Advanced Functional Programming

Software Engineering Chair and Programming Systems Lab

A Scrutiny of the Abstract

Read *A Scrutiny of the Abstract* by Kenneth K. Landes and provide a better abstract of the paper. You can work in teams of two.

Presentations

We expect all participants to give a presentation about an advanced topic in functional programming. The presentation is accompanied by a short written presentation.

- We are planning to present talks in a conference-like setting: all talks will be given on two consecutive days. For each talk we will allocate about 35 minutes, including 10 minutes for discussion.
- The tentative date for presentations is March 22nd and 23rd.
- The written presentation should be an extended abstract, about 5 pages long. We haven't decided on a due date but it will be after the date for giving talks.
- For your grade we will weight your writing assignments (50%), your written presentation (20%), and your talk (30%). No part must be failed but for the writing assignments we will take only the 8 best out of 10 assignments – or a similar ratio – into account.
- We will provide individual supervision for your talk and extended abstract.

Material for Presentations

The papers below serve as a starting point for a presentation about an advanced topic in functional programming.

1. Applications: Koen Claessen and John Hughes, *QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs* [Christian Lindig]
2. Applications: Philip Wadler, *A prettier printer*; Olaf Chitl, *Pretty Printing With Lazy Dequeues* [Christian Lindig]
3. Combinators: Andrew Kennedy, *Pickling Combinators* [Christian Lindig]
4. Laziness: Jerzy Karczmarczuk, *Functional Differentiation of Computer Programs* [Christian Lindig]

5. Continuations: Jacob Matthews et al., *Automatically Restructuring Programs for the Web* [Christian Lindig]

6. Data Structures: Martin Erwig, *A Functional Graph Library* [Jan Schwinghammer]

7. Domain-Specific Languages: P. Bjesse et al., *Lava: Hardware Description in Haskell*, [Jan Schwinghammer]

8. Concurrency: Fabrice Le Fessant, Cédric Fournet, Luc Maranget, and Alan Schmitt, *JoCaml: a Language for Concurrent Distributed and Mobile Programming* [Jan Schwinghammer]

9. Concurrency: John Reppy, *CML: A Higher-Order Concurrent Language* [Jan Schwinghammer]

10. Purity and Laziness: Simon Peyton Jones, Alastair Reid, Tony Hoare, Simon Marlow, and Fergus Henderson, *A Semantics for Imprecise Exceptions* [Andreas Rossberg]

11. Type Classes: Simon Peyton Jones et al., *Bulk types with class* [Andreas Rossberg]

12. Dependent Types: Lennart Augustsson, *Cayenne - A Language with Dependent Types* [Andreas Rossberg]

13. Dynamic Typing: Martin Abadi, Luca Cardelli, Benjamin Pierce, and Gordon Plotkin, *Dynamic Typing in a Statically Typed Language* [Andreas Rossberg]

Homework

1. Find your three favorite topics for a talk and a written presentation. You will be given just one of these but we would like to use these to resolve collisions with other students.

2. Read *Origami Programming* by Jeremy Gibbons; it appeared as a chapter in *The Fun of Programming*, Jeremy Gibbons and Oege De Moor (Editors), pages 41–60, Palgrave Macmillan, 2003.

3. Summarize the paper *in your own words* on one page. Put your name and student ID on your summary and drop off a printout at office **326/45** until Monday, February 6th at noon (12 am). If the door is closed, slide your printout under the door. No Emails.