Resolution for predicate logic

Gilmore's algorithm is correct, but useless in practice.

We upgrade resolution to make it work for predicate logic.

Recall: resolution in propositional logic

Resolution step:

$$\{L_1, \dots, L_n, A\} \qquad \{L'_1, \dots, L'_m, \neg A\}$$

$$\{L_1, \dots, L_n, L'_1, \dots, L'_m\}$$

Mini-example:

$$\{\neg A, B\} \qquad \{A\} \qquad \{\neg B\}$$

$$\{B\}$$

$$\square$$

A set of clauses is unsatisfiable iff the empty clause can be derived.

Adapting Gilmore's Algorithm

Gilmore's Algorithm:

Let F be a closed formula in Skolem form and let $\{F_1, F_2, F_3, \ldots,\}$ be an enumeration of $E(F)$.

$n := 0$;

repeat $n := n + 1$;

until $(F_1 \wedge F_2 \wedge \ldots \wedge F_n)$ is unsatisfiable;

(this can be checked with any calculus for propositional logic)

report "unsatisfiable" and **halt**

"Any calculus" $\leadsto$ use resolution for the unsatisfiability test

Recall: Definition of Res

Definition: Let F be a set of clauses. The set of clauses $Res(F)$ is defined by

$$Res(F) = F \cup \{R \mid R \text{ is a resolvent of two clauses } F\}.$$

We set:

$$\begin{aligned} Res^0(F) &= F \\ Res^{n+1}(F) &= Res(Res^n(F)) \qquad \text{für } n \geq 0 \end{aligned}$$

and define

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F).$$

Ground clauses

A ground term is a term without occurrences of variables.

A ground formula is a formula in which only ground terms occur.

A predicate clause is a disjunction of atomic formulas.

A ground clause is a disjunction of ground atomic formulas.

A ground instance of a predicate clause K is the result of substituting ground terms for the variables of K.

Clause Herbrand expansion

Let $F = \forall y_1 \forall y_2 \ldots \forall y_n F^*$ be a closed formula in Skolem form with matrix F^* in clause form, and let $K_1, \ldots, K_m$ be the set of predicate clauses of F^*.

The clause Herbrand expansion of F is the set of ground clauses

$$CE(F) = \bigcup_{i=1}^{m} \{K_i[y_1/t_1][y_2/t_2] \ldots [y_n/t_n] \mid t_1, t_2, \ldots, t_n \in D(F)\}$$

Lemma: $CE(F)$ is unsatisfiable iff $E(F)$ is unsatisfiable.

Proof: Follows immediately from the definition of satisfiability for sets of formulas.

Ground resolution algorithm

Let $C_1, C_2, C_3, \ldots$ be an enumeration of $CE(F)$.

$n := 0$;

$S := \emptyset$;

repeat

$\quad n := n + 1$;

$\quad S := S \cup \{C_n\}$;

$\quad S := \mathit{Res}^*(S)$

until $\square \in S$

report "unsatisfiable" and **halt**

Ground resolution theorem

Ground Resolution Theorem: A formula $F = \forall y_1 \ldots \forall y_n\ F^*$ with matrix F^* in clause form is unsatisfiable iff there is a set of ground clauses $C_1, \ldots, C_m$ such that:

- C_m is the empty clause, and
- for every $i = 1, \ldots, m$
  - either C_i is a ground instance of a clause $K \in F^*$, i.e., $C_i = K[y_1/t_1] \ldots [y_n/t_n]$ where $t_j \in D(F)$,
  - or C_i is a resolvent of two clauses C_a, C_b with $a < i$ and $b < i$

Proof sketch: If F is unsatisfiable, then $C_1, \ldots, C_m$ can be easily extracted from S by leaving clauses out.

Substitutions

A substitution sub is a (partial) mapping of variables to terms.

An atomic substitution is a substitution which maps one single variable to a term.

$F\,sub$ denotes the result of applying the substitution sub to the formula F.

$t\ sub$ denotes the result of applying the substitution sub to the term t

Substitutions

The concatenation $sub_1 sub_2$ of two substitutions sub_1 and sub_2 is the substitution that maps every variable x to $sub_2(sub_1(x))$.

(First apply sub_1 and then sub_2.)

Substitutions

Two substitutions sub_1, sub_2 are equivalent if $t\, sub_1 = t\, sub_2$ for every term t.

Every substitution is equivalent to a concatenation of atomic substitutions. For instance, the substitution

$$x \mapsto f(h(w)) \quad y \mapsto g(a, h(w)) \quad z \mapsto h(w)$$

is equal to the concatenation

$$[x/f(z)]\,[y/g(a,z)]\,[z/h(w)].$$

Swapping substitutions

Rule for swapping substitutions:

$$[x/t]sub = sub[x/t\,sub] \qquad \text{if } x \text{ does not occur in } sub.$$

Examples:

- $[x/f(y)]\underbrace{[y/g(z)]}_{sub} = [y/g(z)][x/f(g(z))]$
- but $[x/f(y)]\underbrace{[x/g(z)]}_{sub} \neq [x/g(z)][x/f(y)]$
- and $[x/z]\underbrace{[y/x]}_{sub} \neq [y/x][x/z]$

Unifier and most general unifier

Let $\mathbf{L} = \{L_1, \ldots, L_k\}$ be a set of literals of predicate clauses (terms). A substitution sub is a unifier of $\mathbf{L}$ if

$$L_1 sub = L_2 sub = \ldots = L_k sub$$

i.e., if $|\mathbf{L}sub| = 1$, where $\mathbf{L}sub = \{L_1 sub, \ldots, L_k sub\}$.

A unifier sub of $\mathbf{L}$ is a most general unifier of $\mathbf{L}$ if for every unifier sub' of $\mathbf{L}$ there is a substitution s such that $sub' = sub\, s$.

Exercise

Unifiable?			Yes	No
	$P(f(x))$	$P(g(y))$		
	$P(x)$	$P(f(y))$		
	$P(x, f(y))$	$P(f(u), z)$		
	$P(x, f(y))$	$P(f(u), f(z))$		
	$P(x, f(x))$	$P(f(y), y)$		
	$P(x, g(x), g^2(x))$	$P(f(z), w, g(w))$		
$P(x, f(y))$	$P(g(y), f(a))$	$P(g(a), z)$		

Unification algorithm

Input: a set $\mathbf{L} \neq \emptyset$ of literals

$sub := []$ (the empty substitution)

while $|\mathbf{L}sub| > 1$ **do**

Find the first position at which two literals $L_1, L_2 \in \mathbf{L}sub$ differ

if none of the two characters at that position is a variable then
then report "non-unifiable" and **halt**
else let x be the variable and t the term starting at that position
(possibly another variable)
if x occurs in t
then report "non-unifiable" and **halt**
else $sub := sub\,[x/t]$

report "unifiable" and **return** sub

Correctness of the unification algorithm

Lemma: The unification algorithm terminates.

Proof: Every execution of the $\mathbf{while}$-loop (but the last) substitutes a variable x by a term t not containing x, and so the number of variables occurring in $\mathbf{L}sub$ decreases by one.

Lemma: If $\mathbf{L}$ is non-unifiable then the algorithm reports "non-unifiable".

Proof: If $\mathbf{L}$ is non-unifiable then the algorithm can never exit the loop.

Correctness of the unification algorithm

Lemma: If $\mathbf{L}$ is unifiable then the algorithm reports "unifiable" and returns the most general unifier of $\mathbf{L}$ (and so in particular every unifiable set $\mathbf{L}$ has a most general unifier).

Proof: Assume $\mathbf{L}$ is unifiable and let m be the number of iterations of the loop on input $\mathbf{L}$.

Let $sub_0 = []$, for $1 \leq i \leq m$ let sub_i be the value of sub after the i-th iteration of the loop.

We prove for every $0 \leq i \geq m$:

(a) If $1 \leq i \leq m$ the i-th iteration does not report "non-unifiable".

(b) For every (w.l.o.g. ground) unifier sub' of $\mathbf{L}$ there is a substitution s_i such that $sub' = sub_i\, s_i$.

By (a) the algorithm exits the loop normally after m iterations and reports "unifiable". By (b) it returns a most general unifier.

Correctness of the unification algorithm

Proof by induction on i:

Basis ($i = 0$). For (a) there is nothing to prove. For (b) take $s_0 = sub'$.

Step ($i > 0$). By induction hypothesis there is s_{i-1} such that $sub_{i-1}s_{i-1}$ is ground unifier.

For (a), since $|\mathbf{L}sub_{i-1}| > 1$ and $\mathbf{L}sub_{i-1}$ unifiable, x and t exist and x does not occur in t, and so no "non-unifiable" is reported.

For (b), get s_i by removing from s_{i-1} every atomic substitution of the form $[x/t]$. Further, since $sub_{i-1}s_{i-1}$ ground unifier we can assume w.l.o.g. that x does not occur in s_i. We have:

Correctness of the unification algorithm

$$
\begin{array}{cll}
& L\, sub_i\, s_i & \\
= & L\, sub_{i-1}\, [x/t]\, s_i & \text{(algorithm extends } sub_{i-1} \text{ with } [x/t]) \\
= & L\, sub_{i-1}\, s_i\, [x/t\, s_i] & (x \text{ does not occur in } s_i) \\
= & L\, sub_{i-1}\, s_i\, [x/t\, s_{i-1}] & (x \text{ does not occur in } t) \\
= & L\, sub_{i-1}\, s_i\, [x/x\, s_{i-1}] & (t\, s_{i-1} = x\, s_{i-1} \text{ because } sub_{i-1} s_{i-1} \text{ unifier}) \\
= & L\, sub_{i-1}\, s_{i-1} & \text{(definition of } s_i) \\
= & L\, sub' & \text{(induction hypothesis)}
\end{array}
$$

Resolution for predicate logic

A clause R is a resolvent of two predicate clauses K_1, K_2 if the following holds:

- There are renamings of variables s_1, s_2 (particular cases of substitutions) such that no variable occurs in both $K_1\, s_1$ and $K_2\, s_2$.
- There are literals $L_1, \ldots, L_m$ in $K_1\, s_1$ and literals $L'_1, \ldots, L'_n$ in $K_2\, s_2$ such that the set

  $$\mathbf{L} = \{\overline{L_1}, \ldots, \overline{L_n}, L'_1, \ldots, L'_n\}$$

  is unifiable. Let sub be the most general unifier of $\mathbf{L}$.
- $R = ((K_1\, s_1 - \{L_1, \ldots, L_m\}) \cup (K_2\, s_2 - \{L'_1, \ldots, L'_n\}))sub.$

Correctness and completeness

Questions:

- If using predicate resolution $\square$ can be derived from F then F is unsatisfiable (correctness)
- If F is unsatisfiable then predicate resolution can derive the empty clause $\square$ from F (completeness)

Exercise

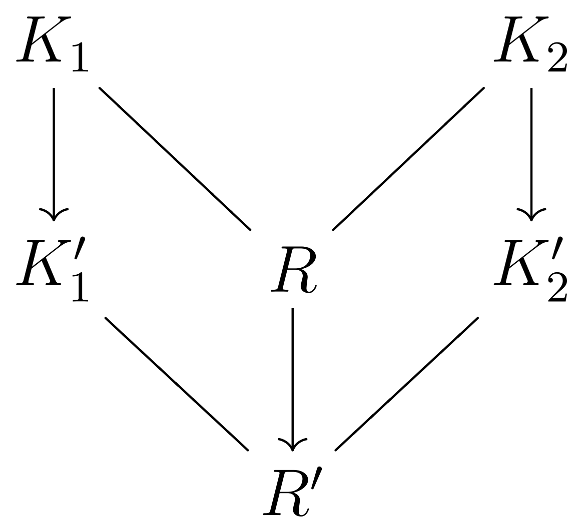
Have the following pairs of predicate clauses a resolvent?
How many resolvents are there?

C_1	C_2	Resolvents
$\{P(x), Q(x, y)\}$	$\{\neg P(f(x))\}$	
$\{Q(g(x)), R(f(x))\}$	$\{\neg Q(f(x))\}$	
$\{P(x), P(f(x))\}$	$\{\neg P(y), Q(y, z)\}$	

Lifting-Lemma

Let C_1, C_2 be predicate clauses and let C_1', C_2' be two ground instances of them that can be resolved into the resolvent R'.

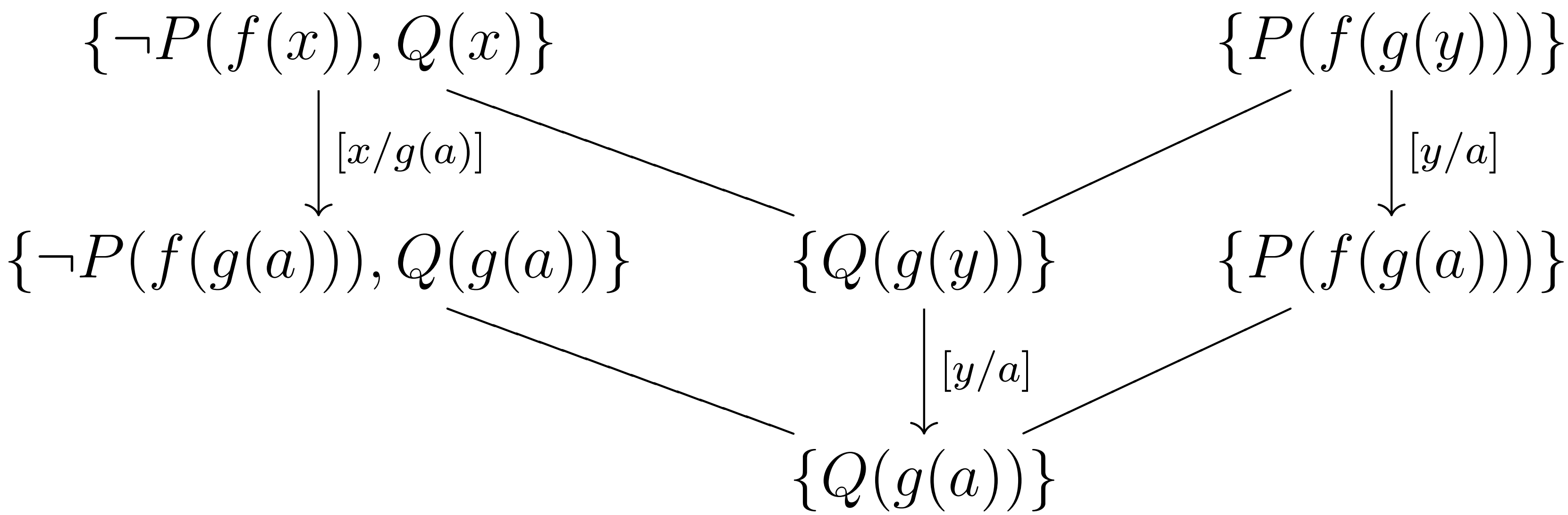
Then there is predicate resolvent R of C_1, C_2 such that R' is a ground instance of R.



—: Resolution

$\rightarrow$: Substitution

Lifting-Lemma: example



Predicate Resolution Theorem

Resolution Theorem of Predicate Logic:

Let F be a closed formula in Skolem form with matrix F^* in predicate clause form. F is unsatisfiable iff $\square \in Res^*(F^*)$.

Universal closure

The universal closure of a formula H with free variables $x_1, \ldots, x_n$ is the formula

$$\forall H = \forall x_1 \forall x_2 \ldots \forall x_n H$$

Let F be a closed formula in Skolem form with matrix F^*. Then

$$F \equiv \forall F^* \equiv \bigwedge_{K \in F^*} \forall K$$

Example:

$$\begin{aligned} F^* &= P(x,y) \wedge \neg Q(y,x) \\ F &\equiv \forall x \forall y (P(x,y) \wedge \neg Q(y,x)) \equiv \forall x \forall y P(x,y) \wedge \forall x \forall y (\neg Q(y,x)) \end{aligned}$$

Exercise

Is the set of clauses

$$\{\{P(f(x))\}, \{\neg P(f(x)), Q(f(x), x)\}, \{\neg Q(f(a), f(f(a)))\},$$
$$\{\neg P(x), Q(x, f(x))\}\}$$

unsatisfiable?

Demo

We consider the following set of predicate clauses (Schöning):

$$\begin{aligned} F &= \{\{\neg P(x), Q(x), R(x, f(x))\}, \{\neg P(x), Q(x), S(f(x))\}, \{T(a)\}, \\ &\quad \{P(a)\}, \{\neg R(a, x), T(x)\}, \{\neg T(x), \neg Q(x)\}, \{\neg T(x), \neg S(x)\}\} \end{aligned}$$

and prove it is unsatisfiable with `otter`.

Refinements of resolution

Problems of predicate resolution:

- Branching degree of the search space too large
- Too many dead ends
- Combinatorial explosion of the search space

Solution:

Strategies and heuristics: forbid certain resolution steps, which narrows the search space.

But: Completeness must be preserved!