Haskell Environment
for
Functioneel Programmeren (192112051)

Original Author: Christiaan Baaij
Revised by: Sebastiaan la Fleur

Kwartiel 1, 2014

1 Introduction

Haskell is a well-known non-strict functional programming language, and is available on all major platforms. The Haskell environment for the functional programming course consists of three parts, which all need to be installed. The first part is the *Haskell Platform*, which contains the Glasgow Haskell Compiler (and Interpreter), and a set of standard libraries and tools. Among these tools is *cabal*, which is Haskell's own package manager. The second part is the `twentefp-number` package. The `twentefp-number` defines the *Number* type, which hides the intricacies of Haskell's *Type Classes* from new users. The third part is the `eventloop-graphics` package, which is a cross-platform graphics library based on websockets and uses the browser. This part will be installed later on in the course. Later on, when a you have a better understanding of Haskell (and *Type Classes*), you can omit using this *Number* type, and use Haskell's own default numeric types. The `eventloop-graphics` package also defines (in module *EventLoop.Graphical*) the *graphicsout* function which displays a value of type *OutGraphical*. Additionally, the module *eventloop-graphics* defines the *eventloop* which forwards keyboard/mouse input to a user-defined function, and displays graphical output returned by that same user-defined function.

During the exam, lab exercises, and the final assignment(s), you can only use the standard *Prelude* module, and/or, the modules exported by the `twentefp-number` and `eventloop-graphics` packages: (*FPPrac.Prelude, EventLoop, EventLoop.Output, EventLoop.Input, EventLoop.Output.Graphical*). If you want to deviate from only using the sanctioned modules, and think you have a good (didactically sound) reason, ask for permission from the lecturer and/or teaching assistants first.

2 Installation on lab machines (Windows 7)

A working version of the *Haskell Platform* is installed on the lab machines, but not all files necessary for the lab exercises are installed on these machines. This section describes the steps required to get the machines in a state required to perform the lab exercises.

- Download `fp.bat` from Blackboard. (If not available, ask a teaching assistant for it)
- Run this file from the command-line every time you login on a machine:
  - Every time it defines/alters certain environment variables so that the system can find the necessary files on the `U:\` drive.

3 Installation on your own machine

This section describes the installation procedure for the Haskell environment used in the functional programming course. Students deviating from this procedure will not get any support from the teaching assistants or teacher if any problems with the Haskell environment occur. The installation consists of a part equal for all platforms, and a part specific for each platform.

3.1 For all platforms

- Download the latest *Haskell Platform* for your platform from `http://hackage.haskell.org/platform/`. Only download a 64-bit version if its is available for your platform, and recommended over the 32-bit version!
- Follow the installation instructions of the Haskell Platform; Linux users: read the subsection below.

3.1.1 Some notes on installing Haskell on Linux

- Installing a GHC version of the 7.* branch most likely suffices for running the graphical environment. Version 7.6.3 is however recommended. Also make sure you install the `cabal(-install)` binary (0.10.* for GHC 7.0.*, 0.12.* for GHC 7.2.*, 0.14.* for GHC 7.4.*, and 0.16.* for GHC 7.6.*). If you fail to install the graphical environment, follow the instructions at the second bullet point.
- If version 2013.2.0.0 of the Haskell Platform is not included in your package manager, follow the instructions found on the Haskell Platform page to install GHC 7.6.3 manually (binary distribution), followed by a manual install (from source) of the Haskell Platform tools and libraries. To install GHC 7.6.3 and Haskell Platform 2013.2.0.0 on LMDE you need the following libraries: `libgmp-dev`, `libz-dev`, `libgl1-mesa-dev`, `libglu1-mesa-dev`, and `freeglut3-dev`.

3.2 Windows XP/Vista/7 (tested on: XP SP3, Win7 SP1 64-bit, Win8.1 64-bit)

- Make sure that the `cabal` executable can be found in your `%PATH%` (should be done by the *Haskell Platform* Installation).
- Run (from the command line): `cabal update`.
- Open, in your favorite text editor, `%APPDATA%\cabal\config` (run: `echo %APPDATA%`, if you don't know which directory `%APPDATA%` is).
- Uncomment the `documentation` line, and set it to `True`. Save the file.
- Run (from the command line): `cabal install twentefp-number`.
- Open (or create if it does not exist) in your favorite text editor `%APPDATA%\ghc\ghci.conf`.
- Add the following two lines to this file, and save it:

```
:set -XNoMonomorphismRestriction
:set -XNoImplicitPrelude
```

3.3 OS X 10.8 (tested on: OS X 10.8.4) & Linux (tested on: Ubuntu 14.02 LTS in VirtualBox)

- Make sure that the `cabal` binary can be found in your `$PATH` (should be done by the *Haskell Platform* Installation).
- Run (from the command line): `cabal update`.
- Open, in your favourite text editor, `$HOME/.cabal/config`.
- Uncomment the `documentation` line, and set it to `True`. Save the file.
- Run (from the command line): `cabal install twentefp-number`.
- Open (or create if it does not exist) in your favourite text editor `$HOME/.ghci`.
- Add the following four lines to this file, and save it:

```
:set -XNoMonomorphismRestriction
:set -XNoImplicitPrelude
```

3.4 Notes on the Installation

As instructed, you have to make sure that generating API documentation is enabled when using `cabal` to install Haskell packages. The reasons you had to update the *GHCi* configuration file are the following:

`:set -XNoMonomorphismRestriction` – lifts certain restriction on the ability to derive polymorphic types for certain functions. If this extension is not set, certain functions might be unexpectedly monomorphic.

`:set -XNoImplicitPrelude` – GHC implicitly imports the *Prelude* module, but the *FPPrac* module defines certain functions that have the same name (such as *take* and *drop*), but a different type (`Number -> [a] -> [a]` instead of `Int -> [a] -> [a]`). So if you don't disable implicitly loading *Prelude*, the compiler will complain that it can find two definitions for a certain function name. Note that *FPPrac* exports all the functions from *Prelude*, except those that share names; see the API documentation for which functions.

4 Using the Haskell Interpreter

We will mostly use the interpreter offered by the Glasgow Haskell Compiler. You can start the interpreter by running `ghci` from the command line. You can immediately evaluate expressions in the interpreter. You can load files using the `:l` command, e.g.: `:l /FP/prac1.hs`. If you changed a file that is loaded in the interpreter, you can reload it by typing `:r`. Instead of opening the interpreter and then loading the file, you can also immediately call the interpreter with the file you want to open, e.g.: `ghci /FP/prac1.hs`. As suggested by the examples, make sure that all your files end with the extension ".hs".

You always start a new module with the line "`module ModuleName where`". A module may use definitions from other modules by including a line "`import ModuleName`". Always make sure that you at least import the `fpprac` prelude in any file that you create by adding the line "`import FPPrac`", otherwise you won't have access to all the standard function and type definitions. Imports must be placed before any other definitions.

As Haskell is a well-known functional programming language many editors should have support (such as syntax-highlighting) for it. Editors that definitely have support for Haskell are: *Vim, Emacs, TextMate,* and *Sublime Text.*

Although the interpreter has a debugger, the teaching assistants offer no support in its use. You can use the `trace` function, as described in the 'dictaat', for debugging purposes.

5 Advanced Users

If you understand what *Type Classes* are, and you want to use the default Haskell numeric datatypes, instead of the `Number` type provided by the `fpprac` package, remove the line `":set -XNoImplicitPrelude"` from your `ghci` configuration file. You should also no longer import the *FPPrac* module in your files.

6 More Information

There is a 'dictaat' for Haskell on Blackboard that can be used as reference documentation. You will also find information on the graphical environment in the 'dictaat'.

When installing the Haskell platform, API documentation for all the libraries that came with it, should be included. And, if you followed, the installation instructions, API documentation for the `fpprac` package should have been automatically generated and added to the API index. The location is of this index file is stated in your `cabal` configuration file (*doc-index-file*).

A lot of additional information about Haskell can of course be found on `http://haskell.org`, but be warned that things can become very advanced, and very technical, quite quickly.