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

Priv.-Doz. T. Noll noll@cs.rwth-aachen.de
J. Heinen heinen@cs.rwth-aachen.de
Ch. Jansen christina.jansen@cs.rwth-aachen.de

9. Exercise sheet *Static Program Analysis 2011*

Due Mon, 4 . July 2011, *before* the exercise course begins.

Exercise 9.1: **(4 + 6 + 3 points)**

In most programming languages (e.g. C and Java) assignments can be used within arithmetic expressions. This results in side effects that should be treated by the abstract interpretation. Consider the following extended syntax of WHILE Programs:

$$\begin{array}{lcl} a & ::= & z \mid x \mid \mathbf{x ::= a} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \star a_2 \in AExp \\ b & ::= & t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\ c & ::= & skip \mid \mathbf{x := a} \mid c_1; c_2 \mid \textsf{if } b \textsf{ then } c_1 \textsf{ else } c_2 \mid \textsf{while } b \textsf{ do } c \in Cmd \end{array}$$

Note that we define two different assignments: a new one $x ::= a$ as arithmetic expression and $x := a$, the one we defined before as command. The use of the $eval_\sigma$ function to evaluate expressions is no longer suitable as side effects could appear in this version of WHILE Programs. Therefore new semantics of WHILE has to be developed. Expressions are evaluated from left to right and are considered as non strict (like in C or Java), i.e. expression are not evaluated unless they are actually used (e.g. for expression $true \vee b$ the expression b is not evaluated as the result is determined by the leading $true$).

(a) Develop rules defining the *concrete* execution relation for extended WHILE Programs.

(b) Develop rules defining the *abstract* execution relation for extended WHILE Programs.

(c) Perform a concrete execution on the following extended WHILE Program using your rules from (a):

$x := 2$
$y := 4$
while $n := y - x < 5$ **do**
  $y := (x := y + n) + x;$