CS 421 — Unification Activity

Why

Unification is a major component of programming language theory. It is the basis of the programming language Prolog, and concepts such as type checking and semantics make heavy use of it. In this activity you will complete the operation of a unification engine written in Haskell.

Examples and Problems

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

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

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

Code

First, review this code with another student. What does it do? How does it work?

```haskell
module Unify where

import qualified Data.HashMap.Strict as H
import Data.Maybe
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) ++ [")"]

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

unifyVar :: Entity -> Env -> Entity
unifyVar x@(Var t) bindings
   | contains t bindings = fromJust $ H.lookup t bindings
   | otherwise = x
unifyVar x _ = x

unify :: Entity -> Entity -> Env -> Env
unify x y bindings = aux (unifyVar x bindings) (unifyVar y bindings) bindings
  where aux (Var s) x bindings = add s x bindings
        aux x (Var s) bindings = -- ???
        aux (Object f ff) (Object g gg) bindings = -- ??



        aux _ _ _ = H.empty
```