ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

D. Basin and P. Müller

Formal Methods and Functional Programming

Exercise Sheet 8: Structural Induction

Submission deadline: May 3rd, 2010

Please submit your solution before **9:15am** on the submission date specified above. Solutions can be submitted via e-mail or by using the boxes to the left of **RZ F1**. Make sure that the first page always contains your name, the exercise sheet number as well as your tutor's name and the weekday (Tuesday or Wednesday) of your exercise group. Don't forget to staple your pages if you submit more than one page.

Assignment 1

A chocolate bar consists of a number of squares $n \geq 1$ arranged in a rectangular pattern. You split the bar into small squares always breaking along the lines between the squares. Prove using strong induction that the minimum number of breaks in order to split the bar into small squares is $n - 1$.

Assignment 2

Use strong induction to prove that every natural number greater than 1 is either a prime or can be written as the product of two or more primes.

Assignment 3

Consider the following language used to represent trees:

```
data Tree = Leaf | Node Tree Tree
```

For instance, `Node (Node Leaf Leaf) Leaf` represents a tree with two non-leaf nodes and three leaves.

On this language we define two functions that respectively count the number of non-leaf nodes and compute the height of the tree:

```
N :: Tree -> Int
N (Leaf) = 0
N (Node t1 t2) = (N t1) + (N t2) + 1
```

```
H :: Tree -> Int
H (Leaf) = 0
H (Node t1 t2) = 1 + max((H t1), (H t2))
```

Prove by induction that $(\texttt{N t}) < 2^{(\texttt{H t})}$.

Assignment 4

Define a substitution function $b[y \mapsto e]$ for Boolean expressions that replaces all occurrences of the variable y in the Boolean expressions b with the arithmetic expression e. Prove that your definition satisfies the equality

$$\mathcal{B}[\![b[y \mapsto e]]\!]\sigma = \mathcal{B}[\![b]\!]\big(\sigma[y \mapsto \mathcal{A}[\![e]\!]\sigma]\big)\,,$$

for all Boolean expressions b, all arithmetic expressions e, all variables y, and all states σ.

Assignment 5

Implement using the programming language Haskell the syntax of the **IMP** language as algebraic (Haskell) data types. Implement also the semantics of boolean and arithmetic expressions (note that you have not to implement a parser for **IMP**, but only the data types representing its syntax, and the semantics has to deal with this data types).

Please mail your solution of this assignment to your tutor. The email addresses of the tutors are:

Alex Summers	`alexander.summers@inf.ethz.ch`
Yannis Kassios	`ioannis.kassios@inf.ethz.ch`
Malte Schwerhoff	`scmalte@student.ethz.ch`

Assignment 6 - Headache of the week

The syntactic category `Bexp'` is defined as the following extension of `Bexp`:

```
Bexp' ::= Aexp | Aexp RelOp Aexp
| '(' 'not' Bexp' ')' | '(' Bexp' 'and' Bexp' ')' | '(' Bexp' 'or' Bexp' ')'
| '(' Bexp' 'implies' Bexp' ')' | '(' Bexp' 'iff' Bexp' ')' ;
```

where `Aexp` and `RelOp` are defined as in the lecture.

- Define a semantic function $\mathcal{B}'$ that extends the semantic function $\mathcal{B}$ introduced in the lecture to the Boolean expressions `Bexp'` (where the value 0 of an arithmetic expression is interpreted as the Boolean false, and true otherwise).
- The (extended) Boolean expressions `b1` and `b2` are *semantically equivalent under* $\mathcal{B}'$ if for all states σ, it holds that $\mathcal{B}'[\![b1]\!]\sigma = \mathcal{B}'[\![b2]\!]\sigma$. Show that for each `b'` of `Bexp'`, there is a Boolean expression `b` of `Bexp` such that `b'` and `b` are semantically equivalent under $\mathcal{B}'$.