

Dr. Matthias Heizmann Dominik Klumpp

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

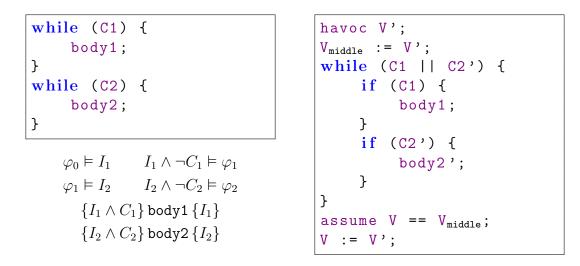
In the lecture, we defined the notion of a control-flow graph of a given statement. This definition is not yet complete: We defined it for simple statements and for the sequential composition. The conditional statement (if/else) and the while statement are still missing. In this exercise, we define the control-flow graph for conditional statements:

Let st_1, st_2 be two statements. Let $G_1 = (Loc^1, \Delta^1, \ell_{init}^1, \ell_{ex}^1)$ be a control-flow graph for st_1 , and let $G_2 = (Loc^2, \Delta^2, \ell_{init}^2, \ell_{ex}^2)$ 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 $23^{\rm rd}$ 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.



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 $\varphi_0, \varphi_1, \varphi_2,$ 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 \mathbf{v} ' for every $\mathbf{v} \in V$, and similarly V_{middle} contains one variable \mathbf{v}_{middle} for every $\mathbf{v} \in V$, such that $\mu_2(\mathbf{v}') = \mu_2(\mathbf{v}_{middle}) = \mu_2(\mathbf{v}) = \mu_1(\mathbf{v})$.

The statement havoc V'; is shorthand for havoc v'; for all $v \in V$, similarly the statement $V_{middle} := V'$; corresponds to $v_{middle} := v'$; for all $v \in V$, and the statement assume $V == V_{middle}$; is shorthand for assume $\bigwedge_{v \in V} (v = v_{middle})$;. 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'; $V_{\text{middle}} := V'; \{I\}$
- $\{I \land (C_1 \lor C'_2)\}$ if (C1) { body1; } if (C2') { body2'; } $\{I\}$
- $\{I \land \neg (C_1 \lor C'_2)\}$ assume V == V_{middle}; V := V'; $\{\varphi_2\}$

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

¹This 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.