Introduction to Software Verification 236342, Homework 1

Yosef Goren, Andrew Elashkin 921130571

November 14, 2022

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 correcness condition on the specification, the correctness is contredicted by the program failing to terminate.

F. Correct.

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 F.

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. $x = 7.5, y = 7.5$ will fail the postcondition.

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

L.

2

A. Precondition q_1 is

$$r = T, r \in \mathbb{N}_0, k \in \mathbb{N}_0, n \in \mathbb{N}_0, F_n = F_{n-1} + F_{n-2}, F_0 = 0, F_1 = 1$$

Postcondition q_2 is:

$$(r = 1 \land T > F_k) \oplus (r = 0 \land T = F_k) \oplus (r = -1 \land T < F_k)$$

B. Precondition q_1 is

$$x = X, r \in \mathbb{N}_0, k \in \mathbb{N}_0, n \in \mathbb{N}_0, F_n = F_{n-1} + F_{n-2}, F_0 = 0, F_1 = 1$$

Postcondition q_2 is:

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

3

A. To enforce the program to not finish on a spesific 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:

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

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

- The state transformer:

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

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