

Formal Methods and Functional Programming

Exercise Sheet 9: Big Step Semantics

Submission deadline: May 9th, 2011

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 (and preferably each sheet) 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 - Last year's exam!

Consider the following **IMP** statement s :

```
while n # 0 do
  a := a+n;
  b := b*n;
  n := n-1
end
```

Let σ be a state such that $\sigma(a) = 0$, $\sigma(b) = 1$, and $\sigma(n) = 2$. Prove using the natural semantics that there is a state σ' with $\sigma'(a) = 3$, $\sigma'(b) = 2$, and $\sigma'(n) = 0$ such that $\langle s, \sigma \rangle \rightarrow \sigma'$.

Hint: provide the complete derivation tree. You have to explicitly write the names of the rules you apply at each derivation step.

Assignment 2 - Last year's exam!

Consider the extension of the **IMP** programming language with the following statement:

$$\text{for}(s_1, b, s_2) \ s \ \text{end}$$

Give derivation rule(s) in natural semantics for this statement. Your rule(s) must not make use of the **while** statement. The semantics of this statement is identical to the **for-loop** statement $\text{for}(s_1; b; s_2) \ \{s\}$ in Java and C#.

Assignment 3

In the lecture, you have seen the proof of the direction from left to right of the following claim:

$$\langle \text{while } b \text{ do } s \text{ end}, \sigma \rangle \rightarrow \sigma' \Leftrightarrow \langle \text{if } b \text{ then } s; \text{while } b \text{ do } s \text{ end else skip end}, \sigma \rangle \rightarrow \sigma'$$

Prove the direction from right to left of the claim.

Assignment 4

In this assignment you will write a simple interpreter for **IMP** programs. You will use the programming language Haskell. A skeleton of the **IMP** interpreter as a literate Haskell file is available at the course web page. The skeleton file contains the data types for arithmetic expressions, Boolean expressions, and statements for representing **IMP** programs in Haskell. Moreover, the skeleton file contains some auxiliary functions. You have to use this skeleton file instead of what you have developed in assignment 5 of exercise sheet 8. The skeleton contains already a parser and several auxiliary functions (e.g., evaluation of arithmetic and Boolean expressions).

Download the skeleton file and complete the definition of the function

```
transNS :: Config -> Config
```

The place where you should insert your code in the skeleton file is marked by the word `TODO`. The function `transNS` should encode the rules of the transition relation from the lecture for the natural semantics. Feel free to extend **IMP**, e.g., with local variables.

Please mail your solution of this assignment together with some test cases to your tutor. The email addresses of the tutors are:

Alex Summers	alexander.summers@inf.ethz.ch
Yannis Kassios	ioannis.kassios@inf.ethz.ch
Pietro Ferrara	pietro.ferrara@inf.ethz.ch
Stefan Heule	stheule@student.ethz.ch

Assignment 5 - Headache of the week

Extend the natural semantics of **IMP** to support exceptions.

You have to introduce two new statements (where $ex \in \text{ExcNames}$ is the name of an exception):

- `throw ex` throws an exception `ex`
- `try s catch(ex) s1` executes `s`, and if it throws an exception `ex` it executes `s1`

Hint: you have to modify the structure of the state on which the natural semantics is defined.