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.

1. \( x := 0; \)
2. \( \textbf{while} \ (x \leq 100) \{ \)
3. \( \quad y := \text{true}; \)
4. \( \quad x := x + 1; \)
5. \( \} \)
6. \( \text{assert} \ y == \text{true}; \)

Exercise 2: Abstraction of a Trace 2 Points
In the lecture we defined an abstraction \( \pi^\# \) of a trace \( \pi \), derived by replacing some of the statements \( st \) with their abstract counterpart \( \text{abstract}(st) \). The intuition is that sometimes a few statements in \( \pi \) are sufficient to make it infeasible. A proof of infeasibility of \( \pi^\# \) is then also a proof of infeasibility of \( \pi \).

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

Show that this is not a good notion of abstraction. In particular, give a trace \( \pi \) and a corresponding abstraction \( \pi^\# \), such that \( \pi^\# \) is infeasible, but \( \pi \) is feasible. Give a proof of infeasibility for \( \pi^\# \), and an execution for \( \pi \).
Exercise 3: Trace Abstraction

Consider the following control-flow graph for a program \( P \), and let \( A_P \) be the corresponding automaton. In this task, you should apply trace abstraction to prove that the program \( P \) is safe.

\[
\begin{align*}
\ell_1 & \quad z := 0 \\
\ell_2 & \quad x := y \\
\ell_3 & \quad w := 17 \\
\ell_4 & \quad x := x + 1 \\
\ell_5 & \quad z := 1 \\
\ell_6 & \quad x := y \\
\ell_7 & \quad w := 17 \\
\ell_8 & \quad z := 1 \\
\end{align*}
\]

Give two error traces \( \pi_1 \) and \( \pi_2 \) and construct corresponding Floyd-Hoare automata \( A_1 \) and \( A_2 \) such that the inclusion \( L(A_P) \subseteq L(A_1) \cup L(A_2) \) holds.

Exercise 4: Well-Founded Relations

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 \( \{0, \ldots, n-1\} \), 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 \( x_1, x_2, \ldots \) 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) \( R_a = \{ (x, x') \in \{true, false\}^2 \mid x = x' \} \)

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

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