General Remarks

- Please hand in your solutions in groups of 3. Either hand in your solutions at the beginning of the exercise class or put them into the box at the chair.
- If you have questions regarding the exercises and/or lecture, feel free to write us an email or visit us at our office.

Exercise 1 (Properties of Galois Connections): (2 Points)

Let $(L, \sqsubseteq_L)$ and $(M, \sqsubseteq_M)$ be complete lattices with $\bot_K$ be the least element of lattice K. Further let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$. Prove or disprove the following statements.

a) $\alpha(\bot_L) = \bot_M$

b) $\gamma(\bot_M) = \bot_L$

c) $\alpha \circ \gamma \circ \alpha = \alpha$

d) $\gamma \circ \alpha \circ \gamma = \gamma$

Exercise 2 (Abstracting Stacks): (3 Points)

Consider stacks over integer values. Given a stack s and a number i s.**push(**i**)** pushes i on the top of s, while s.**pop()** removes the top element without returning it. Moreover, we assume that expression s.**peek()** returns the top most element of s and $t := s$ assigns the stack values from s to the stack t.

1. Extend the *execution relation* given in the lecture for the new statements **push(**i**)** and **pop()**. Consider val_σ to work on stack variables and **peek()** calls as expected.
2. Provide a reasonable Galois connection, with concrete domain $2^{\mathbb{Z}^* \times \mathbb{Z}^*}$ and abstract domain $2^{\{H,S\}} \times 2^{\mathbb{Z}}$, where H in the first component of M states that the top element of both lists are equal and S that one list is a suffix of the other one. The second component of M denotes the difference in length, i.e. for pair of stacks (a, b) the value of the second component would be $|a| - |b|$. Justify your answer!

Exercise 3 (Reaching Definitions Analysis as Abstract Interpretation): (4 Points)

We will now consider the Reaching Definitions Analysis (RDA). Given a labeled WHILE-program, this analysis computes for every program location and every variable all other program locations in which the variable might have been most recently written (i.e. written without being re-written in between). As an example, consider the following program.

```
[x := 2]^1;
[x := 3]^2;
while [y < 10]^3;
    [y := y + 1]^4;
[x := y * 2]^5;
```

$(x, 2)$ is a reaching definition at label 4, because there is a path reaching label 4 such that x is most recently written at label 2. On the other hand $(x, 1)$ is not a reaching definition at label 4. If the most recent definition of a variable is "before the program", this is indicated by a question mark as the label information. For example, for label 5 we have the reaching definitions $\{(y, 4), (y, ?), (x, 2)\}$.

Your task is to implement this analysis in the abstract interpretation framework. Assume that you are given a **labeled** WHILE-program.

Hint: You need to adapt the concrete semantics and the program state to support labelled programs.

a) Formally (re-)define the parts of the concrete semantics of WHILE that need to be changed.

b) Formally adapt the abstract semantics of WHILE.

c) Build the abstract transition system for your abstraction and the following WHILE-program.

```
[x := 2]^1;
[y := x + 2]^2;
while [x > 3]^3
    [y := y + x]^4;
[x := y]^5;
```

d) What do you ultimately have to do with the abstract transition system to derive the desired output of the analysis?