Non-Ground Superposition

Lifting (for Resolution)

Let $c_1 = [\ A\ ] \cup R_1$, let $c_2 = [\ \neg A\ ]\ \cup R_2$ be two ground clauses. The clauses can resolve into $R_1 \cup R_2$.

Let c_1' and c_2' be two non-ground clauses, s.t. c_1 is an instance of c_1', and c_2 is an instance of c_2'.

First make sure that c_1' and c_2' have no variables in common. After this, there exists a single substitution Θ, s.t. $c_1'\Theta = c_1$ and $c_2'\Theta = c_2$.

c_1' and c_2' can be written in the form

$$\begin{cases} c_1' &= [\ A_1'\ ] \cup R_1' \\ c_2' &= [\ \neg A_2'\ ] \cup R_2' \end{cases}$$

with $A_1'\Theta = A, \quad A_2'\Theta = A, \quad R_1'\Theta = R_1, \quad R_2'\Theta = R_2.$

We obviously have $A_1'\Theta = A_2'\Theta$, so that A_1 and A_2 are unifiable. Let Σ be a most general unifier. Using Σ, we can construct the resolvent

$$R_1'\Sigma \cup R_2'\Sigma.$$

We want to show that $R_1 \cup R_2$ is an instance of $R_1'\Sigma \cup R_2'\Sigma$.

By definition of mgu, there exists a Σ', s.t. $\Theta = \Sigma \cdot \Sigma'$. It follows that

$$(R_1'\Sigma \cup R_2'\Sigma)\Sigma' = R_1'(\Sigma \cdot \Sigma') \cup R_2'(\Sigma \cdot \Sigma') = R_1'\Theta \cup R_2'\Theta = R_1 \cup R_2.$$

Lifting of Paramodulation

If we are lucky, then everything is easy:

Ground:

$$\begin{cases} [\ f(a,b) \approx f^{-1}(b,a),\ R(a,b)\ ] \\ [\ p(f(a,b)),\ q(f(a,b))] \end{cases}$$

$$\Rightarrow [\ p(f^{-1}(b,a)),\ R(a,b),\ q(f(a,b))\ ]$$

Non-ground:

$$\begin{cases} [\ f(X,Y) \approx f^{-1}(Y,X),\ R(X,Y)\ ] \\ [\ p(f(a,Z)),\ q(f(a,Z))] \end{cases}$$

$$\Rightarrow [\ p(f^{-1}(Z,a)),\ R(a,Z),\ q(f(a,Z))\ ]$$

Paramodulation into Variables

Now suppose that we have

Non-ground:

$$[\ f(X,Y) \approx f^{-1}(Y,X),\ R(X,Y)\ ]$$
$$[\ p(Z),\ q(Z)\ ]$$

Solution: Change the substitution in the second clause to $\{\Theta := f^1(b,a)\}$.

The resulting clause $[\ p(f^1(b,a)),\ q(f^{-1}(b,a))\ ]$, together with $[\ f(a,b) \approx f^{-1}(b,a),\ R(a,b)\ ]$, make the $[\ p(f^{-1}(b,a)),\ R(a,b),\ q(f(a,b))\ ]$ redundant.

Lifting Ordering Conditions

In the ground case, the four rules (positive superposition, negative superposition, equality factoring, equality resolution) have ordering conditions attached to them.

Lifting ordering conditions is very difficult. Unfortunately, we have to let slip through a lot.

I show the problems with usual resolution:

Lifting Ordering Conditions (2)

Consider the clause set

$$[\ R(0, s(0))\ ]$$
$$[\ R(s(0), s(s(0)))\ ]$$
$$[\ \neg R(0, s(s(0)))\ ]$$
$$[\ \neg R(X, Y),\ \neg R(Y, Z),\ R(X, Z)\ ]$$

Assume that we are using KBO, and no selection function. In order to refute this clause set, we need a refutation in which
$[\ \neg R(X, Y),\ \neg R(Y, Z),\ R(Y, Z)\ ]$ represents
$[\ \neg R(0, s(0)),\ \neg R(s(0), s(s(0))),\ R(0, s(s(0)))\ ]$.

Lifting Ordering Conditions (3)

If we use the ground instance as guide, first $R(X, Z)$ will resolve away.

The result will be $[\ \neg R(0, Y),\ \neg R(Y, s(s(0)))\ ]$. Using the ground instance as guide, we resolve on $\neg R(Y, s(s(0)))$ and obtain $[\ \neg R(0, s(0))\ ]$.

In general, the theorem prover does not know what the instances are. In the example above, one could have additional clauses

$$[\ R(s(s(0)), s(0))\ ],\quad [\ R(s(0), 0)\ ],\quad [\ R(s(s(0)), 0)\ ].$$

Three Ways of Using Ordering Conditions

1. Check order before unification. This is called applying the order a priori.
2. Check order after unification. This is called applying the order a posteriori.
3. Copy ordering assumptions into result. This results in constrained clauses.

A Priori Order Application

We have to resolve $[\ A_1\ ] \cup R_1$ with $[\ \neg A_2\ ] \cup R_2$ when A_1 and A_2 are unifiable, and there exists a substitution Θ, s.t. $A_1\Theta \succ\succ R_1\Theta$ and $\neg A_2\Theta \succeq\succeq R_2\Theta$.

In the transitivity clause, this condition wouldn't be helpful at all, because none of the literals can be a priori blocked.

A Priori Order Application (2)

A priori ordering would be helpful in the following example:

$$[\ p(0)\ ]$$
$$[\ \neg p(s(s(s(0))))\ ]$$
$$[\neg\ p(X),\ p(s(X))\ ]$$

In the clause $[\ \neg p(X),\ p(s(X))\ ]$, one can easily see that for every substitution Θ, one has $\#p(X)\Theta < \#p(s(X))\Theta$. It follows that $p(X)\Theta \prec\prec p(s(X))\Theta$.

Hence we know a priori that it is impossible to resolve on $p(X)$.

A Posteriori Order Application

We have to resolve $[A_1] \cup R_1$ with $[\ \neg A_2\ ] \cup R_2$ when A_1 and A_2 are unifiable with mgu Σ, and there exists a substitution Θ, s.t. $A_1\Sigma\Theta \succ\succ R_1\Sigma\Theta$, and $\neg A_2\Sigma\Theta \succeq\succeq R_2\Sigma\Theta$.

In the following example, a posteriori ordering differs from a priori ordering.

$$[\ \neg p(X, X, Y),\ q(X, Y, Y)\ ]$$

can (using KBO a posteriori) resolve with $[\ q(0, s(0), s(0))\ ]$, but not with $[\ q(s(0), 0, 0)\ ]$.

Using a priori orderings, it can resolve with both.

A Posteriori Order Application (2)

Another example, where a posteriori order is more restrictive than a priori order, is:

$$[\ X + Y \approx Y + X\ ]$$

$$[\ 0 + s(0) < s(s(0)) + 0\ ]$$

A posterori use of KBO will replace $s(s(0)) + 0$ by $0 + s(s(0))$, but it will not replace $0 + s(0)$ by $s(0) + 0$.

A priori application would make both replacements.

Constraint Order Application

Suppose that we want to resolve $[\ \neg R(X,Y),\ \neg R(Y,Z),\ R(X,Z)\ ]$ with itself:

$$[\ \neg R(X,Y),\ \neg R(Y,Z),\ \boxed{R(X,Z)}\ ]$$
$$[\ \boxed{\neg R(X,Z)},\ \neg R(Z,T),\ R(X,T)\ ]$$

When doing this, we are assuming that

$$R(X,Z)\Theta \succ\succ \neg R(X,Y)\Theta,\ \neg R(Y,Z)\Theta$$
$$\neg R(X,Z)\Theta \succeq\succeq \neg R(Z,T)\Theta,\ R(X,T)\Theta$$

One can add these assumptions to the resolvent. The result is:

$$R(X,Z) \succ\succ \neg R(X,Y),\quad R(X,Z) \succ\succ \neg R(Y,Z),$$
$$\neg R(X,Z) \succeq\succeq \neg R(Z,T),\quad \neg R(X,Z) \succeq\succeq R(X,T) \Rightarrow$$
$$[\ \neg R(X,Y),\ \neg R(Y,Z),\ \neg R(Z,T),\ R(X,T)\ ]$$

Constrained Clauses

The first part of the clause is called the constraint. The complete clause is called a constrained clause.

The constraint stores the history of the ordering assumptions that were made in the process of deriving the clause.

The represents the set of instances for which the constraint is true.

Using standard properties of KBO, the clause can be simplified into

$$\#Z \geq \#Y,\ \#X \geq \#Y,\ \#X \geq \#T,\ \#Z \geq \#T \Rightarrow$$
$$[\ \neg R(X,Y),\ \neg R(Y,Z),\ \neg R(Z,T),\ R(X,T)\ ]$$

During simplification, we weakened the constraint. This is logically correct, because we could have derived the clause without any constraint at all.

Constrained Clauses (2)

Constraint ordered resolution is complicated. I know of only one implementation.

The constraints get bigger than the clauses themselves.

Nobody really knows how to solve the constraints.

It is not understood how constraints interact with redundancy.

Maybe there exists a simplification order that is more simple to use with contraints.

The general feeling seems to be that it is not worth the effort.

There is another restriction, called basic superposition, which I didn't write off yet.

Selection Functions

A selection function can 'overrule' the order, and require that a negative literal will be used.

Since there are no restrictions on selection on the ground level, there are no restrictions on the predicate level. (Except that selected literals must always be negative.)

Implementing Redundancy

A ground clause c is redundant in a set of ground clauses S if there exist $c_1, \ldots, c_n \in S$, s.t. each $c_i \succeq\succeq c$, and $c_1, \ldots, c_n$ logically imply c.

A non-ground clause c is redundant in S if each of its instances if redundant in S.

Redundancy is undecidable, and not even enumerable, so we will need to find approximations.

Deletion Rules

- Clause c_1 subsumes c_2 if there exists a substitution Θ, s.t. $c_1\Theta \subseteq c_2$.
- A clause c is a tautology if its grounding $c\Theta$ is a tautology. This is decidable by ground Knuth-Bendix closure procedure.

Simplification Rules

- Let $c = A[t_1] \cup R$ and $[\ t_2 \approx u\ ] \cup S$ be clauses. Assume that there exists a substitution Θ, s.t $t_2\Theta = t_1$, $S\Theta \subseteq c$, and $t_2\Theta \succ u\Theta$.

  Then replace c by $A[u\Theta] \cup R$. This is called demodulation.

- Let $c = [\ A_1\ ] \cup R$ and $[\ \neg A_2\ ] \cup S$ be clauses. Assume that there exists a substitution Θ, s.t. $A_2\Theta = A_1$ and $S\Theta \subseteq c$. Then replace c by R. This is called resolution simplification.

Simplification rules can be repeated as long as possible. If we are lucky, the final result is subsumed or a tautology.

There exist more, more sophisticated redundancy rules, but these are the main ones.

Conclusions

First-Order Theorem proving, its theory, its heuristics, its implementation is very interesting fields.

Some people are dedicating their lives to it. Unfortunatly, it doesn't have the same impact and applicability as e.g. propositional logic, higher-order logic, or SMT.

1. Theorem provers are still too weak.
2. There is an interface problem. There is only one propositional logic, but many first-order logics. (Problem is with type system. Untyped first-order logic is too weak for most applications.)
3. Still, the techniques are fundamental, and one nevers knows what will happen in the future.