SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Sebastian Hahn, M.Sc.

Verification of Real-Time Systems SS 2015

Assignment 5

Deadline: Thursday, June 4, 2015,14:00 (no lecture)
Please send your submission via e-mail to sebastian.hahn@cs.uni-saarland.de, or bring it to room 402 (in building E1.3) on Wednesday, June 3, 2015, until 16:00.

Exercise 5.1: Sign Analysis (8=2+2+2+2 Points)

Design an analysis that keeps track whether a variable has a positive, or negative value, or is zero.

1. Give the lattice you operate on.
2. Derive the abstract operators for addition, subtraction, multiplication and division:

$+^\#$	$\bot$	<0	0	>0	$\top$
$\bot$					
<0					
0					
>0					
$\top$					

$-^\#$	$\bot$	<0	0	>0	$\top$
$\bot$					
<0					
0					
>0					
$\top$					

$*^\#$	$\bot$	<0	0	>0	$\top$
$\bot$					
<0					
0					
>0					
$\top$					

$/^\#$	$\bot$	<0	0	>0	$\top$
$\bot$					
<0					
0					
>0					
$\top$					

3. Derive the abstract operators for $<$ and $=$. Explain how the abstract operators for $\leq$, $\geq$, $>$, and $<>$ can be obtained from these operators.

$<^\#$	$\bot$	<0	0	>0	$\top$
$\bot$					
<0					
0					
>0					
$\top$					

$=^\#$	$\bot$	<0	0	>0	$\top$
$\bot$					
<0					
0					
>0					
$\top$					

4. Build the control-flow graph for the following program and perform your sign analysis. Give the system of (in-)equalities.

```
x = 10;
y = 2;
z = 0;
while (x > 0) {
    y = y + y;
    if (x < y)
        y = 0;
    else
        z = z - x;
    x = x - 1;
}
```

Exercise 5.2: Widening and Narrowing (8=1+2+1+2+2(+3) Points)

In the lecture, we introduced widening as a technique to enforce/speed up termination of analyses. We noticed that applying widening at all program points can yield very imprecise results. However, it seemed enough to apply widening once for each cycle in the control-flow graph.

(a) Explain whether applying widening once per cycle is sufficient.

(b) Build the control-flow graph of the following program that contains an explicit in-bounds check for an array computation.

```
i = 0;
while (i < 42) {
  if (0 <= i & i <= 42) {
    B = A + i;
    M[B] = i;
    i = i + 1;
  } else {
    // Error: Out of Bounds
  }
}
```

(c) Apply simple widening at all program points for an interval analysis of variable i. Discuss the results.

(d) Apply simple widening only at the program point prior to the evaluation of the while condition. Discuss the results.

(e) Apply simple widening only at the program point prior to the evaluation of the if condition. Discuss the results (especially compared to (d)).

(f) *Optional:* Narrowing was presented as a technique to recover precision after applying widening. Perform narrowing starting from the information gained in (c). Discuss the results compared to the previous ones.

Exercise 5.3: Loop Bound Analysis (10=2+6+2 Points)

In this exercise, you should perform a *loop bound analysis* using the approach by Ermedahl et al. Consider the following program:

```
a = 7;
b = 73;
c = 0;
j = 42;
while (j >= INPUT) {
    b = b + a;
    j = j - 14;
    c = 13 * b;
    j = j + a;
}
```

1. Apply program slicing to identify the parts relevant for the loop bound computation.
2. Perform a value analysis for both the interval and the congruence domain. You can assume that `INPUT` has a value in $[0, 10]$. In how many states can the program be at the loop header?
3. Can you tighten the loop bound by incorporating invariant analysis information? Explain.