ECE 653, Assignment 1

Weiqiang Yu
w8yu@uwaterloo.ca
21020022

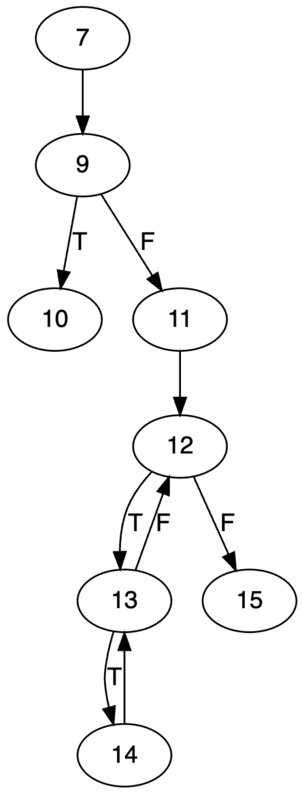
September 2023

1 Q1 Solution

(a) When the input is invalid and raise an exception. For example, input a is None.
(b) When matrix b is a square matrix, no error will occur. For example, b is a 3 * 3 square matrix.
(c) We can find two matrices a and b such that len(a[0]) != len(b[0]). For example, when a = [[0]], b = [[3,2,1], [4,5,6]].
(d) The first error state is the following:

$$a = [[5, 7], [8, 21]]$$
$$b = [[8], [4]$$
$$n = 2$$
$$p = 2$$
$$q = 2$$
$$p1 = 1$$
$$c = undefined$$
$$i = undefined$$
$$j = undefined$$
$$k = undefined$$
$$pc = at\ line\ 8$$

(e)



2 Q2 Solution

(a)

```python
class Ast(object):
    """Base class of AST hierarchy"""
    pass

class Stmt(Ast):
    """A signle statement"""
    pass

class AsgnStmt(Stmt):
    """An assignment statement"""
    def __init__(self, lhs, rhs):
        self.lhs = lhs
        self.rhs = rhs

class IfStmt(Stmt):
    def __init__(self, cond, then_stmt, else_stmt=None):
        self.cond = cond
        self.then_stmt = then_stmt
        self.else_stmt = else_stmt

class RepeatUntilStmt(Stmt):
    def __init__(self, cond, repeat_stmt):
        self.cond = cond
        self.repeat_stmt = repeat_stmt
```

(b)

$$\frac{< s, q >\Downarrow q' \quad < b, q' >\Downarrow true}{< repeat\ s\ until\ b, q >\Downarrow q'}$$

$$\frac{< s, q >\Downarrow q'' \quad < b, q'' >\Downarrow false \quad < repeat\ s\ until\ b, q'' >\Downarrow q'}{< repeat\ s\ until\ b, q >\Downarrow q'}$$

(c)

$$\frac{< x := 2, [] >\Downarrow [x := 2] \quad \dfrac{< x := x-1, [x := 2] >\Downarrow [x := 1] \quad < x \leq 0, [x := 1] >\Downarrow false \quad \dfrac{< x : x-1, [x := 1] >\Downarrow [x := 0] \quad < x \leq 0, [x := 0] >\Downarrow true}{< repeat\ x := x-1\ until\ x \leq 0, [x := 1] >\Downarrow [x := 0]}}{< repeat\ x := x-1\ until\ x \leq 0, [x := 2] >\Downarrow [x := 0]}}{< x := 2;\ repeat\ x := x-1\ until\ x \leq 0, [] >\Downarrow [x := 0]}$$

(d) The proof consists of two parts. We first need to prove that:
If

$$< \text{repeat } S \text{ until } b, q >\Downarrow q' \tag{1}$$

Then

$$< S; \text{if b then skip else (repeat S until b), q} >\Downarrow q' \tag{2}$$

Assume (1) holds and we get state q'' after executing statement S at state q, we know that there exists a derivation tree T for it. Now we need to consider two cases:
case one: If $< b, q'' >\Downarrow$ false, then the derivation tree T has the form:

$$\frac{T_1}{< \text{repeat } S \text{ until } b, q >\Downarrow q'}$$

Where T_1 *is* $< \text{repeat } S \text{ until } b, q'' >\Downarrow q'$.
Using the above premise, we can use the inference rules to construct the derivation tree:

$$\frac{< S, q >\Downarrow q'' \quad \dfrac{< b, q'' >\Downarrow false \quad T_1}{< \text{if b then skip else (repeat S until b)}, q'' >\Downarrow q'}}{< S; \text{if b then skip else (repeat S until b), q} >\Downarrow q'}$$

thereby showing that (1) holds.
In another case, if $< b, q'' >\Downarrow$ true, then we have $q' = q''$, which means that:

$$< \text{repeat } S \text{ until } b, q >\Downarrow q''$$

Since $q' = q''$, we have:

$$\frac{< S, q >\Downarrow q'' \quad \dfrac{< b, q'' >\Downarrow true \quad \overline{< skip, q'' >\Downarrow q''}}{< \text{if b then skip else (repeat S until b)}, q'' >\Downarrow q''}}{< S; \text{if b then skip else (repeat S until b), q} >\Downarrow q''}$$

This completes the first part of the proof.
Next, we need to prove the other way around, which in this case is: If

$$< S; \text{if b then skip else (repeat S until b), q} > \Downarrow q' \tag{3}$$

Then

$$< \text{repeat } S \text{ until } b, q > \Downarrow q' \tag{4}$$

Assume (3) holds and we get state q'' after executing statement S at state q, we know that there exists a derivation tree T for it. Similarly, we shall consider two following cases:
case one: If $< b, q'' > \Downarrow$ false, then the derivation tree T has the form:

$$\frac{T_1}{< S; \text{if b then skip else (repeat S until b), q} > \Downarrow q'}$$

Where $T_1\ is\ < \text{repeat } S \text{ until } b, q'' > \Downarrow q'$.
Using the above premise, we can use the inference rules to construct the derivation tree:

$$\frac{< S, q > \Downarrow q'' \quad < b, q'' > \Downarrow false \quad T_1}{< \text{repeat } S \text{ until } b, q > \Downarrow q'}$$

thereby showing that (3) holds.
In another case, if $< b, q'' > \Downarrow$ true, then according to the inference rule: $\overline{< skip, q'' > \Downarrow q''}$, we have $q' = q''$, which means that:

$$< S; \text{if b then skip else (repeat S until b), q} > \Downarrow q''$$

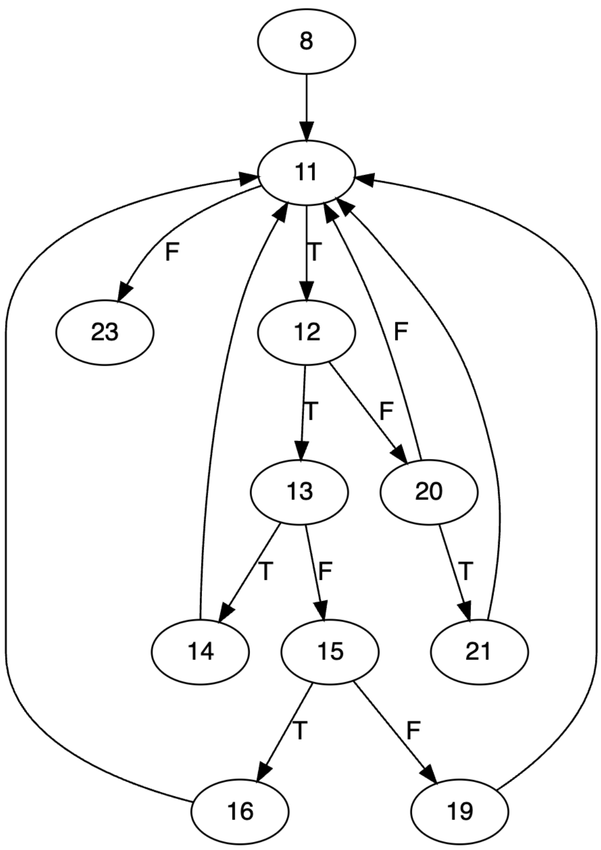
Since $q' = q''$, we have:

$$\frac{< S, q > \Downarrow q'' \quad < b, q'' > \Downarrow true}{< \text{repeat } S \text{ until } b, q > \Downarrow q''}$$

This completes the proof.

3 Q3 Solution

(a)



(b)
$TR_{NC} = \{8, 11, 12, 13, 14, 15, 16, 19, 20, 21, 23\}$
$TR_{EC} = \{[8, 11], [11, 12], [11, 23], [12, 13], [12, 20], [13, 14], [13, 15], [14, 11], [15, 16], [15, 19], [16, 11], [19, 11], [20, 21], [20, 11], [21, 11]\}$
The infeasible test requirements include: $TR_{infeasibleEC} = \{[20, 11]\}$, because the program only has two states, state 0 and state 1. Therefore, when the program counter in at line 20, it is infeasible to find a state other than 1, which means that it will always execute line 21 rather than jumping to line 11.
$TR_{EPC} = \{[8, 11, 12], [8, 11, 23], [11, 12, 13], [11, 12, 20], [12, 13, 14], [12, 13, 15], [12, 20, 21], [12, 20, 11], [13, 14, 11], [13, 15, 16], [13, 15, 19], [14, 11, 23], [14, 11, 12], [15, 16, 11], [15, 19, 11], [16, 11, 23], [16, 11, 12], [19, 11, 23], [19, 11, 12], [20, 21, 11], [20, 11, 23], [20, 11, 12], [21, 11, 23], [21, 11, 12]\}$
The infeasible test requirements include $TR_{EC} = \{[12, 20, 11], [20, 11, 23], [20, 11, 12]\}$.
because we cannot visit edge [20, 11] as discussed before.
(c)It is impossible to find such a case that achieves complete node coverage but not edge coverage. The reason is that if we exclude the infeasible edges, the majority of nodes only have one incoming edge, which means we have to test these edges. We also need to cover edge 8-11 since it is the beginning edge, and all the edges point to node 11, since it is the only node that can visit node 23 and terminate. In this case, when we test all the nodes, all the edges are also tested as well.

4 Q4 Solution

(a) int.py:75 is not covered because WHILE only has $' <=' |' <' |' =' |' >=' |' >'$ relational operators.
int.py:179 - 197 are not covered because we did not execute the int.py as the main program.
parser.py:481 - 482 are not covered. I tried to add `\n` in the context, but it did not work. Not sure how to cover the newline here
parser.py:29 - 41 are not covered because we did not use WhileLangBuffer in this context
parser.py:487 - 586 are not covered because we used the self-defined semantics rather than the generated semantics from the parser.
parser.py:590 - 609 are not covered because we did not execute the parser.py as the main program.
(b) int.py:72 is not covered because WHILE only has $' <=' |' <' |' =' |' >=' |' >'$ relational operators.
int.py:90 is not covered because WHILE only has $'not'|'and'|'or'$ logical operators.
int.py:111 is not covered because WHILE only has $' +' |' -' |' *' |'/'$ arithmetic operators.
int.py:196 is not covered because we did not execute the int.py as the main program.
parser.py:603 is not covered because we did not execute the parser.py as the main program.
The interpreter is sound and complete in this context!