Prof. Dr. Andreas Podelski
Matthias Heizmann

21.12.2011
Submission: 10.1.2012
at the beginning of the lecture

Tutorials for Program Verification
Exercise sheet 9

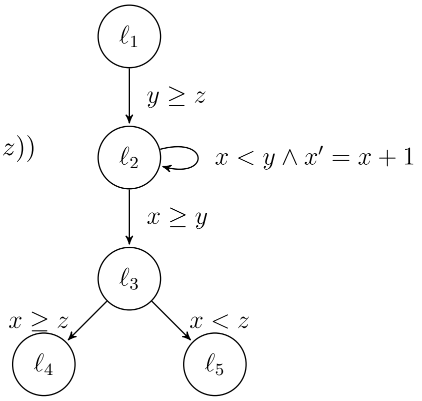
Exercise 1: Counterexample-guided Discovery of Predicates 3+1 points

In the lecture you have seen the function RefinePath which was used in the algorithm AbstRefineLoop and returns a set of predicates Preds given a path $\rho_1, \ldots, \rho_n$.

(a) State a concrete algorithm for RefinePath. Your algorithm may return more than $n + 1$ predicates.

Consider again the program from exercise 4 of the sixth exercise sheet.

$$\begin{aligned}
\rho_1 &= (\mathit{move}(\ell_1, \ell_2) \wedge y \geq z \wedge \mathit{skip}(x, y, z))\\
\rho_2 &= (\mathit{move}(\ell_2, \ell_2) \wedge x + 1 \leq y \wedge x' = x + 1 \wedge \mathit{skip}(y, z))\\
\rho_3 &= (\mathit{move}(\ell_2, \ell_3) \wedge x \geq y \wedge \mathit{skip}(x, y, z))\\
\rho_4 &= (\mathit{move}(\ell_3, \ell_4) \wedge x \geq z \wedge \mathit{skip}(x, y, z))\\
\rho_5 &= (\mathit{move}(\ell_3, \ell_5) \wedge x + 1 \leq z \wedge \mathit{skip}(x, y, z))
\end{aligned}$$



Let Preds_{pc} be the set of all predicates on the program counter.

$$\mathsf{Preds}_{pc} = \{pc = \ell_1, pc = \ell_2, pc = \ell_3, pc = \ell_4, pc = \ell_5\}$$

Given the path $\rho_1\rho_2\rho_3\rho_5$, your algorithm should return a set of predicates Preds such that $\mathsf{Preds} \cup \mathsf{Preds}_{pc}$ is suffient to prove safety of the program i.e., every abstract state returned by AbstReach($\mathsf{Preds} \cup \mathsf{Preds}_{pc}$) is disjoint from φ_{err} (the set of error states φ_{err} is $pc = \ell_5$).

Show that the predicates returned by your algorithm are sufficient to prove safety of the program.

(b) State a different program and some path such that the predicates returned by your algorithm are not sufficient to prove safety of this program. Explain!

Exercise 2: State Space Explosion 2 points + 1 bonus point

Consider the algorithm AbstReach (the version from Monday 12th December). Let $n = |\mathsf{Pred}|$ be the number of predicates. Let $m = |\mathcal{R}|$ be the number of transitions of the program.

(a) How many abstract reachable states (elements of $\mathsf{ReachStates}^{\#}$) are there in the worst case? Explain!

(b) How many times do we check validity of an implication $\varphi \models p$ in the worst case? Explain!

(c) Let us roughly estimate the maximal number of predicates a tool can deal with (in the worst case). Consider the following setting: We have an implementation of AbstReach that may use up to 4 gibibyte, one abstract state needs 32 byte and we neglect the memory necessary for all other data (e.g., the Parent relation). What is the maximal number of predicates n_{max} such that our implementation of AbstReach does not run out of memory. Explain!

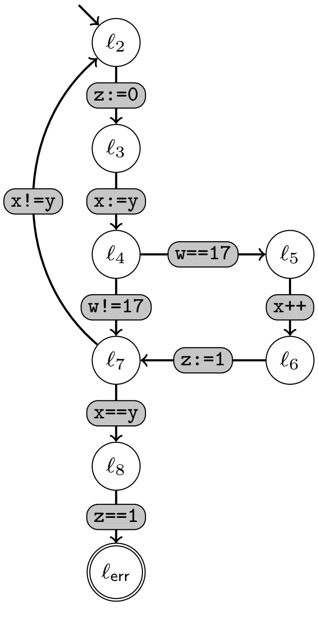
(d) Let us roughly estimate the runtime of AbstReach for n_{max} predicates. Consider the following setting: We have $m = 1000$ relations. The theorem prover always needs exactly one millisecond to decide validity of an implication $\varphi \models p$. If we neglect the runtime of all components but the theorem prover. How much time does it take in the worst case to compute the set of all reachable abstract states? Explain!

(e) Suggest an optimization for the AbstRefineLoop algorithm that can reduce the number of abstract states.

Exercise 3: Execution of Trace Abstraction 3 points

Consider the following program and the corresponding control automaton $\mathcal{A}_{\mathcal{P}}$.

```
        int x, y, z, w;
        void foo()
        {
1:          do {
2:              z = 0;
3:              x = y;
4:              if (w == 17){
5:                  x++;
6:                  z = 1;
                }
7:          } while(x!=y)
8:          assert (z != 1);
        }
```



Give two error traces π_1, π_2 and construct corresponding interpolant automata $\mathcal{A}_1, \mathcal{A}_2$ such that the inclusion $\mathcal{L}(\mathcal{A}_{\mathcal{P}}) \subseteq \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$ holds.

Remark: We call a trace π infeasible if $post(\mathsf{true}, \pi) = \mathsf{false}$

Exercise 4: Interpolant Automata 2 points

Prove that an interpolant automaton accepts only infeasible traces.