Advances in Programming Languages

APL2: Coursework assigment topics

Ian Stark

School of Informatics
The University of Edinburgh

Thursday 14 January 2010
Semester 2 Week 1

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

Links: Web Programming without Tiers

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.

Multiple inheritance

Multiple inheritance in Scala with traits and mixins

The Scala language provides *traits* and *mixins* as modularisation constructs.

Mixin composition solves the infamous *multiple inheritance* ambiguity problem: does a class A that inherits from B and from C implement A.m as B.m or C.m? Java forbids multiple inheritance but provides interfaces. However, interfaces cannot contain implementations, leading to code duplication. Scala's trait and mixin constructs remedy this.

Parallel programming in Haskell

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.

Haskell STM library

Software Transactional Memory in Haskell

The *STM* library for the Glasgow Haskell Compiler (GHC) provides high-level language support for coordinating concurrent computation, where multiple threads act simultaneously on shared datastructures.

Remarkably, STM does this without using locks. Instead, it uses efficient and optimistic *software transactions*, giving freedom from deadlock and promoting non-interfering concurrency. These transactions are modular and composable: small transactions can be glued together to make larger ones. Moreover, implementing this within the Haskell type system gives static guarantees that transactions are used correctly.

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.

Outline

Dates and submission

Week 1 Thursday 14 January: Topic announcement

Week 5 Friday 12 February: Preliminary report

Week 6: Review lecture

Week 8 Friday 5 March: Final report

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.

Review lecture

Coursework review

This will be a lecture to discuss progress and problems with the assigned coursework. It follows submission of the preliminary report, so you will all have chosen topics, and have made some progress with your investigation.

We shall also cover issues with preparation of the report, its content and arrangement, as well as look at some example reports from previous years.

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 books or reading (e.g. library catalogue, author database, . . .)
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.acaffairs.ed.ac.uk/Administration/GuidanceInformation/AcademicBestPractice/Plagiarism/Index.htm
- Informatics policy
  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

- Programming the Web with Links
- Multiple inheritance in Scala with traits and mixins
- Parallel programming in Haskell with par and seq
- Software Transactional Memory in Haskell
- Asynchronous workflows in F#

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