Prof. Dr. Thomas Noll

Christian Dehnert, Benjamin Kaminski

Exercise 1 (Execution Relation): (3 Points)

Consider the following variation of the (if1) execution relation for predicate abstraction.

$$\text{(if1)}\ \frac{\exists \sigma : \sigma \models b \wedge q}{< \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, q > \Rightarrow < c_1, \overline{q \wedge b} >}$$

Does this (if1) executation relation provide an optimisation of predicate abstraction as presented in the lecture? If so, provide an example derivation where this execution relation exhibits less nondeterminism than the one presented in the lecture. If not, show why.

Exercise 2 (Predicate Abstraction): (4 Points)

Consider the following program fragment c:

$[res := 0]^1$;
$[count := 0]^2$;
while $[y > 0]^3$
  $[y := y - 1]^4$;
  $[count := count + 1]^5$;
  $[tmp := x]^6$;
  while $[tmp > 0]^7$
    $[tmp := tmp - 1]^8$;
    $[res := res + 1]^9$;
  if $[x == 0]^{10}$
    if $[!(res == 0)]^{11}$
      $[skip]^{12}$;

Prove that label 12 is not reachable using predicate abstraction. To this aim, choose a suitable set of predicates P and give the abstract transition system of c for the chosen set of predicates.

Exercise 3 (State-of-the-art Verification Tools): (3 Points)

The purpose of this task is to apply existing tools. CPAchecker is a tool that tries to prove that a given error location in a C program is (un)reachable. If you experience technical problems, please read the README of the tool and contact us via e-mail. The source files are available as a zip archive on the course website.

a) Download CPAchecker from its website (`http://cpachecker.sosy-lab.org/#download`) and unzip it. Invoke CPAchecker on the example file that comes with CPAchecker (doc/examples/example.c) using

scripts/cpa.sh -predicateAnalysis doc/examples/example.c

and inspect the output. Is it possible to reach the error location in the given example file?

b) Consider the following C program (source file spa1.c).

```
int nondet();

int f(int x) {
    if (x < 0 || x > 5) {
        return 0;
    }
```

```
    if (x & 1 != 0) {
        return 0;
    }

    int tmp = x;
    while (tmp >= 0) {
        tmp = tmp - 1;
        x = x + 1;
    }
    x = x + 1;
    while (x > 1) {
        x = x - 2;
    }
    if (x == 1) {
        goto ERROR;
    }

    return x;
ERROR:
    return (-1);
}

int main() {
    return f(nondet());
}
```

The function `nondet()` is unknown and potentially returns any value (hence the name). Without calling CPAchecker, do you think the error label is reachable? Then, use CPAchecker to verify your intuition by invoking

scripts/cpa.sh -predicateAnalysis spa1.c -stats

where spa1.c is the name of the input file.

As a proof that you have run the tool, please copy the printed statistics to a file and send them to us via e-mail. What happens if you remove the first 3 lines of the function `f(int x)` and start CPAchecker on the modified example?

c) * This is a bonus exercise. Consider the second example (source file spa2.c).

```
int nondet();

int main() {
    int x = nondet();
    int y = x + 1;

    if (y <= x) {
        goto ERROR;
    }

    return 0;
ERROR:
    return (-1);
}
```

Use CPAchecker to determine whether the error location is reachable. Do you agree with the result (as in "Given any implementation of `nondet()`, if the program is actually compiled[1] and run, the return value of `main()` is never going to be -1.")? Explain your answer.

[1] using a hypothetical standard-compliant C compiler