0. Motivation

Mittwoch, 13. April 2016 10:00

Functional Programming (Slide 14)

Example to illustrate the difference between functional + imperative programming: Slide 3

\[ l = \text{List head} \]

\[ \text{Element value 15 next} \rightarrow \text{Element value 70 next} \rightarrow \text{Element value 36 next null} \]

Corresponds to the list \([15, 70, 36]\)

Characteristics of imperative programs:

- Program consists of instructions executed after another.
- Execution of instruction changes values of variables and of memory. This can lead to side effects (e.g., erasing the list \(l\) in our example).
- Programmer has to consider representation of data in memory → easy to create bugs!

Implementation of "length" in the functional language Haskell:

Define "list length" → needed for declarative programming:
(A) If the list \( l \) is empty, then \( \text{len}(l) = 0 \).
(B) If \( l \) is not empty and \( xs \) is the list \( l \) without its first element, then \( \text{len}(l) = 1 + \text{len}(xs) \).

Syntax for lists in Haskell:

\[ x : xs \] stands for the list resulting from the list \( xs \) by inserting the element \( x \) in front.

\[ 15 : [70,36] = [15,70,36] = 15 : 70 : 36 : [] \]

\[ \text{len} :: [\alpha] \rightarrow \text{Int} \]

\[ \text{len} \ [3] = 0 \]

\[ \text{len} \ (x : xs) = 1 + \text{len} \ xs \]

Execution of a functional program: provide an expression, computer tries to simplify it.

\[ \text{len} \ [15,70,36] = 1 + \text{len} \ [70,36] \]

\[ = 1 + 1 + \text{len} \ [36] \]

\[ = 1 + 1 + 1 + \text{len} \ [ ] \]

\[ = 1 + 1 + 1 + 0 \]

\[ = 3 \]

Characteristics of functional programming:

- Recursion, no loops
- Parametric polymorphism:
  - The same function can be used for arguments of different types.
• No side effects (referential transparency)
• Automatic memory management
• Functions are "first-class" data objects:
  One can write functions that have functions as arguments or results.
• Lazy evaluation (in Haskell, not in all functional languages)

Advantages of functional programming:
• Programs are much shorter and clearer
• Easier to write bug-free programs
• Faster prog. development

Disadvantage: often less efficient, since programmer can’t control "machine-details."

Contents of the lecture:
1. Introduction to Haskell
2. Semantics of Functional Prog. Languages
3. The Lambda Calculus (Implementation of Func. Prog.)
4. Type Checking for Func. Prog.

Organization
- next week: lecture instead of exercise course
- 2 lectures on Tuesday (April 19)
- first exercise sheet issued on Wed (April 20)
  has to be returned on Tue (April 26) before ex. course
- exercises have to be solved in groups of four
  (German or English)
- website: http://verify.ruhr-achen.de/ftp16
  contains transparencies, ex. sheets, notes from the lecture,
  course notes (German), video, literature
- 3 hour lecture (no more lectures at the end of the semester)
- 50% of the points in exercises are needed to participate in exam
- Exam: August 17 / Sept 19 (written exam)
- For exercises: please register via our website until April 20

For exam: please register via
  Campus Office (until May 20)

"Vorzeitige Masterprüfung": register at ZPA
  June 6 - 16

Should we change the times of Tuesday lecture + ex. course?
Currently:  Ex. Course  13:15 - 14:45
            Lecture       15:15 - 16:45

Other possibilities:  10:15 - 11:45
                      12:15 - 13:45
                      14:15 - 15:45
                      16:15 - 17:45

Please enter your preferences until Friday (April 15),
in Doodle Poll.