Programming Languages: Imperative Program Construction
Practicals 0: Non-Looping Constructs and Weakest Precondition

Shin-Cheng Mu

Autumn Term, 2021

Guarded Command Language Basics

1. Which of the following Hoare triples hold?
   (a) $\{x = 7\} skip \{odd\ x\}$;
   (b) $\{x > 60\} x := x \times 2 \{x > 100\}$;
   (c) $\{x > 40\} x := x \times 2 \{x > 100\}$;
   (d) $\{true\} \mathbf{if}\ x \leqslant y \rightarrow y := y - x \mid x \geqslant y \rightarrow x := x - y\ \mathbf{fi} \{x \geqslant 0 \wedge y \geqslant 0\}$;
   (e) $\{even\ x \wedge even\ y\} \mathbf{if}\ x \leqslant y \rightarrow y := y - x \mid x \geqslant y \rightarrow x := x - y\ \mathbf{fi} \{even\ x \wedge even\ y\}$.

Solution: As the first exercise I expect merely that you answer by informal reasoning. What follows is the more formal approach which you will learn later.

(a) The Hoare triple holds because:

$$\begin{array}{ll} & wp\ skip\ (odd\ x) \\ \Leftrightarrow & \quad \{ \text{definition of } wp \} \\ & odd\ x \\ \Leftarrow & x = 7 \ . \end{array}$$

(b) The Hoare triple holds because:

$$\begin{array}{ll} & wp\ (x := x \times 2)\ (x > 100) \\ \Leftrightarrow & \quad \{ \text{definition of } wp \} \\ & x \times 2 > 100 \\ \Leftarrow & x > 60 \ . \end{array}$$

(c) The Hoare triple does not hold because:

$$\begin{array}{ll} & wp\ (x := x \times 2)\ (x > 100) \\ \Leftrightarrow & x \times 2 > 100 \\ \not\Leftarrow & x > 40 \ . \end{array}$$

(d) The annotated **if** statement is

$$\begin{array}{l} \{True\} \\ \mathbf{if}\ x \leqslant y \rightarrow \{x \leqslant y\}\ y := y - x\ \{x \geqslant 0 \wedge y \geqslant 0\} \\ \quad x \geqslant y \rightarrow \{x \geqslant y\}\ x := x - y\ \{x \geqslant 0 \wedge y \geqslant 0\} \\ \mathbf{fi} \\ \{x \geqslant 0 \wedge y \geqslant 0\} \ . \end{array}$$

That $x \leqslant y \vee x \geqslant y$ certainly holds. For the Hoare triple in the first branch we reason:

$$
\begin{aligned}
& (x \geqslant 0 \wedge y \geqslant 0)[y\backslash y - x] \\
\Leftrightarrow\ & x \geqslant 0 \wedge y - x \geqslant 0 \\
\Leftrightarrow\ & x \geqslant 0 \wedge x \leqslant y \\
\not\Leftarrow\ & x \leqslant y \ .
\end{aligned}
$$

The situation with the other branch is similar. The bottom line is that the initial Hoare triple does *not* hold.

The initial Hoare triple would be true if the precondition were $x \geqslant 0 \wedge y \geqslant 0$.

(e) The annotated **if** statement is

$$
\begin{aligned}
& \{even\ x \wedge even\ y\} \\
& \mathbf{if}\ x \leqslant y \to \{even\ x \wedge even\ y \wedge x \leqslant y\}\ y := y - x\ \{even\ x \wedge even\ y\} \\
& \quad\ x \geqslant y \to \{even\ x \wedge even\ y \wedge x \geqslant y\}\ x := x - y\ \{even\ x \wedge even\ y\} \\
& \mathbf{fi} \\
& \{even\ x \wedge even\ y\} \ .
\end{aligned}
$$

That $x \leqslant y \vee x \geqslant y$ certainly holds. For the Hoare triple in the first branch we reason:

$$
\begin{aligned}
& (even\ x \wedge even\ y)[y\backslash y - x] \\
\Leftrightarrow\ & even\ x \wedge even\ (y - x) \\
\Leftrightarrow\ & even\ x \wedge even\ y \\
\Leftarrow\ & even\ x \wedge even\ y \wedge x \leqslant y \ .
\end{aligned}
$$

The situation with the other branch is similar. The bottom line is that the initial Hoare triple does hold.

2. Is it always true that $\{True\}\ x := E\ \{x = E\}$? If you think the answer is yes, explain why. If your answer is no, give a counter example.

Solution: No. For a counterexample, let E be $x + 1$.

When do we do have the property that $\{True\}\ x := E\ \{x = E\}$? Since $(x = E)[x\backslash E] \Leftrightarrow (E = E\ [x\backslash E])$, the Hoare triple holds if and only if $E = E\ [x\backslash E]$. Examples of such E include those that do not contain x, or those that are idempotent funtions on x, for example $E = 0 \uparrow x$.

The actual forward rule for assignment (due to Floyd) is:

$$\{P\}\ x := E\ \{(\exists\, x_0 :: x = E\ [x\backslash x_0] \wedge P\ [x\backslash x_0])\} \ ,$$

where x_0 is a fresh name.

3. Verify:

$$
\begin{aligned}
& \{x = X \wedge y = Y\} \\
& x := x \not\Leftrightarrow y \\
& y := x \not\Leftrightarrow y \\
& x := x \not\Leftrightarrow y \\
& \{x = Y \wedge y = X\}
\end{aligned}
$$

where x and y are boolean and $(\not\Leftrightarrow)$ is the "not equal" or "exclusive or" operator. In fact, the code above works for any $(\otimes)$ that satisfies the properties that for all a, b, and c:

$$\begin{array}{ll} \text{associative}: & a \otimes (b \otimes c) = (a \otimes b) \otimes c\ , \\ \text{unipotent}: & a \otimes a = 1\ , \end{array}$$

where 1 is the unit of $(\otimes)$, that is, $1 \otimes b = b = b \otimes 1$.

Solution: The annotated program is:

$$\begin{array}{l} \{x = X \wedge y = Y, \mathrm{Pf}_2\} \\ x := x \otimes y \\ \{y = Y \wedge x \otimes y = X, \mathrm{Pf}_1\} \\ y := x \otimes y \\ \{x \otimes y = Y \wedge y = X\} \\ x := x \otimes y \\ \{x = Y \wedge y = X\}\ \ . \end{array}$$

Pf_1:

$$\begin{array}{ll} & (x \otimes y = Y \wedge y = X)\,[x \otimes y \,/\, y] \\ \Leftrightarrow & x \otimes (x \otimes y) = Y \wedge x \otimes y = X \\ \Leftrightarrow & \quad \{\ (\otimes) \text{ associative }\} \\ & (x \otimes x) \otimes y = Y \wedge x \otimes y = X \\ \Leftrightarrow & \quad \{\ \text{unipotence }\} \\ & 1 \otimes y = Y \wedge x \otimes y = X \\ \Leftrightarrow & \quad \{\ \text{identity }\} \\ & y = Y \wedge x \otimes y = X\ \ . \end{array}$$

Pf_2:

$$\begin{array}{ll} & (y = Y \wedge x \otimes y = X)\,[x \otimes y \,/\, x] \\ \Leftrightarrow & y = Y \wedge (x \otimes y) \otimes y = X \\ \Leftrightarrow & \quad \{\ (\otimes) \text{ associative }\} \\ & y = Y \wedge x \otimes (y \otimes y) = X \\ \Leftrightarrow & \quad \{\ \text{unipotence }\} \\ & y = Y \wedge x \otimes 1 = X \\ \Leftrightarrow & \quad \{\ \text{identity }\} \\ & y = Y \wedge x = X\ \ . \end{array}$$

4. Verify the following program:

$$\begin{array}{l} \mathbf{var}\ r, b : \mathit{Int} \\ \{0 \leqslant r < 2 \times b\} \\ \mathbf{if}\ b \leqslant r \rightarrow r := r - b \\ \mid\ r < b \rightarrow \mathit{skip} \\ \mathbf{fi} \\ \{0 \leqslant r < b\} \end{array}$$

Solution: The annotated program is:

$$
\begin{array}{l}
\textbf{var}\ r, b : \mathit{Int} \\
\{0 \leqslant r < 2 \times b\} \\
\textbf{if}\ b \leqslant r \rightarrow \{0 \leqslant r < 2 \times b \wedge b \leqslant r\}\ r := r - b\ \{0 \leqslant r < b, \mathrm{Pf}_1\} \\
\mid\ r < b \rightarrow \{0 \leqslant r < 2 \times b \wedge r < b\}\ \mathit{skip}\ \{0 \leqslant r < b, \mathrm{Pf}_2\} \\
\textbf{fi} \\
\{0 \leqslant r < b, \mathrm{Pf}_3\}
\end{array}
$$

Pf_1. We reason:

$$
\begin{array}{ll}
 & (0 \leqslant r < b)\ [r \backslash r - b] \\
\Leftrightarrow & 0 \leqslant r - b < b \\
\Leftrightarrow & b \leqslant r < 2 \times b \\
\Leftarrow & 0 \leqslant r < 2 \times b \wedge b \leqslant r\ \ .
\end{array}
$$

Pf_2. Trivial.

Pf_3. Certainly any proposition implies $b \leqslant r \vee r < b$.

5. Verify:

$$
\begin{array}{l}
\textbf{var}\ x, y : \mathit{Int} \\
\{\mathit{True}\} \\
x, y := x \times x, y \times y \\
\textbf{if}\ x \geqslant y \rightarrow x := x - y \\
\mid\ y \geqslant x \rightarrow y := y - x \\
\textbf{fi} \\
\{x \geqslant 0 \wedge y \geqslant 0\}\ \ .
\end{array}
$$

Solution: For brevity we abbreviate $x \geqslant 0\ \wedge\ y \geqslant 0$ to P. The fully annotated program could be:

$$
\begin{array}{l}
\{\mathit{True}\} \\
x, y := x \times x, y \times y \\
\{P, \mathrm{Pf}_4\} \\
\textbf{if}\ x \geqslant y \rightarrow \{x \geqslant y \wedge P\}\ x := x - y\ \{P, \mathrm{Pf}_1\} \\
\mid\ y \geqslant x \rightarrow \{y \geqslant x \wedge P\}\ y := y - x\ \{P, \mathrm{Pf}_2\} \\
\textbf{fi} \\
\{P, \mathrm{Pf}_3\}\ \ .
\end{array}
$$

To verify the **if** branching, we check that

Pf_1. $\{x \geqslant y\ \wedge\ P\}\ x := x - y\ \{P\}$. The Hoare triple is valid because

$$
\begin{array}{ll}
 & (x \geqslant 0\ \wedge\ y \geqslant 0)[x \backslash x - y] \\
\Leftrightarrow & x - y \geqslant 0\ \wedge\ y \geqslant 0 \\
\Leftrightarrow & x \geqslant y \wedge\ y \geqslant 0 \\
\Leftarrow & x \geqslant y\ \wedge\ x \geqslant 0\ \wedge\ y \geqslant 0.
\end{array}
$$

Pf$_2$. $\{y \geqslant x \ \wedge \ P\}\ y := y - x\ \{P\}$. Omitted.

Pf$_3$. And indeed $x \geqslant y \ \vee \ y \geqslant x$ always holds, thus $P \Rightarrow x \geqslant y \ \vee \ y \geqslant x$.

Do not forget that we have yet to verify $\{true\}\ x, y := x \times x, y \times y\ \{P\}$, which is not difficult either:

Pf$_4$.

$$
\begin{aligned}
& (x \geqslant 0 \ \wedge \ y \geqslant 0)[x, y \backslash x \times x, y \times y] \\
\Leftrightarrow\ & x \times x \geqslant 0 \ \wedge \ y \times y \geqslant 0 \\
\Leftrightarrow\ & true.
\end{aligned}
$$

6. Verify:

$$
\begin{array}{l}
\textbf{var } a, b : Bool \\
\{True\} \\
\textbf{if } \neg\, a \vee b \rightarrow a := \neg\, a \\
\mid\ a \vee \neg\, b \rightarrow b := \neg\, b \\
\textbf{fi} \\
\{a \vee b\}\ .
\end{array}
$$

Solution:

$$
\begin{array}{l}
\textbf{var } a, b : Bool \\
\{True\} \\
\textbf{if } \neg\, a \vee b \rightarrow \{\neg\, a \vee b\}\ a := \neg\, a\ \{a \vee b, \mathrm{Pf}_1\} \\
\mid\ a \vee \neg\, b \rightarrow \{a \vee \neg\, b\}\ b := \neg\, b\ \{a \vee b, \mathrm{Pf}_2\} \\
\textbf{fi} \\
\{a \vee b, \mathrm{Pf}_3\}\ .
\end{array}
$$

Pf$_1$. To verify the first branch:

$$
\begin{aligned}
& (a \vee b)[a \backslash \neg a] \\
\equiv\ & \neg a \vee b.
\end{aligned}
$$

Pf$_2$. The other branch is similar.

Pf$_3$. Certainly $true \Rightarrow \neg a \vee b \vee a \vee \neg b$.

Weakest Precondition

7. Given below is a list of statements and predicates. What are the weakest precondition for the predicates to be true after the statement?

   (a) $x := x \times 2$, $x > 100$;

   (b) $x := x \times 2$, $even\ x$;

   (c) $x := x \times 2$, $x > 100 \wedge even\ x$;

(d) $x := x \times 2$, *odd* x.

(e) *skip*, *odd* x.

Solution:

(a) $x \times 2 > 100$, that is, $x > 50$.

(b) *even* $(x \times 2)$, which simplifies to *True*.

(c) $x \times 2 > 100 \wedge \mathit{even}\ (x \times 2)$, that is, $x > 50$.

(d) *odd* $(x \times 2)$, that is, *False*.

(e) *odd* x.

8. Prove that $(\mathit{wp}\ S\ Q_0 \vee \mathit{wp}\ S\ Q_1) \Rightarrow \mathit{wp}\ S\ (Q_0 \vee Q_1)$.

Solution: Recall from propositional logic that $(P \vee Q) \Rightarrow R$ iff. $(P \Rightarrow R) \wedge (Q \Rightarrow R)$.

$$\begin{array}{ll}
 & (\mathit{wp}\ S\ Q_0 \vee \mathit{wp}\ S\ Q_1) \Rightarrow \mathit{wp}\ S\ (Q_0 \vee Q_1) \\
\Leftrightarrow & \{ \text{ said property above } \} \\
 & (\mathit{wp}\ S\ Q_0 \Rightarrow \mathit{wp}\ S\ (Q_0 \vee Q_1)) \wedge \\
 & (\mathit{wp}\ S\ Q_1 \Rightarrow \mathit{wp}\ S\ (Q_0 \vee Q_1)) \\
\Leftarrow & \{ \text{ Monotonicity } \} \\
 & (Q_0 \Rightarrow (Q_0 \vee Q_1)) \wedge (Q_1 \Rightarrow (Q_0 \vee Q_1)) \\
\Leftrightarrow & \mathit{True}\ \ .
\end{array}$$

9. Recall the definition of Hoare triple in terms of *wp*:

$$\{P\}\,S\,\{Q\} \;=\; P \Rightarrow \mathit{wp}\ S\ Q\ \ .$$

Prove that

1. $(\{P\}\,S\,\{Q\} \wedge (P_0 \Rightarrow P)) \Rightarrow \{P_0\}\,S\,\{Q\}$.
2. $\{P\}\,S\,\{Q\} \wedge \{P\}\,S\,\{R\} \Leftrightarrow \{P\}\,S\,\{Q \wedge R\}$.

Solution:

1. We reason:

$$\begin{array}{ll}
 & \{P_0\}\,S\,\{Q\} \\
\Leftrightarrow & \{ \text{ definition of Hoare triple } \} \\
 & P_0 \Rightarrow \mathit{wp}\ S\ Q \\
\Leftarrow & \{ \text{ since } P_0 \Rightarrow P \ \} \\
 & P \Rightarrow \mathit{wp}\ S\ Q \\
\Leftrightarrow & \{ \text{ definition of Hoare triple } \} \\
 & \{P\}\,S\,\{Q\}\ \ .
\end{array}$$

2. We reason:

$$
\begin{array}{ll}
& \{P\}\, S\, \{Q \wedge R\} \\
\Leftrightarrow & \{ \text{definition of Hoare triple} \} \\
& P \Rightarrow wp\ S\ (Q \wedge R) \\
\Leftrightarrow & \{ \text{distributivity over conjunction} \} \\
& P \Rightarrow (wp\ S\ Q \wedge wp\ S\ R) \\
\Leftrightarrow & \{ \text{since } (P \Rightarrow (X \wedge Y)) \Leftrightarrow (P \Rightarrow X) \wedge (P \Rightarrow Y) \} \\
& (P \Rightarrow wp\ S\ Q) \wedge (P \Rightarrow wp\ S\ R) \\
\Leftrightarrow & \{ \text{definition of Hoare triple} \} \\
& \{P\}\, S\, \{Q\} \wedge \{P\}\, S\, \{R\} \ \ .
\end{array}
$$

10. Recall the weakest precondition of **if**:

$$wp\ (\mathbf{if}\ B_0 \rightarrow S_0 \vee B_1 \rightarrow S_1\ \mathbf{fi})\ Q\ =\ (B_0 \Rightarrow wp\ S_0\ Q) \wedge (B_1 \Rightarrow wp\ S_1\ Q) \wedge (B_0 \vee B_1)\ \ .$$

Prove that

$$
\begin{array}{l}
\{P\}\, \mathbf{if}\ B_0 \rightarrow S_0 \vee B_1 \rightarrow S_1\ \mathbf{fi}\, \{Q\}\ \Leftrightarrow \\
\quad \{P \wedge B_0\}\, S\, \{Q\}\ \wedge\ \{P \wedge B_1\}\, S\, \{Q\}\ \wedge\ (P \Rightarrow (B_0 \vee B_1))\ \ .
\end{array}
$$

Note: having proved so shows that the way we annotate **if** is correct:

$$
\begin{array}{l}
\{P\} \\
\mathbf{if}\ B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{Q\} \\
\ \mid\ B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{Q\} \\
\mathbf{fi} \\
\{Q\}\ \ .
\end{array}
$$

Solution: We reason:

$$
\begin{array}{ll}
& \{P\}\, \mathbf{if}\ B_0 \rightarrow S_0 \vee B_1 \rightarrow S_1\ \mathbf{fi}\, \{Q\} \\
\Leftrightarrow & \{ \text{definition of Hoare triple} \} \\
& P \Rightarrow wp\ (\mathbf{if}\ B_0 \rightarrow S_0 \vee B_1 \rightarrow S_1\ \mathbf{fi})\ Q \\
\Leftrightarrow & \{ \text{definition of } wp \} \\
& P \Rightarrow ((B_0 \Rightarrow wp\ S_0\ Q) \wedge (B_1 \Rightarrow wp\ S_1\ Q) \wedge (B_0 \vee B_1)) \\
\Leftrightarrow & \{ \text{since } (P \Rightarrow (Q \wedge R)) \Leftrightarrow (P \Rightarrow Q) \wedge (P \Rightarrow R) \} \\
& (P \Rightarrow (B_0 \Rightarrow wp\ S_0\ Q)) \wedge \\
& (P \Rightarrow (B_1 \Rightarrow wp\ S_1\ Q)) \wedge \\
& (P \Rightarrow (B_0 \vee B_1)) \\
& \\
\Leftrightarrow & \{ \text{since } (P \Rightarrow (Q \Rightarrow R)) \Leftrightarrow ((P \wedge Q) \Rightarrow R) \} \\
& ((P \wedge B_0) \Rightarrow wp\ S_0\ Q) \wedge \\
& ((P \wedge B_1) \Rightarrow wp\ S_1\ Q) \wedge \\
& (P \Rightarrow (B_0 \vee B_1)) \\
\Leftrightarrow & \{ \text{definition of Hoare triple} \} \\
& \{P \wedge B_0\}\, S_0\, \{Q\} \wedge \\
& \{P \wedge B_1\}\, S_1\, \{Q\} \wedge \\
& (P \Rightarrow (B_0 \vee B_1))\ \ .
\end{array}
$$

11. Recall that $wp\ S\ Q$ stands for "the weakest precondition for program S to terminate in a state satisfying Q". What programs S, if any, satisfy each of the following conditions?

    1. *wp S True* = *True*.
    2. *wp S True* = *False*.
    3. *wp S False* = *True*.
    4. *wp S False* = *False*.

Solution:

1. *wp S True* = *True*: S is a program that always terminates.
2. *wp S True* = *False*: S is a program that never terminates.
3. *wp S False* = *True*: there is no such a program S.
4. *wp S False* = *False*: S can be any program.