Dr. Matthias Heizmann
Dominik Klumpp

Hand in until 10:00 on July 27, 2020
Discussion: July 27, 2020

Tutorial for Program Verification
Exercise Sheet 20

In this exercise sheet we work with the automated verification techniques CEGAR (Predicate Abstraction) and Trace Abstraction. We conclude with a preparation exercise for next week's lecture on termination.

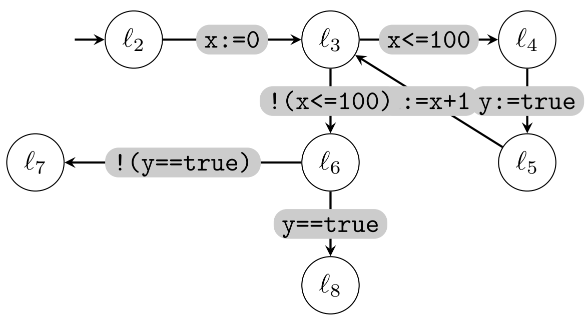
Submit your solution by uploading it as PDF in ILIAS.

Exercise 1: 5 Points

Apply the CEGAR approach to the program below. Whenever you have to provide a sequence of statements you may return any sequence, but we encourage you to take the shortest sequence. Document all intermediate steps.

Hint: If you choose the abstraction of traces wisely, then two iteration steps are sufficient.

```
x := 0;
while (x <= 100) {
  y := true;
  x := x + 1;
}
assert y == true;
```



Exercise 2: Abstraction of a Trace 2 Points

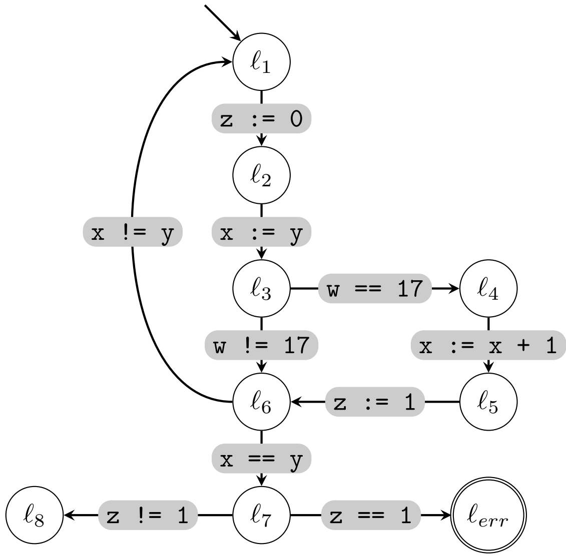
In the lecture we defined an *abstraction* $\pi^\#$ of a trace π, derived by replacing some of the statements *st* with their abstract counterpart *abstract*(*st*). The intuition is that sometimes a few statements in π are sufficient to make it infeasible. A proof of infeasibility of $\pi^\#$ is then also a proof of infeasiblity of π.

In this exercise, we consider a modified concept of abstraction: Instead of replacing assignments with their abstraction (`havoc`), we delete them from the trace entirely.

Show that this is not a good notion of abstraction. In particular, give a trace π and a corresponding abstraction $\pi^\#$, such that $\pi^\#$ is infeasible, but π is feasible. Give a proof of infeasibility for $\pi^\#$, and an execution for π.

Exercise 3: Trace Abstraction 3 Points

Consider the following control-flow graph for a program P, and let $\mathcal{A}_P$ be the corresponding automaton. In this task, you should apply trace abstraction to prove that the program P is safe.



Give two error traces π_1 and π_2 and construct corresponding Floyd-Hoare automata $\mathcal{A}_1$ and $\mathcal{A}_2$ such that the inclusion $L(\mathcal{A}_P) \subseteq L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ holds.

Exercise 4: Well-Founded Relations 3 Points

For the purpose of termination analysis, we will need the notion of a *well-founded* relation. In this exercise, we will introduce the definition and apply it to a few relations.

In the chapter on termination analysis we will work with *infinite sequences*. Analogously to a sequence of length n, which can be seen as a map whose domain is $\{0, \dots n-1\}$, an infinite sequence can be seen as a map whose domain are all natural numbers.

> **Definition (Well-Founded Relation)** Let X be a set. We call a binary relation $R \subseteq X \times X$ *well-founded* if there is no infinite sequence $x_1, x_2, \dots$ such that $(x_i, x_{i+1}) \in R$ for all $i \in \mathbb{N}$.

For each of the following relations, state if it is well-founded or not. If it is not, give an infinite sequence as a counterexample.

(a) $R_a = \{\, (x, x') \in \{\mathbf{true}, \mathbf{false}\}^2 \mid x = x' \,\}$

(b) $R_b = \{\, (x, x') \in \mathbb{N}^2 \mid x > x' \,\}$

(c) $R_c = \{\, (x, x') \in \mathbb{Z}^2 \mid x > x' \,\}$

(d) $R_d = \{\, (x, x') \in \mathbb{Q}^2 \mid x \geq 0 \text{ and } x' \geq 0 \text{ and } x > x' \,\}$

(e) $R_e = \{\, \big((x, y), (x', y')\big) \in (\mathbb{N}^2)^2 \mid x > x' \text{ or } (x = x' \text{ and } y > y') \,\}$

(f) $R_f = \{\, \big((x, y), (x', y')\big) \in (\mathbb{Z}^2)^2 \mid x' = 3 \cdot x \text{ and } y' = 2 \cdot y \text{ and } x \geq 2 \text{ and } y \geq 2\}$