Haskell Introduction

Jean-Pierre Rupp

Haskoin Limited

root@haskoin.com

June 18, 2014

Haskell Qualities

- Referential transparency
- Lazy evaluation
- Type inference
- Immutable data structures
- Pattern Matching
- Property testing
- Made by really smart people (with PhDs)

Functions & Lists

```haskell
-- Functions
square x = x * x

-- Lists
smallNums = [1,2,3,4]
oneMillion = [1..1000000]
nats = [1..]
helloWorld = "hello world!"
abc = 'a':'b':'c':[]
```

Currying

```
-- Functions with multiple parameters
prod x y = x * y

-- Partially-applied functions
multBy4 = prod 4
addOne = (+1)
prodShort = (*)
square = (^2)

-- Anonymous functions
addTwo = \x -> x + 2
```

First-Class Functions

```haskell
-- Apply a function to all elements in list
map (+1) [1..10]           -- Add one
take 5 (map (^2) [1..])    -- Take five squared integers

-- Other list operations
sort "daynet"              -- "adenty"
filter (>10) [5..15]       -- [11,12,13,14,15]
concat ["ad","en","ty"]    -- "adenty"
```

Types

```haskell
-- Simple type
data Soca = OneCent | FiveCent | TenCent | Dollar

-- Polymorphic parametrized algebraic data type
data List a = Nil | Cons a (List a)
Cons 'a' (Cons 'b' (Cons 'c' Nil))   -- Equivalent to "abc"

-- Type alias (and some pattern matching)
type Complex = (Double, Double)
cpxProd :: Complex -> Complex -> Complex
cpxProd (a,b) (c,d) = ((a*c-b*d),(b*c+a*d))
```

Parametrized Types

```haskell
-- Find smallest
data Soca = OneCent | FiveCent | TenCent | Dollar
    deriving Ord

smallest :: Ord a => [a] -> a
smallest [] = undefined
smallest [x] = x
smallest (x:xs) = min x (smallest xs)

smallest [TenCent, FiveCent, Dollar, OneCent] -- OneCent
smallest [5,3,10,9]  -- 3
smallest "adenty"    -- 'a'
```

Monads

```haskell
-- Maybe monad
safeDiv :: Integral a => a -> a -> Maybe a
safeDiv x y = guard (y /= 0) >> return (x `div` y)

-- IO monad
main = do
    putStrLn "What is your name?"
    name <- getLine
    putStrLn ("Nice meeting you " ++ name ++ ".")
```