Dr. Matthias Heizmann
Dominik Klumpp

Hand in until 16:00 on June 16, 2021
Discussion: June 16, 2021

Tutorial for Program Verification
Exercise Sheet 14

In this exercise sheet we work with the control flow graphs. At the end, we have a bonus exercise that deals with havoc and assume, and is meant to further your understanding of these statements. You have extra time to submit solutions to this bonus exercise.

Submit your solution by uploading it as PDF in ILIAS.

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: We defined it for simple statements and for the sequential composition. The conditional statement (`if/else`) and the `while` statement are still missing. In this exercise, we define the control-flow graph for conditional statements:

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 `}`.

> The following bonus exercise is a bit different from the exercises we usually have. While the lecture should give you enough background to understand the correct solution, finding such a correct solution is non-trivial. Don't despair if you get stuck! But if you are interested in the field of program verification and you like riddles, you might enjoy puzzling it out. And you will improve your understanding of the semantics of `havoc` and `assume` at the same time!
>
> To give you enough time to try and solve this exercise, you have time to submit it until 23rd June.

Exercise 2: Loop Transformation with Havoc and Assume 5 Bonus Points

Consider the program $P_1 = (V, \mu_1, T_1)$ such that T_1 is a derivation tree for the program text on the right, where `C1`, `C2` are boolean expressions which only use variables in V, and `body1`, `body2` are program statements which only use variables in V.

```
while (C1) {
    body1;
}
while (C2) {
    body2;
}
```

$$\varphi_0 \models I_1 \qquad I_1 \wedge \neg C_1 \models \varphi_1$$
$$\varphi_1 \models I_2 \qquad I_2 \wedge \neg C_2 \models \varphi_2$$
$$\{I_1 \wedge C_1\}\,\texttt{body1}\,\{I_1\}$$
$$\{I_2 \wedge C_2\}\,\texttt{body2}\,\{I_2\}$$

```
havoc V';
V_middle := V';
while (C1 || C2') {
    if (C1) {
        body1;
    }
    if (C2') {
        body2';
    }
}
assume V == V_middle;
V := V';
```

Let us assume a Hoare proof for the program P_1 and the precondition-postcondition pair $(\{\varphi_0\}, \{\varphi_2\})$, which uses inductive invariants I_1 and I_2. In short, assume that the implications and Hoare triples given above are correct, and that the formulas φ_0, φ_1, φ_2, I_1 and I_2 use only variables in V.

Using `havoc` and `assume` statements, we can transform the program on the left into a new program $P_2 = (V \cup V' \cup V_{middle}, \mu_2, T_2)$, where T_2 is a derivation tree for the program text on the right.[1] The set of variables V' contains one variable `v'` for every `v` $\in V$, and similarly V_{middle} contains one variable $\mathtt{v}_{\mathtt{middle}}$ for every `v` $\in V$, such that $\mu_2(\mathtt{v'}) = \mu_2(\mathtt{v}_{\mathtt{middle}}) = \mu_2(\mathtt{v}) = \mu_1(\mathtt{v})$.

The statement `havoc V';` is shorthand for `havoc v';` for all `v` $\in V$, similarly the statement $\mathtt{V}_{\mathtt{middle}}$ `:= V';` corresponds to $\mathtt{v}_{\mathtt{middle}}$ `:= v';` for all `v` $\in V$, and the statement `assume V ==` $\mathtt{V}_{\mathtt{middle}}$`;` is shorthand for `assume` $\bigwedge_{\mathtt{v} \in V}(\mathtt{v} = \mathtt{v}_{\mathtt{middle}})$;. Finally, `C2'` and `body2';` correspond to `C2` resp. `body2;` where every variable `v` has been replaced by `v'`.

Our goal is to construct a Hoare proof for the program P_2 and the precondition-postcondition pair $(\{\varphi_0\}, \{\varphi_2\})$, using only the formulas from the Hoare proof for P_1, substitutions thereof, and boolean combinations. In particular, give a formula I (a loop invariant) over variables in $V \cup V' \cup V_{middle}$, such that the following Hoare triples are valid:

- $\{\varphi_0\}$ `havoc V';` $\mathtt{V}_{\mathtt{middle}}$ `:= V';` $\{I\}$
- $\{I \wedge (C_1 \vee C_2')\}$ `if (C1) { body1; } if (C2') { body2';}` $\{I\}$
- $\{I \wedge \neg(C_1 \vee C_2')\}$ `assume V ==` $\mathtt{V}_{\mathtt{middle}}$`; V := V';` $\{\varphi_2\}$

You do not have to prove the correctness of your invariant.

[1]This transformation is presented in the paper *"A guess-and-assume approach to loop fusion for program verification"*, A. Imanishi, K. Suenaga and A. Igarashi. PEPM 2018.