Prof. Dr. Christoph Matheja
Theory of Correct Systems @ UOL
Software Systems Engineering @ DTU

Program Verification – Assignment 1

- Submit your solutions online in groups of 1-3 before the deadline.
- The tasks are numbered consecutively across all assignments. If you have questions, you can thus refer to the task number.
- You can use the concepts and notation from the course material without explanation. Any other definitions or notations must be defined before they are used. Similarly, theorems or mathematical properties that are neither introduced in the course nor well-known from typical undergraduate courses, for example discrete math or logic, need to be introduced and proven first. When in doubt about using a property you know from another course, ask us first.

Note: the assignment has been updated on 7/9 to fix typos in T3.

T1: Warmup: verification with CHIP (10 Points)

In this task, we will work with the toy verification tool CHIP:

`http://chip-pv.netlify.app`

Complete each of the following CHIP programs by replacing "TODO" with specifications such that CHIP verifies the program. Try to give preconditions that are true for as many valuations of variables as possible and postconditions that are true for as few valuations of variables as possible. You are not allowed to change other parts of the program.

```
// Program (a)
{ true }
if x < 10 -> skip
[] x <= 10 -> x := 10
fi
{ TODO }
```

```
// Program (b)
{ true }
if x > y -> temp := x; x := y; y := temp
[] y > z -> temp := y; y := z; z := temp
fi;
if
  x > y -> temp := x; x := y; y := temp
fi
{ TODO }
```

```
// Program (c)
{ TODO }
if x > y -> temp := x;
            x := y;
            y := temp
[] x <= y -> skip
fi;
{ TODO }
if x >= y -> z := x
[] y > x -> z := y
fi
{ z = y }
```

Prof. Dr. Christoph Matheja
Theory of Correct Systems @ UOL
Software Systems Engineering @ DTU

T2: Operational semantics (10 Points)

Consider the following memory m:

Variable	$\mathfrak{m}(x)$
x	4
y	-2
z	0

(a) Apply the formal semantics of arithmetic expressions to evaluate the following:

(i) $[x + y + 5](\mathfrak{m})$

(ii) $[y \cdot (x + z)](\mathfrak{m}[y \leftarrow 3])$

(b) Apply the operational semantics of toy programs to execute the command C below on the initial memory $\mathfrak{m}$. That is, compute $\langle C, \mathfrak{m}\rangle \Rightarrow^* \langle \mathsf{done}, \mathfrak{m}'\rangle$ step by step.

$$C: \quad \texttt{if}\ (x > 0)\ \{y := x + y\}\ \texttt{else}\ \{z := x - y\};\ x := z + 1$$

T3: Four flavors of program proofs (40 Points)

In the lectures, we have seen four approaches to writing program proofs, that is, proving that a Floyd-Hoare triple $\{\!\{\ F\ \}\!\}\ C\ \{\!\{\ G\ \}\!\}$ is valid. In this exercise, you will write a program proof using each of those approaches.

(a) Show that the following triple is valid by applying the definition of valid Hoare triple and arguing in terms of the operational semantics.

$$\{\!\{\ 0 < x \land 0 \leq y\ \}\!\}\ y := y + x;\ x := y + y\ \{\!\{\ 0 < x\ \}\!\}$$

(b) Write a proof tree using the rules of Floyd-Hoare logic for the following triple:

$$\{\!\{\ y == 0\ \}\!\}\ \texttt{if}\ (x == 0)\ \{\texttt{skip}\}\ \texttt{else}\ \{y := x + x\};\ z := y + x\ \{\!\{\ z == 3 \cdot x\ \}\!\}$$

(c) Is there a precondition F and a memory $\mathfrak{m}$ with $\mathfrak{m} \models F$ and $\mathfrak{m} \not\models y == 0$ such that $\models \{\!\{\ F\ \}\!\}\ \texttt{if}\ (x == 0)\ \{\texttt{skip}\}\ \texttt{else}\ \{y := x + x\};\ z := y + x\ \{\!\{\ z == 3 \cdot x\ \}\!\}$ holds? Justify your answer.

(d) Write a proof outline for the following triple, where A and B are some integer constants:

$$\{\!\{\ x == 2 * y \land y == 2 * x \land A == 0 \land B == 4 * x\ \}\!\}$$
$$\texttt{if}\ (x > y)\ \{$$
$$\quad z := x - y;$$
$$\quad x := z + y;$$
$$\quad y := z;$$
$$\}\ \texttt{else}\ \{$$
$$\quad z := y - x;$$
$$\quad y := z + x;$$
$$\quad x := z;$$
$$\}$$
$$\{\!\{\ x == B \land y == A\ \}\!\}$$

(e) Check whether the weakest precondition of C and G logically follows from the precondition F. That is, prove $F \models \mathsf{wp}[C](G)$ for the following triple:

$$\{\!\{\ x == B \land y == A \land 0 < x \land 0 < y\ \}\!\}$$
$$\texttt{if}\ (x > y)\ \{$$
$$\quad x := x - y$$
$$\}\ \texttt{else}\ \{$$
$$\quad y := y - x$$
$$\}$$
$$\{\!\{\ x \leq B \land y \leq A\ \}\!\}$$

We have seen five approaches to prove a Floyd-Hoare triple in the course:

T4: Strongest Postconditions (40 Points)

In the lectures, we reduced the task of proving a Floyd-Hoare triple $\{\!\{\ F\ \}\!\}\ C\ \{\!\{\ G\ \}\!\}$ valid to proving a verification condition, namely the logical entailment $F \models \mathsf{wp}[C](G)$, where the weakest precondition $\mathsf{wp}[C](G)$ is computed from the back, i.e. starting at the postcondition, to the front. Alternatively, we can also construct a verification condition by starting at the precondition and computing the *strongest postcondition*. This approach might appear more intuitive at first, because we reason about a program in the same direction as it is executed.

Table 1: Definition of the strongest postcondition by induction on the structure of toy commands for a given precondition F. Recall that $F[\mathtt{x} := a]$ is obtained by substituting every free occurrence of $\mathtt{x}$ in F by a. Moreover, $\mathtt{y}$ is a *fresh* variable.

C	$\mathsf{sp}[C](F)$
`skip`	F
$\mathtt{x} := a$	$\exists \mathtt{y}.\ F[\mathtt{x} := \mathtt{y}] \wedge \mathtt{x} == a[\mathtt{x} := \mathtt{y}]$
$C_1 \,;\, C_2$	$\mathsf{sp}[C_2](\mathsf{sp}[C_1](F))$
`if` (b) `{` C_1 `}` `else` `{` C_2 `}`	$\mathsf{sp}[C_1](b \wedge F) \vee \mathsf{sp}[C_2](\neg b \wedge F)$

The goal of this task is to show that this alternative approach is also sound and complete for toy commands.
Intuitively, the strongest postcondition $\mathsf{sp}[C](F)$ of a toy command C and a precondition F collects all final memories that can be reached by executing C on some initial memory satisfying F. Table 1 defines how to compute strongest postconditions for toycommands.
To verify a triple $\{\!\{\ F\ \}\!\}\ C\ \{\!\{\ G\ \}\!\}$, we then check the verification condition

$$\mathsf{sp}[C](F) \quad \models \quad G.$$

In other words, the set of final memories reachable by executing C from initial memories in F are included in the memories that satisfy the postcondition G.
Show that the above verification condition is equivalent to our verification condition using weakest preconditions. That is, for all toy commands C and toy formulae F, G, we have

$$\mathsf{sp}[C](F) \models G \quad \text{iff} \quad F \models \mathsf{wp}[C](G).$$

To this end, prove the following four properties:

(a) $\vdash \{\!\{\ F\ \}\!\}\ C\ \{\!\{\ \mathsf{sp}[C](F)\ \}\!\}$.

(b) $\mathsf{sp}[C](F) \models G$ implies $F \models \mathsf{wp}[C](G)$.

(c) If $\mathtt{m}' \models \mathsf{sp}[C](F)$, then there exists a memory $\mathtt{m}$ such that $\mathtt{m} \models F$ and $\langle C,\ \mathtt{m}\rangle \Rightarrow^* \langle \mathtt{done},\ \mathtt{m}'\rangle$.

(d) $F \models \mathsf{wp}[C](G)$ implies $\mathsf{sp}[C](F) \models G$.

Hints:

- You can find proofs of similar properties for weakest preconditions in chapter 2.4 of the lecture notes.
- You may use the fact that our toy commands are deterministic and always terminate. That is, for every configuration $\langle C,\ \mathtt{m}\rangle$ there exists a unique memory $\mathtt{m}'$ such that $\langle C,\ \mathtt{m}\rangle \Rightarrow^* \langle \mathtt{done},\ \mathtt{m}'\rangle$.