Introduction to Software Verification 236342, Homework 1

Yosef Goren, Andrew Elashkin

January 15, 2023

1

A. Correct. Since the precondition is false, the postcondition is 'always' satisfied (since it is never tested).

B. Incorrect. Counterexample $x = -100, y = -99$:

- $l_0, -100, -99$
- $l_1, -100, -99$
- $l_2, -100, -99$
- $l_3, -1, -99$
- $l_*, -1, -99$

As can be seen, precondition is satisfied and postcondition is not.

C. Correct. The postcondition is true, so regardless of anything else, for every input selection it will be evaluated as true (the program does not even have to terminate either).

D. Incorrect. Counterexample $x = 1, y = 9$:

- $l_0, 1, 9$
- $l_1, 1, 9$
- $l_2, -8, 9$
- $l_3, -8, 9$
- $l_*, -8, 9$

Postcondition is false, so it is not satisfied.

E. Incorrect. Counterexample $x = 1, y = 3$:

- $l_0, 1, 3$
- $l_1, 1, 3$

- $l_2, 1, 3$
- $l_3, -2, 3$
- $l_4, -2, 3$
- $l_1, -2, -3$
- $l_2, -2, -3$
- $l_3, -2, -3$
- $l_4, -2, -3$
- $l_1, -2, 3$
- $l_2, -2, 3$
- $l_3, -5, 3$
- $l_4, -5, 3$

By running this example we see that the program gets stuck in a loop at labels l_1, l_2, l_3, l_4. And each 4 iterations result with the state being the same as the initial state at l_1. Since this is a total correctness condition on the specification, the correctness is contradicted by the program failing to terminate.

F. Correct. We now require partial correctness, so we check postcondition only for the cases that terminate. That means, that we won't have a problem like we had in case E, since such an input won't be considered, and all the cases that satisfy $x^2 < y^2$ and terminate hold $x^2 < y^2$ postcondition.

G. Incorrect. Since we require total correctness, every computation that satisfies the precondition $z = 5$ should terminate. However, for $x = 1, y = 2, z = 5$ the program never terminates, as we showed in E.

H. Correct. The program does not change the value of z, so every run that terminates and satisfies the precondition $z = 5$ will also hold the postcondition.

I. Correct. The value of $|y|$ never changes during the execution, so the condition will hold for every run that terminates.

J. Incorrect (But correct for $x, y \in \mathbb{N}$). If we assume that the input can be rational numbers, then $x = 7.5, y = 7.5$ will fail the postcondition. However, the assumption holds for the integers.

K. Incorrect. $x = 1, y = 2$ satisfy the precondition, but the program never terminates, as we showed in E.

L. Correct. We check for partial correctness, so we want the program that satisfies the precondition to stuck in infinite loop. This is exactly what happens with all the numbers satisfying the precondition, similar to the example in E.

2

A. Specification is described by the tuple $< q_1, q_2 >$
Where precondition q_1 is

$$r = T, r \in \mathbb{N}_0, k \in \mathbb{N}_0$$

Postcondition q_2 is:

$$k \in \mathbb{N}_0 \wedge (r = 1 \wedge (T > f_k) \wedge \mathcal{F}_k) \vee (r = 0 \wedge (T = f_k) \wedge \mathcal{F}_k) \vee (r = -1 \wedge (T < F_k) \wedge \mathcal{F}_k)$$

$$\mathcal{F}_k = \exists f_0, f_1, ..., f_k (f_0 = 0 \wedge f_1 = 1 \wedge (\bigwedge_{l=2}^{k} f_l = f_{l-1} + f_{l-2})$$

B. Specification is described by the tuple $< q_1, q_2 >$
Precondition q_1 is

$$x = X, x \in \mathbb{N}_0$$

Postcondition q_2 is:

$$\exists Y \in \mathbb{N}_0 \wedge \exists r \in \mathbb{N}_0 \wedge (k \in \mathbb{N}_0 \wedge Fib_k(r)) \wedge Y * r \leq x \wedge (Y+1) * r > x \wedge (y = (Y+1) * r \wedge x = X)$$

3

A. To enforce the program to not finish on a specific set on inputs, we can require that if said inputs have been given and the program finishes - the postcondition fails:

$$\{\forall p \in \mathbb{P}, x \neq p^2\} P \{false\}$$

We can also represent $p \in \mathbb{P}$ more explicitly:

$$p \in \mathbb{P} \Leftrightarrow (\forall x, \forall y, (x \cdot y = p) \rightarrow (|x| = 1 \wedge |y| = 1))$$

(Here we assume the variables are defined over the integers).
So our full specification is:

$$\{\forall p, ((\forall x, \forall y, (x \cdot y = p) \rightarrow (|x| = 1) \wedge |y| = 1) \rightarrow (x \neq p^2))\} P \{false\}$$

B. To require that for a set of inputs a program finishes. we can use the precondition to apply the condition only to the relivant set and use total correctness to require the program to halt on these inputs:

$$< gcd(x, y) = 1 > P < true >$$

Similarly, we can express $gcd(x, y) = 1$ with:

$$(gcd(x, y) = 1) \Leftrightarrow (\forall z, (\exists n, n \cdot z = x) \rightarrow (\forall n, n \cdot z \neq y))$$

And get the full specification:

$$< \forall x, \forall y, (\forall z, (\exists n, n \cdot z = x) \rightarrow (\forall n, n \cdot z \neq y)) > P < true >$$

4

- The reachability condition:

$$R'^0(\bar{x}) = true$$

$$R'^{k+1}(\bar{x}) = \begin{cases} R'^k(\bar{x}) & \text{if } l_{i_k} \in \{[start], [end], [\bar{y} := \bar{e}]\} \\ R'^k(\bar{x}) \wedge B(\bar{x}) & \text{if } l_{i_k} = [B(\bar{X})] \wedge (l_{i_k} \to^T l_{i_{k+1}}) \\ R'^k(\bar{x}) \wedge \neg B(\bar{x}) & \text{if } l_{i_k} = [B(\bar{X})] \wedge (l_{i_k} \to^F l_{i_{k+1}}) \end{cases}$$

- The state transformer:

$$T'^0(\bar{x}) = \bar{x}$$

$$T'^{k+1}(\bar{x}) = \begin{cases} T'^k(\bar{x}) & \text{if } l_{i_k} \in \{[start], [end], [B(\bar{x})]\} \\ T'^k(\bar{x})[\bar{y} \leftarrow \bar{e}] & \text{if } l_{i_k} = [\bar{y} := \bar{e}] \end{cases}$$