Introduction to Functional Programming

Zsók Viktória

Department of Programming Languages and Compilers
Faculty of Informatics
Eötvös Loránd University
Budapest, Hungary
zsv@elte.hu

Overview

Motivation

Functional programming:

- allows programs to be written clearly, concisely
- has a high level of abstraction
- supports reusable software components
- encourages the use of formal verification

What is functional programming?

- the closest programming style to mathematical writing, thinking
- the basic element of the computation is the function

f(a) => `f a`
f(a,b) + cd => `f a b + c * d`
f(g(b)) => `f (g b)`
f(a)g(b) => `f a * g b`

Writing functional programs is FUN

- to motivate you to write functional programs
- to get involved in working with FP
- the Clean compiler can be downloaded from:
  `http://wiki.clean.cs.ru.nl/Clean`
  have FUN

`examples.icl, examples.prj`

```
module examples
import StdEnv
Start = 42 // 42
```

Getting started

Simple examples of Clean functions:

```
inc x = x + 1
double x = x + x
quadruple x = double (double x)
factorial n = prod [1 .. n]
```

Using them:

```
Start = 3+10*2 // 23
Start = sqrt 3.0 // 1.73...
Start = quadruple 2 // 8
Start = factorial 5 // 120
```

Definitions by cases

The cases are guarded by Boolean expressions:

```
abs1 x
| x < 0 = ¬x
| otherwise = x
Start = abs1 -4    // two cases, the result is 4

// otherwise can be omitted
abs2 x
| x < 0 = ¬x
= x
Start = abs2 4    // 4

// more then two guards or cases
signof x
| x > 0 = 1
| x == 0 = 0
| x < 0 = -1
Start =  signof -8  // -1
```

Definitions by recursion

Examples of recursive functions:

```
fac n
| n == 0 = 1
| n > 0 = n * fac (n - 1)
Start = fac 5 // 120
```

```
power x n
| n == 0 = 1
| n > 0 = x * power x (n - 1)
Start = power 2 5 // 32
```

Compositions, function parameters

```
// function composition
twiceof :: (a → a) a → a
twiceof f x = f (f x)
Start = twiceof inc 0 // 2

// Evaluation:
twiceof inc 0
→ inc (inc 0)
→ inc (0+1)
→ inc 1
→ 1+1
→ 2

Twice :: (t→t) → (t→t)
Twice f = f o f
Start = Twice inc 2 // 4

f = g o h o i o j o k   is nicer than   f x = g(h(i(j(k x))))
```