Programming Languages and Compilers (CS 421)

Elsa L Gunter

2112 SC, UIUC

http://www.cs.uiuc.edu/class/fa06/cs421/

Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

Background for Unification

- Terms made from constructors and variables (for the simple first order case)
- Constructors may be applied to arguments (other terms) to make new terms
- Variables and constructors with no arguments are base cases
- Constructors applied to different number of arguments (arity) considered different
- Substitution of terms for variables

Simple Implementation Background

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

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);;
```


Unification Problem

Given a set of pairs of terms ("equations")

$\{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$

(the *unification problem*) does there exist a substitution σ (the *unification solution*) of terms for variables such that

$\sigma(s_i) = \sigma(t_i)$,

for all I = 1, …, n?

Uses for Unification

- Type Inference and type checking
- Pattern matching as in OCAML
  - Can used a simplified version of algorithm
- Logic Programming - Prolog
- Simple parsing

Unification Algorithm

- Let $S = \{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$ be a unification problem.
- Case S = { }: Unif(S) = Identiy function (ie no substitution)
- Case $S = \{(s, t)\} \cup S'$): Four main steps

Unification Algorithm

- Delete: if s = t (they are the same term) then Unif(S) = Unif(S')
- Decompose: if s = f(q_1, … , q_m) and t =f(r_1, … , r_m) (same f, same m!), then Unif(S) = Unif({(q_1, r_1), …, (q_m, r_m)} ∪ S')
- Orient: if t = x is a variable, and s is not a variable, Unif(S) = Unif ({(x,s)} ∪ S')

Unification Algorithm

- Eliminate: if s = x is a variable, and x does not occur in t (the occurs check), then
  - Let φ = x |→ t
  - Let ψ = Unif(φ(S'))
  - Unif(S) = {x |→ ψ(t)} o ψ

Tricks for Efficient Unification

- Don't return substitution, rather do it incrementally
- Make substitution be constant time
  - Requires implementation of terms to use mutable structures (or possibly lazy structures)
  - We haven't discussed these yet

Example

- x,y,z variables, f,g constructors
- S = {(f(x), f(g(y,z))), (g(y,f(y)),x)}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (g(y,f(y)),x)
- S = {(f(x), f(g(y,z))), (g(y,f(y)),x)}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (g(y,f(y))),x)
- Orient is first rule that applies
- S = {(f(x), f(g(y,z))), (g(y,f(y)),x)}

Example

- x,y,z variables, f,g constructors
- S -> {(f(x), f(g(y,z))), (x,g(y,f(y)))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (f(x), f(g(y,z)))
- S -> {(f(x), f(g(y,z))), (x,g(y,f(y)))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (f(x), f(g(y,z)))
- Decompose it (x, g(y,z))
- S -> {(x, g(y,z)), (x,g(y,f(y)))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (x,g(y,f(y)))
- S -> {(x, g(y,z)), (x,g(y,f(y)))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (x,g(y,f(y)))
- Substitute:
- S -> {(g(y,f(y)), g(y,z))}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (g(y,f(y)), g(y,z))
- S -> {(g(y,f(y)), g(y,z))}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (g(y,f(y)), g(y,z))
- Decompose: (y,y) and (f(y), z)
- S -> {(y,y), (f(y),z)}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (y,y)

- S -> {(y,y), (f(y),z)}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (y,y)
- Eliminate
- S -> {(f(y),z)}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (f(y),z)

- S -> {(f(y),z)}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (f(y),z)
- Orient
- S -> {(z,f(y))}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (z,f(y))

- S -> {(z,f(y))}

With {x |→ g(y,f(y))}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (z,f(y))
- Substitute
- S -> { }

With {x |→ {z |→ f(y)} (g(y,f(y)) } o {z |→ f(y)}

Example

- x,y,z variables, f,g constructors
- Pick a pair: (z,f(y))
- Substitute
- S -> { }

With {x |→ g(y,f(y))} o {(z |→ f(y))}

Example

S = {(f(x), f(g(y,z))), (g(y,f(y)),x)}

Solved by {x |→ g(y,f(y))} o {(z |→ f(y))}

f(g(y,f(y))) = f(g(y,f(y)))

x z

and

g(y,f(y)) = g(y,f(y))

x

Example of Failure

- S = {(f(x,g(y)), f(h(y),x))}
- Decompose
- S -> {(x,h(y)), (g(y),x)}
- Orient
- S -> {(x,h(y)), (x,g(y))}
- Substitute
- S -> {(h(y), g(y))} with {x |→ h(y)}
- No rule to apply! Decompose fails!