Resolution Based Theorem Proving

Hans de Nivelle
Max Planck Institut für Informatik
Stuhlsatzenhausweg 85
D-66123, Saarbrücken
Germany

6.12.2000

General

The unsatisfiability problem for FOL is undecidable. There is no algorithm that always terminates, says 'yes' on unsatisfiable formulae, and 'no' on satisfiable formulae.

But the unsatisfiability problem is enumerable. There exists algorithms that terminate and say 'yes' if and exactly if the formula is unsatisfiable.

Resolution is such a enumeration procedure.

State of the Art

Resolution theorem provers find proofs that are really impressive, but still insufficient for applications. It is not so much a matter of strength, but a matter of interface, and some none first order principles need to be added.

There is the TPTP library, which has helped the field quite a lot, but implementers tend to concentrate on it, rather than on the real problems.

Clauses

A *literal* is an atom or its negation. We call a literal *positive* if it is an atom. If it is negated, then it is called *negative*.

A *clause* is a finite set of literals.

The meaning of clause $\{A_1, \ldots, A_p\}$ is

$$\forall X_1 \cdots X_n (A_1 \vee \cdots \vee A_p),$$

where $X_1, \ldots, X_n$ are the variables occurring in the clause.

The meaning of the empty clause $\{\ \}$ is $\perp$, or contradiction.

We consider renamed variants as equal.

Resolution

The principle of resolution is as follows: Take some clause set C, which one supposes inconsistent. Derive logical consequences from it until a contradiction (the empty clause) is derived. If this happens, the original clause set is refuted.

It can be shown that only two rules are sufficient for completeness:

Resolution Let $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$ be clauses, s.t. there is a substitution Θ, for which $A_1\Theta = A_2\Theta$. Then derive $R_1\Theta \cup R_2\Theta$.

Factoring Let $\{A_1, A_2\} \cup R$ be a clause, s.t. there is a substitution Θ, for which $A_1\Theta = A_2\Theta$. Then derive $\{A_1\Theta\} \cup R\Theta$.

Example: Clause set

$\{N(0)\}$
$\{\neg N(X), N(s(X))\}$
$\{\neg\ N(s(s(s(0)))\}$

has many refutations.

Clause set

$\{p(X), p(Y)\}$
$\{\neg p(X), \neg p(Y)\}$

has a refutation. It uses the factoring rule.

Clause Form Transformation

- Replace each occurrence of $A \leftrightarrow B$ by $(A \rightarrow B) \land (B \rightarrow A)$.

- Replace each occurrence of $A \rightarrow B$ by $\neg A \lor B$.

- Rewrite $$\begin{array}{lll} \neg(A \land B) & \text{into} & \neg A \lor \neg B. \\ \neg(A \lor B) & \text{into} & \neg A \land \neg B. \\ \neg\neg A & \text{into} & A. \\ \neg\forall x A & \text{into} & \exists x \neg A. \\ \neg\exists x A & \text{into} & \forall x \neg A. \end{array}$$

- Replace existential quantifiers by fresh function symbols. As long as there are existential quantifiers in F, write F as $F[\exists y A]$, where $\exists y A$ is outermost. Let $x_1, \ldots, x_n$ be the universally quantified variables that have $\exists y A$ in the scope, and that are referred to in A. Then replace $F[\exists y A]$ by
$F[A[y := f(x_1, \ldots, x_n)]\ ]$.
(:= means intelligent substitution)

- Rewrite $\begin{array}{lll} \forall x(A \wedge B) & \text{into} & \forall x A \wedge \forall x B. \\ (A \wedge B) \vee C & \text{into} & (A \vee C) \wedge (B \vee C). \\ A \vee (B \wedge C) & \text{into} & (A \vee B) \wedge (A \vee C). \end{array}$

- Finally change into clause notation. **(1)** Write members of $\wedge$ as distinct clauses. **(2)** Drop universal quantifers. If necessary rename universal quantifiers, in order to avoid name conflicts. **(3)** Replace $\vee$ by set notation.

The General Picture

We now have the general picture. If one hopes to prove that $F_1, \ldots, F_n \models G$, one does this as follows: Construct the Clause Form of

$$F_1 \wedge \cdots \wedge F_n \wedge \neg G.$$

Apply resolution (and factoring) on the result.

The Rest of this Talk

I will eloborate on some subjects:

- Optimized ways of constructing the clause form.
- How to restrict resolution.
- How to include equality.

Optimized Clause Form Transformation

The Clause Form transformation we gave is exponential:

- $A_1 \leftrightarrow (A_2 \leftrightarrow (A_3 \cdots (A_{p-1} \leftrightarrow A_p)))$.

- $(A_1 \wedge B_1) \vee (A_2 \wedge B_2) \vee \cdots (A_p \wedge B_p)$.

The solution to this problem consists of subformula replacement: Let $F[A]$ be a first order formula. Let A be some subformula of $F[A]$, making use of the variables $x_1, \ldots, x_n$. Replace $F[A]$ by

$$F[a(x_1, \ldots, x_n)] \wedge \forall x_1 \cdots x_n (\ a(x_1, \ldots, x_n) \leftrightarrow A).$$

(Here a is a fresh name)

Theorem: Subformulae replacement is 2nd order equivalent. (or satisfiability equivalent)

Various strategies differ in the following choices:

- Which subformulae to replace?
- Use $\leftrightarrow$ or $\rightarrow$?

Why should one bother about exponential behaviour during an undecidable process? Experiments suggests that that one should. It introduces cut formulae.

Refinements

One can use restrictions of resolution without loosing completeness: The most common way to do this is to use an ordering on literals. Definition: Let $\succ$ be an order on literals. $\succ$ is called an $A-order$ if

1. $A \succ B$ iff $\pm A \succ \pm B$.

2. $A \succ B$ implies $A\Theta \succ B\Theta$, for all substitutions Θ.

A-orders are used as follows: In the resolution rule, both resolved literals must be $\succ$-maximal. In the factoring rule, one of A_1, A_2 must be maximal.

Subsumption and Tautologies

Subsumption If for clauses c_1, c_2 there is a substitution Θ, such that $c_1\Theta \subseteq c_2$, then c_1 *subsumes* c_2. In that case c_2 can be deleted.

Tautology A clause is a tautology if it has form $\{A, \neg A\} \cup R$. Tautologies can be deleted.

Splitting The splitting rule can be applied when a clause c can be partitioned into two non-empty parts, that do not have overlapping variables. If c that can be partitioned as $R_1 \cup R_2$, then the prover tries to refute R_1 and R_2 independently.

Equality

We write $t_1 \approx t_2$ for 't_1 is equal to t_2'.

Natural Deduction with equality has two rules: **(1)** From $A[t_1]$ and $t_1 \approx t_2$ derive $A[t_2]$. **(2)** Always derive $t \approx t$.

This inspires to the following resolution-style rules:

Paramodulation Let $\{t_1 \approx t_2\} \cup R_1$ and $\{A[u_1]\} \cup R_2$ be clauses, s.t. there is a substitution Θ, for which $t_1\Theta = u_1\Theta$. Then derive $\{A\Theta[t_2\Theta]\} \cup R_1\Theta \cup R_2\Theta$.

Equality Reflexivity Let $\{t_1 \not\approx t_2\} \cup R$ be a clause, s.t. there is a substitution Θ, for which $t_1\Theta = t_2\Theta$. Then derive $R\Theta$.

The paramodulation rule is incredibly productive.

1. Every equality unifies with every variable: Clause $\{X + 0 \approx X\}$ paramodulates into itself. Result: $\{X + 0 \approx X + 0\}$, $\{(X + 0) + 0 \approx X\}$.

2. Usually an infinite amount of terms can be derived. Continuing in the previous example, we get: $\{(X + 0) + 0 \approx X + 0\}$, $\{X+0 \approx (X+0)+0\}$, $\{((X+0)+0)+0 \approx X + 0\}$, etc. etc.

We will forbid paramodulation into variables, and try to direct equalities, s.t. they can only replace bigger by smaller terms.

Simplification Orders

Let $\succ$ be an order on atoms and on terms. It is called a *simplification order* if the following hold:

- $\succ$ is wellfounded.
- $\succ$ is preserved by substitution: $A_1 \succ A_2$ implies that $A_1\Theta \succ A_2\Theta$.
- $\succ$ is preserved in contexts: If $t_1 \succ t_2$, then $A[t_1] \succ A[t_2]$.

The intuition is: $t_1 \succ t_2$ means 't_1 is more complex than t_2', but there are a few technical problems.

If everything goes well, equality replacement becomes simplification.

The Knuth-Bendix Order

Write $\#t$ for the complexity of t, counting variables as 1. Write $\#_X(t)$ for the number of occurrences of variable X in t.

Let $\succ$ be some order on the functions,constants.

- If $\#t_1 > \#t_2$ and for each variable X, $\#_X(t_1) \geq \#_X(t_2)$, then $t_1 \succ_{KBO} t_2$.

- If both t_1, t_2 are non-variable, $\#t_1 = \#t_2$, and for each variable X, $\#_X(t_1) \geq \#_X(t_2)$, then one can write write $t_1 = f(u_1, \ldots, u_n)$, and $t_2 = g(v_1, \ldots, v_m)$. Then, if $f \succ g$, we put $t_1 \succ_{KBO} t_2$.

  Otherwise if $f = g$ and $n = m$, then if $t_1 = u_1, \ \ldots \ , t_{i-1} = u_{i-1}$, and $t_i \succ_{KBO} t_i$, we put $t_1 \succ_{KBO} t_2$.

The KBO has the following properties:

- KBO is a simplification order.
- It is total on ground terms.
- If t is a subterm of u, then $u \succ t$.

Definition: A multiset is like a set, but it can distinguish how often an element occurs in it.

Definition Let $\succ$ be an order. The *multiset extension* of $\succ$ is defined as follows: $A \succ\succ B$ if $A \neq B$, and whenever an element b occurs more often in B than in A, there is an element a, which occurs more often in A than in B, and for which $a \succ b$.

On finite multisets, $\succ\succ$ is an order. If $\succ$ is well-founded, then $\succ\succ$ is well-founded. If $\succ$ is total, then $\succ\succ$ is total.

Ordering Atoms and Clauses

Assume an ordering $\succ$ on the predicate symbols, including $\approx$, such that $p \succ \approx$, for each symbol p.

Then $\succ_{KBO}$ can be extended to atoms as follows: $p(t_1, \ldots, t_n) \succ_{KBO} q(u_1, \ldots, u_m)$ if $p \succ q$, or $p = q$, and $[t_1, \ldots, t_n] \succ\succ_{KBO} [u_1, \ldots, u_m]$.

It is extended to literals through $\pm A \succ_{KBO} \pm B$ if $A \succ_{KBO} B$, and

$\neg A \succ_{KBO} A$.

Superposition

We define the superposition calculus. For technical reasons, we view clauses as multisets rather than as sets.

Resolution Let $A_1 \vee R_1$ and $\neg A_2 \vee R_2$ be clauses, s.t. A_1 and A_2 are unifiable, and maximal. Let Θ be the most general unifier. Then $R_1\Theta \cup R_2\Theta$ is a resolvent.

Superposition Let $c_1 = t_1 \approx t_2 \vee R_1$ and $c_2 = A[u_1] \vee R_2$ be clauses, s.t. $t_1 \approx t_2$ is maximal in clause c_1, t_1 is maximal in the equality $t_1 \approx t_2$, $A[u_1]$ is maximal in c_2, u_1 is not a variable, u_1 and t_1 have most general unifier Θ. Then $A\Theta[t_2\Theta] \cup R_1\Theta \cup R_2\Theta$ is the result of superposing c_1 into c_2.

Factoring Let $c = A_1 \vee A_2 \vee R$ be a clause, s.t. A_1 is maximal, and A_1, A_2 have most general unifier Θ. Then $A_1\Theta \vee R\Theta$ is a factor of c.

Equality Reflexivity Let $c = t_1 \not\approx t_2 \vee R$ be a clause, s.t. $t_1 \not\approx t_2$ is maximal, and t_1, t_2 have most general unifier Θ. Then $R\Theta$ is obtained by equality reflexivity from c.

Equality Factoring Let $c = t_1 \approx t_2 \vee u_1 \approx u_2 \vee R$ be a clause, s.t. $t_1 \approx t_2$ is maximal in c, t_1 is maximal in $t_1 \approx t_2$, u_1 is maximal in $u_1 \approx u_2$, and t_1, u_1 have most general unifier Θ. Then $t_2\Theta \not\approx u_2\Theta \vee u_1\Theta \approx u_2\Theta \vee R\Theta$ is an equality factor of c.

(This is an ugly rule, needed for technical reasons)

Completeness Proof

We want to prove that the empty clause is derivable from every set of clauses that is unsatisfiable. This is done by constructing models for saturated sets of clauses.

Definition: A clause set C is saturated if it is closed under the derivation rules of the superposition calculus.

Definition: An *abstract interpretation* M is a set of atoms. A clause c is true in an abstract interpretation if either $c \cap M \neq \emptyset$, or there is a negative atom $\neg A$, for which $A \notin M$.

The ground atoms are well-ordered by $\succ_{KBO}$. We write $A_0, A_1, \ldots, A_\omega, \ldots, A_\alpha, \ldots$ with $\alpha < k$ for the ground atoms.

Let C be some set of clauses. We inductively construct an abstract interpretation I_k as follows:

- For limit ordinals α, define $I_\alpha = \bigcup_{\beta<\alpha} I_\beta$.

- For successor ordinals $\alpha + 1$, if there is a clause c, that contains A_α, and that is false in I_α, then $I_{\alpha+1} = I_\alpha \cup \{A_\alpha\}$.

- For successor ordinals $\alpha + 1$, if A_α has form $A[t_1]$, and there are an equality $t_1 \approx t_2$ and an atom $A[t_2]$ in I_α, then $I_{\alpha+1} = I_\alpha \cup \{A_\alpha\}$.

Reducing Counter Examples

It can be shown that the derivation rules reduce counter examples: If there is a clause c that is false under I_k, then by some derivation rule a smaller example can be obtained, unless c is the empty clause.

Equality behaves as it should behave.

Redundancy

The proof also works with a modified notion of saturation.

Definition: A clause set C is saturated if for every clause c, that is derivable from it with the derivation rules, there are clauses $c_1, \ldots, c_n$, s.t. $c_1, \ldots, c_n \models c$, and the $c_i \prec\prec c$.

A set of clauses C makes clause c redundant if for each ground instance $\overline{c}$ of c, there are ground instances $c_1, \ldots, c_n$ of clauses in C, such that each $c_i \prec\prec \overline{c}$, and $c_1, \ldots, c_n \models \overline{c}$.

The completeness proof for resolution allows deletion of redundant clauses.

Both subsumption, and tautology eliminatin are instances of redundancy.

Simplification Rules

Rewriting(Demodulation) If c_1 has the form $\{t_1 \approx t_2\} \cup R_1$, and c_2 has the form $\{A[u]\} \cup R_2$, there exists a substitution Θ, s.t. $R_1\Theta \subseteq R_2$, and $u = t_1\Theta$, and moreover $t_1 \succ t_2$, and $\mid c_1 \mid \leq \mid c_2 \mid$, then c_2 can be deleted, and replaced by $\{A[t_2\Theta]\} \cup R_2$.

Resolution Simplification If c_1 has the form $\{\pm A\} \cup R_1$, c_2 has the form $\{\mp B\} \cup R_2$, there is a substitution Θ, such that $R_1\Theta \subseteq R_2$, $A\Theta = B$, and $\mid c_1 \mid \leq \mid c_2 \mid$, then c_2 can be deleted, and replaced by R_2.

The Future

- Find good ways of handling theories. For example, simple arithmetic, AC.

- Find good ways of handling constraint clauses.

- Find good ways of handling basicness.

- Find more general redundancy checking algorithms that are efficient.

- Improve the interface of existing provers.