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

11. Exercise sheet *Static Program Analysis 2011*

This exercise is **not compulsory** and thus points will not be considered for the exam admission.
The solution should be handed in due Mon, 18. July 2011 if you wish a correction.

Exercise 11.1: **(2 + 2 points)**

Consider the following program:

```
proc A(val x, y, res z) is
    if   x < y   then
        call B(x, x, z);
    else
        call B(x, y − 1, z);
end;

proc B(val x, y, res z) is
    if   y > 0   then
        call A(x, y, z);
    else
        z := y;
end;

call B(x, x, y);
call A(x, y, z);
```

(a) Give the context free grammar defining the valid paths.

(b) Give the resulting path for input variable x = 0 and the corresponding derivation for your grammar from (a).

Exercise 11.2: **(3 + 2 + 2 points)**

In a Detection of Sign Analysis one models numbers by their sign, i.e. by the symbols of {-, 0, +}. Consider the lattice $2^{Var \to \{-,0,+\}}$ as considered in symbolic execution.

(a) Define an instance of the Dataflow Analysis framework for performing Detection of Sign Analysis (without procedures).

(b) Define the interprocedural Dataflow Analysis for the Detection of Sign Analysis.

(c) Realise a Detection of Sign Analysis by MVP solution for the following program (x,y are input variables):

```
proc quad(val x, res z) is
    z := x * x;
end;

proc magic(val x, y, res z) is
    if   x > y   then
         z := x;
    else
         call quad(y, z);
end;

call quad(x, z);
call max(y, z, x);
```

Exercise 11.3: **(2+2 points)**

In the lecture we considered procedures with one call-by-value and one call-by- result parameter. The extension to multiple call-by-value and call-by-result parameters is straightforward. Define the transfer function associated with procedure call, procedure entry, procedure exit and procedure return for the following two extensions, considering the constant propagation:

(a) Call-by-value-result parameters that are parameters that are used as in and output of the procedure (e.g. call by reference).

(b) Local variables declarations: **proc** *name*(**val** ..., **res** ..., **local** ...) **is** ... **end**; where variables are set to 0 on declaration.