Ethan Booker

Dr. Mao Zheng

CS743 Software Verification and Validation

Assignment 6

1)

Let

P - the precondition for the fragment of code (same for both)

Q - the postcondition for the fragment of code (same for both)

C1 - a == b

C2 - b == c

S1 - statement: System.out.print (" All the three are equal");

S2 - statement: System.out.print (" Two of them are equal");

S3 - statement: System.out.print (" All the three are completely different");

{P ∧ C1} inner block {Q} {P ∧ ¬ C1}S2{Q}

Fragment 1:

{ P ∧ C1 ∧ C2 } S1 { Q } (1)

{ P ∧ (C1 v C2) } S2 { Q } (2)

{ P ∧ ¬ (C1 ^ C2) } S3 { Q } (3)

{ P ∧ C1 ∧ C2 } S1 { Q } (4)

{ P ∧ C1 ∧ ¬C2 } S2 { Q } (5)

{ P ∧ ¬C1 ^ C2 } S2 { Q } (6)

{ P ∧ ¬ (C1 ∧ C2) } S3 { Q } (7)

Case 1: C1 and C2 are both true

By equation (1), statement S1 will be executed resulting in post condition Q

By equation (4), statement S1 will be executed resulting in post condition Q

Case 2: C1 and C2 are both false

By equation (3), statement S3 will be executed resulting in post condition Q

By equation (7), statement S3 will be executed resulting in post condition Q

Case 3: C1 is true and C2 is false

By equation (2), statement S2 will be executed resulting in post condition Q

By equation (5), statement S2 will be executed resulting in post condition Q

Case 4: C1 is false and C2 is true

By equation (2), statement S2 will be executed resulting in post condition Q

By equation (6), statement S2 will be executed resulting in post condition Q

We have thus proved that both code fragments result in the same post-condition for the same precondition and the conditions in the if statements and hence they are equivalent.

2) (both K and n are odd at any time) OR (both K and n are even at any time)

* K = blue marbles, M = red marbles, n = max blue marbles at any time, T = K + M

Before entering the loop:

Given that we do not have unlimited blue marbles, we can assume that there is a finite limit of blue marbles at any given time, thus n >= 0, and n can never increase. Since there are K number of initial blue marbles, n >= K must be satisfied. Because there is no procedure to ever add another blue marble then n = K is initially true prior to the loop, regardless of K being even or odd.

Even if the loop does not run, this condition still remains true: n = K prior to the loop [which means they are either both even or both odd]

During the loop:

Let the initial number of total marbles (T = K + M, and n = K) in the jar be > 1.

In each loop, 2 marbles are selected from the jar resulting in one of the two possibilities:

1. If both marbles are the same color then the case is:

We remove 2 blues: K = even - 2 == even | K = odd - 2 == odd → T = T - 2

Or

We remove 2 reds and add 1 red: M = M(even) - 1 = | M = odd - 1 → T = T - 1

This means that K could be 2 less or M could be 1 less, either way, K and n are still both even or odd as they were initially.

1. The other possibility is selecting 1 of each marble: 1 K and 1 M.

This possibility results in removing the red and keeping the blue thus

Remove 1 red: M = M - 1

Blue marbles: K = K, thus K and n still remain either even or odd depending on their initial state.

T = T - 1

So after such iterations, the loop will stop as T will eventually fail the while condition. At any given time however, the blue marbles will remain either even or odd depending on their initial state. K will change by either a 0 or 2 each loop, while n will remain the same thus concluding that any time during the loop K and n are even or K and n are odd.

After the loop:

At this point the number of marbles in the jar will be a 0 or 1. Regardless of the total marbles being a 0 or 1, since the loop kept n and K either both even or both odd at any given time, n and K will still remain to be the same status in terms of even or odds after the loop.

Case 1: K is even

* Initially K = n, but during the loop K = K - 2 ... until K = 0. Since an even number - 2 always results in an even number, n and K are even at any given time in the loop. Once it terminates, K has to be equal to 0 since the loop will only terminate when T <= 1. Since even can never be odd and even - 2 = even, K must be 0 after termination. Since K and n are even at all states, then the even invariance is proven true.

Case 2: K is odd

* Initially, K = n, but during the loop K = K - 2 … until K = 1. Since an odd number - 2 always results in an odd number, n and K are odd at any given time in the loop. Once it terminates, K has to be equal to 1 since the loop will only terminate when T <= 1. Since odd can never be even and odd - 2 = odd, K must be 1 after termination. Since K and n are odd at all states, then the odd invariance is proven true.

Even if initially there are no blue marbles or 1 marble, then n must be n = K as that is the initial value set which means that n = K and n and K are either both even or both odd.

Since both even and odd invariance where both K and n are odd at any time or both K and n are even at any time is indeed an invariant for the while loop in the above algorithm.