Typed Ordered Resolution

Adnan Mohammed Ahmed Balazs Toth

February 6, 2026

Abstract

Ordered Resolution is a proof calculus for reasoning about first-order logic 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 also added a type system to the calculus. We relied on the library for first-order clauses and on the saturation framework.

Contents

theory *Ground-Ordered-Resolution*
imports
First-Order-Clause.Selection-Function
First-Order-Clause.Ground-Order
First-Order-Clause.Literal-Functor
begin

1 Resolution Calculus

locale *ground-ordered-resolution-calculus* =

ground-order **where** $less_t = less_t +$
selection-function select
for
$less_t :: 't \Rightarrow 't \Rightarrow bool$ **and**
select :: *'t clause* $\Rightarrow$ *'t clause*
begin

1.1 Resolution Calculus

inductive *resolution* :: *'t clause* $\Rightarrow$ *'t clause* $\Rightarrow$ *'t clause* $\Rightarrow$ *bool* **where**
resolutionI:
$E =$ *add-mset* l_1 $E' \Longrightarrow$
$D =$ *add-mset* l_2 $D' \Longrightarrow$
$l_1 =$ *Neg* $t \Longrightarrow$
$l_2 =$ *Pos* $t \Longrightarrow$
$C = (E' + D') \Longrightarrow$
resolution D E C
if
$D \prec_c E$
select $E = \{\#\} \wedge$ *is-maximal* l_1 $E \vee$ *is-maximal* l_1 (*select* E)
select $D = \{\#\}$
is-strictly-maximal l_2 D

inductive *factoring* :: *'t clause* $\Rightarrow$ *'t clause* $\Rightarrow$ *bool* **where**
factoringI:
$D =$ *add-mset* l (*add-mset* l D') $\Longrightarrow$
$l =$ *Pos* $t \Longrightarrow$
$C =$ *add-mset* l $D' \Longrightarrow$
factoring D C
if
select $D = \{\#\}$
is-maximal l D

1.2 Ground Layer

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

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

definition *G-Inf* :: *'t clause inference set* **where**
G-Inf =
$\{$*Infer* $[D, E]$ $C \mid D$ E C. *resolution* D E $C\} \cup$
$\{$*Infer* $[D]$ $C \mid D$ C. *factoring* D $C\}$

abbreviation *G-Bot* :: *'t clause set* **where**
G-Bot $\equiv \{\{\#\}\}$

definition *G-entails* :: *'t clause set* $\Rightarrow$ *'t clause set* $\Rightarrow$ *bool* **where**

G-entails N_1 $N_2 \longleftrightarrow (\forall\ I.\ I \models s\ N_1 \longrightarrow I \models s\ N_2)$

1.3 Smaller Conclussions

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

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

end

1.4 Redundancy Criterion

sublocale *ground-ordered-resolution-calculus* ⊆ *consequence-relation* **where**
Bot = *G-Bot* **and**
entails = *G-entails*
⟨*proof*⟩

sublocale *ground-ordered-resolution-calculus* ⊆ *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*⟩

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

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

end
theory *Ground-Ordered-Resolution-Completeness*
imports
Ground-Ordered-Resolution
Relation-Extra
First-Order-Clause.HOL-Extra
begin

1.5 Mode Construction

context *ground-ordered-resolution-calculus*
begin

context
  fixes $N :: 't$ *clause set*
begin

function *epsilon* :: $'t$ *clause* $\Rightarrow$ $'t$ *set* **where**
  epsilon $C = \{A \mid A\ C'.$
    $C \in N \wedge$
    $C =$ *add-mset* (*Pos* A) $C' \wedge$
    select $C = \{\#\} \wedge$
    is-strictly-maximal (*Pos* A) $C \wedge$
    $\neg\ (\bigcup D \in \{D \in N.\ D \prec_c C\}.$ *epsilon* $D) \models C\}$
  ⟨*proof*⟩

termination *epsilon*
⟨*proof*⟩

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

end

lemma *epsilon-eq-empty-or-singleton*: *epsilon* $N\ C = \{\} \vee (\exists A.$ *epsilon* $N\ C = \{A\})$
⟨*proof*⟩

definition *rewrite-sys* **where**
  rewrite-sys $N\ C \equiv (\bigcup D \in \{D \in N.\ D \prec_c C\}.$ *epsilon* $N\ D)$

lemma *rewrite-sys-subset-if-less-cls*: $C \prec_c D \longrightarrow$ *rewrite-sys* $N\ C \subseteq$ *rewrite-sys* $N\ D$
  ⟨*proof*⟩

lemma *mem-epsilonE*:
  assumes *rule-in*: $A \in$ *epsilon* $N\ C$
  obtains C' **where**
    $C \in N$ **and**
    $C =$ *add-mset* (*Pos* A) C' **and**
    select $C = \{\#\}$ **and**
    is-strictly-maximal (*Pos* A) C **and**
    $\neg$ *rewrite-sys* $N\ C \models C$
  ⟨*proof*⟩

lemma *epsilon-unfold*: *epsilon* $N\ C = \{A \mid A\ C'.$
    $C \in N \wedge$
    $C =$ *add-mset* (*Pos* A) $C' \wedge$

select $C = \{\#\} \wedge$
is-strictly-maximal (*Pos* A) $C \wedge$
$\neg$ *rewrite-sys* $N\ C \models C\}$
⟨*proof*⟩

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

lemma
assumes
$D \preceq_c C$ **and**
C-prod: $A \in$ *epsilon* $N\ C$ **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-epsilon*:
assumes *C-prod*: $A_C \in$ *epsilon* $N\ C$ **and** *D-prod*: $A_D \in$ *epsilon* $N\ D$
shows $A_C \prec_t A_D \longleftrightarrow C \prec_c D$
⟨*proof*⟩

lemma *false-cls-if-productive-epsilon*:
assumes *C-prod*: $A \in$ *epsilon* $N\ C$ **and** $D \in N$ **and** $C \prec_c D$
shows $\neg$ *rewrite-sys* $N\ D \models C - \{\#Pos\ A\#\}$
⟨*proof*⟩

lemma *neg-notin-Interp-not-produce*:
Neg $A \in\# C \Longrightarrow A \notin$ *rewrite-sys* $N\ D \cup$ *epsilon* $N\ D \Longrightarrow C \preceq_c D \Longrightarrow A \notin$ *epsilon* $N\ D''$
⟨*proof*⟩

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

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

lemma *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 *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 *lift-entailment-to-Union*:
fixes $N\ D$
assumes
$D\text{-}in$: $D \in N$ **and**
$R_D\text{-}entails\text{-}D$: $rewrite\text{-}sys\ N\ D \models D$
shows
$(\bigcup C \in N.\ epsilon\ N\ C) \models D$
⟨*proof*⟩

lemma *true-cls-if-productive-epsilon*:
assumes $A \in epsilon\ N\ C\ C \prec_c D$
shows $rewrite\text{-}sys\ N\ D \models C$
⟨*proof*⟩

lemma *model-preconstruction*:
fixes
$N :: 't\ clause\ set$ **and**
$C :: 't\ clause$
defines
$entails \equiv \lambda E\ C.\ E \models C$
assumes $saturated\ N$ **and** $\{\#\} \notin N$ **and** $C\text{-}in$: $C \in N$
shows
$epsilon\ N\ C = \{\} \longleftrightarrow entails\ (rewrite\text{-}sys\ N\ C)\ C$
$(\bigcup D \in N.\ epsilon\ N\ D) \models C$
$D \in N \Longrightarrow C \prec_c D \Longrightarrow entails\ (rewrite\text{-}sys\ N\ D)\ C$
⟨*proof*⟩

lemma *model-construction*:
fixes
$N :: 't\ clause\ set$ **and**
$C :: 't\ clause$
defines $entails \equiv \lambda E\ C.\ E \models C$
assumes $saturated\ N$ **and** $\{\#\} \notin N$ **and** $C\text{-}in$: $C \in N$
shows $entails\ (\bigcup D \in N.\ epsilon\ N\ D)\ C$
⟨*proof*⟩

1.6 Static Refutational Completeness

lemma *statically-complete*:

fixes $N :: 't$ *clause set*
assumes *saturated* N **and** *G-entails* N $\{\{\#\}\}$
shows $\{\#\} \in N$
⟨*proof*⟩

sublocale *statically-complete-calculus* **where**
$Bot = G\text{-}Bot$ **and**
$Inf = G\text{-}Inf$ **and**
entails $=$ *G-entails* **and**
Red-I $=$ *Red-I* **and**
Red-F $=$ *Red-F*
⟨*proof*⟩

end

end
theory *Ground-Ordered-Resolution-Soundness*
imports *Ground-Ordered-Resolution*
begin

context *ground-ordered-resolution-calculus*
begin

lemma *soundness-ground-resolution*:
assumes
step: *resolution* D E C
shows *G-entails* $\{D, E\}$ $\{C\}$
⟨*proof*⟩

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

sublocale *sound-inference-system* **where**
$Inf = G\text{-}Inf$ **and**
$Bot = G\text{-}Bot$ **and**
entails $=$ *G-entails*
⟨*proof*⟩

end

end
theory *Ordered-Resolution*
imports
First-Order-Clause.Nonground-Order
First-Order-Clause.Nonground-Selection-Function
First-Order-Clause.Nonground-Typing
First-Order-Clause.Typed-Tiebreakers

Saturation-Framework.Lifting-to-Non-Ground-Calculi

Ground-Ordered-Resolution
begin

locale *ordered-resolution-calculus* =
witnessed-nonground-typing **where**
welltyped = *welltyped* **and** *term-to-ground* = *term-to-ground* :: $'t \Rightarrow 't_G$ **and**
id-subst = *id-subst* :: *'subst* +

nonground-order **where** $less_t = less_t$ +

nonground-selection-function **where**
select = *select* **and** *atom-subst* = $(\cdot t)$ **and** *atom-vars* = *term.vars* **and**
atom-from-ground = *term.from-ground* **and** *atom-to-ground* = *term.to-ground* +
tiebreakers tiebreakers
for
select :: *'t select* **and**
$less_t$ :: $'t \Rightarrow 't \Rightarrow$ *bool* **and**
tiebreakers :: $('t_G, 't)$ *tiebreakers* **and**
welltyped :: $('v$ :: *infinite*, *'ty*$)$ *var-types* $\Rightarrow 't \Rightarrow$ *'ty* $\Rightarrow$ *bool*
begin

inductive *factoring* :: $('t, 'v,$ *'ty*$)$ *typed-clause* $\Rightarrow$ $('t, 'v,$ *'ty*$)$ *typed-clause* $\Rightarrow$ *bool*
where
factoringI:
$D =$ *add-mset* l_1 (*add-mset* l_2 D') $\Longrightarrow$
$l_1 =$ *Pos* $t_1 \Longrightarrow$
$l_2 =$ *Pos* $t_2 \Longrightarrow$
$C =$ (*add-mset* l_1 D') $\cdot$ $\mu \Longrightarrow$
factoring $(\mathcal{V}, D)$ $(\mathcal{V}, C)$
if
select $D = \{\#\}$
is-maximal $(l_1 \cdot l \ \mu)$ $(D \cdot \mu)$
type-preserving-on (*clause.vars* D) $\mathcal{V}$ μ
term.is-imgu μ $\{\{t_1, t_2\}\}$

inductive *resolution* ::
$('t, 'v,$ *'ty*$)$ *typed-clause* $\Rightarrow$
$('t, 'v,$ *'ty*$)$ *typed-clause* $\Rightarrow$
$('t, 'v,$ *'ty*$)$ *typed-clause* $\Rightarrow$ *bool* **where**
resolutionI:
$E =$ *add-mset* l_1 $E' \Longrightarrow$
$D =$ *add-mset* l_2 $D' \Longrightarrow$
$l_1 =$ *Neg* $t_1 \Longrightarrow$
$l_2 =$ *Pos* $t_2 \Longrightarrow$
$C = (E' \cdot \varrho_1 + D' \cdot \varrho_2) \cdot \mu \Longrightarrow$
resolution $(\mathcal{V}_2, D)$ $(\mathcal{V}_1, E)$ $(\mathcal{V}_3, C)$
if

infinite-variables-per-type $\mathcal{V}_1$
infinite-variables-per-type $\mathcal{V}_2$
term.is-renaming ϱ_1
term.is-renaming ϱ_2
clause.vars $(E \cdot \varrho_1) \cap$ *clause.vars* $(D \cdot \varrho_2) = \{\}$
type-preserving-on (*clause.vars* $(E \cdot \varrho_1) \cup$ *clause.vars* $(D \cdot \varrho_2))$ $\mathcal{V}_3$ μ
term.is-imgu μ $\{\{t_1 \cdot t \varrho_1, t_2 \cdot t \varrho_2\}\}$
$\neg$ $(E \cdot \varrho_1 \odot \mu \preceq_c D \cdot \varrho_2 \odot \mu)$
select $E = \{\#\} \Longrightarrow$ *is-maximal* $(l_1 \cdot l \varrho_1 \odot \mu)$ $(E \cdot \varrho_1 \odot \mu)$
select $E \neq \{\#\} \Longrightarrow$ *is-maximal* $(l_1 \cdot l \varrho_1 \odot \mu)$ $((\textit{select}\ E) \cdot \varrho_1 \odot \mu)$
select $D = \{\#\}$
is-strictly-maximal $(l_2 \cdot l \varrho_2 \odot \mu)$ $(D \cdot \varrho_2 \odot \mu)$
$\forall x \in$ *clause.vars* E. $\mathcal{V}_1\ x = \mathcal{V}_3$ (*term.rename* $\varrho_1\ x$)
$\forall x \in$ *clause.vars* D. $\mathcal{V}_2\ x = \mathcal{V}_3$ (*term.rename* $\varrho_2\ x$)
type-preserving-on (*clause.vars* E) $\mathcal{V}_1$ ϱ_1
type-preserving-on (*clause.vars* D) $\mathcal{V}_2$ ϱ_2

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

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

definition *inferences* :: $('t, 'v, 'ty)$ *typed-clause inference set* **where**
inferences $\equiv$ *resolution-inferences* $\cup$ *factoring-inferences*

abbreviation $bottom_F$ :: $('t, 'v, 'ty)$ *typed-clause set* $(\bot_F)$ **where**
$bottom_F \equiv \{(\mathcal{V}, \{\#\}) \mid \mathcal{V}.$ *infinite-variables-per-type* $\mathcal{V}$ }

end

end
theory *Grounded-Ordered-Resolution*
imports
Ordered-Resolution
Ground-Ordered-Resolution

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

Polynomial-Factorization.Missing-List
begin

locale *grounded-ordered-resolution-calculus* =
ordered-resolution-calculus **where**
select = *select* **and** *welltyped* = *welltyped* **and**
term-from-ground = *term-from-ground* :: $'t_G \Rightarrow 't$ **and** *id-subst* = *id-subst* :: $'subst$ +

grounded-selection-function **where**
select = *select* **and** *atom-subst* = $(\cdot t)$ **and** *atom-vars* = *term.vars* **and**
atom-from-ground = *term.from-ground* **and** *atom-to-ground* = *term.to-ground*
and
is-ground-instance = *is-ground-instance*
for
select :: *'t select* **and**
welltyped :: *('v :: infinite, 'ty) var-types* ⇒ *'t* ⇒ *'ty* ⇒ *bool*
begin

sublocale *ground*: *ground-ordered-resolution-calculus* **where**
$less_t = (\prec_{tG})$ **and** $select = select_G$
rewrites
multiset-extension.multiset-extension $(\prec_{tG})$ *ground.literal-to-mset* = $(\prec_{lG})$ **and**
multiset-extension.multiset-extension $(\prec_{lG})$ $(\lambda x.\ x) = (\prec_{cG})$ **and**
$\bigwedge l_G\ C_G$. *ground.is-maximal* $l_G\ C_G \longleftrightarrow$ *ground-is-maximal* $l_G\ C_G$ **and**
$\bigwedge l_G\ C_G$. *ground.is-strictly-maximal* $l_G\ C_G \longleftrightarrow$ *ground-is-strictly-maximal* $l_G\ C_G$
⟨*proof*⟩

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

abbreviation *is-inference-ground-instance-two-premises* **where**
is-inference-ground-instance-two-premises $D\ E\ C\ \iota_G\ \gamma\ \varrho_1\ \varrho_2 \equiv$
case $(D,\ E,\ C)$ *of* $((\mathcal{V}_2,\ D),\ (\mathcal{V}_1,\ E),\ (\mathcal{V}_3,\ C)) \Rightarrow$
term.is-renaming ϱ_1 ∧
term.is-renaming ϱ_2 ∧
clause.vars $(E \cdot \varrho_1) \cap$ *clause.vars* $(D \cdot \varrho_2) = \{\}$ ∧
inference.is-ground (*Infer* $[D \cdot \varrho_2,\ E \cdot \varrho_1]\ C \cdot\iota\ \gamma$) ∧
ι_G = *inference.to-ground* (*Infer* $[D \cdot \varrho_2,\ E \cdot \varrho_1]\ C \cdot\iota\ \gamma$) ∧
type-preserving-on (*clause.vars* C) $\mathcal{V}_3\ \gamma$ ∧
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 $\iota\ \iota_G\ \gamma \equiv$
(*case* ι *of*
Infer $[D]\ C \Rightarrow$ *is-inference-ground-instance-one-premise* $D\ C\ \iota_G\ \gamma$
| *Infer* $[D,\ E]\ C \Rightarrow \exists\, \varrho_1\ \varrho_2$. *is-inference-ground-instance-two-premises* $D\ E\ C\ \iota_G\ \gamma\ \varrho_1\ \varrho_2$

```
    | - ⇒ False)
  ∧ ι_G ∈ ground.G-Inf

definition inference-ground-instances where
  inference-ground-instances ι = { ι_G | ι_G γ. is-inference-ground-instance ι ι_G γ }

lemma is-inference-ground-instance:
  is-inference-ground-instance ι ι_G γ ⟹ ι_G ∈ inference-ground-instances ι
  ⟨proof⟩

lemma is-inference-ground-instance-one-premise:
  assumes is-inference-ground-instance-one-premise D C ι_G γ ι_G ∈ ground.G-Inf
  shows ι_G ∈ 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 ι_G ∈
ground.G-Inf
  shows ι_G ∈ inference-ground-instances (Infer [D, E] C)
  ⟨proof⟩

lemma ground-inference_concl-in-ground-instances:
  assumes ι_G ∈ inference-ground-instances ι
  shows concl-of ι_G ∈ uncurried-ground-instances (concl-of ι)
⟨proof⟩

lemma ground-inference-red-in-ground-instances-of-concl:
  assumes ι_G ∈ inference-ground-instances ι
  shows ι_G ∈ ground.Red-I (uncurried-ground-instances (concl-of ι))
⟨proof⟩

sublocale lifting:
  tiebreaker-lifting
    ⊥_F
    inferences
    ground.G-Bot
    ground.G-entails
    ground.G-Inf
    ground.GRed-I
    ground.GRed-F
    uncurried-ground-instances
    Some ∘ inference-ground-instances
    typed-tiebreakers
⟨proof⟩
end

context ordered-resolution-calculus
begin
```

abbreviation *grounded-inference-ground-instances* **where**
grounded-inference-ground-instances select$_G$ ≡
grounded-ordered-resolution-calculus.inference-ground-instances
(⊙) *apply-subst* (·*t*) *term.vars term.to-ground* (≺$_t$) *id-subst term.from-ground*
select$_G$ *welltyped*

sublocale
lifting-intersection
inferences
{{#}}
select$_{Gs}$
ground-ordered-resolution-calculus.G-Inf (≺$_{tG}$)
λ-. *ground-ordered-resolution-calculus.G-entails*
ground-ordered-resolution-calculus.GRed-I (≺$_{tG}$)
λ-. *ground-ordered-resolution-calculus.GRed-F* (≺$_{tG}$)
⊥$_F$
λ-. *uncurried-ground-instances*
λ*select*$_G$. *Some* ∘ *grounded-inference-ground-instances select*$_G$
typed-tiebreakers
⟨*proof*⟩

end

end
theory *Ordered-Resolution-Soundness*
imports *Grounded-Ordered-Resolution*
begin

1.7 Soundness

context *grounded-ordered-resolution-calculus*
begin

notation *lifting.entails-*$\mathcal{G}$ (**infix** ⊨$_F$ *50*)

lemma *factoring-sound*:
assumes *factoring*: *factoring D C*
shows {*D*} ⊨$_F$ {*C*}
⟨*proof*⟩

lemma *resolution-sound*:
assumes *resolution*: *resolution D E C*
shows {*E*, *D*} ⊨$_F$ {*C*}
⟨*proof*⟩

sublocale *sound-inference-system inferences* ⊥$_F$ (⊨$_F$)
⟨*proof*⟩

end

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

end
theory *Ordered-Resolution-Completeness*
imports
Grounded-Ordered-Resolution
Ground-Ordered-Resolution-Completeness
begin

2 Completeness

context *grounded-ordered-resolution-calculus*
begin

2.1 Liftings

lemma *factoring-lifting*:
fixes
D_G C_G :: $'t_G$ *clause* **and**
D C :: $'t$ *clause* **and**
γ :: $'subst$
defines
[*simp*]: $D_G \equiv$ *clause.to-ground* $(D \cdot \gamma)$ **and**
[*simp*]: $C_G \equiv$ *clause.to-ground* $(C \cdot \gamma)$
assumes
ground-factoring: *ground.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**
type-preserving-γ: *type-preserving-on* $(clause.vars\ D)\ \mathcal{V}\ \gamma$ **and**
$\mathcal{V}$: *infinite-variables-per-type* $\mathcal{V}$
obtains C'
where
factoring $(\mathcal{V}, D)\ (\mathcal{V}, C')$
Infer $[D_G]\ C_G \in$ *inference-ground-instances* $(Infer\ [(\mathcal{V}, D)]\ (\mathcal{V}, C'))$
$C' \cdot \gamma = C \cdot \gamma$
⟨*proof*⟩

lemma *resolution-lifting*:
fixes
E_G D_G C_G :: $'t_G$ *clause* **and**
E D C :: $'t$ *clause* **and**
γ ϱ_1 ϱ_2 :: $'subst$ **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*]: $N_G \equiv$ *ground-instances* $\mathcal{V}_1$ $E \cup$ *ground-instances* $\mathcal{V}_2$ D **and**
[*simp*]: $\iota_G \equiv$ *Infer* $[D_G, E_G]$ C_G
assumes
ground-resolution: *ground.resolution* D_G E_G C_G **and**
ϱ_1: *term.is-renaming* ϱ_1 **and**
ϱ_2: *term.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**
type-preserving-ϱ_1*-*γ: *type-preserving-on* (*clause.vars* E) $\mathcal{V}_1$ $(\varrho_1 \odot \gamma)$ **and**
type-preserving-ϱ_2*-*γ: *type-preserving-on* (*clause.vars* D) $\mathcal{V}_2$ $(\varrho_2 \odot \gamma)$ **and**
type-preserving-ϱ_1: *type-preserving-on* (*clause.vars* E) $\mathcal{V}_1$ ϱ_1 **and**
type-preserving-ϱ_2: *type-preserving-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* $\mathcal{V}_2$
obtains C' $\mathcal{V}_3$
where
resolution $(\mathcal{V}_2, D)$ $(\mathcal{V}_1, E)$ $(\mathcal{V}_3, C')$
$\iota_G \in$ *inference-ground-instances* (*Infer* $[(\mathcal{V}_2, D), (\mathcal{V}_1, E)]$ $(\mathcal{V}_3, C')$)
$C' \cdot \gamma = C \cdot \gamma$
⟨*proof*⟩

2.2 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* $select_G$ $(\bigcup$(*uncurried-ground-instances* ‘ N))
begin

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

lemma *resolution-ground-instance*:
assumes *ground-resolution*: $\iota_G \in$ *ground.resolution-inferences*
obtains ι **where**
$\iota \in$ *Inf-from* N

$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

lemma *ground-instances*:
obtains ι **where**
$\iota \in$ *Inf-from N*
$\iota_G \in$ *inference-ground-instances* ι
⟨*proof*⟩

end

end

context *ordered-resolution-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 *Ordered-Resolution-Welltypedness-Preservation*
imports *Grounded-Ordered-Resolution*
begin

context *ordered-resolution-calculus*
begin

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

lemma *resolution-preserves-typing*:
assumes
resolution: *resolution* $(\mathcal{V}_2,\ D)$ $(\mathcal{V}_1,\ E)$ $(\mathcal{V}_3,\ C)$ **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*⟩

end

end
theory *Untyped-Ordered-Resolution*
imports
First-Order-Clause.Nonground-Order
First-Order-Clause.Nonground-Selection-Function
First-Order-Clause.Tiebreakers

Fresh-Identifiers.Fresh
begin

locale *untyped-ordered-resolution-calculus* =
nonground-order **where**
$less_t = less_t$ **and** *id-subst* = *id-subst* **and** *term-from-ground* = *term-from-ground* :: $'t_G \Rightarrow 't$ **and**
term-vars = *term-vars* +

nonground-selection-function **where**
select = *select* **and** *atom-subst* = $(\cdot t)$ **and** *atom-vars* = *term.vars* **and** *term-vars* = *term-vars* **and**
atom-from-ground = *term.from-ground* **and** *atom-to-ground* = *term.to-ground* **and** *id-subst* = *id-subst* +

tiebreakers tiebreakers +
term: *exists-imgu* **where** *vars* = *term-vars* **and** *subst* = $(\cdot t)$ **and** *id-subst* = *id-subst*
for
select :: $'t$ *select* **and**
$less_t$ :: $'t \Rightarrow 't \Rightarrow$ *bool* **and**
tiebreakers :: $('t_G, 't)$ *tiebreakers* **and**
id-subst :: $'subst$ **and**
term-vars :: $'t \Rightarrow ('v ::$ *infinite*$)$ *set*
begin

inductive *factoring* :: $'t$ *clause* $\Rightarrow$ $'t$ *clause* $\Rightarrow$ *bool* **where**
factoringI:
$D =$ *add-mset* l_1 (*add-mset* l_2 D') $\Longrightarrow$
$l_1 =$ *Pos* $t_1 \Longrightarrow$
$l_2 =$ *Pos* $t_2 \Longrightarrow$
$C = (\text{add-mset } l_1\ D') \cdot \mu \Longrightarrow$
factoring D C
if
select $D = \{\#\}$
is-maximal $(l_1 \cdot l\ \mu)$ $(D \cdot \mu)$
term.is-imgu μ $\{\{t_1, t_2\}\}$

inductive *resolution* :: $'t$ *clause* $\Rightarrow$ $'t$ *clause* $\Rightarrow$ $'t$ *clause* $\Rightarrow$ *bool* **where**
resolutionI:
$E =$ *add-mset* l_1 $E' \Longrightarrow$

$D = \mathit{add\text{-}mset}\ l_2\ D' \Longrightarrow$
$l_1 = \mathit{Neg}\ t_1 \Longrightarrow$
$l_2 = \mathit{Pos}\ t_2 \Longrightarrow$
$C = (E' \cdot \varrho_1 + D' \cdot \varrho_2) \cdot \mu \Longrightarrow$
$\mathit{resolution}\ D\ E\ C$
if
$\mathit{term.is\text{-}renaming}\ \varrho_1$
$\mathit{term.is\text{-}renaming}\ \varrho_2$
$\mathit{clause.vars}\ (E \cdot \varrho_1) \cap \mathit{clause.vars}\ (D \cdot \varrho_2) = \{\}$
$\mathit{term.is\text{-}imgu}\ \mu\ \{\{t_1 \cdot t\ \varrho_1,\ t_2 \cdot t\ \varrho_2\}\}$
$\neg\ (E \cdot \varrho_1 \odot \mu \preceq_c D \cdot \varrho_2 \odot \mu)$
$\mathit{select}\ E = \{\#\} \Longrightarrow \mathit{is\text{-}maximal}\ (l_1 \cdot l\ \varrho_1 \odot \mu)\ (E \cdot \varrho_1 \odot \mu)$
$\mathit{select}\ E \neq \{\#\} \Longrightarrow \mathit{is\text{-}maximal}\ (l_1 \cdot l\ \varrho_1 \odot \mu)\ (\mathit{select}\ E \cdot \varrho_1 \odot \mu)$
$\mathit{select}\ D = \{\#\}$
$\mathit{is\text{-}strictly\text{-}maximal}\ (l_2 \cdot l\ \varrho_2 \odot \mu)\ (D \cdot \varrho_2 \odot \mu)$

abbreviation *factoring-inferences* **where**
$\mathit{factoring\text{-}inferences} \equiv \{\ \mathit{Infer}\ [D]\ C \mid D\ C.\ \mathit{factoring}\ D\ C\ \}$

abbreviation *resolution-inferences* **where**
$\mathit{resolution\text{-}inferences} \equiv \{\ \mathit{Infer}\ [D,\ E]\ C \mid D\ E\ C.\ \mathit{resolution}\ D\ E\ C\ \}$

definition *inferences* :: *'t clause inference set* **where**
$\mathit{inferences} \equiv \mathit{resolution\text{-}inferences} \cup \mathit{factoring\text{-}inferences}$

abbreviation *bottom* :: *'t clause set* **where**
$\mathit{bottom} \equiv \{\{\#\}\}$

end

end
theory *Untyped-Ordered-Resolution-Inference-System*
imports
Untyped-Ordered-Resolution
First-Order-Clause.Untyped-Calculus
Grounded-Ordered-Resolution
begin

context *untyped-ordered-resolution-calculus*
begin

sublocale *typed*: *ordered-resolution-calculus* **where**
$\mathit{welltyped} = \lambda\text{- - }().\ \mathit{True}$
⟨*proof*⟩

declare
typed.term.welltyped-renaming [*simp del*]
typed.term.welltyped-subst-stability [*simp del*]
typed.term.welltyped-subst-stability' [*simp del*]

abbreviation *entails* **where**
entails N N′ ≡ *typed.entails-G* (*empty-typed* ‘ *N*) (*empty-typed* ‘ *N′*)

sublocale *untyped-consequence-relation* **where**
typed-bottom = $\bot_F$ **and** *typed-entails* = *typed.entails-G* **and**
bottom = *bottom* **and** *entails* = *entails*
⟨*proof*⟩

sublocale *untyped-inference-system* **where**
inferences = *inferences* **and** *typed-inferences* = *typed.inferences*
⟨*proof*⟩

end

end
theory *Untyped-Ordered-Resolution-Completeness*
imports
Untyped-Ordered-Resolution-Inference-System
Ordered-Resolution-Completeness
begin

context *untyped-ordered-resolution-calculus*
begin

abbreviation *Red-F* **where**
Red-F N ≡ *snd* ‘ *typed.Red-F-G* (*empty-typed* ‘ *N*)

abbreviation *Red-I* **where**
Red-I N ≡ *remove-types* ‘ *typed.Red-I-G* (*empty-typed* ‘ *N*)

sublocale *untyped-complete-calculus* **where**
typed-bottom = $\bot_F$ **and** *typed-entails* = *typed.entails-G* **and**
typed-inferences = *typed.inferences* **and** *typed-Red-I* = *typed.Red-I-G* **and**
typed-Red-F = *typed.Red-F-G* **and** *bottom* = *bottom* **and** *inferences* = *inferences*
and *Red-F* = *Red-F* **and**
Red-I = *Red-I* **and** *entails* = *entails*
⟨*proof*⟩

end

end
theory *Untyped-Ordered-Resolution-Soundness*
imports
Untyped-Ordered-Resolution-Inference-System
Ordered-Resolution-Soundness
begin

context *untyped-ordered-resolution-calculus*
begin

sublocale *untyped-sound-inference-system* **where**
typed-bottom = $\bot_F$ **and** *typed-entails* = *typed.entails-*$\mathcal{G}$ **and**
typed-inferences = *typed.inferences* **and** *bottom* = *bottom* **and** *inferences* = *inferences* **and**
entails = *entails*
⟨*proof*⟩

end

end
theory *Monomorphic-Ordered-Resolution*
imports
Ordered-Resolution

First-Order-Clause.IsaFoR-Nonground-Clause
First-Order-Clause.Monomorphic-Typing
begin

locale *monomorphic-ordered-resolution-calculus* =
monomorphic-term-typing +

ordered-resolution-calculus **where**
comp-subst = ($\circ_s$) **and** *id-subst* = *Var* **and** *term-subst* = ($\cdot$) **and** *term-vars* = *term.vars* **and**
apply-subst = *apply-subst* **and** *subst-update* = *fun-upd* **and** *subst-updates* = *subst-updates* **and**
term-from-ground = *term.from-ground* **and** *term-to-ground* = *term.to-ground* **and**
welltyped = *welltyped*

end
theory *Ordered-Resolution-Example*
imports
Monomorphic-Ordered-Resolution
First-Order-Clause.IsaFoR-KBO
begin

hide-type *Uprod-Literal-Functor.clause*

abbreviation *trivial-tiebreakers* ::
$'f$ *gterm clause* ⇒ ($'f$,$'v$) *term clause* ⇒ ($'f$,$'v$) *term clause* ⇒ *bool* **where**
trivial-tiebreakers ≡ $\bot$

abbreviation *trivial-select* :: $'a$ *clause* ⇒ $'a$ *clause* **where**

trivial-select - ≡ {#}

abbreviation *unit-typing* **where**
unit-typing - - ≡ *Some* ([], ())

interpretation *unit-types*: *monomorphic-term-typing* **where** $\mathcal{F}$ = *unit-typing*
⟨*proof*⟩

interpretation *example1*: *monomorphic-ordered-resolution-calculus* **where**
select = *trivial-select* :: ((′*f* :: *weighted* , ′*v* :: *infinite*) *term*) *select* **and**
$less_t$ = *less-kbo* **and**
$\mathcal{F}$ = *unit-typing* **and**
tiebreakers = *trivial-tiebreakers*
⟨*proof*⟩

instantiation *nat* :: *infinite*
begin

instance
⟨*proof*⟩

end

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

abbreviation *types* :: *nat* ⇒ *nat* ⇒ (*type list* × *type*) *option* **where**
types f n ≡
let type = *if even f then A else B*
in Some (*replicate n type*, *type*)

interpretation *example-types*: *monomorphic-term-typing* **where** $\mathcal{F}$ = *types*
⟨*proof*⟩

interpretation *example2*: *monomorphic-ordered-resolution-calculus* **where**
select = *trivial-select* :: (*nat*, *nat*) *term select* **and**
$less_t$ = *less-kbo* **and**
$\mathcal{F}$ = *types* **and**
tiebreakers = *trivial-tiebreakers*
⟨*proof*⟩

end