Unification Activity

Mattox Beckman

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to Class	

Purpose

Unification is a core component of many programming language related algorithms. It is important to be able to solve unification problems by hand, as well as to be able to specify to the computer how to solve such a problem.

Your objectives:

- Explain the syntax and usage of ϕ as a substitution operator.
- Identify the proper situations for each of the four unification rules and the results.
- Explain the necessity of the occurs-check.
- Implement the unification rules in HASKELL.

Part 1 — ϕ Day

For the following table, let $\phi = \{x \mapsto 10, y \mapsto 2\}$

Formula	Result
$\phi(\{(x, y)\})$	$\{(10, 2)\}$
$\phi(\{(a, x), (y, z)\})$	$\{(a, 10), (2, z)\}$
$\phi[x \mapsto z](\{(x, y)\})$	$\{(z, 2)\}$
$\phi[z \mapsto 5](\{(a, x), (y, z)\})$	$\{(a, 10), (2, 5)\}$
$\phi[z \mapsto 5][y \mapsto 20](\{(a, x), (y, z)\})$	$\{(a, 10), (20, 5)\}$

Problem 1)

As a team, describe the behavior of ϕ.

Problem 2)

Now, solve these formulas. Let $phi = x \mapsto a, y \mapsto b$

Formula	Result
$\phi(\{(x, y)\})$	
$\phi(\{(a, x), (y, z)\})$	
$\phi[x \mapsto z](\{(x, y)\})$	
$\phi[z \mapsto x](\{(a, x), (y, z)\})$	
$\phi[z \mapsto x][y \mapsto c](\{(a, x), (y, z)\})$	

Part 2 — The Rules

Given a constraint set C, we define $unify(C)$ as...

- If C is empty, return the identity solution. $\phi(s) = s$
- Otherwise, let $(s, t) \in C$ and $C' = C \setminus \{(s, t)\}$.

  Delete If $s = t$ then $unify(C')$

  Orient If t is a variable and s is not, $unify(\{(t, s)\} \cup C')$.

  Decompose If P is a constructor, $s = P(s_1, \ldots, s_n)$ and $t = P(t_1, \ldots, t_n)$ then $unify(C' \cup \{(s_1, t_1), \ldots, (s_n, t_n)\}$

  Eliminate If s is a variable, and s does not occur in t, substitute s with t in C' to get C''. Then let $\phi = unify(C'')$ and retun $\phi[s \mapsto \phi(t)]$.

For these examples we will use $a = b$ to denote unification pairs (a, b).

Problem	Step	Result
$unify(\{g(\alpha, a) = g(b, \beta), h(\gamma, \gamma) = h(f(\alpha), \gamma)\})$	Decompose	$unify(\{h(\gamma, \gamma) = h(f(\alpha), \gamma), \alpha = b, a = \beta\})$
$unify(\{f(\alpha, \alpha) = f(\alpha, \alpha), h(\beta, g(\gamma)) = h(y, \delta)\})$	Delete	$unify(\{h(\beta, g(\gamma)) = h(y, \delta)\})$
$unify(\{f(\alpha) = \delta, g(\alpha) = g(\beta), h(\gamma, x) = h(\beta, \alpha)\})$	Orient	$unify(\{\delta = f(\alpha), g(\alpha) = g(\beta), h(\gamma, x) = h(\beta, \alpha)\})$

Problem 3) Solve the following unification problem, in the order specified above. Label the rule you use for each step.

$$unify(\{g(\alpha, a) = g(b, \beta), h(\gamma, \gamma) = h(f(\alpha), \gamma)\})$$

Problem 4) Solve the following unification problem, in the order specified above. Label the rule you use for each step.

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

Part 3 — It Never Occurred to Me

Problem 5) What happens when we try to solve this?

$$unify(\{f(\alpha) = f(f(\alpha))\})$$

Problem 6) Consider this HASKELL code. What is its type?

```
1 foo a = [foo a]
```

Code

First, review this code with another student. What does it do? How does it work? Write `occurs`, `phi`, and `unify`.

```haskell
import qualified Data.HashMap.Strict as H
import Data.Maybe (fromJust)
import Data.List (intersperse)

data Entity = Var String
            | Object String [Entity]
     deriving (Eq)

instance Show Entity where
show (Var s) = s
show (Object s []) = s
show (Object f xx) = concat $ f : "(" : intersperse "," (map show xx) ++ [")"]

isVar (Var _) = True
isVar _ = False

-- Environment functions

type Env = H.HashMap String Entity

initial :: Env
initial = H.empty

add :: String -> Entity -> Env -> Env
add x y b = H.insert x y b

contains :: String -> Env -> Bool
contains x b = H.member x b

-- Functions you get to write

phi :: Env -> Entity -> Entity
phi env (Var s) = undefined
phi env (Object s xx) = undefined

occurs :: String -> Entity -> Bool
occurs = undefined

unify :: [(Entity,Entity)] -> Env
unify [] = initial
unify ((s,t):c') = undefined
```