Logic programming II

Henrik Boström
Stockholm University

- Unification
- Goal reduction

Substitution and instance

A *substitution* is a finite set of pairs on the form x_i/ t_i, where x_i is a variable and t_i is a term, and $x_i \neq x_j$ for all $i \neq j$, and x_i does not occur in t_j for any i and j.

A term s is an *instance* of a term g if there is a substitution θ such that s = gθ.

s = father(abraham,isaac)
g = father(abraham,X)
θ = {X/isaac}

More definitions

A term t is a *common instance* of two terms t_1 and t_2, if there are substitutions θ_1 and θ_2 such that $t = t_1\theta_1 = t_2\theta_2$.

likes(romeo,juliet) is a common instance of likes(X,juliet) and likes(X,Y).

A term s is *more general* than a term t, if t is an instance of s but s is not an instance of t.
A term s is a *variant* of a term t, if s is an instance of t and t is an instance of s.

s = likes(X,juliet) och t = likes(Y,juliet)

Unifier

A *unifier* θ to two terms t_1 and t_2 is a substitution such that $t_1\theta = t_2\theta$.

If two terms have a unifier, they are said to unify.

p(f(X),Y) and p(W,g(W)) unify.

A unifier is:
θ = {W/f(X), Y/g(f(X))}

The common instance is:
p(f(X),g(f(X)))

Most general unifier

The *most general unifier* (mgu) to two terms is a unifier that results in the most general common instance.

p(X,a) and p(Z,Y)

Substitution	Common instance
{X/a, Z/a, Y/a}	P(a,a)
{X/b, Z/b, Y/a}	P(b,a)
{X/Z, Y/a}	P(Z,a)
{Z/X, Y/a}	P(X,a)

Unification algorithm

Input: two terms t_1 and t_2

Output: an mgu θ to t_1 and t_2 or 'failure'

Let S = [t_1=t_2] and $\theta = \emptyset$.

While S ≠ [] do

Pick first equation E from S.
Call Handle-equation with E, S and θ,
which gives S and θ or 'failure' as output.

In the latter case, exit and return 'failure'.

Return θ.

Handle-equation

Input: equation $s = t$, stack S and substitution θ
Output: stack S and substitution θ or 'failure'

1. If s and t are identical variables or constants, then return S and θ
2. If s is a variable and t is a term*, then replace s with t in the stack and θ and add s/t to θ.
3. If t is a variable and s is a term, then do the above conversely.
4. If s and t are compound terms, where $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$, then put all $s_i = t_i$ on the stack.
5. In all other cases, return 'failure'.

*s must not occur in t – this is called the "occurs check"

Composition

Let $\theta_1 = \{x_1/s_1, \ldots, x_n/s_n\}$ and $\theta_2 = \{y_1/t_1, \ldots, y_m/t_m\}$ be two substitutions such that $x_i \neq y_j$ for all i and j, and x_i does not occur in t_j for any i and j.

Then the *composition* Comp(θ_1,θ_2) of θ_1 and θ_2 =
$\{x_1/s_1\theta_2, \ldots, x_n/s_n\theta_2, y_1/t_1, \ldots, y_m/t_m\}$

$\theta_1 = \{X/Y, Z/f(Y)\}$ och $\theta_2 = \{Y/a\}$
Comp(θ_1,θ_2) = {X/a, Z/f(a), Y/a}

Goal-reduction

Input: a logic program P = C1, ..., Ck and a goal G1, ..., Gn
Output: a substitution or 'no'.

If n=0 then return $\emptyset$.
i := 1
While i $\leq$ k do
    A' :- B1, ..., Bm := a variant of Ci with new variable names
    If there is an mgu θ of G1 and A' then
        Call Goal-reduction with P, (B1, ..., Bm, G2, ..., Gn)θ.
        If a substitution σ is returned then return Comp(θ,σ).
    i := i+1
Return 'no'.

Example

append([],Xs,Xs).
append([X|Xs],Ys,[X|Zs]):- append(Xs,Ys,Zs).

:- append([a,b],[c,d],L). {X/a, Xs/[b], Ys/[c,d], L/[a|Zs]}
:- append([b],[c,d],Zs). {X1/b, Xs1/[], Ys1/[c,d], Zs/[b|Zs1]}
:- append([],[c,d],Zs1). {Zs1/[c,d], Xs2/[c,d]}
□