Introduction to Haskell

Rolf Fagerberg

Spring 2005

Haskell

Functional language (no assignments)

- Purely functional
- Statically typed
- Rich typesystem
- Lazy (infinite data structures OK)

Named after Haskell Brooks Curry (1900–1982, USA, mathematical logic).

Language in development. Haskell-1998: frozen version (used here). Concrete implementation: Hugs interpreter + libraries.

Functions

Math:

$$f(x) = 2x + 5$$ ← definitions
$$g(y, z) = yz^2 + z + 2$$ ⋮
$$\mathrm{abs}(x) = \begin{cases} x & \text{, if } x \geq 0 \\ -x & \text{, otherwise} \end{cases}$$
$$\mathrm{abs}(f(g(3, 2)))$$ ←evaluation

Haskell:

```
f x = 2*x + 5
g y z = y*z^2 + z + 2
abs x
   | x >= 0        =  x
   | otherwise     = -x

abs(f(g 3 2))
```

← definitions ⋮

←evaluation

Types

Math:

$$3.0 \in R$$

$$g : \; R \times R \longrightarrow R$$
$$g(y, z) = yz^2 + z + 2$$

Haskell:

`3.0` is of type `Float`

```
g ::  Float -> Float -> Float
g y z = y*z^2 + z + 2
```

Haskell Basic Elements

Names (identifiers, “variables”) associated with **Values** (integers, booleans, strings, and also functions)

Each value belongs to a **Type** (a domain/set of values)

Definitions associate names with values.

Literals creates basic values.

Functions (including operators: $+$, $*$,$\ldots$) take values to new values

Evaluation of **Expressions** build using basic values and functions. Cf. expression trees.

Haskell

Literals:

`277, -3.141527, 7.89e-6, 'A',‘‘Hello World’’`

Built-In Types

`Int, Bool, Float, Double, Integer, Rational, Complex,...`

Type Constructors (even more to come)

Lists ($\sim$ arrays): `[]`

```
a ::  [Int]
a = [1,2,3]
```

Tuples ($\sim$ records): `()`

```
b ::  (Char,Bool,Int)
b = ('A', True,1)
```

Type Synonyms

```
type String = [Char]
type Coordinate2D = (Float,Float)
```

Some Haskell Syntax

- Off-side rule
- Explicit definition end by “;”
- Comments
- Identifiers: Letter [Letter, Digit, _, ’]*

  Value names, parameters, (type parameters):
  Small initial letter
  Type names, (constructors, modules, type classes):
  Capital initial letter
- Some words reserved (see p. 49).

Recursion

No assignments $\Rightarrow$ no loops

(Loops over lists exist - see *list comprehensions* next time)

Hence, in functional programming, $\boxed{\text{recursion}}$ is used a lot.

Hugs

Interpreter (+ libraries) for Haskell-1998.

Reads **definitions** in script file(s).

Evaluates expressions written in its shell using definitions in script and built-in definitions in standard library `Prelude.hs`

Scripts can be standard (`.hs`) or literate (`.lhs`).