Naga Jayanth Chennupati

21002083

njchennu@uwaterloo.ca

1)

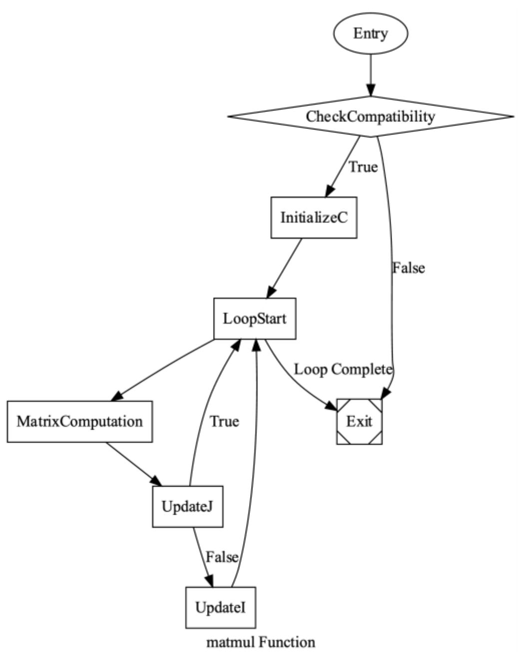
(a) To identify a test case that does not execute the fault, we can choose a test case where the dimensions of matrices **a** and **b** are compatible. For example
a = [[1, 2, 3], [4, 5, 6]]
b = [[7, 8], [9, 10], [11, 12]]

(b) To identify a test case that executes the fault but does not cause an error, we can choose a test case where the dimensions of matrices a and b are incompatible, but the error is not raised. For example
a = [[1, 2, 3], [3, 4, 5]]
b = [[7, 8], [6, 7]]

(c) To identify a test case that results in an error but not in a failure, we need a test case where the error is caught and handled gracefully. This means that the code raises an exception but does not crash the program.
a = [[5, 7], [8, 21]]
b = [[8], [4]]

(d) The first error occurs when the program is at line 9 where p! =p1. Here p is 2 and p2 is 1. The matrices are compatible and can be multiplied with each other. The expected output is [68, 148]. The output from the program is Value Error ("Incompatible dimensions").

(e)



Entry: Line 7 and 8

Check Compatibility: Line 9
False: Skip to **Exit Node**

Initialize C: Line 11

Loop Start: Line 12

Loop Start: Line 13

Matrix Computation: Line 14

Update J: Update loop variable (j)

Update I: Update loop. Variable (i)

Exit: Line 15 and terminates the code if there are any errors

2)
(a)

(2)

(a)
```
Class RepeatUntilStmt:
    def __init__(self, body condition):
        self.body = body #S, a statement
        self.condition = condition #b, a boolean exp
```

(b)

(b) The semantics of the repeat-until loop is that in each iteration

→ S is executed
→ b is evaluated
→ If the current value of b is false, the loop continues to the next iteration
→ If the current value of b is true, the loop terminates (and statements following the loop are executed).

$$\frac{\langle S, q\rangle \Downarrow q' \quad \langle b, q'\rangle \Downarrow true}{\langle repeat\ S\ until\ b, q\rangle \Downarrow q'} \quad \Bigg\} \text{ consider when b is true}$$

$$\underbrace{\frac{\langle S, q\rangle \Downarrow \langle b, q'\rangle \Downarrow false \ \langle repeat\ S\ until\ b, q'\rangle\ q''}{\langle repeat\ S\ until\ b, q\rangle \Downarrow q''}}_{\text{consider when b is false}}$$

(c)

(C) Show that the following judgement is valid

$$\langle x:=2;\ \text{repeat } x:=x-1 \text{ until } x\le 0 [\,]\rangle \Downarrow [x:=0]$$

Prove

$$\langle x:=2;\ \text{repeat } x:=x-1 \text{ until } x\le 0, [\,]\rangle \Downarrow [x:=0]$$

$$\overline{\langle x;[x:2]\rangle \Downarrow 2} \quad \overline{\langle 1,(x=2]\rangle \Downarrow 1} \qquad \overline{\langle x,[x:=1]\rangle \Downarrow 1} \quad \overline{\langle 1,[x:=1]\rangle 1}$$

$$\overline{\langle x,[x:2]\rangle \Downarrow 2} \quad \overline{\langle x-1,[x:2]\rangle \Downarrow 1} \qquad \overline{x,[x:=1] \Downarrow 1} \quad \overline{\langle x-1,(x:=1]\rangle \Downarrow 0}$$

$$\overline{\langle x, q\rangle \Downarrow 2} \qquad \overline{\langle x:=x-1,(x:=2]\rangle \Downarrow [x:=1]} . \quad \overline{\langle x\le 0,[x:=1]\rangle \text{ false } \langle \text{repeat } x:x-1 \text{ until } x\le 0, [x:=1]\rangle \Downarrow (x:=0]}$$

$$\overline{\langle x:=2,[\,]\rangle \Downarrow [x:=2]} \quad \overline{\langle \text{repeat } x:x-1 \text{ until } x\le 0,(x:2]\rangle \Downarrow [x:=0]}$$

$$\overline{\langle x:=2;\ \text{repeat } x:=x-1 \text{ until } x\le 0,[\,]\rangle \Downarrow [x:=0]}$$

(d)

(d) Prove that the statement

repeat S until b

is symmetrically equivalent to

S; if b then skip else (repeat S until b)

The semantics of repeat until loop

When the condition is true:

$$\frac{\langle S,q\rangle \Downarrow q'' \quad \langle b, q''\rangle \Downarrow \text{true}}{\langle \text{repeat } S \text{ until } b, q\rangle \Downarrow q''} \quad \text{—①}$$

When the condition is false:

$$\frac{\langle S,q\rangle \Downarrow q'' \langle b,q''\rangle \text{ false } \langle \text{repeat } S \text{ until } b,q''\rangle \Downarrow q'}{\langle \text{repeat } S \text{ until } b,q\rangle \Downarrow q'} \quad \text{—②}$$

Consider this as the first statement

It has a derivative tree, denoted as T. Depending on the condition b is true or false it can have two forms. So, for the false condition we can rewrite ② as follows

$$\frac{T_1 \qquad T_2}{\langle \text{repeat } S \text{ until } b,q\rangle \Downarrow q'}$$

$\langle S,q\rangle \Downarrow q' \longrightarrow T_1$

$\langle \text{repeat } S \text{ until } b, q'\rangle \Downarrow q'' \longrightarrow T_2$

Semantics S; if b then skip else (repeat S until b)

when the condition is true

$$\frac{\langle S,q\rangle \Downarrow q'' \ \langle b,q''\rangle \Downarrow \text{true} \ \langle \text{skip}.q''\rangle q''}{\langle S; \text{if } b \text{ then skip else (repeat } S \text{ until } b),q\rangle \Downarrow q''} \quad ③$$

In ③ we consider as two parts which is denoted as S_1 and S_2.

In the state q, first S get executed and gets state q''. The code goes to if condition. semantics for if when the condition is true is given below

$$\frac{\langle S_1,q\rangle q''}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, q\rangle \Downarrow q'}$$

The skip statement is being executed. The semantic for skip $\overline{\langle \text{skip}, q\rangle \Downarrow q}$

when the condition is false:

$$\frac{\langle S_2,q\rangle \Downarrow q'' \langle b,q''\rangle \Downarrow \text{false} \ \overline{\langle \text{repeat } S \text{ until } b,q''\rangle \Downarrow q'}}{\langle S; \text{if } b \text{ then skip else (repeat } S \text{ until } b),q\rangle \Downarrow q'} \quad ④$$

In semantics ④ we consider as two parts which is denoted as S_1 and S_2

S_1 is executed in state q and then gets state q''.

semantics for if cond^n when false

$$\frac{\langle S_2,q\rangle \Downarrow q'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2,q\rangle \Downarrow q'}$$

repeat S until b is executed in state q'' and then to state q' when using above rule for S_2.

Hence, proved.

3)
(a)

(b)

- **<u>Node Coverage (TR_{NC}):</u>**

Node coverage aims to ensure that each node in the CFG is visited at least once during testing.

TR_{NC}: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

All nodes in the CFG are reachable, so there are no infeasible test requirements for node coverage.

- **<u>Edge Coverage (TR_{EC}):</u>**

Edge coverage focuses on traversing every edge in the CFG at least once.

TR_{EC}: {(1, 2), (2, 3), (3, 4), (4, 5), (4, 6), (4, 7), (5, 3), (6, 3), (7, 8), (8, 3), (3,9), (9,10), (10, 11)}

Infeasible test requirements for edge coverage:
(5, 3): This edge is infeasible because it represents a loop back to node 3 from node 5, but this loop is controlled by the loop structure in the code and cannot be directly covered in a single test case. Achieving this edge coverage would require multiple iterations of the loop, which is not a typical way to measure edge coverage.

- **<u>Edge-Pair Coverage (TR_{EPC}):</u>**

Edge-pair coverage involves testing pairs of edges in the CFG to ensure that specific paths and transitions are covered.

TR_{EPC}: {(1,2,3), (2,3,9), (2,3,4), (2,3,7), (4,5,3), (4,6,3), (4,7,6), (4,7,8), (5,3,9), (6,3,9), (7,6,3), (7,8,3), (8,3,9), (9,10,11)}