Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Sascha Böhme, Alexander Krauss

WS 2010/11
31. 01. 2011

Semantics of Programming Languages

Exercise Sheet 13

For a change, this exercise should be solved on paper only, not using Isabelle.

Exercise 13.1 Procedure Definedness Check

We consider a language with statically scoped procedures but (for simplicity) without local variables. For this language we can define a small-step semantics that does not require a special procedure environment. Instead, the context of procedure declarations is managed by gradually transforming the program itself. The rules for the basic commands remain unchanged:

$(x ::= a,\ s) \to (SKIP,\ s(x := aval\ a\ s))$

$(SKIP;\ c_2,\ s) \to (c_2,\ s)$

$(c_1,\ s) \to (c_1{}',\ s') \Longrightarrow (c_1;\ c_2,\ s) \to (c_1{}';\ c_2,\ s')$

$bval\ b\ s \Longrightarrow (IF\ b\ THEN\ c_1\ ELSE\ c_2,\ s) \to (c_1,\ s)$

$\neg\ bval\ b\ s \Longrightarrow (IF\ b\ THEN\ c_1\ ELSE\ c_2,\ s) \to (c_2,\ s)$

$(WHILE\ b\ DO\ c,\ s) \to (IF\ b\ THEN\ c;\ WHILE\ b\ DO\ c\ ELSE\ SKIP,\ s)$

Now, procedure declarations distribute over semicolons, and disappear when they surround a *SKIP*. Moreover, we may make an arbitrary step under a procedure declaration:

$(\{PROC\ p = cp;;\ c_1;\ c_2\},\ s) \to (\{PROC\ p = cp;;\ c_1\};\ \{PROC\ p = cp;;\ c_2\},\ s)$

$(\{PROC\ p = cp;;\ SKIP\},\ s) \to (SKIP,\ s)$

$(c,\ s) \to (c',\ t) \Longrightarrow (\{PROC\ p = cp;;\ c\},\ s) \to (\{PROC\ p = cp;;\ c'\},\ t)$

(a) Complete the small-step semantics by formulating the missing rules for *CALL*.

(b) Define a recursive function that checks if a program is well-formed, that is, it contains no calls to procedures that were not defined.

(c) Prove that the evaluation of a well-formed programm cannot get stuck: If c is well-formed and $(c,\ s) \to*\ (c',\ t)$ and *final* $(c',\ t)$ then $c' = SKIP$.

Recall that *final cs* is defined as $\neg\ (\exists\, cs'.\ cs \to cs')$.