

CS421 Lecture 7: Unification¹

Mark Hills

`mhills@cs.uiuc.edu`

University of Illinois at Urbana-Champaign

June 13, 2006

¹Based on slides by Mattox Beckman, as updated by Vikram Adve, Gul Agha, and Elsa Gunter

Objectives

The Problem

The Algorithm

Implementation

Activity

Objectives

Unification is an important technique in programming languages. It is used to implement several important languages features in languages such as OCaml (where it is used for pattern-matching and polymorphic type inference) and Prolog (where it used as the core evaluation process for logic rules). It is also used for type inference in other languages, many modern pointer analysis algorithms, and in theoretical discussions.

After this lecture (and MP3), you should

- ▶ be able to describe the problem of unification;
- ▶ know how to use unification to implement pattern matching;
- ▶ know how to use unification to check types of functions.

The Domain

Terms Have *name* and *arity*

- ▶ The name will be in western alphabet
- ▶ Arity = “number of arguments” — may be zero
- ▶ Terms used as *constructors* to build other terms
- ▶ Constructors with different arities considered different
- ▶ Examples: x , z , $f(x,y)$, $x(y,f,z)$

Note: arguments to terms may have non-zero arity, or may be variables.

The Domain, cont.

Variables Written using Greek alphabet, may be subscripted

- ▶ Represent a target for substitution
- ▶ Examples: $\alpha, \beta_{12}, \gamma_7$

Substitutions Mappings from Variables to Terms

- ▶ Examples: $\sigma = \{\alpha \mapsto f(x, \beta), \beta \mapsto y\}$
- ▶ Substitutions are *applied*: $\sigma(g(\beta)) \rightarrow g(y)$

The Problem

- ▶ Given terms s and t , try to find a substitution σ such that $\sigma(s) = \sigma(t)$.
- ▶ If such a substitution exists, it is said that s and t unify.
- ▶ A *unification problem* is a set of equations $S = \{s_1 = t_1, s_2 = t_2, \dots\}$.
- ▶ A unification problem $S = \{x_1 = t_1, x_2 = t_2, \dots\}$ is in *solved form* if
 - ▶ the terms x_i are distinct variables
 - ▶ none of them occur in t_i .

Our approach: given a unification problem S , we want to find the most general unifier σ that solves it. We will do this by transforming the equations.

Example — Pattern Matching

Pattern Matching can be implemented via unification

```

1 let lst = 3::4::5::[];;
2 match lst with
3   | [] -> ...
4   | x::xs -> ....
  
```

- ▶ We want to unify lst with $[]$ or $x :: xs$.
- ▶ We cannot unify lst with $[]$.
- ▶ Let $S = \{\alpha_x :: \alpha_{xs} = lst; \quad lst = 3::4::5::[]\}$
- ▶ $\sigma(\alpha_{xs}) = 4::5::[]$
- ▶ $\sigma(\alpha_x) = 3$.

Example — Type Checking

Unification is widely used for type checking and type inference

```

1 # let rec ListAssoc lst2 n = match lst2 with ...
2 val ListAssoc : ('a * 'b) list -> 'a -> 'b = <fun>
3 # (ListAssoc [(1,2); (3,5)] 3) < 1.0;; (* ERROR *)
4 # (ListAssoc [(1,2.0); (3,5.0)] 3) < 1.0;; (* bool *)
  
```

- ▶ Let γ be the type of the return value of ListAssoc in last step
- ▶ Function application tells us: $list(\alpha, \beta) = list(int, float)$ and $\alpha = int$
- ▶ The comparison requires $\gamma = float$. The result tells us $\gamma = \beta$.
- ▶ $S = \{list(\alpha, \beta) = list(int, float); \gamma = \beta; \gamma = float\}$
- ▶ Let $\sigma = \{\alpha \mapsto int; \beta \mapsto float; \gamma \mapsto float\}$

Type Inference

MP3 will focus on type inference. Overall, we will:

- ▶ begin by assigning a type variable as the type of the whole expression
- ▶ decompose the expression into component expressions
- ▶ use typing rules to generate constraints on the components and the entire expression
- ▶ recursively gather additional constraints over the structure of the term
- ▶ solve (*unify*) the system of constraints to generate a substitution
- ▶ apply this substitution to the type variable assigned to the whole expression to get the expression type

Four Operations

Start with a unification problem $S = \{s_1 = t_1, s_2 = t_2, \dots\}$ and apply the following transformations as necessary:

Delete A trivial equation $t = t$ can be deleted.

Decompose An equation $f(\overline{t_n}) = f(\overline{u_n})$ can be replaced by the set $\{t_1 = u_1, \dots, t_n = u_n\}$, with f and n same in both

Orient An equation $t = x$ can be replaced by $x = t$ if x is a variable and t is not.

Eliminate an equation $x = t$ can be used to substitute all occurrences of x in the remainder of S . x must not occur in t !

Example 1

(Stolen from “Term Rewriting and All That”)
 $\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$

Example 1

(Stolen from “Term Rewriting and All That”)

$$\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

Example 1

(Stolen from “Term Rewriting and All That”)

$$\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha = f(x), g(f(x), f(x)) = g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

Example 1

(Stolen from “Term Rewriting and All That”)

$$\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha = f(x), g(f(x), f(x)) = g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

$$\{\alpha = f(x), f(x) = f(x), f(x) = \beta\}$$

We can delete the $f(x) = f(x)$ equation.

Example 1

(Stolen from “Term Rewriting and All That”)

$$\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha = f(x), g(f(x), f(x)) = g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

$$\{\alpha = f(x), f(x) = f(x), f(x) = \beta\}$$

We can delete the $f(x) = f(x)$ equation.

$$\{\alpha = f(x), f(x) = \beta\}$$

Now we can reorient to make the variables show up on the left side.

Example 1

(Stolen from “Term Rewriting and All That”)

$$\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$$

We can use the Eliminate method, replace α with $f(x)$ on the right sides of the equations.

$$\{\alpha = f(x), g(f(x), f(x)) = g(f(x), \beta)\}$$

We can use the Decompose method, and get rid of the g functions.

$$\{\alpha = f(x), f(x) = f(x), f(x) = \beta\}$$

We can delete the $f(x) = f(x)$ equation.

$$\{\alpha = f(x), f(x) = \beta\}$$

Now we can reorient to make the variables show up on the left side.

$$\{\alpha = f(x), \beta = f(x)\}$$

Now we are done....

$$S = \{\alpha \mapsto f(x), \beta \mapsto f(x)\}$$

Example 2

Start with $\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

Example 2

Start with $\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

$\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

Use the decompose method to get rid of f .

Example 2

Start with $\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

$\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

Use the decompose method to get rid of f .

$\{\alpha = g(x), g(\beta) = \alpha\}$

Now we can substitute $g(x)$ for α .

Example 2

Start with $\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

$\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

Use the decompose method to get rid of f .

$\{\alpha = g(x), g(\beta) = \alpha\}$

Now we can substitute $g(x)$ for α .

$\{\alpha = g(x), g(\beta) = g(x)\}$

Use decomposition to eliminate g .

Example 2

Start with $\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

$\{f(\alpha, g(\beta)) = f(g(x), \alpha)\}$

Use the decompose method to get rid of f .

$\{\alpha = g(x), g(\beta) = \alpha\}$

Now we can substitute $g(x)$ for α .

$\{\alpha = g(x), g(\beta) = g(x)\}$

Use decomposition to eliminate g .

$\{\alpha = g(x), \beta = x\}$

We are done.

Example 3

Start with $\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

Example 3

Start with $\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

$\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

Use the decompose method to get rid of f .

Example 3

Start with $\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

$\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

Use the decompose method to get rid of f .

$\{\alpha = h(y), g(y) = \alpha\}$

Now orient $g(y) = \alpha$

Example 3

Start with $\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

$\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

Use the decompose method to get rid of f .

$\{\alpha = h(y), g(y) = \alpha\}$

Now orient $g(y) = \alpha$

$\{\alpha = h(y), \alpha = g(y)\}$

Use substitution for α .

Example 3

Start with $\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

$\{f(\alpha, g(y)) = f(h(y), \alpha)\}$

Use the decompose method to get rid of f .

$\{\alpha = h(y), g(y) = \alpha\}$

Now orient $g(y) = \alpha$

$\{\alpha = h(y), \alpha = g(y)\}$

Use substitution for α .

$\{\alpha = h(y), h(y) = g(y)\}$

We're stuck – no rules apply, but we haven't unified.

How to do it....

- ▶ There is a simple algorithm that works well most of the time.

```

1 type exp = Var of string
2           | Term of string * exp list
  
```

- ▶ A unification equation is a pair, a unification problem is a list of pairs.
- ▶ Example: $\{\alpha = f(x), g(\alpha, \alpha) = g(\alpha, \beta)\}$

```

1 let problem = [
2   Var "alpha", Term("f", [Term("x", [])]);
3   Term("g", [Var "alpha"; Var "alpha"]),
4   Term("g", [Var "alpha"; Var "beta"]) ]
  
```

The Rules

- ▶ To unify a problem p ...
 - ▶ If p is $[],$ then you are done.
 - ▶ Otherwise, decompose p into $e :: es.$
- ▶ If e is $\text{Var } n, m,$ then substitute m for $\text{Var } n$ in $es.$ Then “emit” $e :: \text{unify } es$
- ▶ If the second part contains a variable, you need to put it at the end of es rather than emitting it.

```

unify [Var "alpha", Term("f",[Term("x",[[]]))];
      Term("g",[Var "alpha"; Var "alpha"]),
      Term("g",[Var "alpha"; Var "beta"]) ]
  
```

will become...

The Rules

- ▶ To unify a problem p ...
 - ▶ If p is $[],$ then you are done.
 - ▶ Otherwise, decompose p into $e :: es.$
- ▶ If e is $\text{Var } n, m,$ then substitute m for $\text{Var } n$ in $es.$ Then “emit” $e :: \text{unify } es$
- ▶ If the second part contains a variable, you need to put it at the end of es rather than emitting it.

```
(Var "alpha", Term("f", [Term("x", [])])) ::
unify [Term("g", [Term("f", [Term("x", [])]);
      Term("f", [Term("x", [])])]);
      Term("g", [Term("f", [Term("x", [])]);
      Var "beta"])]
```

Decomposing

- ▶ If e consists of a pair $\text{Term}(f, [x_1; x_2; \dots; x_n])$, $\text{Term}(f, [y_1; y_2; \dots; y_n])$...
 - ▶ Note, term names and arities are the same.
- ▶ Then the problem becomes

$$\text{unify}(\left(\bigcup_{i=1}^n [(x_i, y_i)]\right) \cup es)$$

```
(Var "alpha", Term("f", [Term("x", [])])) ::
unify [Term("f", [Term("x", [])]),
      Term("f", [Term("x", [])])] @
unify [Term("f", [Term("x", [])]);
      Var "beta"]])
```

Equalities

- ▶ If e consists of (s, t) , where $s=t$ and s does not contain variables, then discard the pair.

```
(Var "alpha", Term("f", [Term("x", [])])) ::
unify [Term("f", [Term("x", [])]),
Term("f", [Term("x", [])])] @
unify [Term("f", [Term("x", [])]);
  Var "beta"]]
```

- ▶ Finally, if t is a variable and s does not contain variables, flip and emit.

```
[Var "alpha", Term("f", [Term("x", [])]);
  Var "beta", Term("f", [Term("x", [])])]
```

Problem

Try to unify the following:

- ▶ $\{f(\alpha, y) = f(x, \beta)\}$
- ▶ $\{f(\alpha, y) = f(x, \alpha)\}$
- ▶ $\{f(\alpha, \beta) = \gamma; \quad \gamma = f(x, \delta); \quad \beta = g(y)\}$
- ▶ $\{f(\alpha, \beta) = \gamma; \quad \gamma = f(x, \delta); \}$

Answers

- ▶ $\{f(\alpha, y) = f(x, \beta)\}$
Let $\alpha \mapsto x; \beta \mapsto y$
- ▶ $\{f(\alpha, y) = f(x, \alpha)\}$
This one cannot be unified.
- ▶ $\{f(\alpha, \beta) = \gamma; \gamma = f(x, \delta); \beta = g(y)\}$
Let $\{\beta, \delta \mapsto g(y); \alpha \mapsto x; \gamma \mapsto f(x, g(y))\}$
- ▶ $\{f(\alpha, \beta) = \gamma; \gamma = f(x, \delta); \}$
Let $\{\beta \mapsto \delta; \alpha \mapsto x; \gamma \mapsto f(x, \beta)\}$
Unifies, but not completely solved.