CS 421 --- Unification Activity

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to class	
Reflector	Assesses team performance	

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

Time estimate: 10 minutes.

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

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

Problem 1) As a team, describe the behavior of ϕ.

- If there is a mapping $x \mapsto y$ in ϕ, how many times will x be replaced in ϕ's argument?
- If there is a variable x that has no mapping in ϕ, what happens to the occurrences of x in ϕ's argument?
- If there is a mapping $x \mapsto y$ in ϕ, and we call the function $\phi[x \mapsto z]$, on a term x, which mapping wins?

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

Time estimate: 10 minutes

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 $\phi[s \mapsto \phi(t)]$.

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) The Eliminate rule rewrites ϕ to $\phi[s \mapsto \phi(t)]$. Why can't we just rewrite to $\phi[s \mapsto t]$ instead?

Problem 4) In Haskell, function calls like `zipWith xx yy` will truncate the longer of xx and yy if they are not the same size. The decompose rule doesn't do this. Why not?

Problem 5) 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 6) What happens when we try to solve this?

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

Problem 7) Consider this HASKELL code. What is its type?

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

Part 4 --- Show me the Code

Time estimate: 10 minutes.
Problem 8) Review this code with your team. What does it do? How does it work? To liven things up I put in a couple bugs for you to find.

```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 _) = False
isVar _ = True

-- Environment functions

type Env = H.HashMap String Entity

initial :: Env
initial = H.empty

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

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

-- 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
```

Part 5 --- Let's Do This

Problem 9) Write `occurs`.

Problem 10) Write `unify`.

Unification Activity --- Team's Assessment (SII)

Manager or Reflector: Consider the objectives of this activity and your team's experience with it, and then answer the following questions after consulting with your team.

1. What was a **strength** of this activity? List one aspect that helped it achieve its purpose.

2. What is one things we could do to **improve** this activity to make it more effective?

3. What **insights** did you have about the activity, either the content or at the meta level?

Unification Activity--- Reflector's Report

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

1. What was a strength of your team's performance for this activity?

2. What could you do next time to increase your team's performance?

3. What insights did you have about the activity or your team's interaction today?