Advances in Programming Languages

APL4: Coursework assigment topics

Ian Stark

School of Informatics
The University of Edinburgh

Friday 1 October 2010
Semester 1 Week 2

Outline

Course grades

Final grades are based on an exam (80%) and a written coursework assignment (20%).

The exam is in May, and follows the standard "choose two questions out of three" format. Past papers are available through the course web page.

The examinable material for this course is the content of the lectures and their accompanying homework exercises. The following will not be assessed in the examination:

- Any guest lectures;
- The written coursework;
- Further references in the course blog.

The written coursework requires investigation of a topic in programming languages and writing a 10-page report with example code.

Aims of exercises

The aims of the homework exercises set in lectures include:

- To prepare for forthcoming lectures
- To review lecture material
- To give some context for understanding the lectures
- To provide other sources and views on the lecture topic
- To help with learning by exploring the subject

Crucially, these effects arise from doing the exercises. They are not assessed, nor would this necessarily be useful: they are intended to be self-validating (i.e. you can tell when you have succeeded).

Although your coursework reports will be assessed, most of this also applies there: the purpose is for you to find out and learn new things.

Outline

Information flow in Jif

The *Jif* compiler extends the Java language with annotations for static analysis of security properties relating to the flow of information.

These annotations describe restrictions on how information is to be used: which *principals* control which information, and what they trust other principals to do with it. This gives increased assurance that trusted and untrusted information is used only according to explicit security policies.

Programming the Web with Links

The *Links* language unifies the traditional three tiers of web programming: client activity within the web page being viewed; server software directing web site responses; and a back-end database providing content and persistent storage. A single program written in Links is compiled into a different language for each tier and then automatically distributed across them as appropriate.

Links itself is functional, with a range of novel features to present a coherent programming interface for database manipulation, embedded XML, and user interaction flow.

Parallel programming in Haskell with par and seq

The original Haskell '98 language has no specific facilities for concurrent or parallel programming. However, there are several compiler extensions and libraries which make both possible. In particular, operations par and seq allow a programmer to enable parallel or sequential computation of results, and from these build more complex *strategies* for parallel evaluation across multiple cores or even distributed processors.

Asynchronous Workflows in F#

Microsoft's F# language provides several facilities for the building and high-level manipulation of computations and metacomputations. One of these, *workflows*, allows libraries to define domain-specific sublanguages for particular kinds of computation.

Using this, the Async module gives a way to write code that can execute asynchronously when necessary, without needing to explicitly describe any threads or communication. Actions that might potentially block or be long-running will automatically happen in the background, with their results retrieved as they arrive.

Functional Reactive Programming in Flapjax

Functional Reactive Programming (FRP) is a technique for writing programs that interact with their environment over time. For example, *Functional Reactive Animation* draws moving images that interact with a user's mouse. Being functional rather than imperative, a program in FRP describes *what* is desired of a system's behaviour in response to events, rather than *how* it is computed. Simple behaviours and events are then combined into larger, more complex ones.

Flapjax applies this to web programming, scripting the behaviour of web pages that interact with the user on one side, and web services on the other.

Outline

Haskell

The GHC compiler for Haskell does not support dynamically linked libraries on Windows [1].

1. GHC Users Guide. `http://www.haskell.org/ghc/docs/latest/html/users_guide/index.html`

Facebook

Facebook signed up 200 million users in its first five years [NYT].

[NYT] “Is Facebook Growing Up Too Fast?” New York Times

Links

The *Links* programming language compiles client code into Javascript (Cooper et al. 2007).

Ezra Cooper, Sam Lindley, Philip Wadler, and Jeremy Yallop.
Links: Web Programming Without Tiers.
In *Formal Methods for Components and Objects: Proceedings of the 5th International Symposium FMCO 2006.* Lecture Notes in Computer Science 4709, pages 266–296. Springer-Verlag, 2007. DOI: 10.1007/978-3-540-74792-5_12

Moore

Moore's law says that the number of components in an integrated circuit doubles every year[1].

1. Wikipedia

Asynchronous workflows can be used to run code in parallel [TP].

[TP] http://tomasp.net/articles/fsharp-webcast-async.aspx

Outline

Dates and submission

Week 1 Friday 24 September: Topics announced

Week 3 Friday 8 October: Preliminary report due

Week 4 Friday 15 October: Preliminary report returned

Week 5: Coursework week

Week 8 Friday 12 November: Final report due

Week 10 Friday 26 November: Final report returned

Each report should be submitted electronically as a PDF document. The recommended method for creating these is pdflatex with the article document class.

In addition, OpenOffice is freely available for Windows and Linux, installed on Informatics machines, and can write PDF. Mac OS X natively creates PDF. Microsoft provide PDF output as a plugin for Word 2007.

Declaring topic choice

Preliminary report

This document should contain:

- Your student number;
- The topic you have chosen;
- Three suitable references, which you have read; and
- A screenshot by you of the selected system in action.

One reference must be to a published paper; the other two may be too, but could also be white papers, web tutorials, manuals, or similar. In all cases provide enough information for someone else to obtain the document.

To create the screenshot, you will need to have your chosen system downloaded, installed, and running on a suitable machine.

Suggested outline

Heading	Title, date
Abstract	This report describes ...
Introduction	Content summary, overview of report structure
Context	The problem domain
⟨Main topic⟩	What it is, how it works, advantages and limitations
Example	Annotated code, explanation, screenshot
	Salt: the example must in some way concern a sporting activity (e.g. team roster, match results, time trials, . . .)
Resources	For each notable resources used (article, tutorial, manual), give a summary in your own words of what it contains
Related work	Other approaches to the problem
Conclusion	What ⟨topic⟩ does, good and bad points
Bibliography	Full references for all resources used

Total around 10 A4 pages. See the course web pages for further details.

Outline

Plagiarism

University of Edinburgh Undergraduate Assessment Regulations

Regulation 14

14.1 All work submitted for assessment by students is accepted on the understanding that it is the student's own effort without falsification of any kind. Students are expected to offer their own analysis and presentation of information gleaned from research, even when group exercises are carried out. In so far as students rely on sources, they should indicate what these are according to the appropriate convention in their discipline.

Plagiarism

University of Edinburgh Undergraduate Assessment Regulations

Regulation 14

14.2 Plagiarism is the act of copying or including in one's own work, without adequate acknowledgement, intentionally or unintentionally, the work of another. It is academically fraudulent and an offence against University discipline. ...

Plagiarism

University of Edinburgh Undergraduate Assessment Regulations

Regulation 14

14.2 Plagiarism is the act of copying or including in one's own work, without adequate acknowledgement, intentionally or unintentionally, the work of another. It is academically fraudulent and an offence against University discipline. ...

http://www.acaffairs.ed.ac.uk/Regulations/Assessment/09-10/UG.htm#Reg14

See also:

- University guidance
  http://www.docs.sasg.ed.ac.uk/AcademicServices/Discipline/StudentGuidanceUGPGT.pdf
- Informatics statement
  http://www.inf.ed.ac.uk/teaching/plagiarism.html

Suitable working practices

Working practices

- Start with a blank document; all the words must be yours.
- Do not cut and paste from other documents.
  - Except for direct quotations, which must have source declared.
- Do not let others read your text; nor read theirs.

Aims of this assignment

- To learn about the chosen topic
- To improve researching and learning skills
- To demonstrate said knowledge and skills

The tangible outcome is a document, composed and written by you, demonstrating what you have learnt.

Outline

Summary

Topic choices

- Information flow in Jif
- Programming the Web with Links
- Parallel programming in Haskell with par and seq
- Asynchronous workflows in F#
- Functional reactive programming in Flapjax

Coursework and learning

- Homework exercises are there to be done
- Note the essay plan
- All your own work
- The aim of all the coursework is to support learning