

Resolution

Tom Costello
Computer Science Department, Stanford University,
Stanford, CA 94305,
Email costello@cs.Stanford.EDU

Resolution

A literal is an n -ary predicate symbol applied to n terms, or the negation of an n -ary predicate symbol applied to n terms.

A clause is the disjunction of m literals. If L_i are literals, then we write

$$L_1 | L_2 | \dots | L_n$$

for the disjunction of $L_i | i \leq n$

The empty clause is falsity \perp .

The opposite of a literal L without a negation sign is the negation of the literal. The opposite of a literal with a negation sign is the literal without a negation sign.

Propositional Resolution

If $L_1 | \dots | L_n$ and $L'_1 | \dots | L'_m$ are clauses, and L_j is the opposite of L'_k , then the result of resolving $L_1 | \dots | L_n$ and $L'_1 | \dots | L'_m$ is

$$L_1 | \dots | L_{j-1} | L_{j+1} | \dots | L_n | L'_1 | \dots | L'_{k-1} | L'_{k+1} | \dots | L'_m$$

Completeness of Propositional Resolution

If $\Gamma \vdash \phi$ and Γ and $\neg\phi$ are in clause form, then

$$\Gamma, \neg\phi \vdash_{\text{resolution}} \perp$$

That is, they resolve to the empty clause.

Conversion to Clause Form

Any propositional sentence can be converted to clause form by moving negations inwards, and distributing \wedge over \vee , and collecting terms.

$$\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$

$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$$

$$\neg\neg\phi \equiv \phi$$

$$(\phi \vee (\psi \wedge \gamma)) \equiv ((\phi \vee \psi) \wedge (\phi \vee \gamma))$$

$$\phi \vee \phi = \phi$$

$$\phi \wedge \phi = \phi$$

Unification

Two terms (with disjoint variables) are unifiable if there is a substitution of terms for variables that makes the two terms equal.

$$f(g(X, c), Y) \quad f(Z, h(a))$$

can be unified to

$$f(f(X, c), h(a))$$

by substituting $h(a)$ for Y and $g(X, c)$ for Z .

Algorithm for Unification

The Martelli-Montanari algorithm operates on a finite set of equations

$$E = \{s_1 = t_1, \dots, s_n = t_n\}$$

and a substitution θ . Initially $E = \{s = t\}$, where s and t are the terms to be unified, and $\theta = \emptyset$. Then the algorithm chooses from E an arbitrary equation and performs acts according to the following rules:

Equation from E

Action

1 $f(s_1, \dots, s_n) = f(t_1, \dots, t_n)$

replace by the equations

$$s_1 = t_1, \dots, s_n = t_n$$

2 $f(s_1, \dots, s_n) = g(t_1, \dots, t_n)$

fail

3 $X = X$

delete equation

4 $X = t$ or $t = X$ where X does not occur in t

add $X \leftarrow t$ to θ , apply the substitution
 $\{X \leftarrow t\}$ to E and the term in θ

5 $X = t$ or $t = X$ where X occurs in t and $X \neq t$

fail

The above procedure is repeated until E becomes empty, or the procedure fails.

Conversion to Clause Form

Any first order sentence can be converted to clause form by using the propositional methods, and skolemization, which introduces new functions and constants to replace existentials.

The clause form is not satisfiable, if the original sentence is not satisfiable.

$$\forall x_1, \dots, x_n \exists y, \phi(y)$$

$$\phi(SK(x_1, \dots, x_n))$$

$$\exists y, \phi(y)$$

$$\phi(sk)$$

Resolution

If $L_1 | \dots | L_n$ and $L'_1 | \dots | L'_m$ are clauses, and there is a substitution Θ , such that

and $\Theta(L_j)$ is the opposite of $\Theta(L'_k)$, then the result of resolving $L_1 | \dots | L_n$ and $L'_1 | \dots | L'_m$ is

$$\Theta(L_1) | \dots | \Theta(L_{j-1}) | \Theta(L_{j+1}) | \dots$$
$$| \Theta(L_n) | \Theta(L'_1) | \dots | \Theta(L_{k-1}) | \Theta(L_{k+1}) | \dots | \Theta(L'_m)$$