Lehrstuhl für Informatik 2
Modellierung und Verifikation von Software

Semantics and Verification of Software SS2013
Exercise 8 (Hand in on 01.07.2013 before exercise class)

apl. Prof. Dr. Thomas Noll

Kevin van der Pol, Hao Wu

Exercise 1 (Procedure parameters): (2+3 Points)

We extend the procedure declaration with parameters. The context-free grammar is:

$$\begin{aligned} p &::= \textbf{proc } P(v) \textbf{ is } c; p \mid \varepsilon \\ v &::= \textbf{var } x; v \mid \varepsilon \\ c &::= \textbf{skip} \mid x := a \mid c_1; c_2 \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \mid \textbf{while } b \textbf{ do } c \\ &\quad\; \mid \textbf{call } P(s) \mid \textbf{begin } v \; p \; c \textbf{ end} \\ s &::= x; s \mid \varepsilon \end{aligned}$$

a) Extend the definition of the operational semantics of the language with procedures, to also include procedure parameters, for a *call by reference* semantics. Only give the rules that have changed.

b) Write a Swap procedure with two parameters. Show that Swap swaps the values of its parameters, i.e. if $x = X$ and $y = Y$ hold initially, after the execution of $\mathsf{Swap}(x, y)$ it must hold that $x = Y$ and $y = X$.

Exercise 2 (Functions and recursion): (2+3 Points)

We extend the language with functions, which are parameterized procedures that return a value. We restrict ourselves to only one parameter. The context-free grammar is:

$$\begin{aligned} f &::= \textbf{fun } F(\textbf{var } x) \textbf{ is } c; f \mid \varepsilon \\ v &::= \textbf{var } x; v \mid \varepsilon \\ c &::= \textbf{skip} \mid x := a \mid c_1; c_2 \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\ &\quad\; \mid \textbf{while } b \textbf{ do } c \mid \textbf{begin } v \; f \; c \textbf{ end} \mid \textbf{return } a \\ a &::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid F(a) \end{aligned}$$

a) Extend the definition of the operational semantics of the language with procedures, to also include functions, with a *call by value* semantics. Only give the rules that have changed. Note that the **return** command gives a value and not a state and that the **return** command immediately jumps out of the current function. The program flow commands have to be changed accordingly.

b) Write a recursive Factorial function with one parameter, that returns the factorial of the parameter. Show that the Factorial function indeed returns the factorial of the parameter, i.e. if $n = N$ holds, the $\mathsf{Factorial}(n)$ function returns $N!$.