

---

# CS 421 – Spring 2007

## Lecture Notes Set 15: Unification Algorithm

Elsa L. Gunter<sup>1</sup>

Transcribed by: Pooja Mathur

---

**Corresponding to Slides:** 09 – unif

Made available: February 23, 2007

Revision 1.0

---

## 1 Change Log

1.0 Initial Release.

## 2 Background

### 2.1 Basic Idea (slide 2)

The unification algorithm that we are going to learn today is the one for the simple first order case. This is going to be somewhat similar to linear algebra. We will be unifying terms that are made up of constructors and variables. The constructors might be like cons. We want to be able to take pairs of terms and come up with ways of filling up the variables with some substitution, so in the end, the pairs of terms match in a way.

### 2.2 Simple Implementation (slide 3)

This data type captures the idea of what we are looking at. We have a type *term* and have two possibilities with it. Either, there is a variable (and to give the variable a name, we give `Variable` a string as its argument) or there is a constructor, called `Const`, which takes a string for a name and then a list of arguments.

```
type term = Variable of string
          | Const of (string * term list)
```

Now we need to start filling the functions we need and the first one we are going to use is substitution. Here, the idea is that we want to be able to substitute, for some variable, which is called *vn* in the function, a particular term, that in the function is called *residue*, in some other term. So, to do this, we are going to go by the cases. Either there is a variable or a constructor. If we found a variable then we are going to leave behind the residue if the variable is the variable we are supposed to be replacing. So we do the check  $vn = n$  and if it is, we return the residue otherwise we leave behind the variable we found, *n*. Remember in this match case, the *term* was matched with *Variable n*, so when we say we are returning *term* we are really returning the *Variable n*.

The other case is that there is a constructor. If we did find a constructor, we don't want to replace the constructor, but instead, we have to walk inside the list of arguments that the constructor is holding and look for where we can do the replacements.

```
let rec subst vn residue term =
  match term with Variable n →
    if vn = n then residue else term
  | Const (c, tys) →
    Const (c, List.map (subst vn residue) tys);;
```

---

<sup>1</sup>© 2007, Share and Enjoy

So to make sure this is clear, this is substituting one value for one variable. Another step would be to replace a collection of values with a collection of variables.

### 2.3 Uses (slide 5)

Unification can be used for numerous things. It can be helpful in databases, logic programming, parsing algorithms.

Pattern matching is a somewhat simplified version of unification.

We are going to be using it for type checking and type inferencing.

## 3 Unification Problem

### 3.1 The Problem (slide 4)

The problem is a set of equations. Like in linear algebra, if you given a system of equations to be solved. The problem is the system of equations. Well here, the problem is a system of terms, where the terms are supposed to be made equal by filling in values for your variables.

So if you had the set of terms:  $(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)$

We want to do some substitution,  $\sigma$  (or unification solution) from the variables  $s_i$  to terms  $t_i$  such that  $\sigma(s_i) = \sigma(t_i)$

### 3.2 A very simple example

So if you were given two equations:

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

$$g(y) = g(7)$$

$f$  and  $g$  are constructors.  $x$  and  $y$  are variables. How can we make those two equations fit our scheme? What values need to be filled in for  $x$  and  $y$  to make this work?

So to solve the last equation, you set  $y \rightarrow 7$  and now you can rewrite the equations as:

$$f(x) = f(g(7))$$

$$g(7) = g(7)$$

$$y \rightarrow 7$$

We had to replace the  $y$  with  $7$  because we found that match. The last step is to set  $x \rightarrow g(7)$  and we get our solution:

$$f(g(7)) = f(g(7))$$

$$g(7) = g(7)$$

$$y \rightarrow 7, x \rightarrow g(7)$$

While solving this simple example, you probably didn't notice, but you just did most of the steps of unification.

## 4 The Algorithm

### 4.1 Input (slide 6)

Like described earlier, we are going to take our terms and variables in as a set:

$$S = (s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)$$

When actually writing the algorithm, you will have to put the input in some order, but what order that is doesn't matter.

So, we have a set of equations, now how do we go about unifying them? Well, you have to look at the different possibilities for the set.

We could have the case that  $S$  is the empty set, so we don't have to do anything. And the best way of doing nothing here would be replacing everything with itself. So unifying  $S$  would simply be the identity function.

The other case is that there is at least one pair. So  $S = (s, t) \cup S'$  which has 4 main steps.

## 4.2 4 Main Steps (slides 7-8)

One possibility is that  $s$  already equals  $t$  (they are the same term) so that means that  $\text{Unif}(S) = \text{Unif}(S')$ . Since they are the same terms, this particular equation is contributing nothing, so we are just going to throw it away (or **delete** it).

The next possibility is that we have to start looking at the terms themselves. So we have to look in the equations and see which variables occur. So with **decompose** we say that  $s$  and  $t$  are constructors, where  $s = f(q_1, q_2, \dots, q_m)$  and  $t = f(r_1, r_2, \dots, r_m)$  (note that  $s$  and  $t$  are constructors for the same  $f$  so the number of variables  $m$  is the same). Now decomposing  $s$  and  $t$  into their internal variables is what we are going to substitute into the set  $S$ . That is,  $\text{Unif}(S)$  (the set with  $s$  and  $t$ ) =  $\text{Unif}((q_1, r_1), (q_2, r_2), \dots, (q_m, r_m) \cup S')$  (the set with paired variables from the decomposed  $s$  and  $t$ ).

In the next case we will be doing a bit of shuffling around. Think back to linear algebra with you had something like  $0 = 2x + 3y$ , what most people would do is flip the equation around so that it would look like  $2x + 3y = 0$ , so that it lines up with the other equations. That is basically the idea of **orient**. So say that  $s$  is not a variable, but  $t$  is a variable and is equal to  $x$ . So that we are going to swap them around so we have that the variable is equal to the term, instead of the term being equal to the variable.

The last case is called **eliminate**. In this case, we have  $s$  is a variable and  $t$  is something, we don't know what it is. Here, if the value we are looking for,  $x$ , is a variable we need to make sure  $x$  does not appear in the right hand side. If  $x$  did appear on the right hand side, then that would mean the right hand side was an infinite term. That is, if  $x = t(s, x)$  then we could rewrite it as  $t(s, t(s, x))$ ...and this could go on forever. So, once you do the occurs check to make sure there are no infinite terms, you now have a partial solution. That is,  $x$  has to get mapped to  $t$ . So that you have this mapping, you can go back into the set of terms and find any place that uses this, you can substitute. This gives you a new set of equations, that you can solve for.

You have your whole set of equations  $S$  and basically what we have is  $S'$  with the  $x \rightarrow t$  which means we have that  $x$  is being mapped to everything in  $S'$  that has a  $t$ . That is what we are trying to unify next. So, when we are done, we will have filled in all the variables  $s$  that needed to be filled in.

## 4.3 Some Tricks (slide 9)

This algorithm is very slow. Everytime we find a substitution, we have to go through the whole term list and do the substitution, over and over and over again. The best way you could alter this is by making more complicated data structures that get updated incrementally. However, we can't implement our algorithm this way, because we haven't learned this yet.

## 5 An Example (slides 10-28)

$x, y,$  and  $z$  are variables and  $f$  and  $g$  are constructors. We also have that the set  $S$ :

$$S = (f(x), f(g(y,z))), (g(y,f(y)), x)$$

Also note that  $f$  takes one argument and  $g$  takes two.

Now, we have two equations (two pairs) and we want to know if we can solve it. So the first step is to pick a pair. It doesn't matter which. We will arbitrarily pick the second one. So we are looking at  $(g(y,f(y)), x)$ . Which of the four rules applies to it? Here we have that a constructor is equal to a variable, that is,  $g(y,f(y)) = x$ , but we want  $x = g(y,f(y))$ . This is referring to the orient rule, where  $s$  and  $t$  switch sides. Doing so gives us:

$$S = (f(x), f(g(y,z))), (x, g(y,f(y)))$$

Now, let's look at the left side, the  $(f(x), f(g(y,z)))$ . What rule is applied here? In this pair, we have an  $f$  of something is equal to an  $f$  of something. That sounds a lot like what the rule for decompose is looking for. We don't have to go any farther than the outer most constructor and the number of arguments it takes. So that's what we do, we remove the  $f$ 's and then we get  $(x, g(y,z))$ , which gives us a new set:

$$S = (x, g(y,z)), (x, g(y,f(y)))$$

Now we pick another equation, and this time, we will look at  $(x, g(y,f(y)))$ . Now what rule are we looking for? This is a candidate for eliminate, where a variable is equal to a term. But before deciding that we had to make sure that  $x$  did not occur inside  $g$ . You can simply look at  $g(y,f(y))$  and see that this is true. After doing the occurs check, we are now able to do the substitution. Now we have:

$$S = (g(y, f(y)), (g(y, z))) \text{ and} \\ x \mid \rightarrow g(y, f(y))$$

This part was the  $x \rightarrow t$  part that we had before. Also, we after eliminating the term  $(x, g(y, f(y)))$  from the set  $S$ , we replace every other instance of  $x$  in  $S$  with  $g(y, f(y))$ . So in the above equation the  $g(y, f(y))$  was an  $x$  in the equation above that.

Now, with what is left, we can perform decompose, since the outer most constructor is  $g$  for both sides of the equation. From that we get:

$$S = (y, f(y)), (y, z) \text{ and} \\ x \mid \rightarrow g(y, f(y))$$

Then we have that the argument positions get set equal to one another. That is, on the left, we have  $(y, f(y))$  and on the right we have  $(y, z)$ . Well the first argument on the left is  $y$  and the first argument on the right is  $y$ , so that becomes a new pair in  $S$ . Then, using the same idea, the  $f(y)$  is set equal to  $z$ . Note, we did this when we decomposed  $f$ , but since there was only one argument, it wasn't that obvious. Our new  $S$  looks like:

$$S = (y, y), ((f(y)), z) \text{ and} \\ x \mid \rightarrow g(y, f(y))$$

Next step, well, like previous steps, we pick a pair, and this time, we pick the pair  $(y, y)$ . Well, here we have the case where  $s$  is equal to  $t$  and that refers to the delete rule, which means we just throw away the pair and we get:

$$S = ((f(y)), z) \text{ and} \\ x \mid \rightarrow g(y, f(y))$$

We just have one pair to look at and that is  $((f(y)), z)$ . Here we want to get  $z$  on the other side, do we orient the equation and get:

$$S = (z, (f(y))) \text{ and} \\ x \mid \rightarrow g(y, f(y))$$

We again, pick the only pair available and since we have a variable equal to a term in the pair, we have a candidate for eliminate again. That gives:

$$S = \text{and} \\ x \mid \rightarrow z \mid \rightarrow f(y) \ g(y, f(y)) \circ z \mid \rightarrow (f(y))$$

So eliminate involves a substitution that  $z \mid \rightarrow (f(y))$ . We first backsubstitute into ourself, which is the identity function. Then we have to start back substituting so we go into our other substitution from earlier. So we apply the substitution,  $z \mid \rightarrow (f(y))$  to the term  $g(y, f(y))$ . Since there are no variables where we can replace it with  $z$ , only terms, we end up with the solution:

$$x \mid \rightarrow g(y, f(y)) \circ z \mid \rightarrow (f(y))$$

To show our solution is correct, we redo the substitutions. That is, we want to show that:

$$S = (f(x), f(g(y, z))), (g(y, f(y)), x)$$

is solved by:

$$x \mid \rightarrow g(y, f(y)) \circ z \mid \rightarrow (f(y))$$

Well,  $f(x) = f(g(y, f(y))) = f(g(y, z)) = f(g(y, f(y)))$ , so that holds and  $g(y, f(y)) = x = g(y, f(y))$ . So that also holds, so our solution is correct.