Experiment 1 : NFA to DFA

Aim To study and implement a method to convert a Nondeterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA)

Algorithm:

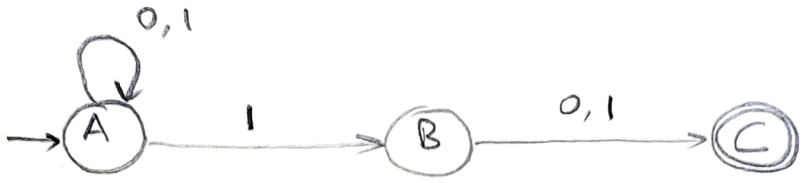
1. Start
2. Get the input NFA from the user
3. Add the first state of NFA to DFA table
4. Add trasitions of the start to DFA transition table.
5. If start state makes transitions to multiple states, then create them as a single state.
6. For new states in DFA transition table, add them to new state and add their transitions.
7. Repeat step 6 till no new states exist.
8. Mark all states which contain the NFA final state as final states for DFA

MANUAL WORKING

Input : NFA transition table

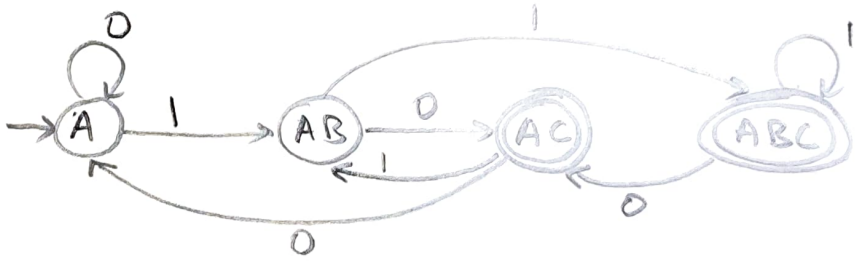
		0	1
→	A	{A}	{A, B}
	B	{C}	{C}
*	C	∅	∅

NFA :



Output :
DFA transition table

		0	1
→	A	A	AB
	AB	AC	ABC
*	AC	A	AB
*	ABC	AC	ABC



Sashrika Surya
RA1911027010092
Section: N2
Lab Batch: 1

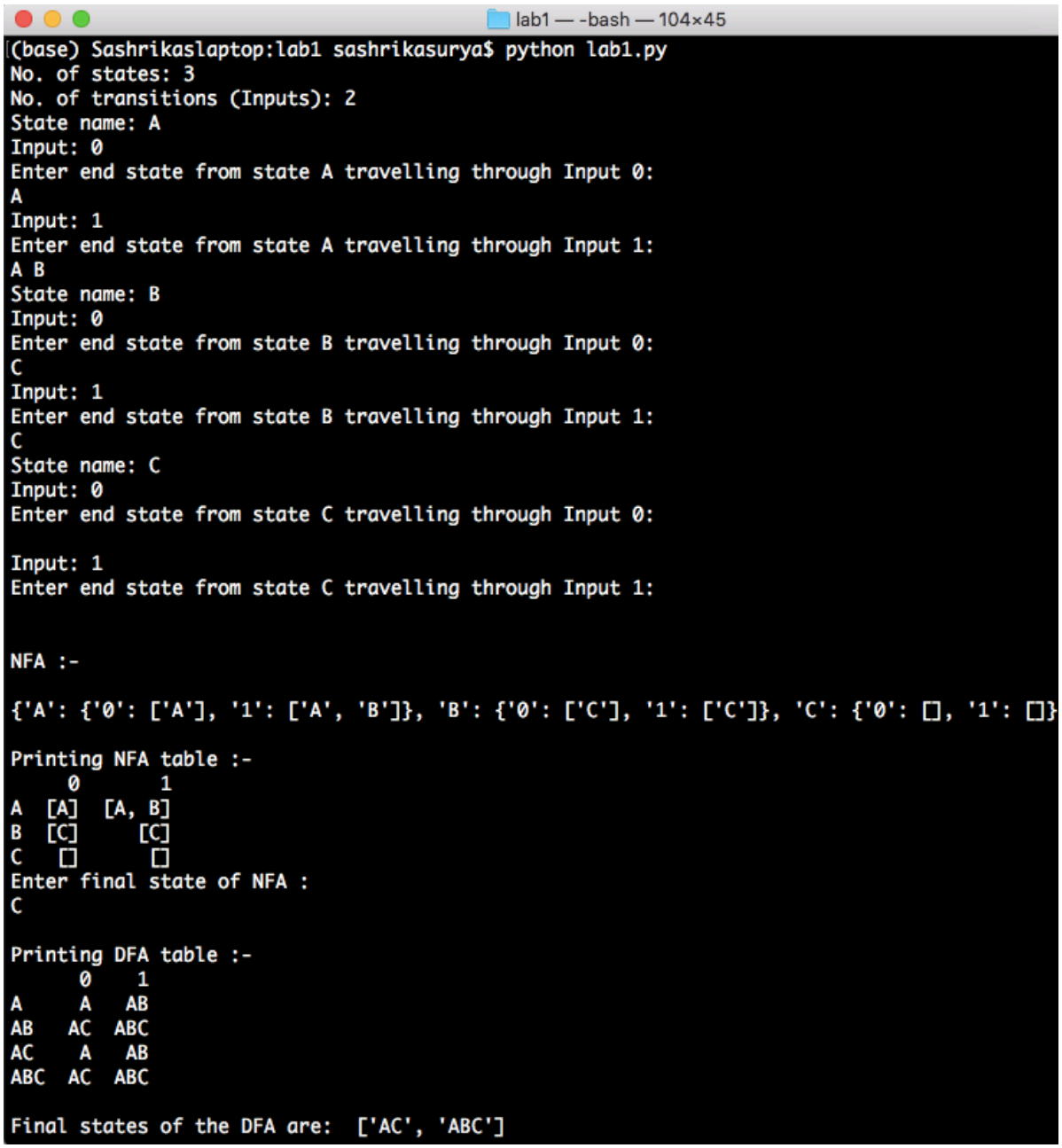
Compiler Design Lab
Experiment 1: NFA to DFA

Code:

```
import pandas as pd

nfa = {}
n = int(input("No. of states: "))
t = int(input("No. of transitions (Inputs): "))
for i in range(n):
    state = input("State name: ")
    nfa[state] = {}
    for j in range(t):
        path = input("Input: ")
        print("Enter end state from state {} travelling through Input {}: ".format(state, path))
        reaching_state = [x for x in input().split()]
        nfa[state][path] = reaching_state

print("\nNFA :- \n")
print(nfa)
print("\nPrinting NFA table :- ")
nfa_table = pd.DataFrame(nfa)
print(nfa_table.transpose())

print("Enter final state of NFA : ")
nfa_final_state = [x for x in input().split()]

new_states_list = []

dfa = {}
keys_list = list(
    list(nfa.keys())[0])
path_list = list(nfa[keys_list[0]].keys())

dfa[keys_list[0]] = {}
for y in range(t):
    var = "".join(nfa[keys_list[0]][
                    path_list[y]])
    dfa[keys_list[0]][path_list[y]] = var
```

```
    if var not in keys_list:
        new_states_list.append(var)
        keys_list.append(var)

while len(new_states_list) != 0:
    dfa[new_states_list[0]] = {}
    for _ in range(len(new_states_list[0])):
        for i in range(len(path_list)):
            temp = []
            for j in range(len(new_states_list[0])):
                temp += nfa[new_states_list[0][j]][path_list[i]]
            s = ""
            s = s.join(temp)
            if s not in keys_list:
                new_states_list.append(s)
                keys_list.append(s)
            dfa[new_states_list[0]][path_list[i]] = s

    new_states_list.remove(new_states_list[0])


print("\nPrinting DFA table :- ")
dfa_table = pd.DataFrame(dfa)
print(dfa_table.transpose())

dfa_states_list = list(dfa.keys())
dfa_final_states = []
for x in dfa_states_list:
    for i in x:
        if i in nfa_final_state:
            dfa_final_states.append(x)
            break

print("\nFinal states of the DFA are: ", dfa_final_states)
```

Output:


lab1 — -bash — 104×45
(base) Sashrikaslaptop:lab1 sashrikasurya$ python lab1.py
No. of states: 3
No. of transitions (Inputs): 2
State name: A
Input: 0
Enter end state from state A travelling through Input 0:
A
Input: 1
Enter end state from state A travelling through Input 1:
A B
State name: B
Input: 0
Enter end state from state B travelling through Input 0:
C
Input: 1
Enter end state from state B travelling through Input 1:
C
State name: C
Input: 0
Enter end state from state C travelling through Input 0:
Input: 1
Enter end state from state C travelling through Input 1:
NFA :-
{'A': {'0': ['A'], '1': ['A', 'B']}, 'B': {'0': ['C'], '1': ['C']}, 'C': {'0': [], '1': []}
Printing NFA table :-
0 1
A [A] [A, B]
B [C] [C]
C [] []
Enter final state of NFA :
C
Printing DFA table :-
0 1
A A AB
AB AC ABC
AC A AB
ABC AC ABC
Final states of the DFA are: ['AC', 'ABC']