Γ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
$\neg(\exists A\ |\in|$ *atms-of-clss* $(N\ |\cup|\ U_{er})$. $A \prec_t$ *atm-of* $L \wedge A\ |\notin|$ *trail-atms* $\Gamma) \Longrightarrow$
$\neg$ *trail-defined-lit* Γ $L \Longrightarrow$
is-pos $L \Longrightarrow$
$\neg$ *trail-false-cls* Γ $\{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$
$\neg(\exists D\ |\in|$ *iefac* $\mathcal{F}$ |'| $(N\ |\cup|\ U_{er})$. $C \prec_c D) \Longrightarrow$
ord-res-8-can-skip-undefined-pos-ultimate N U_{er} $\mathcal{F}$ Γ C

inductive *ord-res-8-can-produce* **where**
$\neg$ *trail-false-cls* Γ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
$\neg$ $(\exists A\ |\in|$ *atms-of-clss* $(N\ |\cup|\ U_{er})$. $A \prec_t$ *atm-of* $L \wedge A\ |\notin|$ *trail-atms* $\Gamma) \Longrightarrow$
$\neg$ *trail-defined-lit* Γ $L \Longrightarrow$
is-pos $L \Longrightarrow$
trail-false-cls Γ $\{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$
linorder-lit.is-greatest-in-mset C $L \Longrightarrow$
ord-res-8-can-produce N U_{er} $\mathcal{F}$ Γ C

inductive *ord-res-8-can-factorize* **where**
$\neg$ *trail-false-cls* Γ $C \Longrightarrow$
linorder-lit.is-maximal-in-mset C $L \Longrightarrow$
$\neg$ $(\exists A\ |\in|$ *atms-of-clss* $(N\ |\cup|\ U_{er})$. $A \prec_t$ *atm-of* $L \wedge A\ |\notin|$ *trail-atms* $\Gamma) \Longrightarrow$
$\neg$ *trail-defined-lit* Γ $L \Longrightarrow$
is-pos $L \Longrightarrow$
trail-false-cls Γ $\{\#K \in\#\ C.\ K \neq L\#\} \Longrightarrow$
$\neg$ *linorder-lit.is-greatest-in-mset* C $L \Longrightarrow$
ord-res-8-can-factorize N U_{er} $\mathcal{F}$ Γ C

definition *is-least-nonskipped-clause* **where**
is-least-nonskipped-clause $N\ U_{er}\ \mathcal{F}\ \Gamma\ C \longleftrightarrow$
linorder-cls.is-least-in-fset $\{|C\ |\in|$ *iefac* $\mathcal{F}\ |`|\ (N\ |\cup|\ U_{er}).$
trail-false-cls $\Gamma\ C\ \vee$
ord-res-8-can-decide-neg $N\ U_{er}\ \mathcal{F}\ \Gamma\ C\ \vee$
ord-res-8-can-skip-undefined-neg $N\ U_{er}\ \mathcal{F}\ \Gamma\ C\ \vee$
ord-res-8-can-skip-undefined-pos-ultimate $N\ U_{er}\ \mathcal{F}\ \Gamma\ C\ \vee$
ord-res-8-can-produce $N\ U_{er}\ \mathcal{F}\ \Gamma\ C\ \vee$
ord-res-8-can-factorize $N\ U_{er}\ \mathcal{F}\ \Gamma\ C|\}\ C$

lemma *is-least-nonskipped-clause-mempty*:
assumes *bot-in*: $\{\#\}\ |\in|$ *iefac* $\mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})$
shows *is-least-nonskipped-clause* $N\ U_{er}\ \mathcal{F}\ \Gamma\ \{\#\}$
⟨*proof*⟩

lemma *nex-is-least-nonskipped-clause-if*:
assumes
no-undef-atom: $\neg\ (\exists\ A|\in|$*atms-of-clss* $(N\ |\cup|\ U_{er}).\ A\ |\notin|$ *trail-atms* $\Gamma)$ **and**
no-false-clause: $\neg$ *fBex* (*iefac* $\mathcal{F}\ |`|\ (N\ |\cup|\ U_{er})$) (*trail-false-cls* Γ)
shows $\nexists\ C.$ *is-least-nonskipped-clause* $N\ U_{er}\ \mathcal{F}\ \Gamma\ C$
⟨*proof*⟩

lemma *MAGIC5*:
assumes *invars*: *ord-res-7-invars* $N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$ **and**
no-more-steps: $\nexists\mathcal{C}'.$ *ord-res-7* $N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}')$
shows $(\forall\ C.\ \mathcal{C} =$ *Some* $C \longleftrightarrow$ *is-least-nonskipped-clause* $N\ U_{er}\ \mathcal{F}\ \Gamma\ C)$
⟨*proof*⟩

lemma *MAGIC6*:
assumes *invars*: *ord-res-7-invars* $N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})$
shows $\exists\mathcal{C}'.$ (*ord-res-7* $N)^{**}\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C})\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}')\ \wedge$
$(\nexists\mathcal{C}''.$ *ord-res-7* $N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}')\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}''))$
⟨*proof*⟩

inductive *ord-res-7-matches-ord-res-8* :: *'f ord-res-7-state* ⇒ *'f ord-res-8-state* ⇒ *bool* **where**
ord-res-7-invars $N\ (U_{er}, \mathcal{F}, \Gamma, \mathcal{C}) \Longrightarrow$
ord-res-8-invars $N\ (U_{er}, \mathcal{F}, \Gamma) \Longrightarrow$
$(\forall\ C.\ \mathcal{C} =$ *Some* $C \longleftrightarrow$ *is-least-nonskipped-clause* $N\ U_{er}\ \mathcal{F}\ \Gamma\ C) \Longrightarrow$
ord-res-7-matches-ord-res-8 $(N,\ U_{er}, \mathcal{F}, \Gamma, \mathcal{C})\ (N,\ U_{er}, \mathcal{F}, \Gamma)$

lemma *ord-res-7-final-iff-ord-res-8-final*:
fixes *S7 S8*
assumes *match*: *ord-res-7-matches-ord-res-8 S7 S8*
shows *ord-res-7-final S7* ⟷ *ord-res-8-final S8*
⟨*proof*⟩

lemma *backward-simulation-between-7-and-8*:

fixes *i S7 S8 S8'*
assumes *match*: *ord-res-7-matches-ord-res-8 S7 S8* **and** *step*: *constant-context ord-res-8 S8 S8'*
shows $\exists$ *S7'*. (*constant-context ord-res-7*)$^{++}$ *S7 S7'* $\wedge$ *ord-res-7-matches-ord-res-8 S7' S8'*
⟨*proof*⟩

theorem *bisimulation-ord-res-7-ord-res-8*:
defines *match* $\equiv$ λ-. *ord-res-7-matches-ord-res-8*
shows $\exists$ (*MATCH* :: *nat* $\times$ *nat* $\Rightarrow$ *'f ord-res-7-state* $\Rightarrow$ *'f ord-res-8-state* $\Rightarrow$ *bool*) $\mathcal{R}$.
bisimulation
(*constant-context ord-res-7*) (*constant-context ord-res-8*)
ord-res-7-final ord-res-8-final
$\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

35 ORD-RES-9 (factorize when propagating)

type-synonym *'f ord-res-9-state* =
'f gclause fset $\times$ *'f gclause fset* $\times$ *'f gclause fset* $\times$ (*'f gliteral* $\times$ *'f gclause option*) *list*

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-8-matches-ord-res-9* :: *'f ord-res-8-state* $\Rightarrow$ *'f ord-res-9-state* $\Rightarrow$ *bool* **where**
ord-res-8-invars N $(U_{er}, \mathcal{F}, \Gamma) \Longrightarrow$
ord-res-8-matches-ord-res-9 $(N, U_{er}, \mathcal{F}, \Gamma)$ $(N, U_{er}, \mathcal{F}, \Gamma)$

lemma *ord-res-8-final-iff-ord-res-9-final*:
fixes *S8 S9*
assumes *match*: *ord-res-8-matches-ord-res-9 S8 S9*
shows *ord-res-8-final S8* $\longleftrightarrow$ *ord-res-8-final S9*
⟨*proof*⟩

lemma *backward-simulation-between-8-and-9*:
fixes *S8 S9 S9'*
assumes *match*: *ord-res-8-matches-ord-res-9 S8 S9* **and** *step*: *constant-context ord-res-9 S9 S9'*
shows $\exists$ *S8'*. (*constant-context ord-res-8*)$^{++}$ *S8 S8'* $\wedge$ *ord-res-8-matches-ord-res-9 S8' S9'*
⟨*proof*⟩

theorem *bisimulation-ord-res-8-ord-res-9*:
defines *match* $\equiv$ λ-. *ord-res-8-matches-ord-res-9*

shows $\exists$ (*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-8-state* ⇒ *'f ord-res-9-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation
(*constant-context ord-res-8*) (*constant-context ord-res-9*)
ord-res-8-final ord-res-8-final
$\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

36 ORD-RES-10 (propagate iff a conflict is produced)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-9-matches-ord-res-10* :: *'f ord-res-9-state* ⇒ *'f ord-res-10-state* ⇒ *bool* **where**
ord-res-8-invars N $(U_{er}, \mathcal{F}, \Gamma_9) \Longrightarrow$
ord-res-10-invars N $(U_{er}, \mathcal{F}, \Gamma_{10}) \Longrightarrow$
list-all2 ($\lambda x\ y.$ *fst x* = *fst y*) $\Gamma_9\ \Gamma_{10} \Longrightarrow$
list-all2 ($\lambda x\ y.$ *snd y* ≠ *None* ⟶ *x* = *y*) $\Gamma_9\ \Gamma_{10} \Longrightarrow$
ord-res-9-matches-ord-res-10 $(N, U_{er}, \mathcal{F}, \Gamma_9)$ $(N, U_{er}, \mathcal{F}, \Gamma_{10})$

lemma *ord-res-9-final-iff-ord-res-10-final*:
fixes *S9 S10*
assumes *match*: *ord-res-9-matches-ord-res-10 S9 S10*
shows *ord-res-8-final S9* ⟷ *ord-res-8-final S10*
⟨*proof*⟩

lemma *backward-simulation-between-9-and-10*:
fixes *S9 S10 S10'*
assumes
match: *ord-res-9-matches-ord-res-10 S9 S10* **and**
step: *constant-context ord-res-10 S10 S10'*
shows $\exists$ *S9'*. (*constant-context ord-res-9*)$^{++}$ *S9 S9'* ∧ *ord-res-9-matches-ord-res-10 S9' S10'*
⟨*proof*⟩

theorem *bisimulation-ord-res-9-ord-res-10*:
defines *match* ≡ λ-. *ord-res-9-matches-ord-res-10*
shows $\exists$ (*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-8-state* ⇒ *'f ord-res-9-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation
(*constant-context ord-res-9*) (*constant-context ord-res-10*)
ord-res-8-final ord-res-8-final
$\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

37 ORD-RES-11 (SCL strategy)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

inductive *ord-res-10-matches-ord-res-11* :: *'f ord-res-10-state* ⇒ *'f ord-res-11-state* ⇒ *bool* **where**
ord-res-10-invars N $(U_{er10}, \mathcal{F}, \Gamma) \Longrightarrow$
ord-res-11-invars N $(U_{er11}, \mathcal{F}, \Gamma, \mathcal{C}) \Longrightarrow$
$U_{er11} = U_{er10} - \{|\{\#\}|\} \Longrightarrow$
if $\{\#\}$ |∈| *iefac* $\mathcal{F}$ |'| (*N* |∪| U_{er10}) *then* $\Gamma = [] \wedge \mathcal{C} =$ *Some* $\{\#\}$ *else* $\mathcal{C} =$ *None* $\Longrightarrow$
ord-res-10-matches-ord-res-11 $(N, U_{er10}, \mathcal{F}, \Gamma)$ $(N, U_{er11}, \mathcal{F}, \Gamma, \mathcal{C})$

lemma *ord-res-10-final-iff-ord-res-11-final*:
fixes *S10 S11*
assumes *match*: *ord-res-10-matches-ord-res-11 S10 S11*
shows *ord-res-8-final S10* ⟷ *ord-res-11-final S11*
⟨*proof*⟩

lemma *forward-simulation-between-10-and-11*:
fixes *S10 S11 S10'*
assumes
match: *ord-res-10-matches-ord-res-11 S10 S11* **and**
step: *constant-context ord-res-10 S10 S10'*
shows ∃ *S11'*. (*constant-context ord-res-11*)$^{++}$ *S11 S11'* ∧ *ord-res-10-matches-ord-res-11 S10' S11'*
⟨*proof*⟩

theorem *bisimulation-ord-res-10-ord-res-11*:
defines *match* ≡ λ-. *ord-res-10-matches-ord-res-11*
shows ∃(*MATCH* :: *nat* × *nat* ⇒ *'f ord-res-10-state* ⇒ *'f ord-res-11-state* ⇒ *bool*) $\mathcal{R}$.
bisimulation
(*constant-context ord-res-10*) (*constant-context ord-res-11*)
ord-res-8-final ord-res-11-final
$\mathcal{R}$ *MATCH*

⟨*proof*⟩

end

lemma *forward-simulation-composition*:
assumes
forward-simulation step1 step2 final1 final2 order1 match1
forward-simulation step2 step3 final2 final3 order2 match2

defines $\mathcal{R} \equiv \lambda i\ i'.$ *lex-prodp order2*$^{++}$ *order1* (*prod.swap i*) (*prod.swap i'*)
shows *forward-simulation step1 step3 final1 final3* $\mathcal{R}$ (*rel-comp match1 match2*)
⟨*proof*⟩

For AFP-devel, delete ⟦*forward-simulation ?step1.0 ?step2.0 ?final1.0 ?final2.0 ?order1.0 ?match1.0*; *forward-simulation ?step2.0 ?step3.0 ?final2.0 ?final3.0 ?order2.0 ?match2.0*⟧ ⟹ *forward-simulation ?step1.0 ?step3.0 ?final1.0 ?final3.0* ($\lambda i\ i'$. *Well-founded.lex-prodp ?order2.0*$^{++}$ *?order1.0* (*prod.swap i*) (*prod.swap i'*)) (*rel-comp ?match1.0 ?match2.0*) as it is available in *VeriComp.Simulation*.

type-synonym *bisim-index-1-2* = *nat* × *nat*
type-synonym *bisim-index-1-3* = *bisim-index-1-2* × (*nat* × *nat*)
type-synonym *bisim-index-1-4* = *bisim-index-1-3* × (*nat* × *nat*)
type-synonym *bisim-index-1-5* = *bisim-index-1-4* × (*nat* × *nat*)
type-synonym *bisim-index-1-6* = *bisim-index-1-5* × (*nat* × *nat*)
type-synonym *bisim-index-1-7* = *bisim-index-1-6* × (*nat* × *nat*)
type-synonym *bisim-index-1-8* = *bisim-index-1-7* × (*nat* × *nat*)
type-synonym *bisim-index-1-9* = *bisim-index-1-8* × (*nat* × *nat*)
type-synonym *bisim-index-1-10* = *bisim-index-1-9* × (*nat* × *nat*)
type-synonym *bisim-index-1-11* = *bisim-index-1-10* × (*nat* × *nat*)

context *simulation-SCLFOL-ground-ordered-resolution* **begin**

theorem *forward-simulation-ord-res-1-ord-res-11*:
obtains
MATCH :: *bisim-index-1-11* ⇒ *'f ord-res-1-state* ⇒ *'f ord-res-11-state* ⇒ *bool*
and
$\mathcal{R}$ :: *bisim-index-1-11* ⇒ *bisim-index-1-11* ⇒ *bool*
where
forward-simulation
ord-res-1 (*constant-context ord-res-11*)
ord-res-1-final ord-res-11-final
$\mathcal{R}$ *MATCH*
⟨*proof*⟩

theorem *backward-simulation-ord-res-1-ord-res-11*:
obtains
MATCH :: *bisim-index-1-11* ⇒ *'f ord-res-1-state* ⇒ *'f ord-res-11-state* ⇒ *bool*
and
$\mathcal{R}$ :: *bisim-index-1-11* ⇒ *bisim-index-1-11* ⇒ *bool*
where
backward-simulation
ord-res-1 (*constant-context ord-res-11*)
ord-res-1-final ord-res-11-final
$\mathcal{R}$ *MATCH*
⟨*proof*⟩

38 ORD-RES-11 is a regular SCL strategy

definition *gtrailelem-of-trailelem* **where**
gtrailelem-of-trailelem ≡ λ(*L*, *opt*).
(*lit-of-glit L*, *map-option* (λ*C*. (*cls-of-gcls* {#*K* ∈# *C*. *K* ≠ *L*#}, *lit-of-glit L*, *Var*)) *opt*)

fun *state-of-gstate* :: - ⇒ (′*f*, ′*v*) *SCL-FOL.state* **where**
state-of-gstate (U_G, -, Γ_G, $\mathcal{C}_G$) =
(*let*
Γ = *map gtrailelem-of-trailelem* Γ_G;
U = *cls-of-gcls* |′| U_G;
$\mathcal{C}$ = *map-option* (λC_G. (*cls-of-gcls* C_G, *Var*)) $\mathcal{C}_G$
in (Γ, *U*, $\mathcal{C}$))

lemma *fst-case-prod-simp*: *fst* (*case p of* (*x*, *y*) ⇒ (*f x*, *g x y*)) = *f* (*fst p*)
⟨*proof*⟩

lemma *trail-false-cls-nonground-iff-trail-false-cls-ground*:
fixes Γ_G **and** D_G :: ′*f gclause*
fixes Γ :: (′*f*, ′*v*) *SCL-FOL.trail* **and** *D* :: (′*f*, ′*v*) *term clause*
defines Γ ≡ *map gtrailelem-of-trailelem* Γ_G **and** *D* ≡ *cls-of-gcls* D_G
shows *trail-false-cls* Γ *D* ⟷ *trail-false-cls* Γ_G D_G
⟨*proof*⟩

theorem *ord-res-11-is-strategy-for-regular-scl*:
fixes
N_G :: ′*f gclause fset* **and**
N :: (′*f*, ′*v*) *term clause fset* **and**
β_G :: ′*f gterm* **and**
β :: (′*f*, ′*v*) *term* **and**
S_G $S_G{}'$ :: ′*f gclause fset* × ′*f gclause fset* × (′*f gliteral* × ′*f gclause option*) *list* × ′*f gclause option* **and**
S *S*′ :: (′*f*, ′*v*) *SCL-FOL.state*
defines
N ≡ *cls-of-gcls* |′| N_G **and**
β ≡ *term-of-gterm* β_G **and**
S ≡ *state-of-gstate* S_G **and**
S′ ≡ *state-of-gstate* $S_G{}'$
assumes
ball-le-β_G: ∀ A_G |∈| *atms-of-clss* N_G. $A_G \preceq_t \beta_G$ **and**
run: (*ord-res-11* N_G)** ({||}, {||}, [], *None*) S_G **and**
step: *ord-res-11* N_G S_G $S_G{}'$
shows
scl-fol.regular-scl N β *S* *S*′
⟨*proof*⟩

end

lemma *wfp-on-antimono-stronger*:
fixes
A :: $'a$ *set* **and** B :: $'b$ *set* **and**
f :: $'a \Rightarrow 'b$ **and**
R :: $'b \Rightarrow 'b \Rightarrow$ *bool* **and** Q :: $'a \Rightarrow 'a \Rightarrow$ *bool*
assumes
wf: *wfp-on* B R **and**
sub: f ‘ $A \subseteq B$ **and**
mono: $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow Q\ x\ y \Longrightarrow R\ (f\ x)\ (f\ y)$
shows *wfp-on* A Q
⟨*proof*⟩

For AFP-devel, delete ⟦*wfp-on ?B ?R*; *?f* ‘ *?A* ⊆ *?B*; $\bigwedge x\ y.$ ⟦$x \in$ *?A*; $y \in$ *?A*; *?Q x y*⟧ $\Longrightarrow$ *?R* (*?f x*) (*?f y*)⟧ $\Longrightarrow$ *wfp-on ?A ?Q* as it is available in *HOL.Wellfounded*.

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

For AFP-devel, delete ⟦*scl-fol-calculus ?renaming-vars ?less-B*; $\bigwedge S\ S'.$ ⟦*?strategy*** *?strategy-init* S; *?strategy* $S\ S'$⟧ $\Longrightarrow$ *scl-fol-calculus.regular-scl ?less-B ?N ?*β (*?proj* S) (*?proj* S'); *?proj ?strategy-init* = *initial-state*⟧ $\Longrightarrow$ *wfp-on* $\{S.$ *?strategy*** *?strategy-init* $S\}$ *?strategy*$^{-1-1}$ as it is available in *Simple-Clause-Learning.Termination*.

corollary (**in** *simulation-SCLFOL-ground-ordered-resolution*) *ord-res-11-termination*:
fixes N :: $'f$ *gclause fset*
shows *wfp-on* $\{S.$ (*ord-res-11* N)** ({||}, {||}, [], *None*) $S\}$ (*ord-res-11* N)$^{-1-1}$
⟨*proof*⟩

corollary (**in** *scl-fol-calculus*) *static-non-subsumption-projectable-strategy*:
fixes *strategy* **and** *strategy-init* **and** *proj*
assumes
run: *strategy*** *strategy-init* S **and**
step: *backtrack* $N\ \beta$ (*proj* S) S' **and**
strategy-restricts-regular-scl:
$\bigwedge S\ S'.$ *strategy*** *strategy-init* $S \Longrightarrow$ *strategy* $S\ S' \Longrightarrow$ *regular-scl* $N\ \beta$ (*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*⟩

For AFP-devel, delete ⟦*scl-fol-calculus ?renaming-vars ?less-B*; *?strategy*** *?strategy-init ?S*; *scl-fol-calculus.backtrack ?N ?*β (*?proj ?S*) *?S*$'$; $\bigwedge S\ S'$. ⟦*?strategy*** *?strategy-init S*; *?strategy S S*$'$⟧ $\Longrightarrow$ *scl-fol-calculus.regular-scl ?less-B ?N ?*β (*?proj S*) (*?proj S*$'$); *?proj ?strategy-init* = *initial-state*⟧ $\Longrightarrow \exists\, C\ \gamma.$ *state-conflict* (*?proj ?S*) = *Some* $(C, \gamma) \wedge \neg (\exists\, D|\!\in\!|$*?N* $|\!\cup\!|$ *state-learned* (*?proj ?S*). *subsumes D C*) as it is available in *Simple-Clause-Learning.Non-Redundancy*.

corollary (**in** *simulation-SCLFOL-ground-ordered-resolution*) *ord-res-11-non-subsumption*:
fixes N_G :: $'f$ *gclause fset* **and** s :: - × - × - × -
defines
$\beta \equiv$ (*THE A. linorder-trm.is-greatest-in-fset* (*atms-of-clss* N_G) A)
assumes
run: (*ord-res-11* N_G)** ({||}, {||}, [], *None*) s **and**
step: *scl-fol.backtrack* (*cls-of-gcls* |'| N_G) (*term-of-gterm* β) (*state-of-gstate* s) s'
shows $\exists\, U_{er}\ \mathcal{F}\ \Gamma\ D.\ s = (U_{er}, \mathcal{F}, \Gamma,$ *Some* $D) \wedge \neg (\exists\, C\ |\!\in\!|\ N_G\ |\!\cup\!|\ U_{er}.\ C \subseteq\!\#\ D)$
⟨*proof*⟩

end

References

[1] M. Bromberger, C. Jain, and C. Weidenbach. SCL(FOL) can simulate non-redundant superposition clause learning. In B. Pientka and C. Tinelli, editors, *Automated Deduction – CADE 29*, pages 134–152, Cham, 2023. Springer Nature Switzerland.