

LAB ASSIGNMENT-1

Submitted for

COMPILER CONSTRUCTION (UCS802)

Submitted by

Aryan Shanker Saxena

102103613

4 COE 22

Submitted to

Mr. Rajesh



Computer Science and Engineering Department

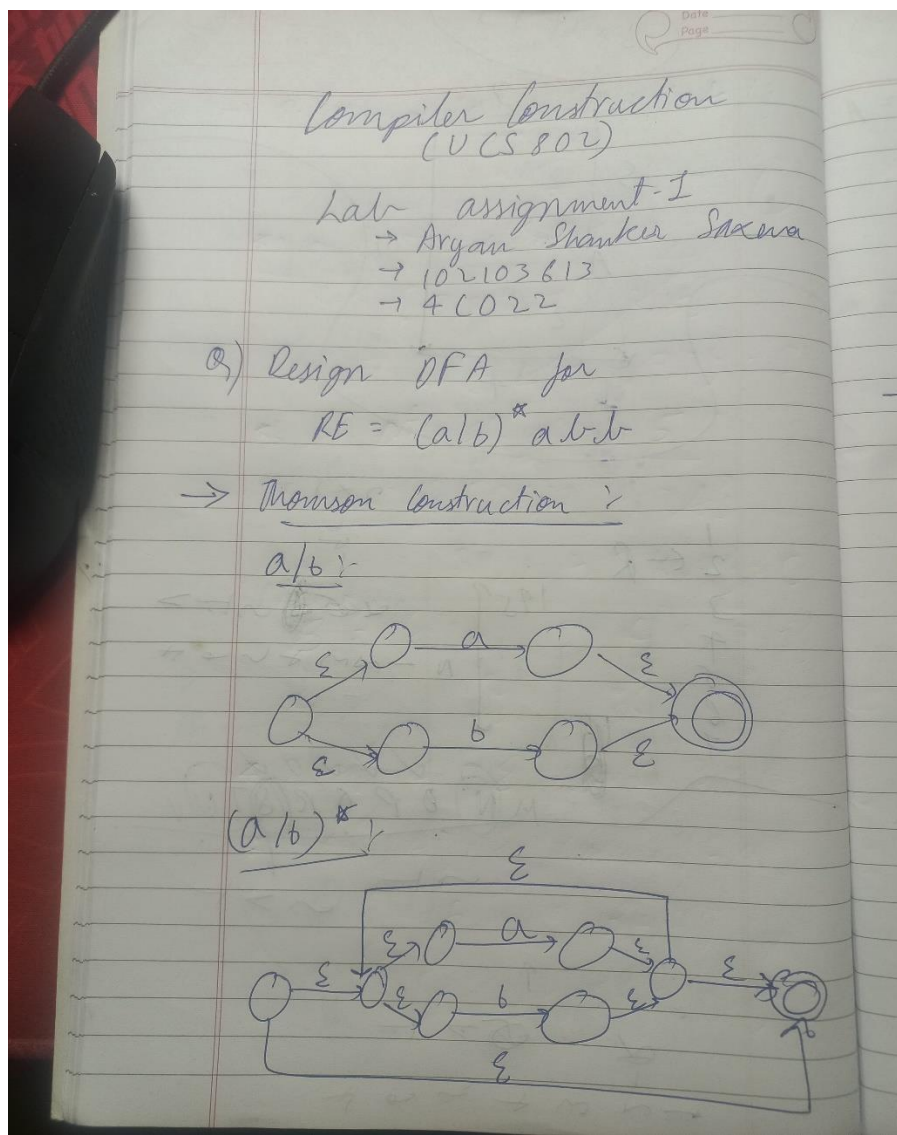
Thapar Institute of Engineering and Technology, Patiala

QUESTION:

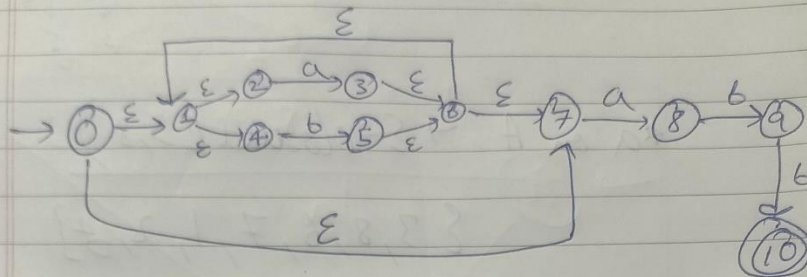
Design a Minimized DFA for the Regular Expression $(a/b)^*abb$ i.e. All strings ending with abb .

Note: THE CODE IS TYPE WRITTEN IT IS JUST THE BACKGROUND OF VSCode THAT MIGHT MAKE IT LOOK LIKE AN IMAGE BUT IT IS PURE TEXT

SOLVED ON PAPER:



$(a/b)^* a b b$



→ DFA Subset Construction:

State	a	b	ϵ_1	ϵ_2
0	—	—	1	7
1	—	—	2	4
2	3	—	—	—
3	—	—	6	—
4	—	5	—	—
5	—	—	6	—
6	—	—	7	1
7	8	—	—	—
8	—	9	—	—
9	—	10	—	—
10	—	—	—	—

classmate
Date _____
Page _____

$$\rightarrow \varepsilon\text{-closure}(0) = \{0, 1, 2, 4, 7\} = A$$

$$\begin{array}{ccc} \downarrow a & \downarrow b & \downarrow a \\ 3 & 5 & 8 \end{array}$$

* a on A $\rightarrow \varepsilon\text{-closure}(3, 8) =$

$$\{3, 8, 6, 7, 1, 2, 4\}$$

$$\rightarrow \{1, 2, 3, 4, 6, 7, 8\} = B$$

* b on A $\rightarrow \varepsilon\text{-closure}(5) =$

$$\{5, 6, 7, 1, 2, 4\}$$

$$= \{1, 2, 4, 5, 6, 7\} = C$$

Now,

$$B = \{1, 2, 3, 4, 6, 7, 8\}$$

$$\begin{array}{ccc} \downarrow a & \downarrow b & \downarrow a \downarrow b \\ 3 & 5 & 8 \ 9 \end{array}$$

* a on B $\rightarrow \varepsilon\text{-closure}(3, 8) = B$

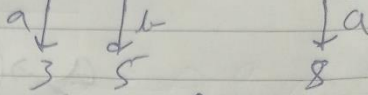
* b on B $\rightarrow \varepsilon\text{-closure}(5, 9) =$

$$\{5, 9, 10, 2, 4, 7\}$$

$$\rightarrow \{1, 2, 4, 5, 6, 7, 9\} = D$$

Now,

$$C = \{1, 2, 4, 5, 6, 7\}$$

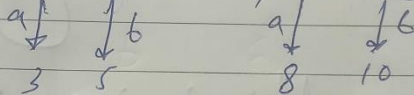


$$* a \text{ on } C \Rightarrow \varepsilon\text{-closure}(3, 8) = B$$

$$* b \text{ on } C \Rightarrow \varepsilon\text{-closure}(5) = C$$

Then,

$$D = \{1, 2, 4, 5, 6, 7, 9\}$$



$$* a \text{ on } D \Rightarrow \varepsilon\text{-closure}(3, 8) = B$$

$$* b \text{ on } D \Rightarrow \varepsilon\text{-closure}(5, 10)$$

$$\Rightarrow \{5, 6, 7, 1, 2, 4, 10\}$$

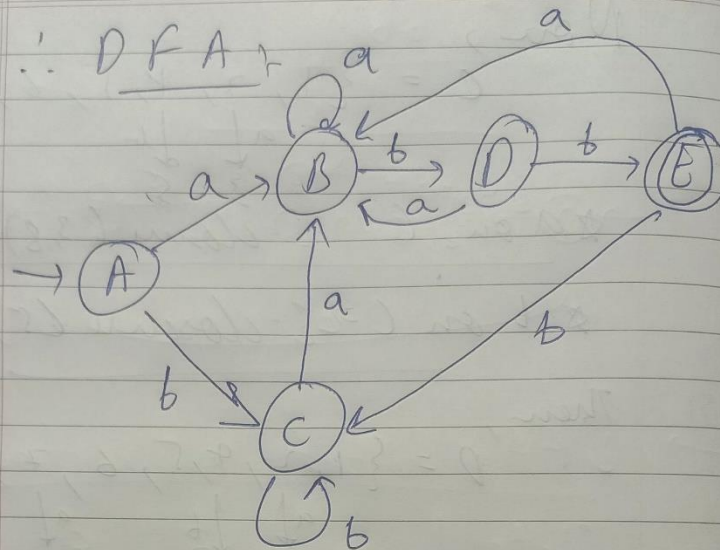
$$\Rightarrow \{1, 2, 4, 5, 6, 7, 10\} = E$$

Finally,

$$E = \{1, 2, 4, 5, 6, 7, 10\}$$

$$a \text{ on } E \Rightarrow \varepsilon\text{-closure}(3, 8) = B$$

$$b \text{ on } E \Rightarrow \varepsilon\text{-closure}(5) = C$$



→ Minimize DFA

State	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

$$\pi_0 = \{ \{E\}, \{A, B, C, D\} \}$$

$$\pi_1 = \{ \{E\}, \{D\}, \{A, B, C\} \}$$

$$\pi_2 = \{ \{E\}, \{D\}, \{B\}, \{A, C\} \}$$

$$\pi_3 = \{ \{E\}, \{D\}, \{B\}, \{C\}, \{A\} \}$$

PYTHON CODE:

```
class NFA:

    def __init__(self):

        self.states = {i: {'a': None, 'b': None, 'ε1': None, 'ε2': None} for i in range(11)}

    def add_transition(self, state, symbol, next_state):

        if symbol in self.states[state]:

            self.states[state][symbol] = next_state

    def print_nfa(self):

        print("State | a | b | ε1 | ε2")

        print("-----")

        for state, transitions in self.states.items():

            print(f" {state} | {transitions['a']} | {transitions['b']} | {transitions['ε1']} | {transitions['ε2']}")

# Initializing the NFA with correct transitions

nfa = NFA()

nfa.add_transition(0, 'ε1', 1)

nfa.add_transition(0, 'ε2', 7)

nfa.add_transition(1, 'ε1', 2)

nfa.add_transition(1, 'ε2', 4)

nfa.add_transition(2, 'a', 3)

nfa.add_transition(3, 'ε1', 6)

nfa.add_transition(4, 'b', 5)

nfa.add_transition(5, 'ε1', 6)

nfa.add_transition(6, 'ε1', 7)

nfa.add_transition(6, 'ε2', 1)

nfa.add_transition(7, 'a', 8)

nfa.add_transition(8, 'b', 9)

nfa.add_transition(9, 'b', 10)

print("Regular Expression: (a/b)*abb")
```

```

print("")

print("Submitted by: Aryan Shanker Saxena (102103613)")

print("")

nfa.print_nfa()

print("")

# epsilon_closure function to use the nfa.states dictionary
def epsilon_closure(nfa_states, state):

    stack = [state]

    closure = set(stack)

    while stack:

        current = stack.pop()

        for next_state in [nfa_states[current]['ε1'], nfa_states[current]['ε2']]:

            if next_state and next_state not in closure:

                closure.add(next_state)

                stack.append(next_state)

    return closure

epsilon_closure_set = epsilon_closure(nfa.states, 0)

print("ε-closure(0):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 1)

print("ε-closure(1):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 2)

print("ε-closure(2):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 3)

print("ε-closure(3):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 4)

print("ε-closure(4):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 5)

print("ε-closure(5):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 6)

print("ε-closure(6):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 7)

print("ε-closure(7):", epsilon_closure_set)

```



```

epsilon_closure_set = epsilon_closure(nfa.states, 8)
print("ε-closure(8):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 9)
print("ε-closure(9):", epsilon_closure_set)

epsilon_closure_set = epsilon_closure(nfa.states, 10)
print("ε-closure(10):", epsilon_closure_set)
print("")

# The subset construction to pass the correct argument
def subset_construction(nfa):
    dfa_states = {}
    unmarked_states = []
    initial_state = epsilon_closure(nfa.states, 0)
    dfa_states[frozenset(initial_state)] = {}

    unmarked_states.append(initial_state)

    while unmarked_states:
        current_set = unmarked_states.pop(0)

        for symbol in ['a', 'b']:
            new_set = set()

            for state in current_set:
                if nfa.states[state][symbol] is not None:
                    new_set.update(epsilon_closure(nfa.states, nfa.states[state][symbol]))

            if new_set and frozenset(new_set) not in dfa_states:
                unmarked_states.append(new_set)
                dfa_states[frozenset(new_set)] = {}

            dfa_states[frozenset(current_set)][symbol] = frozenset(new_set)

    return dfa_states

# Construct DFA from NFA
dfa = subset_construction(nfa)

```

```
accepting_states = set()

for state in dfa:
    if 10 in state:
        accepting_states.add(state)

# Function to check if string is accepted
def check_string(dfa, string):
    current_state = frozenset(epsilon_closure(nfa.states, 0))

    for char in string:
        if char in dfa[current_state]:
            current_state = dfa[current_state][char]
        else:
            return "Not Accepted"

    if current_state in accepting_states:
        return "Accept"
    return "Not Accepted"

result = check_string(dfa, "abbabb")
print(result)
```

OUTPUT:

```
PROBLEMS  OUTPUT  PORTS  SEARCH ERROR  POSTMAN CONSOLE  TERMINAL  DEBUG CONSOLE  SQL CONSOLE

● (base) PS C:\Users\aryan> & C:/Python311/python.exe c:/Users/aryan/OneDrive/Desktop/NFA.py
Regular Expression: (a/b)*abb


Submitted by: Aryan Shanker Saxena (102103613)

State | a | b |  $\epsilon$ 1 |  $\epsilon$ 2
-----
0 | None | None | 1 | 7
1 | None | None | 2 | 4
2 | 3 | None | None | None
3 | None | None | 6 | None
4 | None | 5 | None | None
5 | None | None | 6 | None
6 | None | None | 7 | 1
7 | 8 | None | None | None
8 | None | 9 | None | None
9 | None | 10 | None | None
10 | None | None | None | None

 $\epsilon$ -closure(0): {0, 1, 2, 4, 7}
 $\epsilon$ -closure(1): {1, 2, 4}
 $\epsilon$ -closure(2): {2}
 $\epsilon$ -closure(3): {1, 2, 3, 4, 6, 7}
 $\epsilon$ -closure(4): {4}
 $\epsilon$ -closure(5): {1, 2, 4, 5, 6, 7}
 $\epsilon$ -closure(6): {1, 2, 4, 6, 7}
 $\epsilon$ -closure(7): {7}
 $\epsilon$ -closure(8): {8}
 $\epsilon$ -closure(9): {9}
 $\epsilon$ -closure(10): {10}

Accept
○ (base) PS C:\Users\aryan> |
```

Output when the string does not end in abb:

C: > Users > aryan > OneDrive > Desktop >  NFA.py > ...

116

117 `result = check_string(dfa, "abbabb")`

118 `print(result)`

PROBLEMS

OUTPUT

PORTS

SEARCH ERROR

POSTMAN CONSOLE

TERMINAL

Submitted by: Aryan Shanker Saxena (102103613)

State | a | b | $\epsilon 1$ | $\epsilon 2$

0 | None | None | 1 | 7

1 | None | None | 2 | 4

2 | 3 | None | None | None

3 | None | None | 6 | None

4 | None | 5 | None | None

5 | None | None | 6 | None

6 | None | None | 7 | 1

7 | 8 | None | None | None

8 | None | 9 | None | None

9 | None | 10 | None | None

10 | None | None | None | None

ϵ -closure(0): {0, 1, 2, 4, 7}

ϵ -closure(1): {1, 2, 4}

ϵ -closure(2): {2}

ϵ -closure(3): {1, 2, 3, 4, 6, 7}

ϵ -closure(4): {4}

ϵ -closure(5): {1, 2, 4, 5, 6, 7}

ϵ -closure(6): {1, 2, 4, 6, 7}

ϵ -closure(7): {7}

ϵ -closure(8): {8}

ϵ -closure(9): {9}

ϵ -closure(10): {10}

Not Accepted