

Formal Methods and Functional Programming

Exercise Sheet 1: First Steps in Haskell and Natural Deduction

Submission deadline: February 26th, 2012

Introduction

Important: The exercise groups start *this week*; i.e., February 21th (Tue) and February 22th (Wed). If you have missed the registration during the first lecture, you can send an e-mail to pgrgur@inf.ethz.ch and just visit one of the exercise groups in the first week. You can look up the exercise group assignments on the course web page (www.infsec.ethz.ch/education/ss2012/fmfp).

In this course, you will use the Haskell interpreter GHCi. You can install GHCi on your own computer or use the Haskell platform installed on Windows on the computers in the D-INFK student labs. The GHCi interpreter is included in the Haskell Platform available at hackage.haskell.org/platform/. If you have problems with the Haskell Platform, you can also install the GHC environment www.haskell.org/ghc/ which includes GHCi.

There exists a wealth of resources on Haskell. You find links to almost all of them on www.haskell.org. The most informative ones for you as a becoming Haskell programmer are:

- www.haskell.org/hoogle search the Haskell libraries by name or type
- book.realworldhaskell.org/read this online book contains introductory material on Haskell (in case you do not want to buy Hutton's "Programming in Haskell" or Thompson's "Haskell: The craft of functional programming")
- see the *links on the course homepage* for a printable version of the GHCi Prelude and other material about Haskell

Please send your solutions by e-mail to your tutor as a *Haskell file*. The subject of your e-mail should start with [FMFP]. You can find the e-mail address on the course web page. Hand in your solution no later than the given submission deadline¹.

You can use `--` for single line comments and `{- and -}` to enclose multiline comments. You should use multiline comments for your answers to non-programming exercises. You can find an example of a Haskell file at the course web page. The name of your Haskell file must be

¹The given submission date means that the solution must be sent to your tutor before midnight of Feb 26th.

sheet<nr>_<nethz-username>.hs and the maximal line length is 100 characters. You also must not use TAB characters in your Haskell files. This helps your tutor to sort and print all submissions easily.² In general, provide detailed comments in your solutions in order to help your tutor to understand your solutions.

Assignment 1:

The purpose of this assignment is to get used to GHCi and writing Haskell programs. You do not have to hand in your solution for this assignment; you can find a solution in the file `sheet1_johndo.hs`, which is available from the course web page.

Important prompt commands in GHCi are

<code>:?</code>	help
<code>:load <filename> or :l <filename></code>	load the file filename
<code>:reload or :r</code>	repeat the last load command
<code>:quit or :q</code>	quit

We recommend to use a decent text editor that supports syntax highlighting for editing your Haskell files. See the Haskell links on the course homepage for suggestions for all major operating systems.

- (a) Download the file `gcd.hs` from the course web page and load it into GHCi. Use GHCi to calculate the greatest common divisor of 139629 and 83496. What happens if one of the arguments of the function `gcd` is negative? What happens if one of the arguments is 0?

Generalize the `gcd` function to a function `gcdInt :: Int -> Int -> Int` such that `gcdInt x y = gcd x' y'`, where `x'` is the absolute value of `x` and `y'` is the absolute value of `y`. Does your function terminate for all inputs?

- (b) GHCi has already many predefined functions. These functions are defined in the `Prelude`, which is automatically loaded when you start GHCi. You can find a link to the standard Prelude files on the course homepage.

Look at the `Prelude` and check whether you can simplify your solution in (a) by using some of the predefined functions.

- (c) You can query GHCi for the type of a function with the prompt command `:t`. For instance, GHCi will output `Int -> Int -> Int` if you type in `:t gcd`.

What happens if you apply `gcd` to the floats 3.6 and 7.2? Change the type of `gcd` to `Float -> Float -> Float`. What is now the output of `gcd 3.6 7.2`? What is the output of `gcd 3.6 7.1999999`? What function does this new typed `gcd` calculate?

Assignment 2:

Complex numbers can be represented as pairs of reals: the first coordinate of a pair represents the real part of the complex number and the second coordinate represents the imaginary part. In Haskell, we can use pairs of type `Float` for complex numbers.

²TAB characters are also prone to result in strange GHCi error messages, as Haskell is layout sensitive.

- (a) Write functions `re :: (Float, Float) -> Float` and `im :: (Float, Float) -> Float` that return the real part and the imaginary part of a complex number, respectively.
- (b) Write a function `conj :: (Float, Float) -> (Float, Float)` that conjugates a complex number.
- (c) Write functions for addition and multiplication of complex numbers, and write a function that returns the absolute value of a complex number.

Assignment 3:

The Fibonacci numbers are defined as

$$fib(n) = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ fib(n-1) + fib(n-2) & \text{otherwise.} \end{cases}$$

Louis Reasoner writes the following Haskell program for computing Fibonacci numbers:

```
fibLouis :: Int -> Int
fibLouis 0 = 0
fibLouis 1 = 1
fibLouis n = fibLouis (n-1) + fibLouis (n-2)
```

Eva La Tour writes another Haskell program for the Fibonacci numbers:

```
fibEva :: Int -> Int
fibEva n = fst (aux n)
  where aux 0 = (0,1)
        aux n = next (aux (n-1))
        next (a,b) = (b, a+b)
```

- (a) Complete the evaluation steps in Haskell given below for `fibLouis 4`.

```
fibLouis 4 =
(fibLouis (4-1) + fibLouis (4-2)) =
(fibLouis 3 + fibLouis (4-2)) =
...
```

- (b) Complete the evaluation steps for `fibEva 4`.

```
fibEva 4 =
fst (aux 4) =
...
```

Assignment 4:

Recall that \rightarrow is right-associative, while \wedge and \vee are left-associative. Moreover, \neg binds stronger than \wedge , which binds stronger than \vee , which in turn binds stronger than \rightarrow . Hence, the formula

$A \wedge B \vee C \rightarrow \neg E \rightarrow C \vee A \wedge B$ is parenthesized as $((A \wedge B) \vee C) \rightarrow ((\neg E) \rightarrow (C \vee (A \wedge B)))$.

We recommend to always parenthesize formulas before proving them using natural deduction. This simplifies matching the inference rules and you avoid trivial parsing errors.

(a) Parenthesize the following formulas.

(i) $A \vee B \rightarrow C \rightarrow A \wedge C \vee B \wedge C$

(ii) $(A \rightarrow B \rightarrow C) \rightarrow A \wedge B \rightarrow C$

(b) Prove that the formulas in (a) are tautologies in intuitionistic logic using natural deduction. Give complete proof trees and label each rule application with the rule's name. For your convenience, the rules for natural deduction in intuitionistic logic are copied below.

$$\begin{array}{c}
 \frac{}{\Gamma, A \vdash A} \text{ axiom} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow I \qquad \frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \rightarrow E \\
 \\
 \frac{\Gamma \vdash \perp}{\Gamma \vdash A} \perp E \qquad \frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} \neg I \qquad \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash B} \neg E \\
 \\
 \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge I \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge EL \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge ER \\
 \\
 \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee IL \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee IR \qquad \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee E
 \end{array}$$

(c) We define $A \leftrightarrow B$ as $(A \rightarrow B) \wedge (B \rightarrow A)$. Provide suitable introduction and elimination rules for \leftrightarrow and use them to prove the validity of $(A \leftrightarrow B) \rightarrow (B \leftrightarrow A)$.

Assignment 5 (headache of the week³):

Recall that one way to make the above inference system complete for classical logic is to add an axiom formalizing the "law of excluded middle" (lat. "tertium non datur").

$$\frac{}{\Gamma \vdash A \vee \neg A} \text{ TND}$$

Prove that the formula $((A \rightarrow B) \rightarrow A) \rightarrow A$ is valid in classical logic.

³Our weekly headaches are challenging problems. They are meant as supplements to test your FMFP skills.