Dr. Matthias Heizmann
Dominik Klumpp

Hand in until 16:00 on July 21, 2021
Discussion: July 21, 2021

Tutorial for Program Verification
Exercise Sheet 24

Exercise 1: Easter Bunny 2 Bonus Points

Let `k` be an integer variable and `a` be an array that has integer indices and integer values. Is the following program terminating?

```
while (a[a[k]] >= 0) {
  a[a[k]] := a[k]] - 1;
}
```

A story that motivates this program:
The array `a` models some street with infinitely many farmhouses. Each farmhouse has a finite number of eggs and because of the zombie apocalypse no new eggs can be produced. The easter bunny is stealing eggs in this street according to a strategy given below and we will ask ourselves if the easter bunny will eventually fail to find new eggs.

The array `a` maps house numbers to the number of eggs that are available in the house. The easter bunny picks initially an integer `k`, and steals eggs iteratively (one at a time) according to the following strategy. In each iteration, the easter bunny first visits farm number `k` and checks the amount of eggs that are available at that farm. Then he visits another farm, namely the farm whose house number coincides with this amount. If no egg is available at this farm, the easter bunny gives up. Otherwise he steals one egg and continues with the next iteration.

Is this a bad strategy because the easter bunny will eventually visit always the same farm? Or is there some street and some choice of k in which the easter bunny can continue forever?

Please note that the easter bunny does not neccesarily steal eggs from the same farm. If he takes the egg from farm number k he will alter the farm that he visits next.

Exercise 2: Concurrent Programs 2 Bonus Points

In the lecture on Wednesday we will see *concurrent programs*. We will not introduce concurrent programs formally and only give an informal explanation in this exercise. A concurrent program is a program that consists of several procedure that are executed at the same time. Below is a concurrent program that consists of two procedures. Initially, the computer starts both programs and in every step the computer decides nondeterministically if it executes the next statement of the first procedure or the next statement of the second procedure.
We assume that `x` is a global variable that can be read/written by both procedures and that the precondition is $\mathtt{x} = 0$.

```
procedure proc1() {
  var y1 : int;
  var i1 : int;
  i1 := 0;
  while (i1 < 10) {
    y1 := x;
    x := y1 + 1;
    i1 := i1 + 1;
  }
  assert x >= 10;
}
```

```
procedure proc2() {
  var y2 : int;
  var i2 : int;
  i2 := 0;
  while (i2 < 10) {
    y2 := x;
    x := y2 + 1;
    i2 := i2 + 1;
  }
  assert x >= 10;
}
```

(a) Let us first consider the first procedure in isolation with the semantics that we learned in the lecture. Can the assert statement ever fail? If yes, give a counterexample. If no, give a loop invariant that is sufficient to prove that the program is correct wrt. the assert statement.

(b) Let us now consider the concurrent program given by both procedures. Can one of the assert statements every fail? If yes, give a counterexample. If no, explain why the assert statements can never fail.