Haskell

PENG

Created: 20260913, Version: 20260913

Contents

1 Introduction

`https://www.engineering.upenn.edu/~cis1940/spring13/`

Haskell is a *lazy*, *functional* programming language created in the late 1980s.

1.1 Functional

Definition 1.1 (functional)**.**

- *functions are values*
- *evaluating expressions rather than executing instructions*

1.2 Pure

Definition 1.2 (referentially transparent)**.**

- *everything (var, data structures) is immutable*
- *expressions have no side effects*
- *same function and arguments = same output*

benefits:

- replace equals by equals
- parallelism

1.3 Lazy

Definition 1.3 (lazy)**.** *Haskell expressions are not evaluated until their results are actually needed.*

consequences:

- define a control structure by defining a function
- define infinite data structures
- time and space usage become more complicated

1.4 Statically Typed

Definition 1.4 (statically typed)**.** *Each Haskell expression has a type, and types are all checked at compile-time.*

1.5 Themes

- static type system
- abstraction principle: *every idea, algorithm, and piece of data should occur exactly once in your code*
- wholemeal programming: *first solve a more general problem, then transform it into more specialized ones*

1.6 Declarations and variables

Definition 1.5 (=)**.** *= denotes definition such that variables are not mutable*

Example 1.6.
```
x :: Int
x = 3
x = 4 -- error, multiple declaration of x
y :: Int
y = y + 1 -- <<loop>>
```

1.7 Basic Types

- `Int` : at least up to $\pm 2^{29}$
- `Integer`
- `Double`
- `Float`
- `Bool`
- `Char`
- `String`

1.8 Arithmetic

Example 1.7.

```
mod 18 5
18 `mod` 5
2 / 2 -- floating-point division
div 2 2 -- integer division
2 `div` 2
```

1.9 Boolean Logic

Definition 1.8 (boolean logic)**.** `&&`, `||`, `not`

Definition 1.9 (equality and order)**.**

- `==` *or* `/=`
- `>`, `<`, `>=`, `<=`

Definition 1.10 (if-expression)**.** `if b then t else f`

Remark 1.11. *For if-expression, the* `else` *part is required.*

1.10 Defining Basic Functions