# Unification

## **1. Unification**

Unification is a two way matching process.

A **substitution** w = {x1/t1,...,xn/tn} is a mapping, where x1,...,xn are *distinct* variables. Given a term t and a substitution w,

w(t)

denotes the term obtained from t by simultaneously replacing each occurrence of the variable xi in w by the term ti.

In the following, we use capital letters to denote variables.

Although here we use "terms" (the objects that are composed of function symbols, variables, and constants), this definition of substitution also applies to other objects, such as atoms, a conjunction of atoms, etc.

For example, let

w = {X/b, Y/f(Z)}.

When it is applied to the term

f(X,g(Y))

we will get

f(b,g(f(Z))).

We may write this as

w[f(X,g(Y))] = f(b,g(f(Z)).

A **unifier** of two terms C1 and C2 is a substitution such that

w(C1) = w(C2)

That is, a unifier of two terms makes the two identical after the substitution.

* This definition can be easily extended to the case of an arbitrary number of terms.
* If there exists a unifier for objects t1 and t2, we then say t1 and t2 are *unifiable*.

Examples.

(a) Let t1 = f(a,X), t2 = f(Y,b), and w={X/b,Y/a}.   
  
 Then  
 w(t1) = w(t2).   
  
 So w is a unifier of t1 and t2.  
  
 (b) t1 = p(f(g(X,a)), X)  
 t2 = p(Z, b)  
  
 w = {X/b, Z/f(g(b,a))}  
   
 w(t1) = p(f(g(b,a)), b)  
 w(t2) = p(f(g(b,a)), b)   
  
 Thus, t1 and t2 are unifiable.  
  
 (c) t1 = p(f(X,X),Y)  
 t2 = p(f(a,Z),b)  
 w = {X/a, Z/a, Y/b}  
  
 w is a unifier of t1 and t2.  
  
 (d) t1 = p(X,X)  
 t2 = p(a,b)  
  
 There exists no unifier for the two atoms.   
  
 (e) t1 = p(f(X,Y), Z)  
 t2 = p(Z, f(a,Y))  
  
 w = {Z/f(a,Y), X/a}  
  
 w(t1) = w(p(f(X,Y), Z)) = p(f(a,Y), f(a,Y))  
 w(t2) = w(p(Z, f(a,Y)) = p(f(a,Y), f(a,Y))  
  
 Yes, w is a unifier of the two atoms.

## **2. The Most General Unifier**

In example (e) above, where

t1 = p(f(X,Y), Z)  
 t2 = p(Z, f(a,Y))

we know

w = {Z/f(a,Y), X/a}

is a unifier. There is however a different unifier

w' = {Z/f(a,b), X/a, Y/b}.

This is a unifier since

w'(t1) = w'(t2) = p(f(a,b), f(a,b))

The difference between w and w' is the last substitute Y/b in w' (which is already applied to the Y in f(a,Y)). In other words, w' can be obtained from w by replacing the ocurrences of Y in w by b.

On the reverse, w cannot be obtained from w': no matter how you replace occurrences of variables in the substitutes of w', you won't end up with w.

Because of this, we say that w is *more general than* w'. In fact, if two terms t1 and t2 are unifiable, there always exists a unique (up to variable renaming) *most general*unifier.

## **3. An Unification Algorithm**

We are then interested in an algorithm, hopefully an efficient one, for generating the most general unifier for two given terms.

Let us explain a unification algorithm by an example.

Example. Let  
 t1 = p(f(g(X,a)), X)  
 t2 = p(f(Y), b)

The unification process can be treated as a process of solving a system of equations. To start, we want to solve the system of the single equation

S0 = {p(f(g(X,a)), X) == p(f(Y), b))} w0 = { }

By this notation we mean that we want to find out substitutes such that both sides can be made identical. The empty set at the right is to be used to hold substitutes generated in this process.

To solve this system of equation is equivalent to solving the following system of equations:

S1= {f(g(X,a)) == f(Y), X == b} w1 = { }

This means we need to solve both of these equations in order to solve the original equation. The order in which these equations are solved is not important. But one may find it easier to solve the simple ones first. In this case, X == b.

This equation can be easily solved by having X bound to b. We now get a partial solution:

w2 = {X/b}

This means we have to commit X to b. This commitment should be consistent and applied to our system as well as to previously obtained substitutes. Thus, we get

S2 = w2(S1)   
 = {f(g(b,a)) == f(Y), b == b}

The second equation is solved. Continuing, we get

S3 = {g(b,a) == Y, b == b}   
 w3 = {Y/g(b,a), X/b}  
   
 S4 = w3(S3)   
 = {g(b,a) == g(b,a), b == b}

Thus, the original is solved with the substitution

{Y/g(b,a), X/b}

It can be shown that a substitution generated this way is the most general unifier for the terms in both sides of the original equation.

The following sequence shows that different orders in which equations are solved would generate the same solution

S0 = {p(f(g(X,a), X) == p(f(Y), b))} w0 = { }  
 S1 = {f(g(X,a)) == f(Y), X == b}   
 S2 = {g(X,a) == Y, X == b}   
 w1 = {Y/g(X,a)}  
 S3 = w1[ {g(X,a) == Y, X == b} ]   
 = {g(X,a) == g(X,a), X == b}   
 w2 = {Y/g(b,a), X/b}  
  
 S3 = w2[{g(X,a) == g(X,a), X == b}]   
 = {g(b,a) == g(b,a), b == b}

## **4. Occurs Check**

Consider unifying

t1 = likes(X, Y)   
 t2 = likes(g(Y), f(X))  
  
 S0 = {likes(X,Y) == likes(g(Y), f(X))} w0 = { }  
 S1 = {X == g(Y), Y == f(X)}   
 w1 = {X/g(Y)}  
 S2 = w1(S1)   
 = {g(Y) == g(Y), Y == f(g(Y))}  
 ^^^^^^^^^^^^  
 Y occurs in a term containing it.

An equation that has a variable on one side and a term containing that variable on the other side cannot be solved no matter what the substitute is. So the unification fails.

The above phenomenon is called "occurs check". In many Prolog systems, for efficiency reasons, occurs checks are not performed. They simply go ahead to generate a "unifier".

In this example, we would get,

{Y/f(g(Y)), X/g(f(g(Y)))}

We can see that it cannot be used to solve the second equation.

Therefore, Prolog without the occurs check is not a faithful implementation of Horn clause logic.

SWI Prolog omits the occurs check by default, but it has a switch to turn it on. See the discussion on https://en.wikipedia.org/wiki/Occurs\_check and the manual at http://www.swi-prolog.org/pldoc/man?predicate=unify\_with\_occurs\_check/2 .

Last modified: Monday, 29 February 2016, 9:05 PM