CS 421 --- Unification Activity

<table>
<thead>
<tr>
<th>Name</th>
<th>Netid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please write your name/netid legibly in dark ink. Hand in one copy per team. Do not staple or mangle the corners.

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 $\phi$ 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 --- $\phi$ Day

Time estimate: 10 minutes.

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

<table>
<thead>
<tr>
<th>Formula</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi({(x, y)})$</td>
<td>${(10, 2)}$</td>
</tr>
<tr>
<td>$\phi({(a, x), (y, z)})$</td>
<td>${(a, 10), (2, z)}$</td>
</tr>
<tr>
<td>$\phi[x \mapsto z]({(x, y)})$</td>
<td>${(z, 2)}$</td>
</tr>
<tr>
<td>$\phi[z \mapsto 5]({(a, x), (x, z)})$</td>
<td>${(a, 10), (10, 5)}$</td>
</tr>
<tr>
<td>$\phi[z \mapsto 5][y \mapsto 20]({(a, x), (y, z)})$</td>
<td>${(a, 10), (20, 5)}$</td>
</tr>
</tbody>
</table>

Problem 1) As a team, describe the behavior of $\phi$.

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

Problem 2) Now, solve these formulas. Let $\phi = \{ x \mapsto a, y \mapsto b \}$

<table>
<thead>
<tr>
<th>Formula</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi({(x, y)})$</td>
<td>${(a, b)}$</td>
</tr>
<tr>
<td>$\phi({(a, x), (y, z)})$</td>
<td>${(a, a), (b, z)}$</td>
</tr>
<tr>
<td>$\phi[x \mapsto z]({(x, y)})$</td>
<td>${(a, z), (b, z)}$</td>
</tr>
<tr>
<td>$\phi[z \mapsto x]({(a, x), (y, z)})$</td>
<td>${(a, a), (y, z)}$</td>
</tr>
<tr>
<td>$\phi[z \mapsto x][y \mapsto c]({(a, x), (y, z)})$</td>
<td>${(a, a), (c, z)}$</td>
</tr>
</tbody>
</table>
Part 2 --- The Rules

Time estimate: 10 minutes
Given a constraint set $C$, we define $\text{unify}(C)$ 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 $\text{unify}(C')$
**Orient**  If $t$ is a variable and $s$ is not, $\text{unify}((t, s) \cup C')$.
**Decompose**  If $P$ is a constructor, $s = P(s_1, \ldots, s_n)$ and $t = P(t_1, \ldots, t_n)$ then $\text{unify}(C' \cup \{(s_1, t_1), \ldots, (s_n, t_n)\})$.
**Eliminate**  If $s$ is a variable, and $s$ does not occur in $t$, substitute $s$ with $t$ in $C'$ to get $C''$. Then let $\phi = \text{unify}(C'')$ and return $\phi[s \mapsto \phi(t)]$.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Step</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{unify}{g(\alpha, a) = g(b, \beta), h(\gamma, \gamma) = h(f(\alpha), \gamma)}$</td>
<td>Decompose</td>
<td>$\text{unify}{h(\gamma, \gamma) = h(f(\alpha), \gamma), \alpha = b, a = \beta}$</td>
</tr>
<tr>
<td>$\text{unify}{f(\alpha) = f(\alpha), h(\beta, g(\gamma)) = h(\gamma, \delta)}$</td>
<td>Delete</td>
<td>$\text{unify}{h(\beta, g(\gamma)) = h(\gamma, \delta)}$</td>
</tr>
<tr>
<td>$\text{unify}{f(\alpha) = \delta, g(\alpha) = g(\beta), h(\gamma, x) = h(\beta, \alpha)}$</td>
<td>Orient</td>
<td>$\text{unify}{\delta = f(\alpha), g(\alpha) = g(\beta), h(\gamma, x) = h(\beta, \alpha)}$</td>
</tr>
</tbody>
</table>

**Problem 3** The Eliminate rule rewrites $\phi$ 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.

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

\[ \circ \text{foo} \ a = \text{[foo a]} \]
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.