The Superposition Calculus

Hans de Nivelle.

The goal is to extend resolution to equality.

We first discuss the problems with explicit axiomatization of equality, and after that introduce the superposition calculus.

We give highlights of its completeness proof.

All the essential ingredients of the superposition calculus (and also of its completeness proof) have been covered in the slide series propres and predres.

Equality

Definition: We introduce atoms of form $t_1 \approx t_2$, denoting t_1 equals t_2.

We assume that $t_1 \approx t_2$ and $t_2 \approx t_1$ are the same atom.

Explicit Axiomatization

A possible approach to equality is explicit axiomatization:

EQREFL $[X \approx X]$,

EQTRANS $[X \not\approx Y,\ Y \not\approx Z,\ X \approx Z]$,

EQFUNC For each function symbol, introduce the clause

$$[X_1 \not\approx Y_1, \ldots, X_n \not\approx Y_n,\ f(X_1, \ldots, X_n) \approx f(Y_1, \ldots, Y_n)],$$

EQPRED For each predicate symbol, introduce the clause

$$[X_1 \not\approx Y_1, \ldots, X_n \not\approx Y_n,\ \neg p(X_1, \ldots, X_n),\ p(Y_1, \ldots, Y_n)].$$

This works in theory and one can solve some easy problems like this.

For practical use, these axioms are too productive, because they can resolve with each other, and with atoms in the problem in too may ways.

Use of a selection function, or A-ordering that not change this.

Superposition/ Paramodulation

In order to deal with equality directly, resolution is replaced by superposition.

Analogeous to resolution without equality, superposition is defined on the ground level, and after that lifted to the variable level.

Informally, ground superposition is the following rule:

From $[\ A[t_1]\ ] \cup R_1$ and $[\ t_1 \approx t_2\ ] \cup R_2$ derive $[\ A[t_2]\ ] \cup R_1 \cup R_2$.

There is an order $\succ$ on terms and equalities which is used to restrict the possible inferences, (similar to A-orders)

This order decides which equalities can be used for rewriting, which literals can be rewritten into, and also in which direction the equality $t_1 \approx t_2$ can be applied.

Reduction Orders

Definition: A reduction order is a total order on ground terms that satisfies the following additional conditions:

WF In every non-empty set of terms T over a finite signature there is at least one minimal term t. This is a term $t \in T$, such that there is no $t' \in T$ with $t \succ t'$.

CONT The order $\succ$ is preserved in contexts: If $t_1 \succ t_2$, then $t[t_1] \succ t[t_2]$.

T The truth constant **true** is minimal: If $t \neq \mathbf{true}$, then $t \succ \mathbf{true}$.

Condition WF ensures that $\succ$ can be used for transfinite induction.

In propositional logic, signatures were finite, so that WF is implicit.

The reason for condition T will come later.

Obtaining reduction orders is much harder than obtaining L-orders, mainly due to condition WF.

Simple lexicographic comparison on prefix representations is not WF.

If one puts $a \succ s$, then $a \succ s(a) \succ s(s(a)) \succ \cdots$, so that the set $\{a, s(a), s(s(a)), s^3(a), s^4(a), \ldots\}$ has no minimal element.

The simplest reduction orders are Knuth-Bendix orders. These are obtained by first comparing weight, and if the weights are equal, then using lexicographic comparison to break the tie.

Knuth-Bendix Orders

Assume a function # that assigns a weight to every symbol.
Assume a total order $\succ$ on the symbols.

The weight function # can be extended to terms in the obvious way.

The Knuth-Bendix Order $\succ$ is defined as follows:

$t_1 \succ t_2$ if

1. $\#t_1 > \#t_2$ or
2. $\#t_1 = \#t_2$ and $t_1 \succ t_2$, using the lexicographic extension of $\succ$ on the prefix representations.

(Note: This is not the original definition by KB. You will get acquainted with the original version in the exercises)

Ground Superposition

POSSUPERPOS Let $[t_1 \approx t_2] \cup R_1$ and $[u_1[t_1] \approx u_2] \cup R_2$ be clauses, s.t.

$$t_1 \succ t_2, \quad u_1[t_1] \succ u_2,$$

$$(t_1 \approx t_2) \succ R_1, \quad (u_1[t_1] \approx u_2) \succ R_2.$$

Then the clause $[u_1[t_2] \approx u_2] \cup R_1 \cup R_2$ is obtained by positive superposition.

NEGSUPERPOS Let $[t_1 \approx t_2] \cup R_1$ and $[u_1[t_1] \not\approx u_2] \cup R_2$ be clauses, s.t.

$$t_1 \succ t_2, \quad u_1[t_1] \succ u_2,$$

$$(t_1 \approx t_2) \succ R_1, \quad (u_1[t_1] \not\approx u_2) \succ R_2.$$

Then the clause $[u_1[t_2] \not\approx u_2] \cup R_1 \cup R_2$ is obtained by negative superposition.

Ground Superposition (2)

EQREFL Let $[t \not\approx t] \cup R$ be a clause, s.t.

$$(t \not\approx t) \succeq R.$$

Then R is obtained by equality reflexivity.

EQFACT Let $[t_1 \approx t_2,\ t_1 \approx t_3\ ] \cup R$ be a clause, s.t.

$$t_1 \succ t_2 \text{ and } t_2 \succeq t_3.$$

Then the clause

$$[\ t_1 \approx t_2,\ t_2 \not\approx t_3\ ] \cup R$$

is obtained by equality factoring.

Selection Functions

Selection functions can be defined in the same way as for propositional resolution. The selection function $\Sigma(c)$ is either empty, or it selects a subset of negative literals that override the order. The selection function has no influence on which side of an equality is used, also not for negative equalities.

Elimination of Non-Equality

Positive non-equality literals of form $p(t_1, \ldots, t_n)$ are replaced by $p(t_1, \ldots, t_n) \approx \mathbf{true}$.

Negative non-equality literals of form $\neg p(t_1, \ldots, t_n)$ are replaced by $p(t_1, \ldots, t_n) \not\approx \mathbf{true}$.

The only predicate symbol is equality!

From $[p(t_1, \ldots, t_n) \approx \mathbf{true}] \cup R_1$ and $[p(t_1, \ldots, t_n) \not\approx \mathbf{true}] \cup R_2$, one can obtain $[\mathbf{true} \not\approx \mathbf{true}] \cup R_1 \cup R_2$ by negative superposition.

This can be simplified into

$$R_1 \cup R_2.$$

From $[p(t_1, \ldots, t_n) \approx \mathbf{true},\ p(t_1, \ldots, t_n) \approx \mathbf{true}] \cup R$ one can derive $[p(t_1, \ldots, t_n) \approx \mathbf{true},\ \mathbf{true} \not\approx \mathbf{true}] \cup R$ by equality factoring.

This can be simplified into

$$[\ p(t_1, \ldots, t_n) \approx \mathbf{true}\ ] \cup R.$$

Extending the Reduction Order

First, the reduction order $\succ$ is extended to equalities/disequalities as follows:

1. Sort (dis)equality: $t_1 \approx t_2$, s.t. $t_1 \succ t_2$.
2. Then write in infix notation: $t_1 \approx t_2$ or $t_1 \not\approx t_2$. Assume that $\approx \prec \not\approx$.
3. Finally, compare atoms from left to right.

(Note that the extension to negative atoms differs from the extension of A-orders. This is needed in the completeness proof)

Ranking/Redundancy

A ranking for clauses can be obtained in the same way as for propositional resolution. (Clauses are sorted, and then compared from left to right)

Definition: Clauses $C_1, \ldots, C_n$ make clause D redundant if D is a logical consequence of $C_1, \ldots, C_n$ and all of $C_1, \ldots, C_n$ have rank lower than D.

Semantics of equality is taken into account.

Examples of Redundancy

- $[a \approx b]$ makes $[s(a) \approx s(b)]$ redundant.
- $[a \approx b]$ and $[b \approx c]$ make $[s(a) \approx s(c)]$ redundant.
- $[a \approx b]$ and $[p(b) \approx \mathbf{true}]$ make $[p(a) \approx \mathbf{true}]$ redundant, if $a \succ b$.
- If $a \succ b \succ c$, then $b \approx c$ and $a \approx c$ make $a \approx b$ redundant.

Non-ground Superposition

POSSUPERPOS Let $[t_1 \approx t_2] \cup R_1$ and $[u_1[t_1'] \approx u_2] \cup R_2$ be clauses without overlapping variables.

Assume that t_1' is not a variable.

Assume that the system of constraints

$$t_1\Theta = t_1'\Theta,$$

$$t_1\Theta \succ t_2\Theta, \quad u_1[t_1']\Theta \succ u_2\Theta,$$

$$(t_1\Theta \approx t_2\Theta) \succ R_1\Theta, \quad (u_1[t_1']\Theta \approx u_2\Theta) \succ R_2\Theta$$

has a solution.

Then let Θ be the mgu of t_1 and t_1'.

The clause

$$[u_1[t_2]\Theta \approx u_2\Theta] \cup R_1\Theta \cup R_2\Theta$$

is obtained by positive superposition.

Non-ground Superposition (2)

NEGSUPERPOS Let $[t_1 \approx t_2] \cup R_1$ and $[u_1[t_1'] \not\approx u_2] \cup R_2$ be clauses without overlapping variables.

Assume that t_1' is not a variable.

Assume that the system of constraints

$$t_1\Theta = t_1'\Theta,$$

$$t_1\Theta \succ t_2\Theta, \quad u_1[t_1']\Theta \succ u_2\Theta,$$

$$(t_1\Theta \approx t_2\Theta) \succ R_1\Theta, \quad (u_1[t_1']\Theta \not\approx u_2\Theta) \succ R_2\Theta$$

has a solution.

Then let Θ be the mgu of t_1 and t_1'. The clause

$$[u_1[t_2]\Theta \not\approx u_2\Theta] \cup R_1\Theta \cup R_2\Theta$$

is obtained by negative superposition.

Non-ground Superposition (3)

EQREFL Let $[t_1 \not\approx t_2] \cup R$ be a clause, s.t. the system of constraints

$$t_1\Theta = t_2\Theta, \quad (t_1\Theta \not\approx t_2\Theta) \succeq R\Theta$$

has a solution. Let Θ be the mgu of t_1 and t_2. Then the clause $R\Theta$ is obtained by equality reflexivity.

EQFACT Let $[t_1 \approx t_2,\ t_1' \approx t_3\ ] \cup R$ be a clause, s.t. the system of constraints

$$t_1\Theta = t_1'\Theta, \quad t_1\Theta \succ t_2\Theta, \quad t_2\Theta \succeq t_3\Theta$$

has a solution.

Let Θ be the mgu of t_1 and t_1'. Then the clause

$$[\ t_1\Theta \approx t_2\Theta,\ t_2\Theta \not\approx t_3\Theta\ ] \cup R\Theta$$

is obtained by equality factoring.

Non-ground Superposition (4)

In the rules, the condition that t_1' is not a variable, is very useful. If t_1' is a variable, then every possible t_1 unifies with t_1'.

On the ground level, EQREFL is a simplification rule. On the predicate level, it is not.

Redundancy (2)

As with predicate logic, checking for redundancy is hard. Therefore, only rude appoximations have been implemented until now.

DEMODULATION If c has form $[t_1 \approx t_2] \cup R_1$, $\quad d$ has form $[\ A[t]\ ] \cup S$, there exists a substitution Θ, s.t.

$$t_1\Theta = t, \text{ and } R_1\Theta \text{ implies } S.$$

And, for every substitution Σ, $\quad c\Theta\Sigma$ is ranked before $d\Theta\Sigma$, then d is deleted and replaced by

$$[\ A[t_2\Theta]\ ] \cup S.$$

(For 'implies', usually 'is a subset of' is taken)

Redundancy (3)

TAUTOLOGY If c has form

$$[\, t_1 \not\approx u_1, \ldots, t_n \not\approx u_n, \quad v_1 \approx w_1, \ldots, w_m \approx w_m \,]$$

and for some $1 \leq i \leq m$,

$$t_1 \approx u_1, \ldots, t_n \approx u_n \vdash v_i \approx w_i$$

is true, then c is a tautology. Testing this is easy, and the algorithm is very elegant.

Tautologies do not occur often by themselves, but often DEMODULATION creates a tautology.

Redundancy (4)

EQREFL Let c be a clause of form $[t \not\approx t] \cup R$. It can be replaced by R.

SUPERPOS When the from clause is a unit clause, and the other clause is not instantiated, then both POSSUPERPOS and NEGSUPERPOS are simplification rules.

Completeness

The completeness proof has the same structure as the completeness proof for propositional, A-ordered resolution, but only infinitely more complicated.

Let S be a saturated set. As with propositional resolution, the set is sorted, using the ranking based on $\succ$. (Here WF is needed) After that, we inspect the clauses, and construct an increasing sequence of interpretations $I_0, I_1, I_2, \ldots$.

Interpretations

Definition An interpretation I is a set of rewrite rules, s.t.

- For each rewrite rule $(t_1 \Rightarrow t_2) \in I, \quad t_1 \succ t_2$.
- For each pair of rewrite rules $(t_1 \Rightarrow t_2), (u_1 \Rightarrow u_2) \in I$, if t_1 is a subterm of u_1, then $t_1 = u_1$ and $t_2 = u_2$.

Given a term t, the normal form of t under I, written as $I(t)$ is the term that one obtains if one applies rewrite rules from I on t as long as possible.

Theorem: Let I be an interpretation, let t be a term. The term t has exactly one normal form.

proof: By WF and confluence.

Given an interpretation I, a clause C is true in I, if either

- it contains an equality $t_1 \approx t_2$, s.t. $I(t_1) = I(t_2)$, or
- it contains a disequality $t_1 \not\approx t_2$, s.t. $I(t_1) \neq I(t_2)$.

Model Construction

Let S be a saturated set. We construct a sequence of interpretations $I_0, I_1, \ldots,$ as follows:

$I_0 = \{\ \}$.

At each rank i, let $C_i \in S$ be the clause with rank i.

- If C_i is false in I_i, and the maximal literal in C_i is a positive equality of form $t_1 \approx t_2$ with $t_1 \neq t_2$, no literal is selected in C_i, then one may assume without loss of generality that $t_1 \succ t_2$.

  If there no equality $t_1 \approx t_3$ in C, such that $I_i(t_2) = I_i(t_3)$, and $I_i(t_1) = t_1$, then put $I_{i+1} = I_i \cup \{t_1 \Rightarrow t_2\}$.

- Otherwise, put $I_{i+1} = I_i$.

We have to show that I_n is an interpretation.

It is easy to see that for every $t_1 \Rightarrow t_2 \in I_n, \quad t_1 \succ t_2$.

Suppose that there are rules $t_1 \Rightarrow t_2, \;\; u_1 \Rightarrow u_2 \in I$, s.t. t_1 is a subterm of u_1, and either $t_1 \neq u_1$, or $t_2 \neq u_2$.

By the construction, there is a clause $[t_1 \approx t_2] \cup R_1$ which caused the insertion of $t_1 \Rightarrow t_2$.

There also is a clause $[u_1 \approx u_2] \cup R_2$ which caused the insertion of $u_1 \Rightarrow u_2$.

Clearly, $[t_1 \approx t_2] \cup R_1$ is ranked before $[u_1 \approx u_2] \cup R_2$.

Let i be the rank of $[u_1 \approx u_2] \cup R_2$. Because $(t_1 \Rightarrow t_2) \in I_i$, it must be the case that $I_n(u_1) \neq u_1$. This contradicts the fact that $(u_1 \Rightarrow u_2)$ was added.

Lemma A

For every rewrite rule $t_1 \Rightarrow t_2 \ \in I_n$, there is a clause of form $[t_1 \approx t_2] \cup R \ \in S$, s.t. $(t_1 \approx t_2) \succ R$, nothing is selected in $[t_1 \approx t_2] \cup R$, and R is false in I_n.

proof It is not too hard to prove that for every rewrite rule $t_1 \Rightarrow t_2 \in I_n$, there exists a clause c_i, which has form $[t_1 \approx t_2] \cup R$, nothing is selected in c_i, and $t_1 \approx t_2$ is maximal in c_i.

If adding $t_1 \Rightarrow t_2$ accidentily makes an equality in R true, this equality must have form $t_1 \approx t_3$, and also $I(t_2) = I(t_3)$. Then $t_1 \Rightarrow t_2$ would not have been added. It follows that R is false in I_{i+1}.

In order to show that R remains false up to I_n, one needs an argument similar to the argument for propositional reasoning.

(Somewhere on this slide, the reason for equality factoring is hiding. Where?)

Summary of the rest of the proof:

All the remaining clauses c_i that did not contribute to I_n, can be used in an inference with a conclusion c_j that is ranked before c_i. (or made redundant by clauses that are ranked before c_i)

Using induction, we assume that c_j is true in I_n, and show that this implies that c_i is true in I_n.

This is done by a case analysis on the reason why c_i did not contribute to I_n.

Lifting

As for logic without equality, lifting theorems can be proven for the rules of the superposition calculus.

For EQREFL, and EQFACT, lifting is analogeous to lifting for resolution.

For POSSUPERPOS and NEGSUPERPOS, there is a problem:

Failure of Lifting for Superposition

$$[a \approx b] \text{ and } [s(a) \approx s(c),\ p(s(a)) \approx \mathbf{true}]$$

$$\Rightarrow [s(b) \approx s(c),\ p(s(a)) \approx \mathbf{true}].$$

Now look at the variable clauses:

$$[a \approx b] \text{ and } [s(X) \approx s(c),\ p(s(X)) \approx \mathbf{true}].$$

$$[a \approx b] \text{ and } [X \approx s(c),\ p(X) \approx \mathbf{true}].$$

In the first clause, superposition is forbidden because the position of $s(a)$ has changed into a variable. In the second clause, the position of a is completely gone.

Solution: Change the substitution!

If equality replacement in the ground clause takes place at a position which is still present in the variable clause, one can construct a unifier, and there is no problem.

Otherwise, it takes place on a position that is 'inside' or 'at' a variable X.

Saving of Lifting for Superposition

$$[a \approx b] \text{ and } [s(a) \approx s(c),\ p(s(a)) \approx \mathbf{true}]$$

$$\Rightarrow [s(b) \approx s(c),\ p(s(a)) \approx \mathbf{true}].$$

For variable clause

$$[a \approx b] \text{ and } [s(X) \approx s(c),\ p(s(X)) \approx \mathbf{true}],$$

replace instance $\{X := a\}$ by $\{X := b\}$.

This results in the ground clause

$$[s(b) \approx s(c),\ p(s(b)) \approx \mathbf{true}].$$

This clause makes $[s(b) \approx s(c),\ p(s(a)) \approx \mathbf{true}]$ redundant.

In

$$[X \approx s(c),\ p(X) \approx \mathbf{true}],$$

replace $\{X := s(a)\}$ by $\{X := s(b)\}$.

Let $C_1 = [t_1 \approx t_2] \cup R_1$ be a clause.

Let $C_2 = [\ u_1[X] \approx u_2[X]\ ] \cup R_2(X)$ be a clause with instance

$$[\ u_1[\ t[t_1]\ ] \approx u_2[\ t[t_1]\ ]\ ] \cup R_2(\ t[t_1]\ )$$

using substitution $X := t[t_1]$.

Chanching the substitution into $X := t[t_2]$ results in the ground clause

$$[\ u_1[\ t[t_2]\ ] \approx u_2[\ t[t_2]\ ]\ ] \cup R_2(\ t[t_2]\ ).$$

This clause, together with C_1 makes the result of positive superposition redundant:

$$[\ u_1[\ t[t_2]\ ] \approx u_2[\ t[t_1]\ ]\ ] \cup R_1 \cup R_2(\ t[t_1]\ )$$

Summary, Conclusion

We defined superposition, which is a specialized, highly-optimized, and complicated method for dealing with predicate logic with equality.

Currently, it is the best known method.