Lab Assignment-1

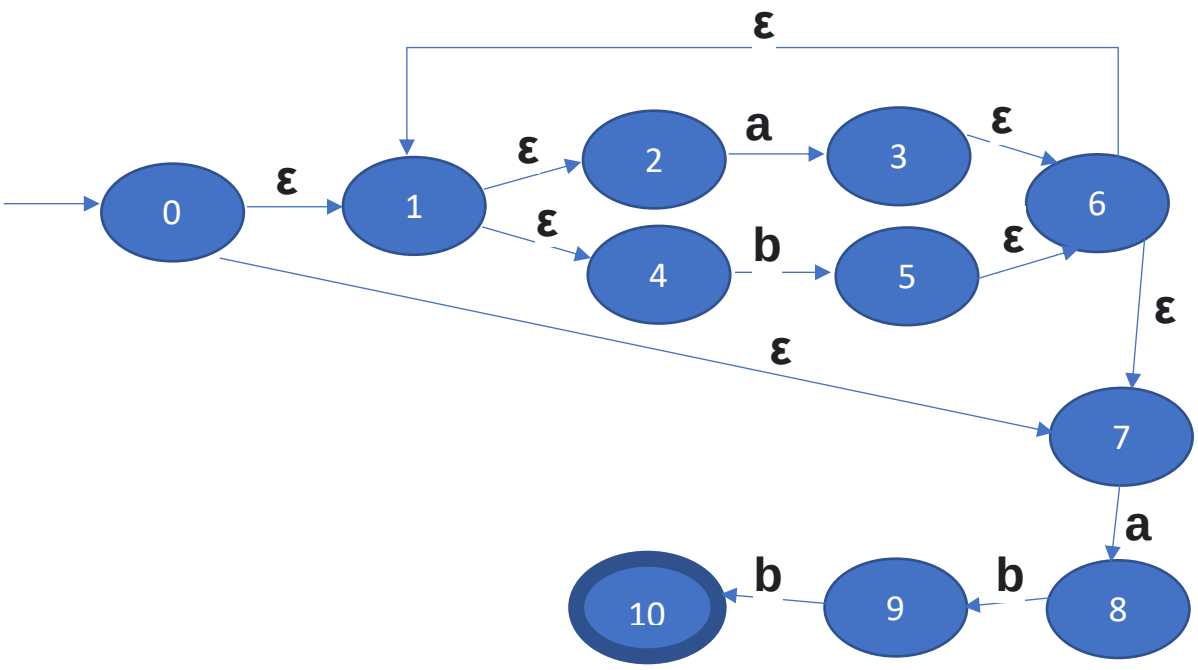
COMPILER CONSTRUCTION-UCS802

Ques 1. Design a Minimized DFA for the Regular Expression (a/b)*abb i.e. All strings ending with abb.

ANS.

This involves three steps:

Step 1. **Generate the NFA using Thomson' s Construction**



Step 2. **Generate the DFA using Subset Construction**

Now, we have to generate table of size 11x4 because of 11 states.

State	a	b	ε_1	ε_2
0	-	-	1	7
1	-	-	2	4
2	3	-	-	-
3	-	-	6	-

4	-	5	-	-
5	-	-	6	-
6	-	-	1	7
7	8	-	-	-
8	-	9	-	-
9	-	10	-	-
10	-	-	-	-

Algorithm for constructing DFA:

```
ec={}
stack=[]
states=11
start=0
final=10
ndfa={0:{'a':[],'b':[],'eps':[1,7]},1:{'a':[],'b':[],'eps':[2,4]},2:{'a':[3],
'b':[],'eps':[]},3:{'a':[],'b':[],'eps':[6]},4:{'a':[],'b':[5],'eps':[]},5:{'
a':[],'b':[],'eps':[6]},6:{'a':[],'b':[],'eps':[1,7]},7:{'a':[8],'b':[],'eps'
:[]},8:{'a':[],'b':[9],'eps':[]},9:{'a':[],'b':[10],'eps':[]},10:{'a':[],'b':
[],'eps':[]}}
for state in range(states):
    stack.append(state)
    ec[state]=[state]
    while stack:
        t=stack.pop()
        for i in ndfa[t]['eps']:
            if i not in ec[state]:
                ec[state].append(i)
                stack.append(i)
print('Epsilon Closure-',ec)

ds=[ec[0]]
input=['a','b']
dfa={}
mapping={}
complete=[]
char1='A'

while ds:
    p=sorted(ds[0])
    t=frozenset(ds[0])
    complete.append(ds[0])
    ds.pop(0)
    if t not in mapping.items():
        mapping[char1]=p
    else:
        continue
    if char1 not in dfa:
        dfa[char1]={'a':None,'b':None}
    for i in input:
```

```
        list1 = []
        s2=None
        for j in t:
            if ndfa[j][i]:
                s2=ndfa[j][i][0]
                if s2 not in list1:
                    list1.append(s2)
        list2=[]
        for state in list1:
            list2+=ec[state]
        list2.sort()
        if list2 not in ds and list2 not in complete:
            ds.append(list2)
        dfa[char1][i]=list2
    char1=chr(ord(char1)+1)
print(mapping)
starting_state=[]
final_state=[]
for i in dfa:
    a=dfa[i]['a']
    b=dfa[i]['b']
    def get_key(val):
        for key, value in mapping.items():
            if val == value:
                return key
    dfa[i]['a']=get_key(a)
    dfa[i]['b']=get_key(b)
for i in mapping:
    if start in mapping[i]:
        starting_state.append(i)
    if final in mapping[i]:
        final_state.append(i)
print('Final DFA is-',dfa)
print('Starting States are-','.join(i for i in starting_state))
print('Final States are-','.join(i for i in final_state))
```

OUTPUT:

Epsilon Closure- {0: [0, 1, 7, 2, 4], 1: [1, 2, 4], 2: [2], 3: [3, 6, 1, 7, 2, 4], 4: [4], 5: [5, 6, 1, 7, 2, 4], 6: [6, 1, 7, 2, 4], 7: [7], 8: [8], 9: [9], 10: [10]}

{'A': [0, 1, 2, 4, 7], 'B': [1, 2, 3, 4, 6, 7, 8], 'C': [1, 2, 4, 5, 6, 7], 'D': [1, 2, 4, 5, 6, 7, 9], 'E': [1, 2, 4, 5, 6, 7, 10]}

Final DFA is- {'A': {'a': 'B', 'b': 'C'}, 'B': {'a': 'B', 'b': 'D'}, 'C': {'a': 'B', 'b': 'C'}, 'D': {'a': 'B', 'b': 'E'}, 'E': {'a': 'B', 'b': 'C'}}

Starting States are- A

Final States are- E

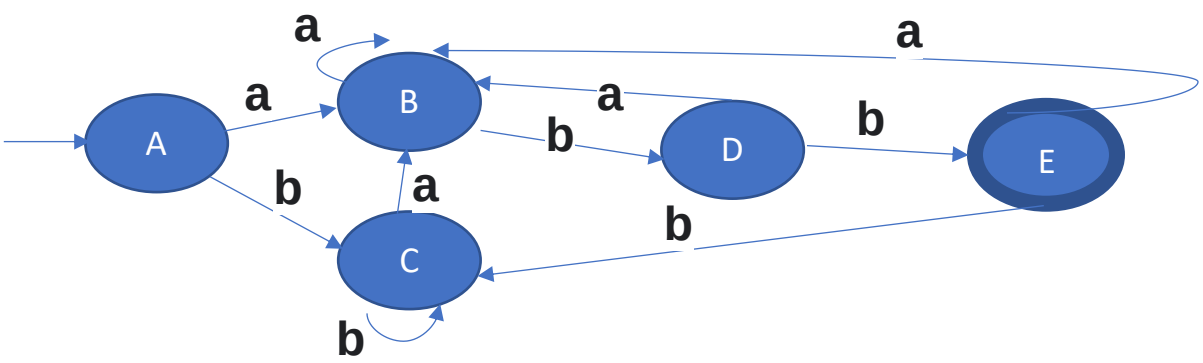
```
Epsilon Closure- {0: [0, 1, 7, 2, 4], 1: [1, 2, 4], 2: [2], 3: [3, 6, 1, 7, 2, 4], 4: [4], 5: [5, 6, 1, 7, 2, 4], 6: [6, 1, 7, 2, 4], 7:
[7], 8: [8], 9: [9], 10: [10]}


{'A': [0, 1, 2, 4, 7], 'B': [1, 2, 3, 4, 6, 7, 8], 'C': [1, 2, 4, 5, 6, 7], 'D': [1, 2, 4, 5, 6, 7, 9], 'E': [1, 2, 4, 5, 6, 7, 10]}


Final DFA is- {'A': {'a': 'B', 'b': 'C'}, 'B': {'a': 'B', 'b': 'D'}, 'C': {'a': 'B', 'b': 'C'}, 'D': {'a': 'B', 'b': 'E'}, 'E': {'a': 'B',
'b': 'C'}}


Starting States are- A
Final States are- E
```

Generated DFA is:



Step 3. Minimize the DFA

We will minimize the DFA using Table Filling Algorithm

```
dfa={'A': {'a': 'B', 'b': 'C'}, 'B': {'a': 'B', 'b': 'D'}, 'C': {'a': 'B',
'b': 'C'}, 'D': {'a': 'B', 'b': 'E'}, 'E': {'a': 'B', 'b': 'C'}}
states=['A','B','C','D','E']
final=['E']
nonfinal=['A','B','C','D']
table=[[0 for _ in range(5)] for _ in range(5)]
count=0
for i in range(5):
    for j in range(i):
        c1=states[i] in final
        c2=states[j] in nonfinal
        c3=states[i] in nonfinal
        c4=states[j] in final
        table[i][j]=c1&c2 or c3&c4
print(table)
while True:
    for i in range(5):
        for j in range(i):
            c1=dfa[states[i]]['a']
            c2=dfa[states[j]]['a']
            c3=dfa[states[i]]['b']
```

```
            c4=dfa[states[j]]['b']
            if (table[ord(c1)-ord('A')][ord(c2)-ord('A')] or table[ord(c2)-
ord('A')][ord(c1)-ord('A')]) and not table[i][j]:
                table[i][j]=True
                count+=1
            elif (table[ord(c3)-ord('A')][ord(c4)-ord('A')] or
table[ord(c4)-ord('A')][ord(c3)-ord('A')]) and not table[i][j]:
                table[i][j]=True
                count+=1
    if count==0:
        break
    else:
        count=0
unmarked=[]
for i in range(5):
    for j in range(i):
        if not table[i][j]:
            unmarked.append(states[i])
            unmarked.append(states[j])
print(set(unmarked))
```

OUTPUT:

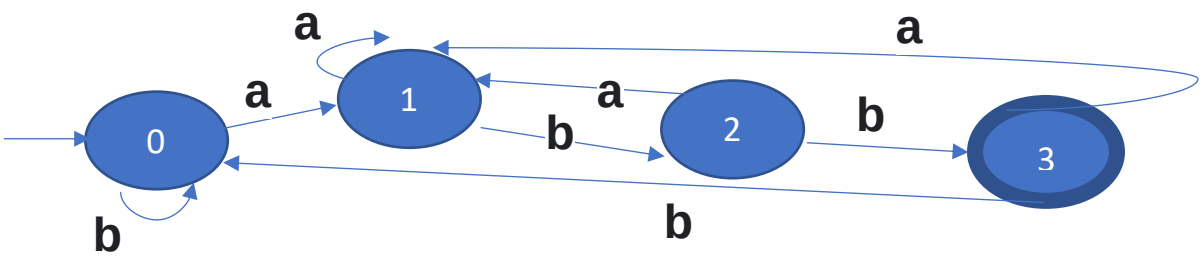
[[0, 0, 0, 0, 0], [False, 0, 0, 0, 0], [False, False, 0, 0, 0], [False, False, False, 0, 0], [True, True, True, True, 0]]

{'C', 'A'}

```
[[0, 0, 0, 0, 0], [False, 0, 0, 0, 0], [False, False, 0, 0, 0], [False, False, False, 0, 0], [True, True, True, True, 0]]

{'A', 'C'}
```

Now combining the states A and C together we get



Step 4. Check each string is it accepted or not

Algorithm for checking each string

```
def start(c):
    if (c == 'a'):
        dfa = 1
    elif (c == 'b'):
        dfa = 0
    else:
        dfa = -1
    return dfa

def state1(c):
    if (c == 'a'):
        dfa = 1
    elif (c == 'b'):
        dfa = 2
    else:
        dfa = -1
    return dfa

def state2(c):
    if (c == 'b'):
        dfa = 3
    elif (c == 'a'):
        dfa = 1
    else:
        dfa = -1
    return dfa

def state3(c):
    if (c == 'b'):
        dfa = 0
    elif (c == 'a'):
        dfa = 1
    else:
        dfa = -1
    return dfa

def state4(c):
    dfa = -1
    return dfa

def isAccepted(String):
    l = len(String)
    dfa = 0
    for i in range(l):
        if (dfa == 0):
            dfa = start(String[i])

        elif (dfa == 1):
            dfa = state1(String[i])

        elif (dfa == 2):
            dfa = state2(String[i])
```

```
        elif (dfa == 3):
            dfa = state3(String[i])

        elif (dfa == 4):
            dfa = state4(String[i])
        else:
            return 0
    if (dfa == 3):
        return 1
    else:
        return 0

list1 = ["aaaabb", "aabbb", "abbabb"]
for i in list1:
    if (isAccepted(i)):
        print("ACCEPTED")
    else:
        print("NOT ACCEPTED")
```

INPUT:

aaaabb
aabbb
abbabb

OUTPUT:

ACCEPTED
NOT ACCEPTED
ACCEPTED

```
ACCEPTED
NOT ACCEPTED
ACCEPTED
```