Dr. Matthias Heizmann
Tanja Schindler
Dominik Klumpp

Hand in until 10:00 on June 24, 2019
Discussion: June 24, 2019

Tutorial for Program Verification
Exercise Sheet 14

Exercise 1: CFG for Conditional Statement 2 Points

In the lecture, we defined the notion of a control-flow graph of a given statement. This definition is not yet complete, the case of the conditional-statement was left out. Complete the definition:

Let st_1, st_2 be two statements. Let $G_1 = (Loc^1, \Delta^1, \ell^1_{\mathsf{init}}, \ell^1_{\mathsf{ex}})$ be a control-flow graph for st_1, and let $G_2 = (Loc^2, \Delta^2, \ell^2_{\mathsf{init}}, \ell^2_{\mathsf{ex}})$ be a control-flow graph for st_2 such that Loc^1 and Loc^2 are disjoint. Define a control-flow graph for `if (expr)` $\{\ st_1\ \}$ `else` $\{\ st_2\ \}$.

Exercise 2: From Programs to CFGs 2 Points

For each of the programs given below, draw a control-flow graph.

(a) Code of program P_{pow}:

```
e := 1;
z := 0;
while (z < y) {
  e := e * x;
  z := z + 1;
}
```

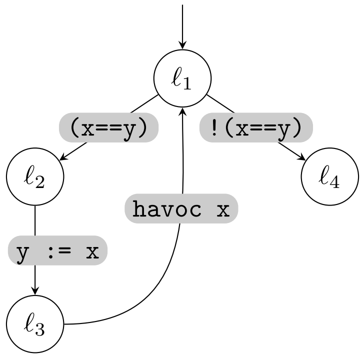
(b) Code of program P_{findmin}:

```
i := lo;
min := a[lo, lo];
while (i <= hi) {
  j := lo;
  while (j <= hi) {
    if (a[i, j] < min) {
      min := a[i, j];
    }
    j := j + 1;
  }
  i := i + 1;
}
```

Exercise 3: Program Configurations 2 Points

Consider the program $P = (V, \mu, \mathcal{T})$ with $V = \{x, y\}$, $\mu(x) = \mu(y) = \{\textbf{true}, \textbf{false}\}$ and $\mathcal{T}$ a derivation tree for the statement below on the left. On the right, a CFG for P is shown.

```
while (x == y) {
  y := x;
  havoc x;
}
```



Draw the reachability graph for this control-flow graph and the precondition-postcondition-pair $(x, x \to \neg y)$.

Exercise 4: Existence of Program Executions 2 Points

Prove the following lemma that has been added to the slides.

> **Lemma** (RelAndExec.2) Let $G = (Loc, \Delta, \ell_{\mathsf{init}}, \ell_{\mathsf{ex}})$ be a control-flow graph for the sequential composition $st_1 st_2$. There exists a program execution $(\ell_0, s_0), \ldots, (\ell_n, s_n)$ with $\ell_0 = \ell_{\mathsf{init}}$ and $\ell_n = \ell_{\mathsf{ex}}$, iff $(s_0, s_n) \in [\![st_1 st_2]\!]$.