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

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

Due to Mon., 26. Apr. 2010, *before* the exercise course begins.

Exercise 1.1: **(1+2+1 points)**

In this exercise we will make ourselves familiar with the WHILE programming language introduced in the lecture and extend it to handle not only arithmetic and boolean expressions, but strings as well.

(a) Write a WHILE program calculating the sequence (a_n) by $a_n = 2^{2^{n-1}}$ for $n = 1, 2, 3, \ldots$. (*Hint:* Note that 2^0 needs special treatment.)

(b) Extend the WHILE syntax in a way that it includes string expressions, do not forget to give appropriate semantics, too. The extended syntax should at least contain support for strings of arbitrary length and string concatenation as well as an equality test for strings.

(c) Adapt your implementation from (a) to store the sequence calculated up to (a_n) in a string variable at any time.

Exercise 1.2: **(1+1+1 points)**

The explicit calculation function $a_n = 2^{2^{n-1}}$ given in exercise 1.1 describes the non-linear recursive sequence $a_n = (a_{n-1})^2$ for $a_1 = 2$.

(a) Prove by mathematical induction that this is indeed the case for all natural numbers $n \geq 1$, i.e. show:

$$2^{2^{n-1}} = (a_{n-1})^2 \text{ for } a_1 = 2, \forall\, n \in \mathbb{N} \tag{1}$$

(b) What could go wrong when you try to prove, that the resulting sequence of your program from exercise 1.1(a) and a program implementing the non-linear recursive sequence $a_n = (a_{n-1})^2, a_1 = 2$ are equivalent.

(c) Show that the following proposition holds:

For all states σ, σ', $\langle \textbf{while true do skip}, \sigma \rangle \nrightarrow \sigma'$.

Exercise 1.3: **(2+2 points)**

In the lecture we have defined so–called *bigstep semantics* for expressions, i.e., a relation $\rightarrow \subseteq (AExp \cup BExp) \times \Sigma \times (\mathbb{Z} \cup \mathbb{B})$ which yields the value of an expression within one step: $\langle (3+3) * (9-2), \sigma \rangle \rightarrow 42$. (Thus the intermediate results of the computation are "hidden" in the derivation tree.)
Alternatively it is possible to explicitly represent the intermediate steps by defining *single–step semantics*:
$\langle (3+3) * (9-2), \sigma \rangle \rightarrow \langle 6 * (9-2), \sigma \rangle \rightarrow \langle 6 * 7, \sigma \rangle \rightarrow \langle 42, \sigma \rangle \rightarrow 42$. Give a complete specification of the single–step relation

(a) $\rightarrow_1^a \subseteq (AExp \times \Sigma) \times (AExp \times \Sigma \cup \mathbb{Z})$ for arithmetic expressions and

(b) $\rightarrow_1^b \subseteq (BExp \times \Sigma) \times (BExp \times \Sigma \cup \mathbb{B})$ for Boolean expressions.