Proof theory for FO Logic

Most of the propositional logics can be generalized to include quantification. (I.e., Natural deduction.)

However, in FO Logic it is very difficult to search for a proof, so it will be necessary in automated theorem proving and AI applications to have a very simple proof theory.

Resolution, once again, is popular for this reason.

How to use Resolution as a FO theorem prover?

- The first key step is the conversion of an arbitrary first-order formula to a CNF formula. This might sound impossible, but now we have terms that can "stand in" for quantifiers.

- The second key step is to generalize Resolution to apply to (quantifier-free) CNF formulas, but now containing terms.

Conversion to Normal Form

1. Put the formula in Prenex normal form–that is, with all quantifiers out in front. This is done by: moving negation inwards, using unique variable names in quantification, and then moving quantifiers to the left.

2. Skolemize to eliminate all existentially quantified variables, and then drop the universally quantified variables.

3. Convert to CNF using distributivity.

Example

$$\forall x((A(x) \vee \neg \exists y B(x,y)) \wedge \exists z C(z,x))$$

(1) $\forall x \exists z \forall y((A(x) \vee \neg B(x,y)) \wedge C(z,x))$

(2) $\forall x \forall y((A(x) \vee \neg B(x,y)) \wedge C(f(x),x))$

(3) $(A(x) \vee B(x,y)) \wedge C(f(x),x)$

<u>Unification</u>

The job of unification is to take two atomic sentences p and q and return the most general substitution that makes p look like q.

Algorithm for Unification:

At stage s, algorithm has a set E_s of equations and a substituion σ_s. The equatoins in E_s are of the form $\alpha = \beta$ where α, β may be formulas or terms. The meaning is that the sought-for unifier must be a substitution making α and β idential.

(1) Initially, E_0 is the $k-1$ original equations and σ_0 is the identity.

(2) Given E_s, σ_s, the algorithm does one of the following to choose E_{s+1} and σ_{s+1}:

a. If E_s is empty, we are done and σ_s is the most general unifier.

b. if E_s contains an equation of the form

$$F(t_1, .., t_i) = F(t'_1, ..., t'_i)$$

then $\sigma_{s+1} = \sigma_s$, and E_{s+1} is obtained by removing the above equation, and adding the equations $t_1 = t'_1$,...,$t_i = t'_i$.

c. If E_s contains an equation of the form $x = t$, where x is a variable and t is a term. If t is equal to x, then this equation is discarded from E_s to obtain E_{s+1}. If t is nontrivial and contains x, then the algorithm halts and fails. Otherwise, let $[x/t]$ denote the substitution mapping x to t and define E_{s+1} to be the set of equations $s[x/t] = s'[x/t]$ such that $s = s'$ is in E_s, and define $\sigma_{s+1} = \sigma_s[x/t]$.

Unification Example

$$Unify[P(x, F(x, A)), P(H(y), z)]$$

1. Initially E_0 contains $P(x, F(x, A)) = P(H(y), z))$, and σ_0 is the identity ($x := x, y := y, z := z$).

2. $E_1 = \{x = H(y), F(x, A) = z\}, \sigma_1 = \sigma_0$

3. $E_2 = \{F(H(y), A) = z\}, \sigma_2 : x := H(y); y := y; z := z$.

4. $E_3 = \{\}, \sigma_3 : x := H(y); y := y; z := F(H(y), A)$.

Generalized Resolution

There are two rules:

1. Substitution: from C, derive $C[x/t]$ where $[x/t]$ denotes the substitution mapping x to t.
2. Resolution: from $(a \vee C_1)$, and $(\neg a \vee C_2)$, derive $(C_1 \vee C_2)$.

The above two rules are sometimes combined into the following single Resolution rule:

From $(a \vee C_1)$, and $(\neg a' \vee C_2)$, derive

$$(C_1 \vee C_2)[Unify(a, a')]$$

where $Unify(a, a')$ is the substitution giving the most general unifier of a and a'.

Example from book

The above is refutable:

1. Jack owns a dog
2. Every dog owner is an animal lover
3. No animal lover kills an animal
4. Either Jack or Curiosity killed the cat, who is named Tuna
5. Curiosity did not kill the cat.

Conversion to FO logic:

1. $\exists x Dog(x) \wedge Owns(Jack, x)$
2. $\forall x \forall y (Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$
3. $\forall x \forall y (AnimalLover(x) \wedge Animal(y) \rightarrow \neg Kills(x, y)$
4. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

5. $Cat(Tuna)$

6. $\forall x Cat(x) \rightarrow Animal(x)$

7. $\neg Kills(Curiosity, Tuna)$

Conversion to Normal form:

1. $Dog(D)$, $Owns(Jack, D)$

2. $\neg Dog(y) \vee \neg Owns(x, y) \vee AnimalLover(x)$

3. $\neg AnimalLover(x) \vee \neg Animal(y) \vee \neg Kills(x, y)$

4. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

5. $Cat(Tuna)$

6. $\neg Cat(x) \vee Animal(x)$

Example cont'd