Lehrstuhl für Informatik 2
Modellierung und Verifikation von Software

Static Program Analysis WS2014/15
Sheet 2 (Hand in until 03.11.2014)

Prof. Dr. Thomas Noll

Christian Dehnert, Benjamin Kaminski

Exercise 1 (Analyses Properties): (6 Points)

a) In the lecture it was indicated that the solution to the equation system of static analyses need not always be unique. Give a labeled WHILE program c whose corresponding equation system for Available Expression Analysis does not possess a unique solution. Provide the equation system as well as two different solutions.

b) For both Available Expression Analysis (AEA) and Live Variable Analysis (LVA) we assumed the program to have isolated entries or exits, respectively. Is this restriction necessary for AEA and LVA as defined in the lecture, i.e. do the analyses need to be changed in case the program does not fulfill the property and if so *how*? Justify your answer.

c) Imagine a *Possibly Available Expression Analysis* (PAEA) that tries to determine which expressions are possibly available at some program location. In other words, the analysis works similarly as AEA but uses the set union instead of the set intersection as the *combination operator*. Is it in this case necessary for the program to have an isolated entry? Justify your answer.

d) Furthermore, does the isolated entry/exit property limit the expressivity of the WHILE language? For this, either provide an example that cannot be transformed to some equivalent program with isolated entries/exits or give a general transformation technique that establishes that property for a given program.

Exercise 2 (Live Variables Analysis): (4 Points)

Perform a *live variable analysis* for the following program:

```
x := 4;
y := 1;
while (y < 10)
    z := x + y * 2;
    if (z > 314)
        y := 42;
    else
        y := x - y;
    z := x + 4;
    if (x > 3)
        skip;
    else
        y := 4;
    z := x + x;
    y := 2 * x;
z := x * y;
```

a) Extend the program to a labeled WHILE-program and give *kill*- and *gen*-sets for every block.

b) Provide the resulting equation system for LVA as well as its least solution.

c) State the resulting program after dead code elimination.