Software-Modellierung und Verifikation
Informatik 2
Prof. J.-P. Katoen
RWTH Aachen

Priv.-Doz. T. Noll noll@cs.rwth-aachen.de
C. Jansen christina.jansen@cs.rwth-aachen.de

10. Exercise sheet *Semantics and Verification of Software SoSe2010*

Due to Monday, 5th July 2010, *before* the exercise course begins.

Exercise 10.1: **(1+3 points)**

Consider the following program fragment c calculating the product of a and b storing the result in the variable *product*.

```
begin
 var b;
 procedure mult is product := a * b;
 b := 1;
 begin
  var b;
  b := 2;
  call mult;
 end
end
```

(a) Provide an initial variable environment ρ_0 and program state σ_0, for which it holds that $\forall\, \pi_0$ (supposing static scope semantics are used):

$$(\rho_0, \pi_0) \vdash \langle c, \sigma_0 \rangle \to \sigma_0[0/2, 1/2, 2/1]$$

(b) Proof your claim from part a) using operational semantics given in lecture 15.

Exercise 10.2: **(2+2 points)**

Consider the following modification to our *WHILE* language, where procedures have (exactly) one parameter:

$$\begin{array}{rcl} p & ::= & \textbf{proc}\ P(x)\ \textbf{is}\ c;\ p \mid \varepsilon \in \mathbf{PDec} \\ c & ::= & \ldots \mid \textbf{call}\ p(a) \in \mathbf{Cmd} \end{array}$$

Lift the operational semantics to meet the extended language, i.e. define new *call* and *block* rules

(a) for a call by value parameter.

(b) for a call by reference parameter. (Without restriction you can assume that a in **call** $p(a)$ is indeed a variable here.)

Exercise 10.3: **(2+2 points)**

Taking the extended WHILE language introduced in lecture 15 as basis, give denotational semantics for this language restricted to non-recursive (no direct or indirect recursion!) procedures.

Now assume we allow recursive procedures in our WHILE language. Which problems do occur in your semantics now? How could they be solved?