A Modular Formalization of Superposition

Martin Desharnais Balazs Toth

March 17, 2025

Abstract

Superposition is an efficient proof calculus for reasoning about first-order logic with equality that is implemented in many automatic theorem provers. It works by saturating the given set of clauses and is refutationally complete, meaning that if the set is inconsistent, the saturation will contain a contradiction. In this formalization, we restructured the completeness proof to cleanly separate the ground (i.e., variable-free) and nonground aspects. We relied on the IsaFoR library for first-order terms and on the Isabelle saturation framework. A paper describing this formalization was published at the 15th International Conference on Interactive Theorem Proving (ITP 2024) [1].

Contents

theory *Ground-Critical-Pairs*
imports *First-Order-Clause.Term-Rewrite-System*
begin

definition *ground-critical-pairs* :: *'f gterm rel* ⇒ *'f gterm rel* **where**
ground-critical-pairs R = {(*ctxt*⟨r_2⟩$_G$, r_1) | *ctxt l* r_1 r_2. (*ctxt*⟨*l*⟩$_G$, r_1) ∈ *R* ∧ (*l*, r_2) ∈ *R*}

locale *ground-critical-pair-theorem* =
fixes *f-type* :: *'f itself*
assumes *ground-critical-pair-theorem*:
⋀*R* :: *'f gterm rel*.
WCR (*rewrite-inside-gctxt R*) ⟷ *ground-critical-pairs R* ⊆ (*rewrite-inside-gctxt R*)$^{\downarrow}$

end
theory *Ground-Superposition*
imports
Ground-Critical-Pairs
First-Order-Clause.Selection-Function
First-Order-Clause.Ground-Order
First-Order-Clause.Ground-Clause
begin

1 Superposition Calculus

locale *ground-superposition-calculus* =
ground-order-with-equality **where** *less*$_t$ = *less*$_t$ +
selection-function select +
ground-critical-pair-theorem TYPE('*f*)
for
less$_t$ :: *'f gterm* ⇒ *'f gterm* ⇒ *bool* **and**
select :: *'f gatom clause* ⇒ *'f gatom clause*
begin

1.1 Ground Rules

inductive *superposition* ::
'f gatom clause ⇒ *'f gatom clause* ⇒ *'f gatom clause* ⇒ *bool*
where
superpositionI:
E = *add-mset* L_E *E'* ⟹
D = *add-mset* L_D *D'* ⟹
D ≺$_c$ *E* ⟹
$\mathcal{P}$ ∈ {*Pos*, *Neg*} ⟹
L_E = $\mathcal{P}$ (*Upair* *κ*⟨*t*⟩$_G$ *u*) ⟹
L_D = *t* ≈ *t'* ⟹
u ≺$_t$ *κ*⟨*t*⟩$_G$ ⟹
t' ≺$_t$ *t* ⟹
($\mathcal{P}$ = *Pos* ∧ *select E* = {#} ∧ *is-strictly-maximal* L_E *E*) ∨
($\mathcal{P}$ = *Neg* ∧ (*select E* = {#} ∧ *is-maximal* L_E *E* ∨ *is-maximal* L_E (*select E*)))
⟹
select D = {#} ⟹

is-strictly-maximal L_D $D \Longrightarrow$
$C =$ *add-mset* $(\mathcal{P}$ $(Upair$ $\kappa\langle t' \rangle_G$ $u))$ $(E' + D') \Longrightarrow$
superposition D E C

inductive *eq-resolution* :: *'f gatom clause* ⇒ *'f gatom clause* ⇒ *bool* **where**
eq-resolutionI:
$D =$ *add-mset* L $D' \Longrightarrow$
$L = t$ $!\approx$ $t \Longrightarrow$
select $D = \{\#\} \wedge$ *is-maximal* L $D \vee$ *is-maximal* L $(select$ $D) \Longrightarrow$
$C = D' \Longrightarrow$
eq-resolution D C

inductive *eq-factoring* :: *'f gatom clause* ⇒ *'f gatom clause* ⇒ *bool* **where**
eq-factoringI:
$D =$ *add-mset* L_1 $(add\text{-}mset$ L_2 $D') \Longrightarrow$
$L_1 = t \approx t' \Longrightarrow$
$L_2 = t \approx t'' \Longrightarrow$
select $D = \{\#\} \Longrightarrow$
is-maximal L_1 $D \Longrightarrow$
$t' \prec_t t \Longrightarrow$
$C =$ *add-mset* $(t' !\approx t'')$ $(add\text{-}mset$ $(t \approx t'')$ $D') \Longrightarrow$
eq-factoring D C

abbreviation *eq-resolution-inferences* **where**
eq-resolution-inferences ≡ $\{Infer$ $[D]$ $C \mid D$ $C.$ *eq-resolution* D $C\}$

abbreviation *eq-factoring-inferences* **where**
eq-factoring-inferences ≡ $\{Infer$ $[D]$ $C \mid D$ $C.$ *eq-factoring* D $C\}$

abbreviation *superposition-inferences* **where**
superposition-inferences ≡ $\{Infer$ $[D,$ $E]$ $C \mid D$ E $C.$ *superposition* D E $C\}$

1.1.1 Alternative Specification of the Superposition Rule

inductive *superposition'* ::
'f gatom clause ⇒ *'f gatom clause* ⇒ *'f gatom clause* ⇒ *bool*
where
superposition'I:
$P_1 =$ *add-mset* L_1 $P_1' \Longrightarrow$
$P_2 =$ *add-mset* L_2 $P_2' \Longrightarrow$
$P_2 \prec_c P_1 \Longrightarrow$
$\mathcal{P} \in \{Pos,$ $Neg\} \Longrightarrow$
$L_1 = \mathcal{P}$ $(Upair$ $s\langle t \rangle_G$ $s') \Longrightarrow$
$L_2 = t \approx t' \Longrightarrow$
$s' \prec_t s\langle t \rangle_G \Longrightarrow$
$t' \prec_t t \Longrightarrow$
$(\mathcal{P} = Pos \longrightarrow$ *select* $P_1 = \{\#\} \wedge$ *is-strictly-maximal* L_1 $P_1) \Longrightarrow$
$(\mathcal{P} = Neg \longrightarrow (select$ $P_1 = \{\#\} \wedge$ *is-maximal* L_1 $P_1 \vee$ *is-maximal* L_1 $(select$ $P_1))) \Longrightarrow$

$\mathit{select}\ P_2 = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ L_2\ P_2 \Longrightarrow$
$C = \mathit{add\text{-}mset}\ (\mathcal{P}\ (\mathit{Upair}\ s\langle t'\rangle_G\ s'))\ (P_1' + P_2') \Longrightarrow$
$\mathit{superposition}'\ P_2\ P_1\ C$

lemma $\mathit{superposition}' = \mathit{superposition}$
⟨*proof*⟩

inductive *pos-superposition* ::
$'f\ \mathit{gatom\ clause} \Rightarrow 'f\ \mathit{gatom\ clause} \Rightarrow 'f\ \mathit{gatom\ clause} \Rightarrow \mathit{bool}$
where
pos-superpositionI:
$P_1 = \mathit{add\text{-}mset}\ L_1\ P_1' \Longrightarrow$
$P_2 = \mathit{add\text{-}mset}\ L_2\ P_2' \Longrightarrow$
$P_2 \prec_c P_1 \Longrightarrow$
$L_1 = s\langle t\rangle_G \approx s' \Longrightarrow$
$L_2 = t \approx t' \Longrightarrow$
$s' \prec_t s\langle t\rangle_G \Longrightarrow$
$t' \prec_t t \Longrightarrow$
$\mathit{select}\ P_1 = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ L_1\ P_1 \Longrightarrow$
$\mathit{select}\ P_2 = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ L_2\ P_2 \Longrightarrow$
$C = \mathit{add\text{-}mset}\ (s\langle t'\rangle_G \approx s')\ (P_1' + P_2') \Longrightarrow$
$\mathit{pos\text{-}superposition}\ P_2\ P_1\ C$

lemma *superposition-if-pos-superposition*:
assumes *step*: $\mathit{pos\text{-}superposition}\ P_2\ P_1\ C$
shows $\mathit{superposition}\ P_2\ P_1\ C$
⟨*proof*⟩

inductive *neg-superposition* ::
$'f\ \mathit{gatom\ clause} \Rightarrow 'f\ \mathit{gatom\ clause} \Rightarrow 'f\ \mathit{gatom\ clause} \Rightarrow \mathit{bool}$
where
neg-superpositionI:
$P_1 = \mathit{add\text{-}mset}\ L_1\ P_1' \Longrightarrow$
$P_2 = \mathit{add\text{-}mset}\ L_2\ P_2' \Longrightarrow$
$P_2 \prec_c P_1 \Longrightarrow$
$L_1 = s\langle t\rangle_G\ !\approx\ s' \Longrightarrow$
$L_2 = t \approx t' \Longrightarrow$
$s' \prec_t s\langle t\rangle_G \Longrightarrow$
$t' \prec_t t \Longrightarrow$
$\mathit{select}\ P_1 = \{\#\} \wedge \mathit{is\text{-}maximal}\ L_1\ P_1 \vee \mathit{is\text{-}maximal}\ L_1\ (\mathit{select}\ P_1) \Longrightarrow$
$\mathit{select}\ P_2 = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ L_2\ P_2 \Longrightarrow$
$C = \mathit{add\text{-}mset}\ (\mathit{Neg}\ (\mathit{Upair}\ s\langle t'\rangle_G\ s'))\ (P_1' + P_2') \Longrightarrow$
$\mathit{neg\text{-}superposition}\ P_2\ P_1\ C$

lemma *superposition-if-neg-superposition*:

assumes *neg-superposition* P_2 P_1 C
shows *superposition* P_2 P_1 C
⟨*proof*⟩

lemma *superposition-iff-pos-or-neg*:
superposition P_2 P_1 $C \longleftrightarrow$
pos-superposition P_2 P_1 $C \vee$ *neg-superposition* P_2 P_1 C
⟨*proof*⟩

1.2 Ground Layer

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

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

definition *G-entails* :: *'f gatom clause set* ⇒ *'f gatom clause set* ⇒ *bool* **where**
G-entails N_1 $N_2 \longleftrightarrow$ (∀ (*I* :: *'f gterm rel*). *refl I* ⟶ *trans I* ⟶ *sym I* ⟶
compatible-with-gctxt I ⟶ *upair* ‘ *I* ⊨*s* N_1 ⟶ *upair* ‘ *I* ⊨*s* N_2)

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

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

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

sublocale *consequence-relation* **where** *Bot* = *G-Bot* **and** *entails* = *G-entails*
⟨*proof*⟩

1.3 Redundancy Criterion

sublocale *calculus-with-finitary-standard-redundancy* **where**
Inf = *G-Inf* **and**
Bot = *G-Bot* **and**
entails = *G-entails* **and**
less = ($\prec_c$)

defines *GRed-I* = *Red-I* **and** *GRed-F* = *Red-F*
⟨*proof*⟩

lemma *redundant-infer-singleton*:
assumes ∃ *D*∈*N*. *G-entails* (*insert D* (*set* (*side-prems-of* ι))) {*concl-of* ι} ∧ *D* ≺$_c$ *main-prem-of* ι
shows *redundant-infer N* ι
⟨*proof*⟩

end

end
theory *Abstract-Rewriting-Extra*
imports
First-Order-Clause.Transitive-Closure-Extra
Abstract−Rewriting.Abstract-Rewriting
begin

lemma *mem-join-union-iff-mem-join-lhs*:
assumes
⋀*z*. (*x*, *z*) ∈ *A** ⟹ *z* ∉ *Domain B* **and**
⋀*z*. (*y*, *z*) ∈ *A** ⟹ *z* ∉ *Domain B*
shows (*x*, *y*) ∈ (*A* ∪ *B*)$^↓$ ⟷ (*x*, *y*) ∈ *A*$^↓$
⟨*proof*⟩

lemma *mem-join-union-iff-mem-join-rhs*:
assumes
⋀*z*. (*x*, *z*) ∈ *B** ⟹ *z* ∉ *Domain A* **and**
⋀*z*. (*y*, *z*) ∈ *B** ⟹ *z* ∉ *Domain A*
shows (*x*, *y*) ∈ (*A* ∪ *B*)$^↓$ ⟷ (*x*, *y*) ∈ *B*$^↓$
⟨*proof*⟩

lemma *refl-join*: *refl* (*r*$^↓$)
⟨*proof*⟩

lemma *trans-join*:
assumes *strongly-norm*: *SN r* **and** *confluent*: *WCR r*
shows *trans* (*r*$^↓$)
⟨*proof*⟩

end
theory *Relation-Extra*
imports *Main*
begin

lemma *partition-set-around-element*:
assumes *tot*: *totalp-on N R* **and** *x-in*: *x* ∈ *N*
shows *N* = {*y* ∈ *N*. *R y x*} ∪ {*x*} ∪ {*y* ∈ *N*. *R x y*}
⟨*proof*⟩

end
theory *Ground-Superposition-Completeness*
imports
Ground-Superposition

First-Order-Clause.HOL-Extra
Abstract-Rewriting-Extra
Relation-Extra
begin

1.4 Model Construction

context *ground-superposition-calculus* **begin**

function *epsilon* :: - ⇒ *'f gatom clause* ⇒ *'f gterm rel* **where**
epsilon N C = {(*s*, *t*)| *s t C'*.
C ∈ *N* ∧
C = *add-mset* (*Pos* (*Upair s t*)) *C'* ∧
select C = {#} ∧
is-strictly-maximal (*Pos* (*Upair s t*)) *C* ∧
t ≺*t* *s* ∧
(*let* R_C = (⋃ *D* ∈ {*D* ∈ *N*. *D* ≺*c* *C*}. *epsilon* {*E* ∈ *N*. *E* ⪯*c* *D*} *D*) *in*
¬ *upair* ‘ $(\textit{rewrite-inside-gctxt}\ R_C)^{\downarrow}$ ⊨ *C* ∧
¬ *upair* ‘ $(\textit{rewrite-inside-gctxt}\ (\textit{insert}\ (s,\ t)\ R_C))^{\downarrow}$ ⊨ *C'* ∧
s ∈ *NF* (*rewrite-inside-gctxt* R_C))}
⟨*proof*⟩

termination *epsilon*
⟨*proof*⟩

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

lemma *epsilon-filter-le-conv*: *epsilon* {*D* ∈ *N*. *D* ⪯*c* *C*} *C* = *epsilon N C*
⟨*proof*⟩

end

lemma (**in** *ground-superposition-calculus*) *epsilon-eq-empty-or-singleton*:
epsilon N C = {} ∨ (∃ *s t*. *epsilon N C* = {(*s*, *t*)})
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *card-epsilon-le-one*:
card (*epsilon N C*) ≤ *1*
⟨*proof*⟩

definition (**in** *ground-superposition-calculus*) *rewrite-sys* **where**
rewrite-sys N C ≡ (⋃ *D* ∈ {*D* ∈ *N*. *D* ≺*c* *C*}. *epsilon* {*E* ∈ *N*. *E* ⪯*c* *D*} *D*)

definition (**in** *ground-superposition-calculus*) *rewrite-sys'* **where**
rewrite-sys' $N \equiv (\bigcup C \in N.\ epsilon\ N\ C)$

lemma (**in** *ground-superposition-calculus*) *rewrite-sys-alt*: *rewrite-sys'* $\{D \in N.\ D \prec_c C\} =$ *rewrite-sys* $N\ C$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *mem-epsilonE*:
assumes *rule-in*: *rule* $\in$ *epsilon* $N\ C$
obtains $l\ r\ C'$ **where**
$C \in N$ **and**
rule $= (l,\ r)$ **and**
$C =$ *add-mset* (*Pos* (*Upair* $l\ r$)) C' **and**
select $C = \{\#\}$ **and**
is-strictly-maximal (*Pos* (*Upair* $l\ r$)) C **and**
$r \prec_t l$ **and**
$\neg$ *upair* ‘ (*rewrite-inside-gctxt* (*rewrite-sys* $N\ C$))$^{\downarrow} \models C$ **and**
$\neg$ *upair* ‘ (*rewrite-inside-gctxt* (*insert* $(l,\ r)$ (*rewrite-sys* $N\ C$)))$^{\downarrow} \models C'$ **and**
$l \in$ *NF* (*rewrite-inside-gctxt* (*rewrite-sys* $N\ C$))
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *mem-epsilon-iff*:
$(l,\ r) \in$ *epsilon* $N\ C \longleftrightarrow$
$(\exists C'.\ C \in N \wedge C =$ *add-mset* (*Pos* (*Upair* $l\ r$)) $C' \wedge$ *select* $C = \{\#\} \wedge$
is-strictly-maximal (*Pos* (*Upair* $l\ r$)) $C \wedge r \prec_t l \wedge$
$\neg$ *upair* ‘ (*rewrite-inside-gctxt* (*rewrite-sys'* $\{D \in N.\ D \prec_c C\}$))$^{\downarrow} \models C \wedge$
$\neg$ *upair* ‘ (*rewrite-inside-gctxt* (*insert* $(l,\ r)$ (*rewrite-sys'* $\{D \in N.\ D \prec_c C\}$)))$^{\downarrow} \models C' \wedge$
$l \in$ *NF* (*rewrite-inside-gctxt* (*rewrite-sys'* $\{D \in N.\ D \prec_c C\}$)))
(**is** *?LHS* $\longleftrightarrow$ *?RHS*)
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *rhs-lt-lhs-if-mem-rewrite-sys*:
assumes $(t1,\ t2) \in$ *rewrite-sys* $N\ C$
shows $t2 \prec_t t1$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *rhs-less-trm-lhs-if-mem-rewrite-inside-gctxt-rewrite-sys*:
assumes *rule-in*: $(t1,\ t2) \in$ *rewrite-inside-gctxt* (*rewrite-sys* $N\ C$)
shows $t2 \prec_t t1$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *rhs-lesseq-trm-lhs-if-mem-rtrancl-rewrite-inside-gctxt-rewrite-sys*:
assumes *rule-in*: $(t1,\ t2) \in$ (*rewrite-inside-gctxt* (*rewrite-sys* $N\ C$))*
shows $t2 \preceq_t t1$
⟨*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 (**in** *ground-superposition-calculus*) *rewrite-sys-subset-if-less-cls*:
$C \prec_c D \Longrightarrow$ *rewrite-sys* $N\ C \subseteq$ *rewrite-sys* $N\ D$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *mem-rewrite-sys-if-less-cls*:
assumes $D \in N$ **and** $D \prec_c C$ **and** $(u,\ v) \in$ *epsilon* $N\ D$
shows $(u,\ v) \in$ *rewrite-sys* $N\ C$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *less-trm-iff-less-cls-if-lhs-epsilon*:
assumes E_C: *epsilon* $N\ C = \{(s,\ t)\}$ **and** E_D: *epsilon* $N\ D = \{(u,\ v)\}$
shows $u \prec_t s \longleftrightarrow D \prec_c C$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *termination-rewrite-sys*: *wf* ((*rewrite-sys* $N\ C)^{-1}$)
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *termination-Union-rewrite-sys*:
wf $((\bigcup D \in N.$ *rewrite-sys* $N\ D)^{-1}$)
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *no-crit-pairs*:
$\{(t1,\ t2) \in$ *ground-critical-pairs* $(\bigcup$ (*epsilon N2* ‘ $N)).$ *t1* $\neq$ *t2*$\} = \{\}$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *WCR-Union-rewrite-sys*:
WCR (*rewrite-inside-gctxt* $(\bigcup D \in N.$ *epsilon N2* $D))$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*)
assumes
$D \preceq_c C$ **and**
E_C-*eq*: *epsilon* $N\ C = \{(s,\ t)\}$ **and**
L-in: $L \in\# D$ **and**
topmost-trms-of-L: *mset-uprod* (*atm-of L*) $= \{\#u,\ v\#\}$
shows
lesseq-trm-if-pos: *is-pos* $L \Longrightarrow u \preceq_t s$ **and**
less-trm-if-neg: *is-neg* $L \Longrightarrow u \prec_t s$
⟨*proof*⟩

lemma (**in** *ground-order*) *less-trm-const-lhs-if-mem-rewrite-inside-gctxt*:
fixes *t t1 t2 r*
assumes
rule-in: $(t1,\ t2) \in$ *rewrite-inside-gctxt* r **and**

ball-lt-lhs: $\bigwedge t1\ t2.\ (t1,\ t2) \in r \Longrightarrow t \prec_t t1$
shows $t \prec_t t1$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *split-Union-epsilon*:
assumes *D-in*: $D \in N$
shows $(\bigcup C \in N.\ epsilon\ N\ C) =$
$rewrite\text{-}sys\ N\ D \cup epsilon\ N\ D \cup (\bigcup C \in \{C \in N.\ D \prec_c C\}.\ epsilon\ N\ C)$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *split-Union-epsilon′*:
assumes *D-in*: $D \in N$
shows $(\bigcup C \in N.\ epsilon\ N\ C) = rewrite\text{-}sys\ N\ D \cup (\bigcup C \in \{C \in N.\ D \preceq_c C\}.\ epsilon\ N\ C)$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *split-rewrite-sys*:
assumes $C \in N$ **and** *D-in*: $D \in N$ **and** $D \prec_c C$
shows $rewrite\text{-}sys\ N\ C = rewrite\text{-}sys\ N\ D \cup (\bigcup C' \in \{C' \in N.\ D \preceq_c C' \wedge C' \prec_c C\}.\ epsilon\ N\ C')$
⟨*proof*⟩

lemma (**in** *ground-order*) *mem-join-union-iff-mem-join-lhs′*:
assumes
ball-R_1-rhs-lt-lhs: $\bigwedge t1\ t2.\ (t1,\ t2) \in R_1 \Longrightarrow t2 \prec_t t1$ **and**
ball-R_2-lt-lhs: $\bigwedge t1\ t2.\ (t1,\ t2) \in R_2 \Longrightarrow s \prec_t t1 \wedge t \prec_t t1$
shows $(s,\ t) \in (R_1 \cup R_2)^{\downarrow} \longleftrightarrow (s,\ t) \in R_1{}^{\downarrow}$
⟨*proof*⟩

lemma (**in** *ground-order*) *mem-join-union-iff-mem-join-rhs′*:
assumes
ball-R_1-rhs-lt-lhs: $\bigwedge t1\ t2.\ (t1,\ t2) \in R_2 \Longrightarrow t2 \prec_t t1$ **and**
ball-R_2-lt-lhs: $\bigwedge t1\ t2.\ (t1,\ t2) \in R_1 \Longrightarrow s \prec_t t1 \wedge t \prec_t t1$
shows $(s,\ t) \in (R_1 \cup R_2)^{\downarrow} \longleftrightarrow (s,\ t) \in R_2{}^{\downarrow}$
⟨*proof*⟩

lemma (**in** *ground-order*) *mem-join-union-iff-mem-join-lhs″*:
assumes
Range-R_1-lt-Domain-R_2: $\bigwedge t1\ t2.\ t1 \in Range\ R_1 \Longrightarrow t2 \in Domain\ R_2 \Longrightarrow t1 \prec_t t2$ **and**
s-lt-Domain-R_2: $\bigwedge t2.\ t2 \in Domain\ R_2 \Longrightarrow s \prec_t t2$ **and**
t-lt-Domain-R_2: $\bigwedge t2.\ t2 \in Domain\ R_2 \Longrightarrow t \prec_t t2$
shows $(s,\ t) \in (R_1 \cup R_2)^{\downarrow} \longleftrightarrow (s,\ t) \in R_1{}^{\downarrow}$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *lift-entailment-to-Union*:
fixes $N\ D$
defines $R_D \equiv rewrite\text{-}sys\ N\ D$
assumes

D-in: $D \in N$ **and**
R_D-entails-D: *upair* ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ R_D)^{\downarrow} \models D$
shows
upair ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\bigcup D \in N.\ \mathit{epsilon}\ N\ D))^{\downarrow} \models D$ **and**
$\bigwedge C.\ C \in N \Longrightarrow D \prec_c C \Longrightarrow \mathit{upair}$ ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\mathit{rewrite\text{-}sys}\ N\ C))^{\downarrow} \models D$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*)
assumes *productive*: $\mathit{epsilon}\ N\ C = \{(l,\ r)\}$
shows
true-cls-if-productive-epsilon:
upair ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\bigcup D \in N.\ \mathit{epsilon}\ N\ D))^{\downarrow} \models C$
$\bigwedge D.\ D \in N \Longrightarrow C \prec_c D \Longrightarrow \mathit{upair}$ ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\mathit{rewrite\text{-}sys}\ N\ D))^{\downarrow} \models C$ **and**
false-cls-if-productive-epsilon:
$\neg\ \mathit{upair}$ ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\bigcup D \in N.\ \mathit{epsilon}\ N\ D))^{\downarrow} \models C - \{\#\mathit{Pos}\ (\mathit{Upair}\ l\ r)\#\}$
$\bigwedge D.\ D \in N \Longrightarrow C \prec_c D \Longrightarrow \neg\ \mathit{upair}$ ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\mathit{rewrite\text{-}sys}\ N\ D))^{\downarrow} \models C - \{\#\mathit{Pos}\ (\mathit{Upair}\ l\ r)\#\}$
⟨*proof*⟩

lemma *from-neq-double-rtrancl-to-eqE*:
assumes $x \neq y$ **and** $(x,\ z) \in r^*$ **and** $(y,\ z) \in r^*$
obtains
w **where** $(x,\ w) \in r$ **and** $(w,\ z) \in r^*$ |
w **where** $(y,\ w) \in r$ **and** $(w,\ z) \in r^*$
⟨*proof*⟩

lemma *ex-step-if-joinable*:
assumes *asymp* R $(x,\ z) \in r^*$ **and** $(y,\ z) \in r^*$
shows
$R^{==}\ z\ y \Longrightarrow R\ y\ x \Longrightarrow \exists\, w.\ (x,\ w) \in r \wedge (w,\ z) \in r^*$
$R^{==}\ z\ x \Longrightarrow R\ x\ y \Longrightarrow \exists\, w.\ (y,\ w) \in r \wedge (w,\ z) \in r^*$
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *trans-join-rewrite-inside-gctxt-rewrite-sys*:
trans $((\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\mathit{rewrite\text{-}sys}\ N\ C))^{\downarrow})$
⟨*proof*⟩

lemma (**in** *ground-order*) *true-cls-insert-and-not-true-clsE*:
assumes
upair ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\mathit{insert}\ r\ R))^{\downarrow} \models C$ **and**
$\neg\ \mathit{upair}$ ‘ $(\mathit{rewrite\text{-}inside\text{-}gctxt}\ R)^{\downarrow} \models C$
obtains $t\ t'$ **where**
$\mathit{Pos}\ (\mathit{Upair}\ t\ t') \in\#\ C$ **and**
$t \prec_t t'$ **and**
$(t,\ t') \in (\mathit{rewrite\text{-}inside\text{-}gctxt}\ (\mathit{insert}\ r\ R))^{\downarrow}$ **and**
$(t,\ t') \notin (\mathit{rewrite\text{-}inside\text{-}gctxt}\ R)^{\downarrow}$

⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *model-preconstruction*:
fixes
N :: *'f gatom clause set* **and**
C :: *'f gatom clause*
defines
entails ≡ λ*E C*. *upair* ‘ (*rewrite-inside-gctxt E*)$^{\downarrow}$ ⊨ *C*
assumes *saturated N* **and** {#} ∉ *N* **and** *C-in*: *C* ∈ *N*
shows
epsilon N C = {} ⟷ *entails* (*rewrite-sys N C*) *C*
⋀*D*. *D* ∈ *N* ⟹ *C* ≺$_c$ *D* ⟹ *entails* (*rewrite-sys N D*) *C*
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *model-construction*:
fixes
N :: *'f gatom clause set* **and**
C :: *'f gatom clause*
defines
entails ≡ λ*E C*. *upair* ‘ (*rewrite-inside-gctxt E*)$^{\downarrow}$ ⊨ *C*
assumes *saturated N* **and** {#} ∉ *N* **and** *C-in*: *C* ∈ *N*
shows *entails* (⋃ *D* ∈ *N*. *epsilon N D*) *C*
⟨*proof*⟩

1.5 Static Refutational Completeness

lemma (**in** *ground-superposition-calculus*) *statically-complete*:
fixes *N* :: *'f gatom clause set*
assumes *saturated N* **and** *G-entails N* {{#}}
shows {#} ∈ *N*
⟨*proof*⟩

sublocale *ground-superposition-calculus* ⊆ *statically-complete-calculus* **where**
Bot = *G-Bot* **and**
Inf = *G-Inf* **and**
entails = *G-entails* **and**
Red-I = *Red-I* **and**
Red-F = *Red-F*
⟨*proof*⟩

end
theory *Ground-Superposition-Soundness*
imports *Ground-Superposition*
begin

lemma (**in** *ground-superposition-calculus*) *soundness-ground-superposition*:
assumes
step: *superposition P1 P2 C*
shows *G-entails* {*P1*, *P2*} {*C*}

⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *soundness-ground-eq-resolution*:
assumes *step*: *eq-resolution P C*
shows *G-entails* {*P*} {*C*}
⟨*proof*⟩

lemma (**in** *ground-superposition-calculus*) *soundness-ground-eq-factoring*:
assumes *step*: *eq-factoring P C*
shows *G-entails* {*P*} {*C*}
⟨*proof*⟩

sublocale *ground-superposition-calculus* ⊆ *sound-inference-system* **where**
Inf = *G-Inf* **and**
Bot = *G-Bot* **and**
entails = *G-entails*
⟨*proof*⟩

end
theory *Ground-Superposition-Welltypedness-Preservation*
imports
Ground-Superposition
First-Order-Clause.Ground-Typing
begin

context *ground-superposition-calculus*
begin

sublocale *ground-typing* **where** $\mathcal{F} = \mathcal{F} :: ('f, 'ty)$ *fun-types*
⟨*proof*⟩

context
fixes $\mathcal{F} :: ('f, 'ty)$ *fun-types*
begin

lemma *ground-superposition-preserves-typing*:
assumes
superposition D E C
clause.is-welltyped D
clause.is-welltyped E
shows *clause.is-welltyped C*
⟨*proof*⟩

lemma *ground-eq-resolution-preserves-typing*:
assumes *eq-resolution D C clause.is-welltyped D*
shows *clause.is-welltyped C*
⟨*proof*⟩

lemma *ground-eq-factoring-preserves-typing*:

assumes *eq-factoring D C*
shows *clause.is-welltyped D* $\longleftrightarrow$ *clause.is-welltyped C*
⟨*proof*⟩

end

end

end
theory *Superposition*
imports
First-Order-Clause.Nonground-Order
First-Order-Clause.Nonground-Selection-Function
First-Order-Clause.Nonground-Typing
First-Order-Clause.Typed-Tiebreakers
First-Order-Clause.Welltyped-IMGU

Ground-Superposition
begin

2 Nonground Layer

locale *superposition-calculus* =
nonground-inhabited-typing $\mathcal{F}$ +
nonground-equality-order $less_t$ +
nonground-selection-function select +
typed-tiebreakers tiebreakers +
ground-critical-pair-theorem $TYPE('f)$
for
select :: $('f, 'v ::$ *infinite*$)$ *select* **and**
$less_t$ :: $('f, 'v)$ *term* $\Rightarrow$ $('f, 'v)$ *term* $\Rightarrow$ *bool* **and**
$\mathcal{F}$ :: $('f, 'ty)$ *fun-types* **and**
tiebreakers :: $('f, 'v)$ *tiebreakers* +
assumes
types-ordLeq-variables: $|UNIV :: 'ty\ set| \leq_o |UNIV :: 'v\ set|$
begin

interpretation *term-order-notation*⟨*proof*⟩

inductive *eq-resolution* :: $('f, 'v, 'ty)$ *typed-clause* $\Rightarrow$ $('f, 'v, 'ty)$ *typed-clause* $\Rightarrow$ *bool* **where**
eq-resolutionI:
$D = add\text{-}mset\ l\ D' \Longrightarrow$
$l = t\ !\approx\ t' \Longrightarrow$
welltyped-imgu-on (*clause.vars D*) $\mathcal{V}$ *t t'* $\mu \Longrightarrow$
select $D = \{\#\} \wedge$ *is-maximal* $(l \cdot l\ \mu)\ (D \cdot \mu) \vee$ *is-maximal* $(l \cdot l\ \mu)$ (*select* $D \cdot \mu) \Longrightarrow$
$C = D' \cdot \mu \Longrightarrow$
eq-resolution $(D, \mathcal{V})\ (C, \mathcal{V})$

inductive *eq-factoring* :: ('*f*, '*v*, '*ty*) *typed-clause* ⇒ ('*f*, '*v*, '*ty*) *typed-clause* ⇒ *bool* **where**
eq-factoringI:
$D = \mathit{add\text{-}mset}\ l_1\ (\mathit{add\text{-}mset}\ l_2\ D') \Longrightarrow$
$l_1 = t_1 \approx {t_1}' \Longrightarrow$
$l_2 = t_2 \approx {t_2}' \Longrightarrow$
$\mathit{select}\ D = \{\#\} \Longrightarrow$
$\mathit{is\text{-}maximal}\ (l_1 \cdot l\ \mu)\ (D \cdot \mu) \Longrightarrow$
$\neg\ (t_1 \cdot t\ \mu \preceq_t {t_1}' \cdot t\ \mu) \Longrightarrow$
$\mathit{welltyped\text{-}imgu\text{-}on}\ (\mathit{clause.vars}\ D)\ \mathcal{V}\ t_1\ t_2\ \mu \Longrightarrow$
$C = \mathit{add\text{-}mset}\ (t_1 \approx {t_2}')\ (\mathit{add\text{-}mset}\ ({t_1}' \not\approx {t_2}')\ D') \cdot \mu \Longrightarrow$
$\mathit{eq\text{-}factoring}\ (D, \mathcal{V})\ (C, \mathcal{V})$

inductive *superposition* ::
('*f*, '*v*, '*ty*) *typed-clause* ⇒ ('*f*, '*v*, '*ty*) *typed-clause* ⇒ ('*f*, '*v*, '*ty*) *typed-clause* ⇒ *bool*
where
superpositionI:
$\mathit{infinite\text{-}variables\text{-}per\text{-}type}\ \mathcal{V}_1 \Longrightarrow$
$\mathit{infinite\text{-}variables\text{-}per\text{-}type}\ \mathcal{V}_2 \Longrightarrow$
$\mathit{term\text{-}subst.is\text{-}renaming}\ \varrho_1 \Longrightarrow$
$\mathit{term\text{-}subst.is\text{-}renaming}\ \varrho_2 \Longrightarrow$
$\mathit{clause.vars}\ (E \cdot \varrho_1) \cap \mathit{clause.vars}\ (D \cdot \varrho_2) = \{\} \Longrightarrow$
$E = \mathit{add\text{-}mset}\ l_1\ E' \Longrightarrow$
$D = \mathit{add\text{-}mset}\ l_2\ D' \Longrightarrow$
$\mathcal{P} \in \{\mathit{Pos}, \mathit{Neg}\} \Longrightarrow$
$l_1 = \mathcal{P}\ (\mathit{Upair}\ c_1\langle t_1\rangle\ {t_1}') \Longrightarrow$
$l_2 = t_2 \approx {t_2}' \Longrightarrow$
$\neg\ \mathit{is\text{-}Var}\ t_1 \Longrightarrow$
$\mathit{welltyped\text{-}imgu\text{-}on}\ (\mathit{clause.vars}\ (E \cdot \varrho_1) \cup \mathit{clause.vars}\ (D \cdot \varrho_2))\ \mathcal{V}_3\ (t_1 \cdot t\ \varrho_1)\ (t_2 \cdot t\ \varrho_2)\ \mu \Longrightarrow$
$\forall x \in \mathit{clause.vars}\ E.\ \mathcal{V}_1\ x = \mathcal{V}_3\ (\mathit{term.rename}\ \varrho_1\ x) \Longrightarrow$
$\forall x \in \mathit{clause.vars}\ D.\ \mathcal{V}_2\ x = \mathcal{V}_3\ (\mathit{term.rename}\ \varrho_2\ x) \Longrightarrow$
$\mathit{term.subst.is\text{-}welltyped\text{-}on}\ (\mathit{clause.vars}\ E)\ \mathcal{V}_1\ \varrho_1 \Longrightarrow$
$\mathit{term.subst.is\text{-}welltyped\text{-}on}\ (\mathit{clause.vars}\ D)\ \mathcal{V}_2\ \varrho_2 \Longrightarrow$
$(\bigwedge \tau\ \tau'.\ \mathit{typed}\ \mathcal{V}_2\ t_2\ \tau \Longrightarrow \mathit{typed}\ \mathcal{V}_2\ {t_2}'\ \tau' \Longrightarrow \tau = \tau') \Longrightarrow$
$\neg\ (E \cdot \varrho_1 \odot \mu \preceq_c D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$(\mathcal{P} = \mathit{Pos} \Longrightarrow \mathit{select}\ E = \{\#\}) \Longrightarrow$
$(\mathcal{P} = \mathit{Pos} \Longrightarrow \mathit{is\text{-}strictly\text{-}maximal}\ (l_1 \cdot l\ \varrho_1 \odot \mu)\ (E \cdot \varrho_1 \odot \mu)) \Longrightarrow$
$(\mathcal{P} = \mathit{Neg} \Longrightarrow \mathit{select}\ E = \{\#\} \Longrightarrow \mathit{is\text{-}maximal}\ (l_1 \cdot l\ \varrho_1 \odot \mu)\ (E \cdot \varrho_1 \odot \mu)) \Longrightarrow$
$(\mathcal{P} = \mathit{Neg} \Longrightarrow \mathit{select}\ E \neq \{\#\} \Longrightarrow \mathit{is\text{-}maximal}\ (l_1 \cdot l\ \varrho_1 \odot \mu)\ ((\mathit{select}\ E) \cdot \varrho_1 \odot \mu)) \Longrightarrow$
$\mathit{select}\ D = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ (l_2 \cdot l\ \varrho_2 \odot \mu)\ (D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$\neg\ (c_1\langle t_1\rangle \cdot t\ \varrho_1 \odot \mu \preceq_t {t_1}' \cdot t\ \varrho_1 \odot \mu) \Longrightarrow$
$\neg\ (t_2 \cdot t\ \varrho_2 \odot \mu \preceq_t {t_2}' \cdot t\ \varrho_2 \odot \mu) \Longrightarrow$
$C = \mathit{add\text{-}mset}\ (\mathcal{P}\ (\mathit{Upair}\ (c_1 \cdot t_c\ \varrho_1)\langle {t_2}' \cdot t\ \varrho_2\rangle\ ({t_1}' \cdot t\ \varrho_1)))\ (E' \cdot \varrho_1 + D' \cdot \varrho_2) \cdot \mu \Longrightarrow$

superposition $(D, \mathcal{V}_2)$ $(E, \mathcal{V}_1)$ $(C, \mathcal{V}_3)$

abbreviation *eq-factoring-inferences* **where**
eq-factoring-inferences $\equiv$ { *Infer* $[D]$ C | D C. *eq-factoring* D C }

abbreviation *eq-resolution-inferences* **where**
eq-resolution-inferences $\equiv$ { *Infer* $[D]$ C | D C. *eq-resolution* D C }

abbreviation *superposition-inferences* **where**
superposition-inferences $\equiv$ { *Infer* $[D, E]$ C | D E C. *superposition* D E C}

definition *inferences* :: $('f, 'v, 'ty)$ *typed-clause inference set* **where**
inferences $\equiv$ *superposition-inferences* $\cup$ *eq-resolution-inferences* $\cup$ *eq-factoring-inferences*

abbreviation $bottom_F$ :: $('f, 'v, 'ty)$ *typed-clause set* $(\bot_F)$ **where**
$bottom_F \equiv \{(\{\#\}, \mathcal{V}) \mid \mathcal{V}.$ *infinite-variables-per-type* $\mathcal{V}$ }

2.0.1 Alternative Specification of the Superposition Rule

inductive *superposition'* ::
$('f, 'v, 'ty)$ *typed-clause* $\Rightarrow$ $('f, 'v, 'ty)$ *typed-clause* $\Rightarrow$ $('f, 'v, 'ty)$ *typed-clause* $\Rightarrow$ *bool*
where
superposition'I:
infinite-variables-per-type $\mathcal{V}_1 \Longrightarrow$
infinite-variables-per-type $\mathcal{V}_2 \Longrightarrow$
term-subst.is-renaming $\varrho_1 \Longrightarrow$
term-subst.is-renaming $\varrho_2 \Longrightarrow$
clause.vars $(E \cdot \varrho_1) \cap$ *clause.vars* $(D \cdot \varrho_2) = \{\} \Longrightarrow$
$E =$ *add-mset* l_1 $E' \Longrightarrow$
$D =$ *add-mset* l_2 $D' \Longrightarrow$
$\mathcal{P} \in \{Pos, Neg\} \Longrightarrow$
$l_1 = \mathcal{P}$ (*Upair* $c_1\langle t_1\rangle$ $t_1{}'$) $\Longrightarrow$
$l_2 = t_2 \approx t_2{}' \Longrightarrow$
$\neg$ *is-Var* $t_1 \Longrightarrow$
welltyped-imgu-on (*clause.vars* $(E \cdot \varrho_1) \cup$ *clause.vars* $(D \cdot \varrho_2)$) $\mathcal{V}_3$ $(t_1 \cdot t \varrho_1)$ $(t_2 \cdot t \varrho_2)$ $\mu \Longrightarrow$
$\forall x \in$ *clause.vars* E. $\mathcal{V}_1$ $x = \mathcal{V}_3$ (*term.rename* ϱ_1 x) $\Longrightarrow$
$\forall x \in$ *clause.vars* D. $\mathcal{V}_2$ $x = \mathcal{V}_3$ (*term.rename* ϱ_2 x) $\Longrightarrow$
term.subst.is-welltyped-on (*clause.vars* E) $\mathcal{V}_1$ $\varrho_1 \Longrightarrow$
term.subst.is-welltyped-on (*clause.vars* D) $\mathcal{V}_2$ $\varrho_2 \Longrightarrow$
$(\bigwedge \tau\ \tau'.$ *typed* $\mathcal{V}_2$ t_2 $\tau \Longrightarrow$ *typed* $\mathcal{V}_2$ $t_2{}'$ $\tau' \Longrightarrow \tau = \tau') \Longrightarrow$
$\neg$ $(E \cdot \varrho_1 \odot \mu \preceq_c D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$(\mathcal{P} = Pos \wedge$ *select* $E = \{\#\} \wedge$ *is-strictly-maximal* $(l_1 \cdot l \varrho_1 \odot \mu)$ $(E \cdot \varrho_1 \odot \mu) \vee$
$\mathcal{P} = Neg \wedge$ (*select* $E = \{\#\} \wedge$ *is-maximal* $(l_1 \cdot l \varrho_1 \odot \mu)$ $(E \cdot \varrho_1 \odot \mu) \vee$
is-maximal $(l_1 \cdot l \varrho_1 \odot \mu)$ ((*select* E) $\cdot \varrho_1 \odot \mu$))) $\Longrightarrow$
select $D = \{\#\} \Longrightarrow$
is-strictly-maximal $(l_2 \cdot l \varrho_2 \odot \mu)$ $(D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$\neg$ $(c_1\langle t_1\rangle \cdot t \varrho_1 \odot \mu \preceq_t t_1{}' \cdot t \varrho_1 \odot \mu) \Longrightarrow$

$\neg\ (t_2 \cdot t\ \varrho_2 \odot \mu \preceq_t t_2' \cdot t\ \varrho_2 \odot \mu) \Longrightarrow$
$C = \mathit{add\text{-}mset}\ (\mathcal{P}\ (\mathit{Upair}\ (c_1 \cdot t_c\ \varrho_1)\langle t_2' \cdot t\ \varrho_2\rangle\ (t_1' \cdot t\ \varrho_1)))\ (E' \cdot \varrho_1 + D' \cdot \varrho_2) \cdot \mu \Longrightarrow$
$\mathit{superposition}'\ (D, \mathcal{V}_2)\ (E, \mathcal{V}_1)\ (C, \mathcal{V}_3)$

lemma *superposition-eq-superposition'*: $\mathit{superposition} = \mathit{superposition}'$
⟨*proof*⟩

inductive *pos-superposition* ::
$('f, 'v, 'ty)$ *typed-clause* ⇒ $('f, 'v, 'ty)$ *typed-clause* ⇒ $('f, 'v, 'ty)$ *typed-clause* ⇒ *bool*
where
pos-superpositionI:
infinite-variables-per-type $\mathcal{V}_1 \Longrightarrow$
infinite-variables-per-type $\mathcal{V}_2 \Longrightarrow$
term-subst.is-renaming $\varrho_1 \Longrightarrow$
term-subst.is-renaming $\varrho_2 \Longrightarrow$
clause.vars $(E \cdot \varrho_1) \cap$ *clause.vars* $(D \cdot \varrho_2) = \{\} \Longrightarrow$
$E = \mathit{add\text{-}mset}\ l_1\ E' \Longrightarrow$
$D = \mathit{add\text{-}mset}\ l_2\ D' \Longrightarrow$
$l_1 = c_1\langle t_1\rangle \approx t_1' \Longrightarrow$
$l_2 = t_2 \approx t_2' \Longrightarrow$
$\neg$ *is-Var* $t_1 \Longrightarrow$
welltyped-imgu-on (*clause.vars* $(E \cdot \varrho_1) \cup$ *clause.vars* $(D \cdot \varrho_2))\ \mathcal{V}_3\ (t_1 \cdot t\ \varrho_1)\ (t_2 \cdot t\ \varrho_2)\ \mu \Longrightarrow$
$\forall x \in \mathit{clause.vars}\ E.\ \mathcal{V}_1\ x = \mathcal{V}_3\ (\mathit{term.rename}\ \varrho_1\ x) \Longrightarrow$
$\forall x \in \mathit{clause.vars}\ D.\ \mathcal{V}_2\ x = \mathcal{V}_3\ (\mathit{term.rename}\ \varrho_2\ x) \Longrightarrow$
term.subst.is-welltyped-on (*clause.vars* E) $\mathcal{V}_1\ \varrho_1 \Longrightarrow$
term.subst.is-welltyped-on (*clause.vars* D) $\mathcal{V}_2\ \varrho_2 \Longrightarrow$
$(\bigwedge \tau\ \tau'.\ \mathit{typed}\ \mathcal{V}_2\ t_2\ \tau \Longrightarrow \mathit{typed}\ \mathcal{V}_2\ t_2'\ \tau' \Longrightarrow \tau = \tau') \Longrightarrow$
$\neg\ (E \cdot \varrho_1 \odot \mu \preceq_c D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$\mathit{select}\ E = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ (l_1 \cdot l\ \varrho_1 \odot \mu)\ (E \cdot \varrho_1 \odot \mu) \Longrightarrow$
$\mathit{select}\ D = \{\#\} \Longrightarrow$
$\mathit{is\text{-}strictly\text{-}maximal}\ (l_2 \cdot l\ \varrho_2 \odot \mu)\ (D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$\neg\ (c_1\langle t_1\rangle \cdot t\ \varrho_1 \odot \mu \preceq_t t_1' \cdot t\ \varrho_1 \odot \mu) \Longrightarrow$
$\neg\ (t_2 \cdot t\ \varrho_2 \odot \mu \preceq_t t_2' \cdot t\ \varrho_2 \odot \mu) \Longrightarrow$
$C = \mathit{add\text{-}mset}\ ((c_1 \cdot t_c\ \varrho_1)\langle t_2' \cdot t\ \varrho_2\rangle \approx (t_1' \cdot t\ \varrho_1))\ (E' \cdot \varrho_1 + D' \cdot \varrho_2) \cdot \mu \Longrightarrow$
$\mathit{pos\text{-}superposition}\ (D, \mathcal{V}_2)\ (E, \mathcal{V}_1)\ (C, \mathcal{V}_3)$

lemma *superposition-if-pos-superposition*:
assumes *pos-superposition D E C*
shows *superposition D E C*
⟨*proof*⟩

inductive *neg-superposition* ::
$('f, 'v, 'ty)$ *typed-clause* ⇒ $('f, 'v, 'ty)$ *typed-clause* ⇒ $('f, 'v, 'ty)$ *typed-clause* ⇒ *bool*
where

neg-superpositionI:
infinite-variables-per-type $\mathcal{V}_1 \Longrightarrow$
infinite-variables-per-type $\mathcal{V}_2 \Longrightarrow$
term-subst.is-renaming $\varrho_1 \Longrightarrow$
term-subst.is-renaming $\varrho_2 \Longrightarrow$
clause.vars $(E \cdot \varrho_1) \cap$ *clause.vars* $(D \cdot \varrho_2) = \{\} \Longrightarrow$
$E =$ *add-mset* $l_1\ E' \Longrightarrow$
$D =$ *add-mset* $l_2\ D' \Longrightarrow$
$l_1 = c_1\langle t_1 \rangle\ !\approx\ {t_1}' \Longrightarrow$
$l_2 = t_2 \approx {t_2}' \Longrightarrow$
$\neg$ *is-Var* $t_1 \Longrightarrow$
welltyped-imgu-on (*clause.vars* $(E \cdot \varrho_1) \cup$ *clause.vars* $(D \cdot \varrho_2))\ \mathcal{V}_3\ (t_1 \cdot t\ \varrho_1)\ (t_2 \cdot t\ \varrho_2)\ \mu \Longrightarrow$
$\forall x \in$ *clause.vars* $E.\ \mathcal{V}_1\ x = \mathcal{V}_3\ (\textit{term.rename}\ \varrho_1\ x) \Longrightarrow$
$\forall x \in$ *clause.vars* $D.\ \mathcal{V}_2\ x = \mathcal{V}_3\ (\textit{term.rename}\ \varrho_2\ x) \Longrightarrow$
term.subst.is-welltyped-on (*clause.vars* E) $\mathcal{V}_1\ \varrho_1 \Longrightarrow$
term.subst.is-welltyped-on (*clause.vars* D) $\mathcal{V}_2\ \varrho_2 \Longrightarrow$
$(\bigwedge \tau\ \tau'.$ *typed* $\mathcal{V}_2\ t_2\ \tau \Longrightarrow$ *typed* $\mathcal{V}_2\ {t_2}'\ \tau' \Longrightarrow \tau = \tau') \Longrightarrow$
$\neg\ (E \cdot \varrho_1 \odot \mu \preceq_c D \cdot \varrho_2 \odot \mu) \Longrightarrow$
(*select* $E = \{\#\} \Longrightarrow$ *is-maximal* $(l_1 \cdot l\ \varrho_1 \odot \mu)\ (E \cdot \varrho_1 \odot \mu)) \Longrightarrow$
(*select* $E \neq \{\#\} \Longrightarrow$ *is-maximal* $(l_1 \cdot l\ \varrho_1 \odot \mu)\ ((\textit{select}\ E) \cdot \varrho_1 \odot \mu)) \Longrightarrow$
select $D = \{\#\} \Longrightarrow$
is-strictly-maximal $(l_2 \cdot l\ \varrho_2 \odot \mu)\ (D \cdot \varrho_2 \odot \mu) \Longrightarrow$
$\neg\ (c_1\langle t_1 \rangle \cdot t\ \varrho_1 \odot \mu \preceq_t {t_1}' \cdot t\ \varrho_1 \odot \mu) \Longrightarrow$
$\neg\ (t_2 \cdot t\ \varrho_2 \odot \mu \preceq_t {t_2}' \cdot t\ \varrho_2 \odot \mu) \Longrightarrow$
$C =$ *add-mset* $((c_1 \cdot t_c\ \varrho_1)\langle {t_2}' \cdot t\ \varrho_2 \rangle\ !\approx\ ({t_1}' \cdot t\ \varrho_1))\ (E' \cdot \varrho_1 + D' \cdot \varrho_2) \cdot \mu \Longrightarrow$
neg-superposition $(D, \mathcal{V}_2)\ (E, \mathcal{V}_1)\ (C, \mathcal{V}_3)$

lemma *superposition-if-neg-superposition*:
assumes *neg-superposition E D C*
shows *superposition E D C*
⟨*proof*⟩

lemma *superposition-iff-pos-or-neg*:
superposition D E C $\longleftrightarrow$ *pos-superposition D E C* $\vee$ *neg-superposition D E C*
⟨*proof*⟩

end

end
theory *Grounded-Superposition*
imports
Superposition
Ground-Superposition

First-Order-Clause.Grounded-Selection-Function
First-Order-Clause.Nonground-Inference
Saturation-Framework.Lifting-to-Non-Ground-Calculi
begin

locale *grounded-superposition-calculus* =
  superposition-calculus **where** *select* = *select* **and** $\mathcal{F} = \mathcal{F}$ +
  grounded-selection-function **where** *select* = *select* **and** $\mathcal{F} = \mathcal{F}$
  for
    select :: ('*f*, '*v* :: *infinite*) *select* **and**
    $\mathcal{F}$ :: ('*f*, '*ty*) *fun-types*
begin

sublocale *nonground-inference*⟨*proof*⟩

sublocale *ground*: *ground-superposition-calculus* **where**
  $less_t = (\prec_{tG})$ **and** *select* = $select_G$
rewrites
  multiset-extension.multiset-extension $(\prec_{tG})$ *mset-lit* = $(\prec_{lG})$ **and**
  multiset-extension.multiset-extension $(\prec_{lG})$ $(\lambda x.\ x) = (\prec_{cG})$ **and**
 $\bigwedge l\ C$. *ground.is-maximal l C* $\longleftrightarrow$ *is-maximal* (*literal.from-ground l*) (*clause.from-ground C*) **and**
  $\bigwedge l\ C$. *ground.is-strictly-maximal l C* $\longleftrightarrow$
    is-strictly-maximal (*literal.from-ground l*) (*clause.from-ground C*)
  ⟨*proof*⟩

abbreviation *is-inference-ground-instance-one-premise* **where**
  is-inference-ground-instance-one-premise D C ι_G $\gamma \equiv$
    case (*D*, *C*) *of* ((*D*, $\mathcal{V}'$), (*C*, $\mathcal{V}$)) ⇒
     inference.is-ground (*Infer* [*D*] *C* $\cdot\iota$ γ) ∧
     ι_G = *inference.to-ground* (*Infer* [*D*] *C* $\cdot\iota$ γ) ∧
     clause.is-welltyped $\mathcal{V}$ *D* ∧
     term.subst.is-welltyped-on (*clause.vars C*) $\mathcal{V}$ γ ∧
     clause.is-welltyped $\mathcal{V}$ *C* ∧
     $\mathcal{V} = \mathcal{V}'$ ∧
     infinite-variables-per-type $\mathcal{V}$

abbreviation *is-inference-ground-instance-two-premises* **where**
  is-inference-ground-instance-two-premises D E C ι_G γ ϱ_1 ϱ_2 ≡
    case (*D*, *E*, *C*) *of* ((*D*, $\mathcal{V}_2$), (*E*, $\mathcal{V}_1$), (*C*, $\mathcal{V}_3$)) ⇒
        term-subst.is-renaming ϱ_1
      ∧ *term-subst.is-renaming* ϱ_2
      ∧ *clause.vars* (*E* · ϱ_1) ∩ *clause.vars* (*D* · ϱ_2) = {}
      ∧ *inference.is-ground* (*Infer* [*D* · ϱ_2, *E* · ϱ_1] *C* $\cdot\iota$ γ)
      ∧ ι_G = *inference.to-ground* (*Infer* [*D* · ϱ_2, *E* · ϱ_1] *C* $\cdot\iota$ γ)
      ∧ *clause.is-welltyped* $\mathcal{V}_1$ *E*
      ∧ *clause.is-welltyped* $\mathcal{V}_2$ *D*
      ∧ *term.subst.is-welltyped-on* (*clause.vars C*) $\mathcal{V}_3$ γ
      ∧ *clause.is-welltyped* $\mathcal{V}_3$ *C*
      ∧ *infinite-variables-per-type* $\mathcal{V}_1$
      ∧ *infinite-variables-per-type* $\mathcal{V}_2$
      ∧ *infinite-variables-per-type* $\mathcal{V}_3$

abbreviation *is-inference-ground-instance* **where**
is-inference-ground-instance ι ι_G $\gamma \equiv$
(*case* ι *of*
Infer [D] $C \Rightarrow$ *is-inference-ground-instance-one-premise* D C ι_G γ
| *Infer* [D, E] $C \Rightarrow \exists\, \varrho_1$ ϱ_2. *is-inference-ground-instance-two-premises* D E C ι_G γ ϱ_1 ϱ_2
| - $\Rightarrow$ *False*)
$\wedge$ $\iota_G \in$ *ground.G-Inf*

definition *inference-ground-instances* **where**
inference-ground-instances $\iota = \{\ \iota_G \mid \iota_G\ \gamma.$ *is-inference-ground-instance* ι ι_G $\gamma\ \}$

lemma *is-inference-ground-instance*:
is-inference-ground-instance ι ι_G $\gamma \Longrightarrow \iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

lemma *is-inference-ground-instance-one-premise*:
assumes *is-inference-ground-instance-one-premise* D C ι_G γ $\iota_G \in$ *ground.G-Inf*
shows $\iota_G \in$ *inference-ground-instances* (*Infer* [D] C)
⟨*proof*⟩

lemma *is-inference-ground-instance-two-premises*:
assumes *is-inference-ground-instance-two-premises* D E C ι_G γ ϱ_1 ϱ_2 $\iota_G \in$ *ground.G-Inf*
shows $\iota_G \in$ *inference-ground-instances* (*Infer* [D, E] C)
⟨*proof*⟩

lemma *ground-inference_concl-in-welltyped-ground-instances*:
assumes $\iota_G \in$ *inference-ground-instances* ι
shows *concl-of* $\iota_G \in$ *clause.welltyped-ground-instances* (*concl-of* ι)
⟨*proof*⟩

lemma *ground-inference-red-in-welltyped-ground-instances-of-concl*:
assumes $\iota_G \in$ *inference-ground-instances* ι
shows $\iota_G \in$ *ground.Red-I* (*clause.welltyped-ground-instances* (*concl-of* ι))
⟨*proof*⟩

thm *option.sel*

sublocale *lifting*:
tiebreaker-lifting
$\bot_F$
inferences
ground.G-Bot
ground.G-entails
ground.G-Inf
ground.GRed-I
ground.GRed-F
clause.welltyped-ground-instances

Some ∘ *inference-ground-instances*
typed-tiebreakers
⟨*proof*⟩

end

context *superposition-calculus*
begin

sublocale
lifting-intersection
inferences
{{#}}
$select_{Gs}$
ground-superposition-calculus.G-Inf ($\prec_{tG}$)
λ-. *ground-superposition-calculus.G-entails*
ground-superposition-calculus.GRed-I ($\prec_{tG}$)
λ-. *ground-superposition-calculus.GRed-F* ($\prec_{tG}$)
$\bot_F$
λ-. *clause.welltyped-ground-instances*
$\lambda select_G$. *Some* ∘ (*grounded-superposition-calculus.inference-ground-instances*
($\prec_t$) $select_G$ $\mathcal{F}$)
typed-tiebreakers
⟨*proof*⟩

end

end
theory *Superposition-Welltypedness-Preservation*
imports *Superposition*
begin

context *superposition-calculus*
begin

lemma *eq-resolution-preserves-typing*:
assumes *eq-resolution* (D, $\mathcal{V}$) (C, $\mathcal{V}$)
shows *clause.is-welltyped* $\mathcal{V}$ D ⟷ *clause.is-welltyped* $\mathcal{V}$ C
⟨*proof*⟩

lemma *eq-factoring-preserves-typing*:
assumes *eq-factoring* (D, $\mathcal{V}$) (C, $\mathcal{V}$)
shows *clause.is-welltyped* $\mathcal{V}$ D ⟷ *clause.is-welltyped* $\mathcal{V}$ C
⟨*proof*⟩

lemma *superposition-preserves-typing-C*:
assumes
superposition: *superposition* (D, $\mathcal{V}_2$) (E, $\mathcal{V}_1$) (C, $\mathcal{V}_3$) **and**
D-is-welltyped: *clause.is-welltyped* $\mathcal{V}_2$ D **and**

E-is-welltyped: *clause.is-welltyped* $\mathcal{V}_1$ *E*
shows *clause.is-welltyped* $\mathcal{V}_3$ *C*
⟨*proof*⟩

lemma *superposition-preserves-typing-D*:
assumes
superposition: *superposition* (*D*, $\mathcal{V}_2$) (*E*, $\mathcal{V}_1$) (*C*, $\mathcal{V}_3$) **and**
C-is-welltyped: *clause.is-welltyped* $\mathcal{V}_3$ *C*
shows *clause.is-welltyped* $\mathcal{V}_2$ *D*
⟨*proof*⟩

lemma *superposition-preserves-typing-E*:
assumes
superposition: *superposition* (*D*, $\mathcal{V}_2$) (*E*, $\mathcal{V}_1$) (*C*, $\mathcal{V}_3$) **and**
C-is-welltyped: *clause.is-welltyped* $\mathcal{V}_3$ *C*
shows *clause.is-welltyped* $\mathcal{V}_1$ *E*
⟨*proof*⟩

lemma *superposition-preserves-typing*:
assumes *superposition* (*D*, $\mathcal{V}_2$) (*E*, $\mathcal{V}_1$) (*C*, $\mathcal{V}_3$)
shows *clause.is-welltyped* $\mathcal{V}_2$ *D* ∧ *clause.is-welltyped* $\mathcal{V}_1$ *E* ⟷ *clause.is-welltyped* $\mathcal{V}_3$ *C*
⟨*proof*⟩

end

end
theory *Superposition-Completeness*
imports
Grounded-Superposition
Ground-Superposition-Completeness
Superposition-Welltypedness-Preservation

First-Order-Clause.Nonground-Entailment
begin

3 Completeness

context *grounded-superposition-calculus*
begin

3.1 Liftings

lemma *eq-resolution-lifting*:
fixes
D_G C_G :: '*f ground-atom clause* **and**
D C :: ('*f*, '*v*) *atom clause* **and**
γ :: ('*f*, '*v*) *subst*
defines

[*simp*]: $D_G \equiv$ *clause.to-ground* $(D \cdot \gamma)$ **and**
[*simp*]: $C_G \equiv$ *clause.to-ground* $(C \cdot \gamma)$
assumes
ground-eq-resolution: *ground.eq-resolution* D_G C_G **and**
D-grounding: *clause.is-ground* $(D \cdot \gamma)$ **and**
C-grounding: *clause.is-ground* $(C \cdot \gamma)$ **and**
select: *clause.from-ground* $(select_G\ D_G) = (select\ D) \cdot \gamma$ **and**
D-is-welltyped: *clause.is-welltyped* $\mathcal{V}$ D **and**
γ*-is-welltyped*: *term.subst.is-welltyped-on* $(clause.vars\ D)$ $\mathcal{V}$ γ **and**
$\mathcal{V}$: *infinite-variables-per-type* $\mathcal{V}$
obtains C'
where
eq-resolution $(D, \mathcal{V})$ $(C', \mathcal{V})$
Infer $[D_G]$ $C_G \in$ *inference-ground-instances* $(Infer\ [(D, \mathcal{V})]\ (C', \mathcal{V}))$
$C' \cdot \gamma = C \cdot \gamma$
⟨*proof*⟩

lemma *eq-factoring-lifting*:
fixes
D_G C_G :: $'f$ *ground-atom clause* **and**
D C :: $('f, 'v)$ *atom clause* **and**
γ :: $('f, 'v)$ *subst*
defines
[*simp*]: $D_G \equiv$ *clause.to-ground* $(D \cdot \gamma)$ **and**
[*simp*]: $C_G \equiv$ *clause.to-ground* $(C \cdot \gamma)$
assumes
ground-eq-factoring: *ground.eq-factoring* D_G C_G **and**
D-grounding: *clause.is-ground* $(D \cdot \gamma)$ **and**
C-grounding: *clause.is-ground* $(C \cdot \gamma)$ **and**
select: *clause.from-ground* $(select_G\ D_G) = (select\ D) \cdot \gamma$ **and**
D-is-welltyped: *clause.is-welltyped* $\mathcal{V}$ D **and**
γ*-is-welltyped*: *term.subst.is-welltyped-on* $(clause.vars\ D)$ $\mathcal{V}$ γ **and**
$\mathcal{V}$: *infinite-variables-per-type* $\mathcal{V}$
obtains C'
where
eq-factoring $(D, \mathcal{V})$ $(C', \mathcal{V})$
Infer $[D_G]$ $C_G \in$ *inference-ground-instances* $(Infer\ [(D, \mathcal{V})]\ (C', \mathcal{V}))$
$C' \cdot \gamma = C \cdot \gamma$
⟨*proof*⟩

lemma *superposition-lifting*:
fixes
E_G D_G C_G :: $'f$ *ground-atom clause* **and**
E D C :: $('f, 'v)$ *atom clause* **and**
γ ϱ_1 ϱ_2 :: $('f, 'v)$ *subst* **and**
$\mathcal{V}_1$ $\mathcal{V}_2$ :: $('v, 'ty)$ *var-types*
defines
[*simp*]: $E_G \equiv$ *clause.to-ground* $(E \cdot \varrho_1 \odot \gamma)$ **and**
[*simp*]: $D_G \equiv$ *clause.to-ground* $(D \cdot \varrho_2 \odot \gamma)$ **and**

[*simp*]: $C_G \equiv$ *clause.to-ground* $(C \cdot \gamma)$ **and**
[*simp*]: $N_G \equiv$ *clause.welltyped-ground-instances* $(E, \mathcal{V}_1) \cup$
clause.welltyped-ground-instances $(D, \mathcal{V}_2)$ **and**
[*simp*]: $\iota_G \equiv$ *Infer* $[D_G, E_G]$ C_G
assumes
ground-superposition: *ground.superposition* D_G E_G C_G **and**
ϱ_1: *term-subst.is-renaming* ϱ_1 **and**
ϱ_2: *term-subst.is-renaming* ϱ_2 **and**
rename-apart: *clause.vars* $(E \cdot \varrho_1) \cap$ *clause.vars* $(D \cdot \varrho_2) = \{\}$ **and**
E-grounding: *clause.is-ground* $(E \cdot \varrho_1 \odot \gamma)$ **and**
D-grounding: *clause.is-ground* $(D \cdot \varrho_2 \odot \gamma)$ **and**
C-grounding: *clause.is-ground* $(C \cdot \gamma)$ **and**
select-from-E: *clause.from-ground* $(select_G\ E_G) = (select\ E) \cdot \varrho_1 \odot \gamma$ **and**
select-from-D: *clause.from-ground* $(select_G\ D_G) = (select\ D) \cdot \varrho_2 \odot \gamma$ **and**
E-is-welltyped: *clause.is-welltyped* $\mathcal{V}_1$ E **and**
D-is-welltyped: *clause.is-welltyped* $\mathcal{V}_2$ D **and**
ϱ_1-γ-*is-welltyped*: *term.subst.is-welltyped-on* (*clause.vars* E) $\mathcal{V}_1$ $(\varrho_1 \odot \gamma)$ **and**
ϱ_2-γ-*is-welltyped*: *term.subst.is-welltyped-on* (*clause.vars* D) $\mathcal{V}_2$ $(\varrho_2 \odot \gamma)$ **and**
ϱ_1-*is-welltyped*: *term.subst.is-welltyped-on* (*clause.vars* E) $\mathcal{V}_1$ ϱ_1 **and**
ϱ_2-*is-welltyped*: *term.subst.is-welltyped-on* (*clause.vars* D) $\mathcal{V}_2$ ϱ_2 **and**
$\mathcal{V}_1$: *infinite-variables-per-type* $\mathcal{V}_1$ **and**
$\mathcal{V}_2$: *infinite-variables-per-type* $\mathcal{V}_2$ **and**
not-redundant: $\iota_G \notin$ *ground.Red-I* N_G
obtains C' $\mathcal{V}_3$
where
superposition $(D, \mathcal{V}_2)$ $(E, \mathcal{V}_1)$ $(C', \mathcal{V}_3)$
$\iota_G \in$ *inference-ground-instances* (*Infer* $[(D, \mathcal{V}_2), (E, \mathcal{V}_1)]$ $(C', \mathcal{V}_3)$)
$C' \cdot \gamma = C \cdot \gamma$
⟨*proof*⟩

3.2 Ground instances

context
fixes ι_G N
assumes
subst-stability: *subst-stability-on* N **and**
ι_G-*Inf-from*: $\iota_G \in$ *ground.Inf-from-q* $select_G$ $(\bigcup$(*clause.welltyped-ground-instances* ‘ N))
begin

lemma *single-premise-ground-instance*:
assumes
ground-inference: $\iota_G \in$ $\{$*Infer* $[D]$ C | D C. *ground-inference* D $C\}$ **and**
lifting: $\bigwedge D\ \gamma\ C\ \mathcal{V}$ *thesis*. $[\![$
ground-inference (*clause.to-ground* $(D \cdot \gamma)$) (*clause.to-ground* $(C \cdot \gamma)$);
clause.is-ground $(D \cdot \gamma)$;
clause.is-ground $(C \cdot \gamma)$;
clause.from-ground $(select_G$ (*clause.to-ground* $(D \cdot \gamma))) = select\ D \cdot \gamma$;
clause.is-welltyped $\mathcal{V}$ D; *term.subst.is-welltyped-on* (*clause.vars* D) $\mathcal{V}$ γ;

infinite-variables-per-type $\mathcal{V}$;
$\bigwedge C'$. ⟦
inference $(D, \mathcal{V})$ $(C', \mathcal{V})$;
Infer [*clause.to-ground* $(D \cdot \gamma)$] (*clause.to-ground* $(C \cdot \gamma)$)
$\in$ *inference-ground-instances* (*Infer* $[(D, \mathcal{V})]$ $(C', \mathcal{V})$);
$C' \cdot \gamma = C \cdot \gamma$⟧ $\Longrightarrow$ *thesis*⟧
$\Longrightarrow$ *thesis* **and**
inference-eq: *inference* = *eq-factoring* $\vee$ *inference* = *eq-resolution*
obtains ι **where**
$\iota \in$ *Inf-from N*
$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

lemma *eq-resolution-ground-instance*:
assumes *ground-eq-resolution*: $\iota_G \in$ *ground.eq-resolution-inferences*
obtains ι **where**
$\iota \in$ *Inf-from N*
$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

lemma *eq-factoring-ground-instance*:
assumes *ground-eq-factoring*: $\iota_G \in$ *ground.eq-factoring-inferences*
obtains ι **where**
$\iota \in$ *Inf-from N*
$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

lemma *superposition-ground-instance*:
assumes
ground-superposition: $\iota_G \in$ *ground.superposition-inferences* **and**
not-redundant: $\iota_G \notin$ *ground.GRed-I* ($\bigcup$ (*clause.welltyped-ground-instances* ‘ *N*))
obtains ι **where**
$\iota \in$ *Inf-from N*
$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

lemma *ground-instances*:
assumes *not-redundant*: $\iota_G \notin$ *ground.Red-I* ($\bigcup$ (*clause.welltyped-ground-instances* ‘ *N*))
obtains ι **where**
$\iota \in$ *Inf-from N*
$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

end

end

context *superposition-calculus*

begin

lemma *overapproximation*:
  obtains $select_G$ **where**
    ground-Inf-overapproximated $select_G$ *premises*
    is-grounding $select_G$
⟨*proof*⟩

sublocale *statically-complete-calculus* $\bot_F$ *inferences entails-*$\mathcal{G}$ *Red-I-*$\mathcal{G}$ *Red-F-*$\mathcal{G}$
⟨*proof*⟩

end

end
theory *Superposition-Soundness*
  imports
    First-Order-Clause.Nonground-Entailment

    Grounded-Superposition
    Superposition-Welltypedness-Preservation
begin

3.3 Soundness

context *grounded-superposition-calculus*
begin

notation *lifting.entails-*$\mathcal{G}$ (**infix** $\models\!\models_F$ *50*)

lemma *eq-resolution-sound*:
  assumes *eq-resolution*: *eq-resolution D C*
  shows $\{D\} \models\!\models_F \{C\}$
  ⟨*proof*⟩

lemma *eq-factoring-sound*:
  assumes *eq-factoring*: *eq-factoring D C*
  shows $\{D\} \models\!\models_F \{C\}$
  ⟨*proof*⟩

lemma *superposition-sound*:
  assumes *superposition*: *superposition D E C*
  shows $\{E, D\} \models\!\models_F \{C\}$
  ⟨*proof*⟩

end

sublocale *grounded-superposition-calculus* $\subseteq$ *sound-inference-system inferences* $\bot_F$ ($\models\!\models_F$)
⟨*proof*⟩

sublocale *superposition-calculus* $\subseteq$ *sound-inference-system inferences* $\bot_F$ *entails-*$\mathcal{G}$ ⟨*proof*⟩

end

4 Integration of IsaFoR Terms and the Knuth–Bendix Order

This theory implements the abstract interface for atoms and substitutions using the IsaFoR library.

theory *IsaFoR-Term-Copy*
imports
First-Order-Terms.Unification
HOL−Cardinals.Wellorder-Extension
Knuth-Bendix-Order.KBO
begin

This part extends and integrates and the Knuth–Bendix order defined in IsaFoR.

record *'f weights* =
w :: *'f* × *nat* ⇒ *nat*
w0 :: *nat*
pr-strict :: *'f* × *nat* ⇒ *'f* × *nat* ⇒ *bool*
least :: *'f* ⇒ *bool*
scf :: *'f* × *nat* ⇒ *nat* ⇒ *nat*

class *weighted* =
fixes *weights* :: *'a weights*
assumes *weights-adm*:
admissible-kbo
(*w weights*) (*w0 weights*) (*pr-strict weights*) ((*pr-strict weights*)$^{==}$) (*least weights*) (*scf weights*)
and *pr-strict-total*: *fi* = *gj* ∨ *pr-strict weights fi gj* ∨ *pr-strict weights gj fi*
and *pr-strict-asymp*: *asymp* (*pr-strict weights*)
and *scf-ok*: *i* < *n* ⟹ *scf weights* (*f*, *n*) *i* ≤ *1*

instantiation *unit* :: *weighted* **begin**

definition *weights-unit* :: *unit weights* **where** *weights-unit* =
(|*w* = *Suc* ∘ *snd*, *w0* = *1*, *pr-strict* = λ(-, *n*) (-, *m*). *n* > *m*, *least* = λ-. *True*, *scf* = λ- -. *1*|)

instance
⟨*proof*⟩
end

global-interpretation *KBO*:
admissible-kbo
w (*weights* :: *'f* :: *weighted weights*) *w0* (*weights* :: *'f* :: *weighted weights*)
pr-strict weights ((*pr-strict weights*)$^{==}$) *least weights scf weights*
defines *weight* = *KBO.weight*
and *kbo* = *KBO.kbo*
⟨*proof*⟩

lemma *kbo-code*[*code*]: *kbo s t* =
(*let wt* = *weight t*; *ws* = *weight s in*
if vars-term-ms (*KBO.SCF t*) ⊆# *vars-term-ms* (*KBO.SCF s*) ∧ *wt* ≤ *ws*
then
(*if wt* < *ws then* (*True*, *True*)
else
(*case s of*
Var y ⇒ (*False*, *case t of Var x* ⇒ *True* | *Fun g ts* ⇒ *ts* = [] ∧ *least weights*
g)
| *Fun f ss* ⇒
(*case t of*
Var x ⇒ (*True*, *True*)
| *Fun g ts* ⇒
if pr-strict weights (*f*, *length ss*) (*g*, *length ts*) *then* (*True*, *True*)
else if (*f*, *length ss*) = (*g*, *length ts*) *then lex-ext-unbounded kbo ss ts*
else (*False*, *False*))))
else (*False*, *False*))
⟨*proof*⟩

definition *less-kbo s t* = *fst* (*kbo t s*)

lemma *less-kbo-gtotal*: *ground s* ⟹ *ground t* ⟹ *s* = *t* ∨ *less-kbo s t* ∨ *less-kbo*
t s
⟨*proof*⟩

lemma *less-kbo-subst*:
fixes *σ* :: (*'f* :: *weighted*, *'v*) *subst*
shows *less-kbo s t* ⟹ *less-kbo* (*s* · *σ*) (*t* · *σ*)
⟨*proof*⟩

lemma *wfP-less-kbo*: *wfP less-kbo*
⟨*proof*⟩

end
theory *Superposition-Example*
imports
Superposition
IsaFoR-Term-Copy
VeriComp.Well-founded
begin

sublocale *nonground-term-with-context* $\subseteq$
  nonground-term-order less-kbo :: $('f :: weighted, 'v)$ *term* $\Rightarrow$ $('f, 'v)$ *term* $\Rightarrow$ *bool*
⟨*proof*⟩

abbreviation *trivial-tiebreakers* ::
  $'f$ *gatom clause* $\Rightarrow$ $('f, 'v)$ *atom clause* $\Rightarrow$ $('f, 'v)$ *atom clause* $\Rightarrow$ *bool* **where**
  trivial-tiebreakers - - - $\equiv$ *False*

lemma *trivial-tiebreakers*: *wellfounded-strict-order* (*trivial-tiebreakers* C_G)
  ⟨*proof*⟩

locale *trivial-superposition-example* =
  ground-critical-pair-theorem $TYPE('f :: weighted)$
begin

sublocale *nonground-term-with-context*⟨*proof*⟩

abbreviation *trivial-select* :: $'a$ *clause* $\Rightarrow$ $'a$ *clause* **where**
  trivial-select - $\equiv$ $\{\#\}$

abbreviation *unit-types* **where**
  unit-types - $\equiv$ ([], ())

sublocale *selection-function trivial-select*
  ⟨*proof*⟩

sublocale
  superposition-calculus
    trivial-select :: $('f$, $'v :: infinite)$ *select*
    less-kbo
    unit-types
    trivial-tiebreakers
  ⟨*proof*⟩

end

context *nonground-equality-order*
begin

abbreviation *select-max* **where**
  select-max C $\equiv$
    if $\exists\, l \in \# C.$ *is-maximal l C* $\wedge$ *is-neg l*
    then $\{\#$*SOME l. is-maximal l C* $\wedge$ *is-neg l*$\#\}$
    else $\{\#\}$

sublocale *select-max*: *selection-function select-max*
⟨*proof*⟩

end

datatype *type* = *A* | *B*

lemma *UNIV-type* [*simp*]: (*UNIV* :: *type set*) = {*A*, *B*}
⟨*proof*⟩

lemma *UNIV-type-ordLeq-UNIV-nat*: |*UNIV* :: *type set*| ≤*o* |*UNIV* :: *nat set*|
⟨*proof*⟩

definition *pr-strict* :: ('*f* :: *wellorder* × *nat*) ⇒ - ⇒ *bool* **where**
pr-strict = *lex-prodp* ((<) :: '*f* ⇒ '*f* ⇒ *bool*) ((<) :: *nat* ⇒ *nat* ⇒ *bool*)

lemma *wfp-pr-strict*: *wfp pr-strict*
⟨*proof*⟩

lemma *transp-pr-strict*: *transp pr-strict*
⟨*proof*⟩

definition *least* **where**
least x ⟷ (∀ *y*. *x* ≤ *y*)

definition *weight* :: '*f* × *nat* ⇒ *nat* **where**
weight p = *1*

abbreviation *weights* **where** *weights* ≡
(|*w* = *weight*, *w0* = *1*, *pr-strict* = *pr-strict*$^{-1-1}$, *least* = *least*, *scf* = λ- -. *1*|)

interpretation *weighted weights*
⟨*proof*⟩

instantiation *nat* :: *weighted* **begin**

definition *weights-nat* :: *nat weights* **where** *weights-nat* ≡ *weights*

instance
⟨*proof*⟩

end

instantiation *nat* :: *infinite* **begin**

instance
⟨*proof*⟩

end

fun *repeat* :: *nat* ⇒ *'a* ⇒ *'a list* **where**
  repeat 0 - = []
| *repeat (Suc n) x = x # repeat n x*

abbreviation *types* :: *nat* ⇒ *type list* × *type* **where**
  types n ≡
    let type = if even n then A else B
    in (repeat (n div 2) type, type)

lemma *types-inhabited*: ∃*f*. *types f* = ([], *τ*)
⟨*proof*⟩

locale *superposition-example* =
  ground-critical-pair-theorem TYPE(*nat*)
begin

sublocale *wellfounded-strict-order trivial-tiebreakers* C_G
  ⟨*proof*⟩

sublocale *nonground-term-with-context* ⟨*proof*⟩

sublocale *nonground-equality-order less-kbo*
  ⟨*proof*⟩

sublocale
  superposition-calculus
    select-max :: (*nat*, *nat*) *select*
    less-kbo
    types
    trivial-tiebreakers
⟨*proof*⟩

end

end

References

[1] M. Desharnais, B. Toth, U. Waldmann, J. Blanchette, and S. Tourret. A Modular Formalization of Superposition in Isabelle/HOL. In Y. Bertot, T. Kutsia, and M. Norrish, editors, *15th International Conference on Interactive Theorem Proving (ITP 2024)*, volume 309 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.