Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann

Preference deadline: July 5, 2017
Discussion: July 9, 2017

Tutorial for Program Verification
Exercise Sheet 10

Exercise 1: Properties of $\mathit{post}^\#$ — 1 Point

Give a counterexample for those of the following propositions that are wrong.

(a) $\mathit{post}^\#(\varphi, \rho_1 \circ \rho_2) \subseteq \mathit{post}^\#(\mathit{post}^\#(\varphi, \rho_2), \rho_1)$

(b) $\mathit{post}^\#(\varphi, \rho_1 \circ \rho_2) \supseteq \mathit{post}^\#(\mathit{post}^\#(\varphi, \rho_2), \rho_1)$

(c) $\mathit{post}^\#(\varphi, \rho_1 \vee \rho_2) \subseteq \mathit{post}^\#(\varphi, \rho_1) \vee \mathit{post}^\#(\varphi, \rho_2)$

(d) $\mathit{post}^\#(\varphi, \rho_1 \vee \rho_2) \supseteq \mathit{post}^\#(\varphi, \rho_1) \vee \mathit{post}^\#(\varphi, \rho_2)$

(e) $\mathit{post}^\#(\varphi_1 \vee \varphi_2, \rho) \subseteq \mathit{post}^\#(\varphi_1, \rho_1) \vee \mathit{post}^\#(\varphi_2, \rho)$

(f) $\mathit{post}^\#(\varphi_1 \vee \varphi_2, \rho) \supseteq \mathit{post}^\#(\varphi_1, \rho_1) \vee \mathit{post}^\#(\varphi_2, \rho)$

Exercise 2: AbstReach — 2 Points

Consider the following program.

```
    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);
    }
```

Solve the following tasks without explicitly executing the procedure AbstReach (unless you have a lot of free time and paper).

(a) Is the program safe? Give an intuitive argument.

(b) Give three predicates (in addition to the predicates on the program counter) such that the corresponding abstraction is sufficient to prove safety. Give the corresponding abstract reachability graph (in an informal representation where the edges are labeled by line numbers).

(c) Give the abstract reachability graph that corresponds to the abstraction for the set of predicates $Pred_0$ which contains only the predicates on the program counter. Take the shortest counterexample path. Add one predicate p_1 to eliminate this first counterexample.

(d) Give the abstract reachability graph that corresponds to the abstraction for the set of predicates $Pred_1 := Pred_0 \cup \{p_1\}$. Take again the shortest counterexample path. Add two predicates p_2 and p_3 to eliminate this counterexample. (Did you get the three predicates from (b)?)

Exercise 3: State space explosion 2 Points

Consider the procedure AbstReach. Let $n := |Preds|$ 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?

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

(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? Consider the worst case scenario from part (a).

(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? Consider the worst case scenario from part (b).

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