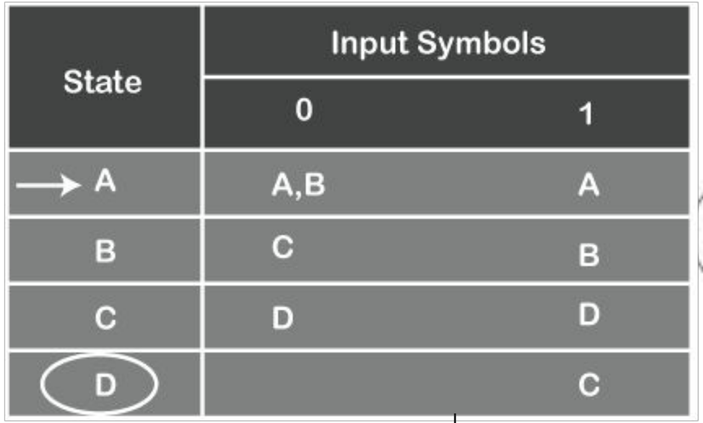
State Transition Table Visualization Conversions from NFA To DFA

By Shravan

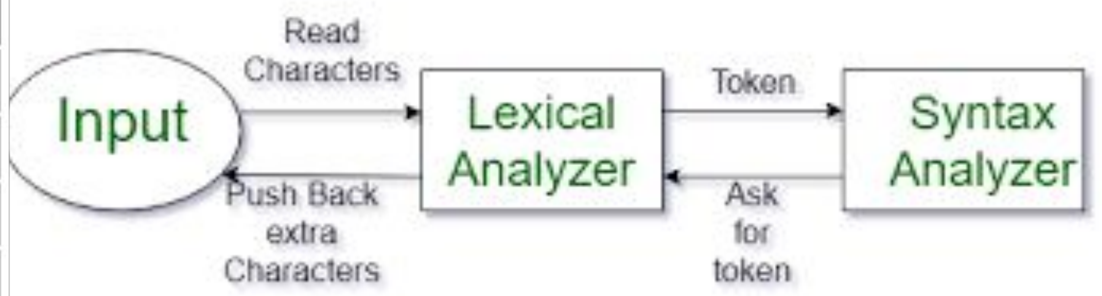
Introduction

The project will focus on Conversion of NFA (Non-Deterministic Finite Automata) to DFA (Deterministic Finite Automata) using State Transition Table. NFA, when an input is inserted, the machine will go from the current state to multiple states. Vice versa, in DFA, the machine will go from the current state to only one state, when an input is given. One of the proposed solution will be using dataset of NFA to DFA with given State Transition Table to visualize the process. Another one is to use Lexical Analysis to Produce Python code for conversions from NFA to DFA.

NFA

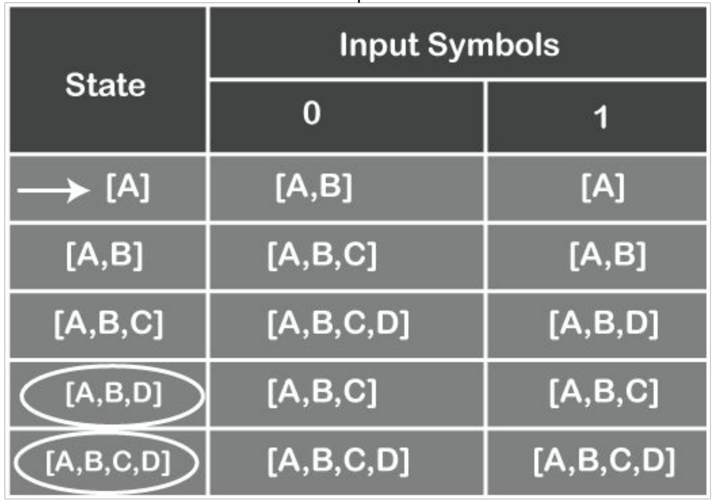


State	Input Symbols	
	0	1
→ A	A,B	A
B	C	B
C	D	D
D		C



To

DFA



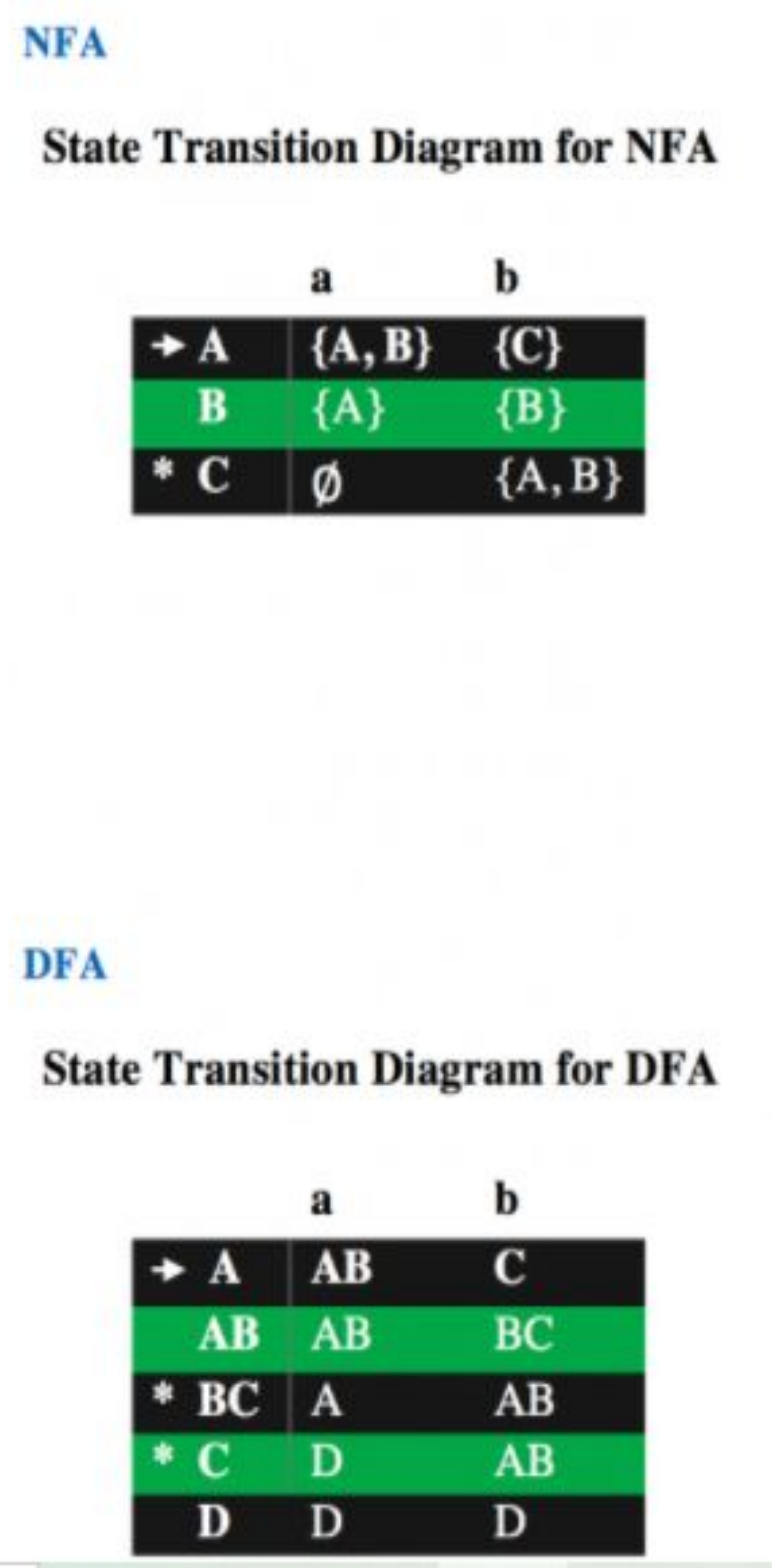
State	Input Symbols	
	0	1
→ [A]	[A,B]	[A]
[A,B]	[A,B,C]	[A,B]
[A,B,C]	[A,B,C,D]	[A,B,D]
[A,B,D]	[A,B,C]	[A,B,C]
[A,B,C,D]	[A,B,C,D]	[A,B,C,D]

Related work

1. A-DFA: A Time- and Space-Efficient DFA Compression Algorithm for Fast Regular Expression Evaluation (researchgate.net)
2. A technique for converting NFA's and DFA's to regular expressions in Lex (rowan.edu)
3. jan_2020_dfa_submatches_extraction.pdf (nitely.github.io)

Implementation#1

For this I will use a dataset that has NFA To DFA Conversion. But the main implementation will be using State Transition diagram as a process on converting NFA to DFA.



NFA

State Transition Diagram for NFA

	a	b
→ A	{A, B}	{C}
B	{A}	{B}
* C	∅	{A, B}

DFA

State Transition Diagram for DFA

	a	b
→ A	AB	C
AB	AB	BC
* BC	A	AB
* C	D	AB
D	D	D

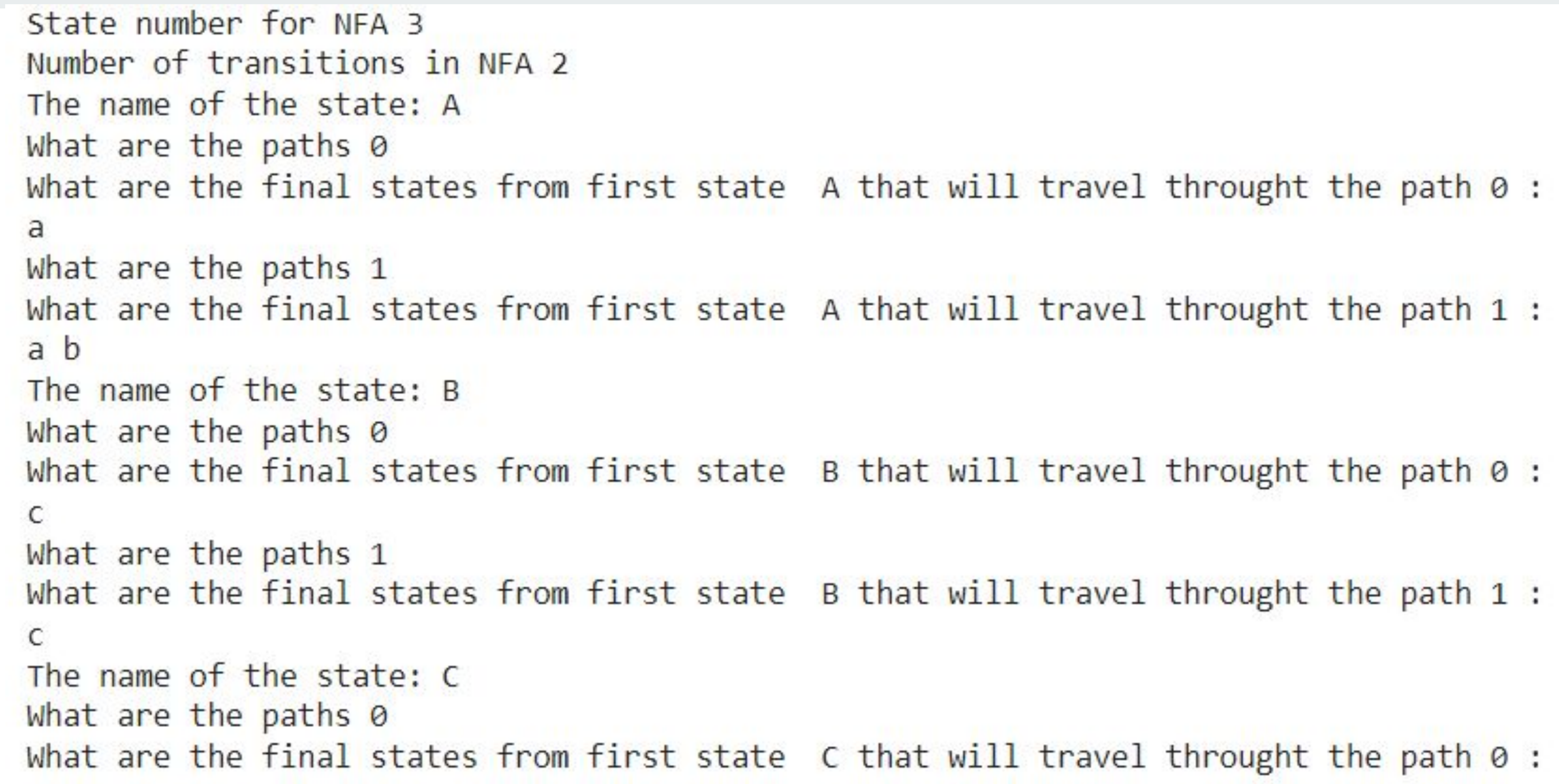
Implementation#2

Here is the code progress I have in store thus far. So far I have implemented the NFA table in the code as well as the tokens for NFA. Likewise, I will further improvise on creating a dfa table and create a token for it as well as showing the end result of DFA solution.

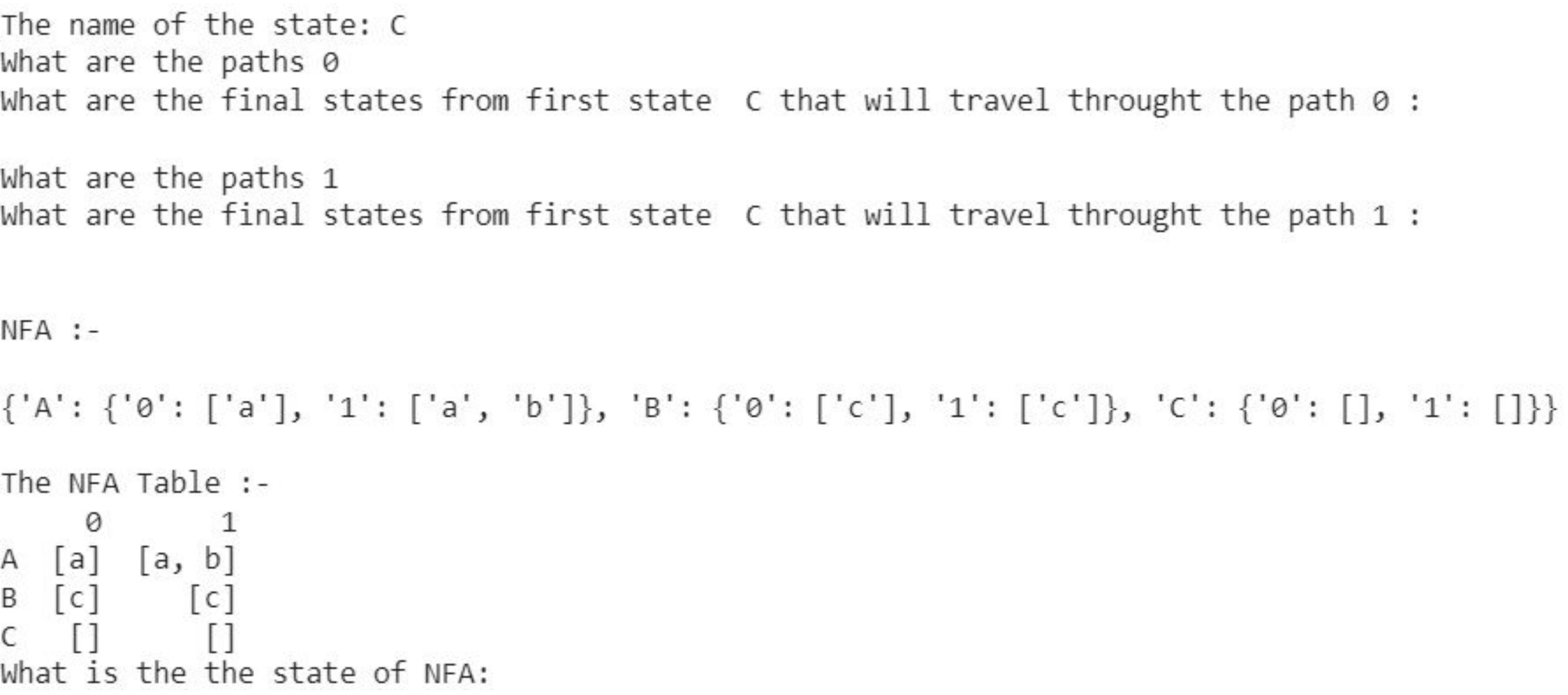
```python
import pandas as pd

nfa = {}
o = int(input("State number for NFA "))
#input the overall State number for NFA. Ex: If you want to create state a,b, and c put in 3.
p = int(input("Number of transitions in NFA "))  #Input Number of transitions in NFA eg: if you want to create a and b for transition type 2.
for i in range(o):
    state = input("The name of the state: ") #This is where you name the state. eg: A,b,c,1,2,3,4,etc....
    nfa[state] = {}
    for j in range(p):
        path = input("What are the paths ") #create the path on top of the transition. Eg: path 0 in transition one and path 1 in transition two.
        print("What are the final states from first state  {} that will travel throught the path {} : ".format(state, path))
        reaching_state = [x for x in input().split()] #the end state for NFA Table.
        nfa[state][path] = reaching_state #This is to make sure that the end state is displayed in the paths
print("\nNFA :- \n")
print(nfa)
print("\nThe NFA Table :- ") #Display the NFA table
nfa_table = pd.DataFrame(nfa)
print(nfa_table.transpose())

print("What is the the state of NFA: ")
nfa_final_state = [x for x in input().split()] #Input the final state.
```

Implementation#3

Here I try to do some testings for the NFA table, And I was able to display the NFA code output. But I still need to put the code for dfa table in here.



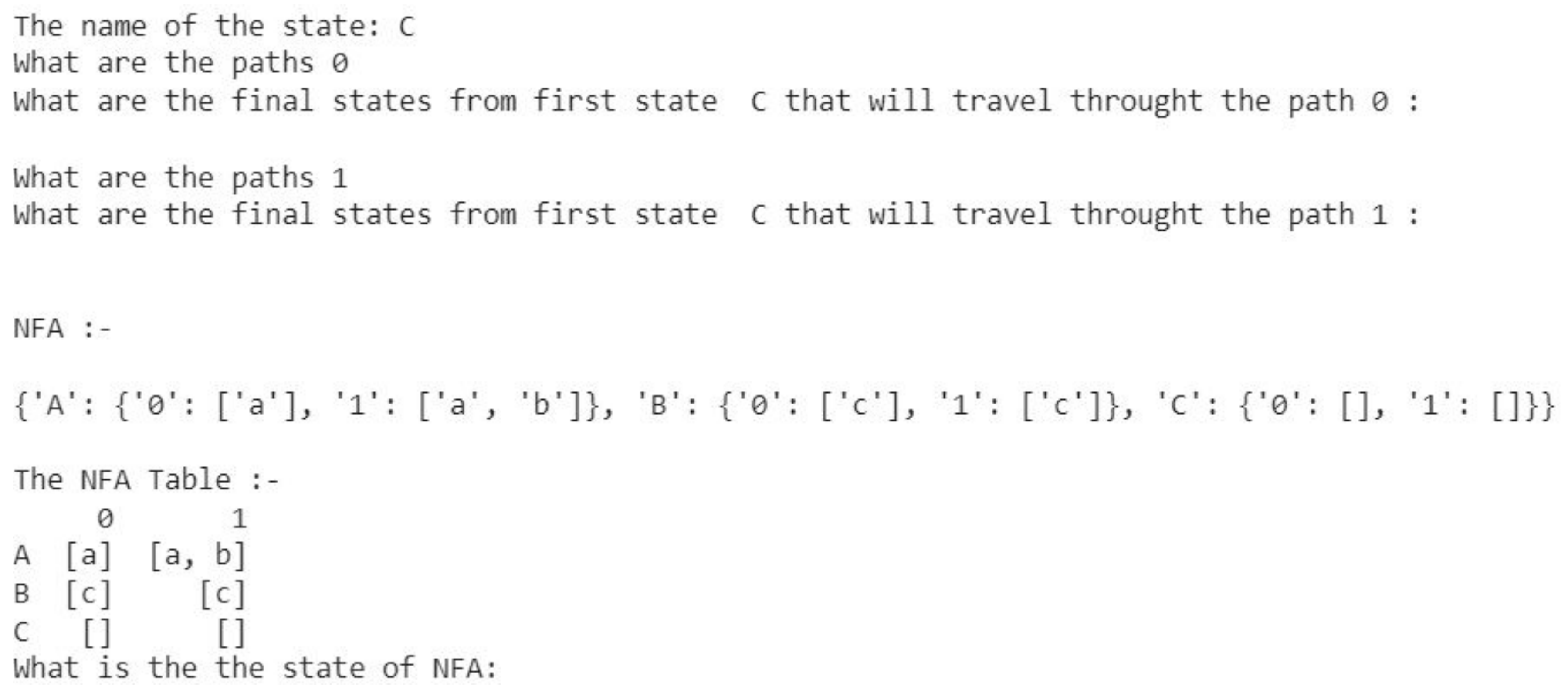
```
State number for NFA 3
Number of transitions in NFA 2
The name of the state: A
What are the paths 0
What are the final states from first state  A that will travel throught the path 0 :
a
What are the paths 1
What are the final states from first state  A that will travel throught the path 1 :
a b
The name of the state: B
What are the paths 0
What are the final states from first state  B that will travel throught the path 0 :
c
What are the paths 1
What are the final states from first state  B that will travel throught the path 1 :
c
The name of the state: C
What are the paths 0
What are the final states from first state  C that will travel throught the path 0 :
```



```
The name of the state: C
What are the paths 0
What are the final states from first state  C that will travel throught the path 0 :

What are the paths 1
What are the final states from first state  C that will travel throught the path 1 :


NFA :-

{'A': {'0': ['a'], '1': ['a', 'b']}, 'B': {'0': ['c'], '1': ['c']}, 'C': {'0': [], '1': []}}

The NFA Table :-
     0       1
A  [a]  [a, b]
B  [c]     [c]
C   []      []
What is the the state of NFA:
```

Evaluation and Discussion

No, the problem is not solved yet because there is still alot of stuff to put such as the Final state of NFA as well as the State Transition table for DFA.



```
The name of the state: C
What are the paths 0
What are the final states from first state  C that will travel throught the path 0 :

What are the paths 1
What are the final states from first state  C that will travel throught the path 1 :


NFA :-

{'A': {'0': ['a'], '1': ['a', 'b']}, 'B': {'0': ['c'], '1': ['c']}, 'C': {'0': [], '1': []}}

The NFA Table :-
     0       1
A  [a]  [a, b]
B  [c]     [c]
C   []      []
What is the the state of NFA:
```

Conclusion And Recommendation

Overall, the code still need to be improved to solve the problem. Improvement such as additional features of DFA tables, tokens, as well as showing the end result of DFA being coveted from the series of input in the NFA.

References

1. Converting Epsilon-NFA to DFA using Python and Graphviz - GeeksforGeeks
2. Implement NFA algorithm (based on Python) (programmer.ink)
3. Welcome to PySimpleAutomata’s documentation! — PySimpleAutomata 0.5.0 documentation
4. 2. Finite Automata — Computational Models (kentdlee.github.io)

Appendix

1. Srinivasan32/CompilationTechniqueProject (github.com)