Chapter 7: Predicate Calculus: Resolution

November 7, 2008

Outline

7.1 Functions and Terms

- We will now allow the language for predicate logic to contain functions symbols. They will be interpreted as functions (of appropriate arity) in the domain of an interpretation.

Example

Consider the formula

$$p(x, y) \rightarrow p(x, f(y))$$

where p is a binary predicate symbol, and f is a unary function symbol.

Two possible interpretations of this formula

1. $I_1 = (\mathbb{N}, \{<\}, \{succ(x)\})$, where $succ(x) = x + 1$
2. $I_2 = (\{0,1\}^*, \{substr\}, \{f\})$, where the relation $substr(w_1, w_2)$ means that w_1 is a substring of w_2, and $f(w) = w0$, is the function appending 0 to the right end of word w.

Terms

Suppose

$\mathcal{V}$: variables
$\mathcal{A}$: constant symbols
$\mathcal{P}$: predicate symbols
$\mathcal{F}$: function symbols

$$
\begin{aligned}
\mathit{term} &::= x, && x \in \mathcal{V} \\
\mathit{term} &::= a, && a \in \mathcal{A} \\
\mathit{term} &::= f(\mathit{termList}), && f \in \mathcal{F} \\
\mathit{termList} &::= \mathit{term} \\
\mathit{termList} &::= \mathit{term}, \mathit{termList} \\
\mathit{atomicFormula} &::= p(\mathit{termList}), && p \in \mathcal{P}
\end{aligned}
$$

Examples

Suppose a, b are constant symbols, p a binary predicate symbol, f is a binary function symbol, and g a unary function symbol.

1. Examples of terms:

$$a, g(a), f(x, y), f(x, g(a)), f(f(a, x), b), f(f(x, y), f(b, g(a))), \ldots$$

2. Examples of atomic formulas:

$$p(a, a), p(f(x, y), g(y)), p(g(b), f(a, g(a))), \ldots$$

Example

Consider the formula

$$\forall x \forall y (p(x, y) \rightarrow p(f(x, a), f(y, a))$$

This formula is satisfiable but not valid.

An interpretation in which the formula is true:

$$I_1 = (\mathbb{Z}, \{<\}, \{+\}, \{0\})$$

An interpretation in which the formula is false:

$$I_2 = (\mathbb{Z}, \{<\}, \{\cdot\}, \{0\})$$

Definition

A term or atom is ground if it contains no variables. A formula is ground if it has no quantifiers and no variables.
A' is a ground instance of a quantifier-free formula A if it can be obtained from A by substituting ground terms for free variables.

Examples

1. Examples of ground terms: $f(a,a), g(b), f(f(a,b),g(a)), \ldots$
2. Examples of ground formulas:
   $\neg p(a,a), p(f(a,b),b) \rightarrow p(a,a), \ldots$
3. The formula $\neg p(f(a,b),b) \vee p(a,f(a,a))$ is a ground instance of the formula $\neg p(f(x,b),y) \vee p(x,f(x,x))$

Remark

We note that the set of all ground terms in some language can be systematically enumerated; i.e. there is an algorithm which produces a listing:

$$t_0, t_1, t_2, \ldots, t_n, \ldots$$

of all ground terms in that language.

This listing can be produced using diagonalization method (for details, see pp.141-142 in the textbook).

7.2 Clausal Form

Definition

We say that a formula A is in prenex conjunctive normal form (PCNF), if it has the form

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \; M(x_1, x_2, \ldots, x_n)$$

where Q_i $(i = 1, 2, \ldots, n)$ are quantifiers and $M(x_1, x_2, \ldots, x_n)$ is a quantifier-free formula in CNF, called the **matrix** of A.

- We say that a closed formula is in clausal form if it is in PCNF and all quantifiers Q_i are $\forall$.
- A **literal** is an atomic formula or its negation; e.g.

$$p(x, f(a, y)), \neg p(g(x), f(x, g(x)))$$

A literal is said to be ground if it contains no variables.
A **clause** is a disjunction of literals; e.g.

$$\neg p(x, y) \vee p(x, f(x))$$

- C' is a ground clause if it is a ground instance of a clause C; e.g. if

$$C = \neg p(x, y) \vee p(x, f(x)),$$

the substitution $x \leftarrow f(a), y \leftarrow a$ produces the ground clause

$$C' = \neg p(f(a), a) \vee p(f(a), f(f(a)))$$

Example

The following formula is in clausal form:

$$\forall x \forall y \forall z[(p(x, f(y)) \vee q(z)) \wedge (\neg q(f(x)) \vee \neg p(f(y), z))]$$

We will write this clausal form in the following way:

$$\{\{p(x, f(y)), q(z)\}, \{\neg q(f(x)), \neg p(f(y), z)\}\}$$

An even more simplified version of this clausal form is:

$$\{pxfyqz, \neg qfx\neg pfyz\}$$

[In order to properly parse the clauses, we need to keep in mind the arities of all predicate and function symbols.]

Definition

We write

$$A \approx B$$

to denote the fact that a formula A is satisfiable if and only if B is satisfiable.

Theorem

(Skolem) Suppose A is a closed formula. Then, there exists a formula A' in clausal form such that

$$A \approx A'$$

Proof (Algorithm for converting A to A').
We start with a particular formula

$$\forall x\ (p(x) \to q(x)) \to (\forall x\ p(x) \to \forall x\ q(x))$$

and we will illustrate each step of the algorithm by showing how it applies to this particular example.

Input: Closed formula A
Output: formula A' in clausal form, such that $A \approx A'$

(1) Rename bound variables (if necessary) so that no variable appears in the scope of two different quantifiers:

$$\forall x\ (p(x) \to q(x)) \to (\forall y\ p(y) \to \forall z\ q(z))$$

(2) Eliminate all propositional connectives, except for $\neg$, $\wedge$, and $\vee$:

$$\neg\forall x(\neg p(x) \vee q(x)) \vee (\neg\forall y\ p(y) \vee \forall z\ q(z))$$

(3) Push all $\neg$ inward and eliminate double negations:

$$\exists x(p(x) \wedge \neg q(x)) \vee (\exists y \neg p(y) \vee \forall z\ q(z))$$

(4) Extract quantifiers from the matrix:

$$\begin{aligned}
&\exists x(p(x) \wedge \neg q(x)) \vee \exists y(\neg p(y) \vee \forall z\ q(z)) \\
&\equiv \exists x(p(x) \wedge \neg q(x)) \vee \exists y \forall z(\neg p(y) \vee q(z)) \\
&\equiv \exists x \exists y \forall z((p(x) \wedge \neg q(x)) \vee (\neg p(y) \vee q(z)))
\end{aligned}$$

(5) Transform the matrix of the formula into CNF:

$$\exists x \exists y \forall z((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z)))$$

(6) Eliminate existential quantifiers:

- If we have $\exists x\ A(x)$, replace x with a new constant symbol a.
- If we have $\forall x_1 \ldots \forall x_n \exists y\ A(x_1, \ldots, x_n, y)$, replace y with $f(x_1, x_2, \ldots, x_n)$, where f is a new n-ary function symbol.

$$\begin{aligned}
&\exists x \exists y \forall z((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z))) \\
&\approx \exists y \forall z((p(a) \vee \neg p(y) \vee q(z)) \wedge (\neg q(a) \vee \neg p(y) \vee q(z))) \\
&\approx \forall z((p(a) \vee \neg p(b) \vee q(z)) \wedge (\neg q(a) \vee \neg p(b) \vee q(z)))
\end{aligned}$$

□

Example

Transform the formula

$$A = \exists x \forall y p(x, y) \rightarrow \forall y \exists x p(x, y)$$

into a clause form $A' \approx A$.

Solution:

$$\begin{aligned}
&\exists x \forall y\ p(x, y) \rightarrow \forall y \exists x\ p(x, y) \\
&\equiv \exists x \forall y\ p(x, y) \rightarrow \forall w \exists z\ p(z, w) \\
&\equiv \neg \exists x \forall y\ p(x, y) \vee \forall w \exists z\ p(z, w) \\
&\equiv \forall x \exists y\ \neg p(x, y) \vee \forall w \exists z\ p(z, w) \\
&\equiv \forall x \exists y \forall w \exists z\ (\neg p(x, y) \vee p(z, w)) \\
&\approx \forall x \forall w \exists z\ (\neg p(x, f(x)) \vee p(z, w)) \\
&\approx \forall x \forall w\ (\neg p(x, f(x)) \vee p(g(x, w), w))
\end{aligned}$$

□

- The method of converting a closed formula A into a clausal form A' so that $A \approx A'$ is also called Skolemization of A.
- One way to minimize the arity of the new function symbols introduced during this method is to perform the steps of the algorithm in the following sequence:
  1. first push all quantifiers inward
  2. eliminate all $\exists$'s
  3. extract all $\forall$ to the front of the formula.

7.3 Herbrand Models

- Once we allow function symbols in the language of predicate logic, the models can become rather complicated since, given any function symbol, there are a large number of functions on the domain that can interpret it.
- We will show that, if a set of clauses has a model (i.e. is satisfiable), it has a particular **canonical** (or, generic) model.

Definition

Let S be a set of clauses and

$\mathcal{A}$: set of constant symbols appearing in S
$\mathcal{F}$: set of function symbols appearing in S

We define H_S, the Herbrand universe for S inductively, as follows:

- $a \in H_S$, for every $a \in \mathcal{A}$
- $f(t_1, t_2, \ldots, t_n)$, for every n-ary function symbol $f \in \mathcal{F}$, and $t_1, t_2, \ldots, t_n \in H_S$

If there are no constant symbols in S ($\mathcal{A} = \emptyset$), we initialize the inductive definition by putting some arbitrary new constant symbol a in H_S.

Remark

The Herbrand universe H_S for S will be infinite as soon as there is a function symbol in S.

Examples

(a) If $S_1 = \{pxy\neg qa, qa\neg pbx\}$, we have

$$H_S = \{a, b\}$$

(b) If $S_2 = \{\neg pxf(y), pwg(w)\}$,

$$H_S = \{a, f(a), g(a), f(f(a)), f(g(a)), g(f(a)), g(g(a)), \ldots\}$$

(c) For $S_3 = \{\neg paf(x, y), pbf(x, y)\}$

$$H_S = \{a, b, f(a, a), f(a, b), f(b, a), f(b, b),$$

$$f(a, f(a, a)), f(f(a, a), a), \ldots\}$$

Definition

Suppose H_S is the Herbrand universe for a set of clauses. The Herbrand base B_S is the set of ground atomic formulas formed using the predicate symbols in S and terms from H_S.

Example

For $S_3 = \{\neg paf(x, y), pbf(x, y)\}$, the Herbrand base is

$$B_S = \{p(a, f(a, a)), p(a, f(a, b)), p(a, f(b, a)), p(a, f(b, b)),$$

$$\ldots,$$

$$p(a, f(f(a, a), a)), \ldots, p(b, f(a, a)), p(b, f(a, b)), p(b, f(b, a), p(b, f(b, b)$$

Definition

An Herbrand model for a set of clauses S consists of the Herbrand universe H_S as the domain, the function symbols and constants are interpreted by themselves, and the relations are interpreted in some way which would make all the clauses in S true. In other words, the true relations form some subset of B_S which makes all clauses true.

Example

For $S_3 = \{\neg paf(x, y), pbf(x, y)\}$, we saw that Herbrand universe is

$$H_S = \{a, b, f(a, a), f(a, b), f(b, a), f(b, b),$$

$$f(a, f(a, a)), f(f(a, a), a), \ldots\}$$

The relations in Herbrand model are given as

$$v(p(a, f(a, a))) = \mathsf{F},\ v(p(a, f(b, a))) = \mathsf{F},\ v(p(a, f(a, b))) = \mathsf{F}$$

$$v(p(a, f(b, b))) = \mathsf{F},$$

$$v(p(b, f(a, a))) = \mathsf{T},\ v(p(b, f(b, a))) = \mathsf{T}$$

$$v(p(b, f(a, b))) = \mathsf{T},\ v(p(b, f(b, b))) = \mathsf{T}, \ldots$$

Theorem

Let S be a set of clauses. S has a model if and only if it has an Herbrand model.

7.4 Herbrand's Theorem

Theorem

(Herbrand's Theorem - Semantic Form) A set of clauses S of predicate logic is unsatisfiable if and only if a finite set of ground instances of clauses from S is unsatisfiable.

Example

Consider the following unsatisfiable formula:

$$\neg[\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall x\ p(x) \rightarrow \forall x\ q(x))]$$

We saw that its clausal form is

$$\{\neg p(x) \vee q(x), p(y), \neg q(z)\}$$

One set of ground instances for this set of clauses is:

$$\{\neg p(a) \vee q(a), p(a), \neg q(a)\}$$

obtained by substitution

$$x \leftarrow a, \quad y \leftarrow a, z \leftarrow a$$

Now, we can use any method for proving unsatisfiability from propositional logic (semantic tableaux, resolution, etc.)

1.	$\neg p(a) \vee q(a)$	given
2.	$p(a)$	given
3.	$q(a)$	Resolution 1,2
4.	$\neg q(a)$	given
5.	$\square$	Resolution 3,4

Semi-Decision Procedure for Checking Validity

1. Negate the formula.
2. Transform the formula into a clausal form.
3. Generate a finite set of ground instances of clauses.
4. Check if the set of ground clauses from (3) is unsatisfiable.

Step 3 is highly problematic; namely, there are infinitely many ground terms (if there is at least one function symbol) so it may be difficult to find a correct substitution for resolution. In fact, if the set of clauses is satisfiable, there is no such substitution, so our search for it may run forever. This is not surprising, however, considering that we already know that the validity in predicate logic is undecidable (Sec.5.8).

7.5 Ground Resolution

- This is a simpler version of the general resolution method which will be covered in Section 7.8, but is highly ineffective, for the reasons described at the end of the previous section.

Definition

Suppose C_1, C_2 are ground clauses containing a pair of clashing literals l, l^c, say, $l \in C_1$ and $l^c \in C_2$. The resolvent of C_1 and C_2 is the clause

$$Res(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\})$$

C_1 and C_2 are said to be the **parent clauses** in resolution.

Theorem

The resolvent C is satisfiable if and only if C_1 and C_2 are simultaneously satisfiable.

7.6 Substitutions

Definition

Suppose $x_1, x_2, \ldots, x_n$ are distinct variables and $t_1, t_2, \ldots, t_n$ terms such that t_i is not equal to x_i ($i = 1, 2, \ldots, n$). The substitution

$$\{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_n \leftarrow t_n\}$$

is the mapping assigning the term t_i to the variable x_i.

- Generally, we will use Greek letters $\lambda, \mu, \sigma, \tau, \ldots$ to denote substitutions.

Definition

An expression is any term, literal, a clause, or a set of clauses.
If E is an expression and

$$\sigma = \{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_n \leftarrow t_n\}$$

is a substitution, the instance $E\sigma$ is obtained by simultaneously applying the substitution σ to all occurrences of x_i's in E.

Example

Suppose the expression E is the clause

$$E = p(x, y) \vee \neg q(f(x))$$

and σ is the substitution

$$\sigma = \{x \leftarrow a, y \leftarrow f(x)\}$$

Then the instance $E\sigma$ of E is:

$$E\sigma = p(a, f(x)) \vee \neg q(f(a))$$

Definition

Suppose

$$\theta = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}, \quad \sigma = \{y_1 \leftarrow s_1, \ldots, y_k \leftarrow s_k\}$$

are two substitutions with $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_k\}$. The **composition** of θ and σ is the substitution

$$\theta\sigma = \{x_i \leftarrow t_i\sigma \ : \ x_i \neq t_i\sigma\} \cup \{y_j \leftarrow s_j \ : \ y_j \in Y, y_j \notin X\}$$

In plain English: first apply the substitution σ to the terms t_i that appear in θ. If some substitutions become $x_i \leftarrow x_i$, delete them. Finally, append all $y_j \leftarrow s_j$ to the list, for which y_j is not one of the variables x_i in θ.

Example

We are given two substitutions

$$\theta = \{x \leftarrow f(y), y \leftarrow f(a), z \leftarrow u\}$$
$$\sigma = \{y \leftarrow g(a), u \leftarrow z, v \leftarrow f(f(a))\}$$

Then,

$$\theta\sigma = \{x \leftarrow f(g(a)), y \leftarrow f(a), u \leftarrow z, v \leftarrow f(f(a))\}$$

If we are given a 5-ary relation $E = p(x, y, z, u, v)$ it is easy to see that

$$E\theta = p(f(y), f(a), u, u, v)$$
$$(E\theta)\sigma = p(f(g(a)), f(a), z, z, f(f(a)))$$
$$E(\theta\sigma) = p(f(g(a)), f(a), z, z, f(f(a)))$$

Lemma

If E is an expression and θ and σ are two substitutions,

$$E(\theta\sigma) = (E\theta)\sigma$$

Lemma

The composition of substitutions is associative: if θ, σ, and λ are substitutions,

$$\theta(\sigma\lambda) = (\theta\sigma)\lambda$$

Remark

In general, the composition of two substitutions is not commutative; i.e.

$$\theta\sigma \neq \sigma\theta$$

7.7 Unification

Consider the pair of literals

$$p(f(x), g(y)), \qquad \neg p(f(f(a)), g(z)).$$

This pair cannot be resolved using the method of ground resolution. However, the substitution

$$\{x \leftarrow f(a), y \leftarrow a, z \leftarrow a\}$$

will turn the pair into

$$p(f(f(a)), g(a)), \qquad \neg p(f(f(a)), g(a))$$

to which ground resolution can be applied.
In fact, a simpler substitution

$$\{x \leftarrow f(a), y \leftarrow z\}$$

will also accomplish this, even though we will not end up with a pair of ground literals, but

$$p(f(f(a)), g(z)), \qquad \neg p(f(f(a)), g(z)),$$

which are still clashing.

Definition

Given a set of atoms, a unifier is a substitution which makes the atoms identical. A most general unifier, or m.g.u, for short, is a unifier μ such that, if θ is any unifier for the set of atoms

$$\theta = \mu\lambda,$$

for some substitution λ.
In other words, every unifier for that set of atoms can be obtained from an m.g.u. through further substitution.

Example

The unifier

$$\mu = \{x \leftarrow f(a), y \leftarrow z\}$$

is an m.g.u. for the set of atoms $p(f(x), g(y))$ and $p(f(f(a)), g(z))$ from the beginning of the section. In fact,

$$\{x \leftarrow f(a), y \leftarrow a, z \leftarrow a\} = \mu\{z \leftarrow a\}$$

Question: When is it impossible to unify two atomic formulas?

1. if they start with different predicate symbols (obvious).
2. a trickier obstacle: consider the pair of atoms

$$p(a, x), \qquad p(a, f(x))$$

No matter what substitution we attempt to use, the problem is that we will never be able to make the terms x and $f(x)$ identical.
The reason for this is that the two terms we are attempting to unify **contain the same variable**.

- If we want to unify two atoms

$$p(t_1, t_2, \ldots, t_n), \quad p(t'_1, t'_2, \ldots, t'_n)$$

  we are trying to unify pairs of terms t_1 and t'_1, t_2 and t'_2, $\ldots$, t_n and t'_n.

- We can view this problem as attempting to solve a system of n term equations:

$$\begin{aligned} t_1 &= t'_1 \\ t_2 &= t'_2 \\ &\vdots \\ t_n &= t'_n \end{aligned}$$

Example

The problem of trying to unify the pair of atoms

$$p(x, f(y)), \quad p(f(f(y)), g(a))$$

can be viewed as solving the system of two term equations:

$$x = f(f(y))$$
$$f(y) = g(a)$$

Based on previous remarks, this system cannot be solved since the terms in the second equation start with different function symbols.

Remark

So far we have considered the problem of unifying a pair of atomic formulas. Everything mentioned so far and what follows in the rest of this section can be easily adapted to the problem of unifying a set of three or more atomic formulas.

Definition

A set of term equations is in solved form if:

1. all equations are of the form

$$x = t, \quad x - \text{ variable }, t - \text{ term not containing } x$$

2. every variable x which appears in the left-hand side of one of the equations does not appear in any other equation in the system.

If the solved form of the system is

$$x_1 = t_1, \; x_2 = t_2, \ldots, x_n = t_n$$

the solving substitution is

$$\sigma = \{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_n \leftarrow t_n\}$$

Unification Algorithm

Input: a set of term equations
Output: a set of term equations in solved form or the answer "not unifiable".

(1) Transform every equation of the form $t = x$ (x-variable, t-term which is not a variable) into $x = t$;

(2) Delete all trivial equations of the form $x = x$ (x-variable)

(3) Suppose $t' = t''$ is an equation where neither term t', t'' is a variable. If starting function symbols of t' and t'' are not the same, output "not unifiable"; otherwise, if

$$t' = f(t'_1, \ldots, t'_k), \quad t'' = f(t''_1, \ldots, t''_k),$$

replace the equation with k new equations

$$t'_1 = t''_1, \ t'_2 = t''_2, \ldots, t'_k = t''_k$$

(4) If $x = t$ is an equation such that x appears elsewhere in the system:

- if x appears in t, output “not unifiable”
- if x appears in other equations, replace all occurrences of x in those equation with t.

Example

Consider the system of term equations:

$$g(y) = x$$
$$f(x, h(x), y) = f(g(z), w, z)$$

We start by using Rule 3 to replace the second equation with three equations:

$$g(y) = x$$
$$x = g(z)$$
$$h(x) = w$$
$$y = z$$

Next, we use Rule 1 to rewrite the first and the third equation:

$$
\begin{aligned}
x &= g(y) \\
x &= g(z) \\
w &= h(x) \\
y &= z
\end{aligned}
$$

Using the first equation, we eliminate x from the right-hand sides of the remaining equations:

$$
\begin{aligned}
x &= g(y) \\
g(y) &= g(z) \\
w &= h(g(y)) \\
y &= z
\end{aligned}
$$

Next, using Rule 3, we can rewrite the second equation:

$$
\begin{aligned}
x &= g(y) \\
y &= z \\
w &= h(g(y)) \\
y &= z
\end{aligned}
$$

So, we see that the last equation is redundant (it is identical to the second equation) and we can delete it:

$$
\begin{aligned}
x &= g(y) \\
y &= z \\
w &= h(g(y))
\end{aligned}
$$

Finally, we use the second equation to eliminate y from the right-hand sides of other equations:

$$x = g(z)$$
$$y = z$$
$$w = h(g(z))$$

This system is in solved form, so we have an m.g.u:

$$\mu = \{x \leftarrow g(z), y \leftarrow z, w \leftarrow h(g(z))\}$$

- Suppose A and A' are two atoms starting with the same predicate symbol.
- Let k be the first position within these two atoms, considered as strings, where they differ (if such a position exists)
- The pair of terms t, t' in A, A', respectively, which start at the k-th position in these two atoms are called the **disagreement set** for A and A'.

Robinson's Unification Algorithm

Input: two atoms A and A' starting with the same predicate symbol.

Output: an m.g.u. for A and A' or the answer "not unifiable".

1. Initialize $A_0 := A$, $A'_0 := A'$.
2. Suppose A_i and A'_i have been constructed.
3. Let $\{t, t'\}$ be the disagreement set for A and A'. If one term is a variable x_{i+1} and the other one is a term t_{i+1} such that x_{i+1} does not appear in t_{i+1}, define

$$\sigma_{i+1} = \{x_{i+1} \leftarrow t_{i+1}\}, \quad A_{i+1} := A_i\sigma_{i+1}, \quad A'_{i+1} := A'_i\sigma_{i+1}$$

4. If, at some point, it is impossible to complete Step 3, output "not unifiable". If, at the step n, we have $A_n = A'_n$, output "unifiable" and an m.g.u is

$$\mu = \sigma_1\sigma_2 \ldots \sigma_n$$

Example

Consider the pair of atoms

$$A = p(g(y), f(x, h(x), y)), \qquad A' = p(x, f(g(z), w, z))$$

The disagreement set is $\{g(y), x\}$, so

$$\sigma_1 = \{x \leftarrow g(y)\}$$

and

$$A_1 = p(g(y), f(g(y), h(g(y)), y)), \qquad A'_1 = p(g(y), f(g(z), w, z))$$

The disagreement set for A_1 and A'_1 is $\{y, z\}$ so we can take

$$\sigma_2 = \{y \leftarrow z\}$$

and we get

$$A_2 = p(g(z), f(g(z), h(g(z)), z), \qquad A_2 = p(g(z), f(g(z), w, z))$$

Finally, the disagreement set for A_2 and A'_2 is $\{h(g(z)), w\}$ so we take the substitution

$$\sigma_3 = \{w \leftarrow h(g(z))\}$$

After that, we have

$$A_3 = p(g(z), f(g(z), h(g(z)), z), \qquad A'_3 = p(g(z), f(g(z), h(g(z)), z)$$

Since $A_3 = A'_3$, the output is "unifiable" and an m.g.u. is

$$\mu = \sigma_1\sigma_2\sigma_3 = \{x \leftarrow g(y)\}\{y \leftarrow z\}\{w \leftarrow h(g(z))\}$$

$$= \{x \leftarrow g(z), y \leftarrow z, w \leftarrow h(g(z))\}$$

[This is the same m.g.u. as the one obtained by solving the system of term equations.]

7.8 General Resolution

Definition

Suppose $L = \{l_1, l_2, \ldots, l_m\}$ is a set of literals. Then, we define its complement as $L^c = \{l_1^c, l_2^c, \ldots, l_m^c\}$.

General Resolution Rule: Suppose C_1 and C_2 are two clauses **with no variables in common**. Let

$$L_1 = \{l_{11}, \ldots, l_{1m}\} \subseteq C_1$$
$$L_2 = \{l_{21}, \ldots, l_{2n}\} \subseteq C_2$$

so that L_1 and L_2^c can be unified using an m.g.u. σ. Then, C_1 and C_2 are said to be **clashing clauses** and their **resolvent** is

$$Res(C_1, C_2) = (C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

Example

Consider the two clauses

$$C_1 = p(f(x), g(y)) \vee q(x, y)$$
$$C_2 = \neg p(f(f(a), g(z)) \vee q(f(a), g(z))$$

These two clauses contain the following literals:

$$L_1 = \{p(f(x), g(y))\}$$
$$L_2 = \{\neg p(f(f(a), g(z))\}$$

so that $L_2^c = \{p(f(f(a)), g(z))\}$.

It is easy to check that the literals in L_1 and L_2^c are unifiable with an m.g.u.

$$\sigma = \{x \leftarrow f(a), y \leftarrow z\}$$

so the clauses C_1 and C_2 are clashing, and their resolvent is:

$$\begin{aligned} Res(C_1, C_2) &= (C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma) \\ &= \{q(f(a), z)\} \cup \{q(f(a), g(z))\} \end{aligned}$$

so that we get the clause

$$q(f(a), z) \vee q(f(a), g(z))$$

□

- Most often, the two clauses we are trying to resolve will have variables in common, so we cannot use the Resolution Rule as stated before.
- Before we attempt to resolve such pairs of clauses, we have to standardize apart; i.e. we have to rename the variables so that the clauses no longer have any variables in common.

Example

Consider the two literals

$$p(f(x), y), \quad \neg p(x, a)$$

The two atoms contain the same variable x, so we have to standardize them apart before attempting to resolve; e.g. we can replace x in the second atom with z:

$$p(f(x), y), \quad \neg p(z, a)$$

It is easy to see that an m.g.u. is

$$\sigma = \{z \leftarrow f(x), y \leftarrow a\}$$

The m.g.u. transforms the pair of literals into:

$$p(f(x), a), \quad \neg p(f(x), a)$$

Now, we can apply resolution to this pair of literals to obtain the empty clause $\square$.

General Resolution Procedure

Input: a set of clauses S
Output: "satisfiable" or "not satisfiable", in the case the algorithm terminates.

- Set $S_0 := S$
- Suppose S_i has been constructed.
- Choose a pair of clashing clauses C_1, C_2 in S_i and find

$$C = Res(C_1, C_2)$$

  (we may need to standardize apart certain clauses during this step)
- If $C = \square$, terminate the procedure and output "not satisfiable"; otherwise, put

$$S_{i+1} := S_i \cup \{C\}$$

- If $S_{i+1} = S_i$, for all possible pairs of clashing clauses in S_i, terminate the procedure and output "satisfiable".

Example

We will show, using the general resolution procedure, that the following set of clauses is unsatisfiable:

1. $\neg p(x) \vee q(x) \vee r(x, f(x))$
2. $\neg p(x) \vee q(x) \vee s(f(x))$
3. $t(a)$
4. $p(a)$
5. $\neg r(a, y) \vee t(y)$
6. $\neg t(x) \vee \neg q(x)$
7. $\neg t(x) \vee \neg s(x)$

8.	$\neg q(a)$	3,6	$y \leftarrow a$
9.	$\neg p(a) \vee s(f(a))$	2,8	$x \leftarrow a$
10.	$\neg p(a) \vee r(a, f(a))$	1,8	$x \leftarrow a$
11.	$s(f(a))$	4,9	
12.	$r(a, f(a))$	4,10	
13.	$t(f(a))$	5,12	$y \leftarrow f(a)$
14.	$\neg t(f(a))$	11,7	$x \leftarrow f(a)$
15.	$\square$	13,14	

$\square$

Theorem

(Soundness of Resolution) If the empty clause □ is derived by general resolution, then S is unsatisfiable.

Theorem

(Completeness of Resolution) If a set of clauses S is unsatisfiable, then the empty clause □ can be derived from S using general resolution.

Example

In this example, we will prove that the set of clauses (1)-(4) is unsatisfiable. However, we will se that it is not necessary to use ground clauses in derivation.

1. $\neg p(x, y) \vee p(y, x)$
2. $\neg p(x, y) \vee \neg p(y, z) \vee p(x, z)$
3. $p(x, f(x))$
4. $\neg p(x, x)$

3'.	$p(x', f(x'))$	3 Rename x to x'
5.	$p(f(x), x)$	1,3 $x' \leftarrow x, y \leftarrow f(x)$
3".	$p(x'', f(x''))$	3 Rename x to x''
6.	$\neg p(f(x), z) \vee p(x, z)$	3",2 $y \leftarrow f(x), x'' \leftarrow x$
5"'.	$p(f(x'''), x''')$	5 Rename x to x'''
7.	$p(x, x)$	6,5 $z \leftarrow x, x''' \leftarrow x$
4"".	$\neg p(x'''', x'''')$	4 Rename x to x''''
8.	$\square$	7,4"" $x'''' \leftarrow x$

The composition of all unifiers in steps (3')-(8) is:

$$\sigma = \{y \leftarrow f(x), z \leftarrow x, x' \leftarrow x, x'' \leftarrow x, x''' \leftarrow x, x'''' \leftarrow x\}$$

which, restricted to the original variables x, y nad z is

$$\sigma = \{y \leftarrow f(x), z \leftarrow x\}$$

$\square$