CS 240H NOTES: FUNCTIONAL SYSTEMS IN HASKELL

ARUN DEBRAY

JUNE 3, 2014

Contents

1. Basics: 4/1/14

This course is taught by David Mazières and Bryan O'Sullivan, who together have done a lot of systems programming and research and Haskell. Meanwhile, the CA, David Terei, is a member of the Haskell language standards committee...

The goal of this class is to learn how to use Haskell to program systems with reduced upfront cost. Haskell is typically taught in the context of programming language research, but this course will adopt a systems perspective. It's a surprisingly flexible and effective language, and since it was created by and for programming language resources, there are lots of interesting things to do, and it's extremely flexible (if you want to try something new, even if it's syntactical, this is as easy as using a library, unlike most programming languages).

The first week of this class will cover the basics of Haskell, though having some prior experience or a supplement (e.g. *Real World Haskell* or *Learn You a Haskell*) is helpful. After the basics, we will cover more advanced techniques. The class grade will be based on attendance and participation, with scribing one of the lectures, and also three warm-up programming assignments and a large final project and presentation, in groups of one to three people.

Now, let's talk about Haskell.

In order to use Haskell, one will want to install the Haskell platform or `cabal`, along with the Haskell compiler, `ghc`. The simplest program is

```haskell
main = putStrLn "Hello, world!"
```

Unsurprisingly, this is a "Hello, world!" program. One can compile it, e.g. `ghc -o hello hello.hs`, but also load it into an interpreter `ghci` (in this regard, Haskell is much like Lisp).

The first thing you've noticed is the equals sign, which makes a binding, e.g.

```haskell
x = 2         -- Two hyphens introduce a comment.
y = 3         -- Comments go until the end of a line.
main = let z = x + y -- let introduces local bindings.
       in print z
```