CS410
Advanced Functional Programming

Conor McBride
Mathematically Structured Programming Group
Department of Computer and Information Sciences
University of Strathclyde

September 16, 2013

Chapter 1

Introduction

1.1 Language and Tools

For the most part, we'll be using the experimental language, Agda Norell [2008], which is a bit like Haskell (and implemented in Haskell), but has a more expressive type system and a rather fabulous environment for typed programming. Much of what we learn here can be ported back to Haskell with a bit of bodging and fudging (and perhaps some stylish twists), but it's the programming environment that makes it worth exploring the ideas in this class via Agda.

The bad news, for some of you at any rate, is that the Agda programming environment is tightly coupled to the Emacs editor. If you don't like Emacs, tough luck. You may have a job getting all this stuff to work on whatever machines you use outside the department, but the toolchain all works fine on departmental machines.

Teaching materials, exercise files, lecture scripts, and so on, will all pile up in the repository `https://github.com/pigworker/CS410-13`, so you'll need to get with the git programme. We'll fix it so you each have your own place to put your official branch of the repo where I can get at it. All work and feedback will be mediated via your git repository.

1.2 Lectures, Lab, Tutorials

Monday: Lecture, 1–2pm LT714; Lab, 3–5pm LT1301

Thursday: Tutorial, 2–3pm, GH811 (this will usually be conducted by one of my graduate students)

Friday: Lecture, 11am–12pm, GH811

Scheduled interruptions of service: Monday 30 September, University closed; Week 4 (14–18 October), I'm at a working group meeting; Friday 8 November, I'm examining a PhD. We can't do anything about the University closing, but I'll try to find fun people for you to hang out with on the other dates.

1.3 Twitter @CS410afp

This class has a twitter feed. Largely, this is so that I can post pictures of the whiteboard. I don't use it for essential communications about class business, so you need neither join twitter nor follow this user. You can access all the relevant stuff just by surfing into `http://twitter.com/CS410afp`. This user, unlike my personal account, will follow back all class members who follow it, unless you ask it not to.

1.4 Hoop Jumping

CS410 Advanced Functional Programming is a level 4 class worth 20 credits. It is assessed *entirely* by coursework. 75% of the marks will be for semester 1 coursework, set in several small chunks in the autumn term, and one middle-sized chunk to be submitted by end of semester in January. The other 25% of the marks will be for the semester 2 coursework, which will consist of one large open-ended task, to be undertaken starting no sooner than the final year project deadline and finishing as near to the mark upload deadline as I think I can manage. I reserve the right to set one-lab exercises under exam conditions and to conduct oral examinations, by way of ensuring that credit is obtained by individual work.

1.5 Getting Agda Going on Departmental Machines

Step 1. Use Linux. Get yourself a shell. (It's going to be that sort of a deal, all the way along. Welcome back to the 1970s.)

Step 2. Ensure that your `PATH` environment variable includes the directory where Haskell's `cabal` build manager puts executables. Under normal circumstances, this is readily achieved by ensuring that your `.profile` file contains the line:

```
export PATH=$HOME/.cabal/bin:$PATH
```

After you've edited `.profile`, grab a fresh shell window before continuing.

Step 3. Ensure that you are in sync with the Haskell package database by issuing the command:

```
cabal update
```

Step 4. Install Agda by issuing the command:

```
cabal install agda
```

Yes, that's a lower case 'a' in 'agda'.

Step 5. Wait.

Step 6. Wait some more.

Step 7. Assuming all of that worked just fine, set up the Emacs interactive environment with the command:

```
agda-mode setup; agda-mode compile
```

Step 8. Get this repository. Navigate to where in your file system you want to keep it and do

```
git clone https://github.com/pigworker/CS410-13.git
```

Step 9. Navigate into the repo.

```
cd CS410-13
```

Step 10. Start an emacs session involving an Agda file, e.g., by the command:

```
emacs Hello.agda &
```

1.6 Making These Notes

The sources for these notes are included in the repo along with everything else. They're built using the excellent `lhs2TeX` tool, developed by Andres Löh and Ralf Hinze. This, also, can be summoned via the Haskell package manager.

```
cabal install lhs2tex
```

With that done, the default action of `make` is to build these notes as `CS410.pdf`.

Bibliography

Ulf Norell. Dependently typed programming in agda. In Pieter W. M. Koopman, Rinus Plasmeijer, and S. Doaitse Swierstra, editors, *Advanced Functional Programming*, volume 5832 of *LNCS*, pages 230–266. Springer, 2008.