State Transition Table Visualization Conversions from NFA To DFA

Course: Compilation Technique

Course ID: COMP6062001

Instructor: Mrs Seraphine

Group Members:

Shravan Srinivasan(2440042872)

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 solutions will be using a dataset of NFA to DFA with a 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. Since I am doing it individually, there is only limited of one job to do.

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

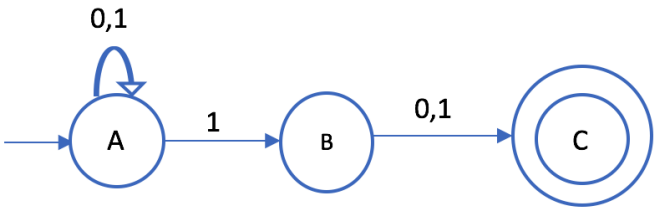
- For this I will use a dataset that has NFA To DFA Conversion. But the main implementation will be using the State Transition diagram as a process on converting NFA to DFA.

  NFA Table

NFA

State Transition Diagram for NFA

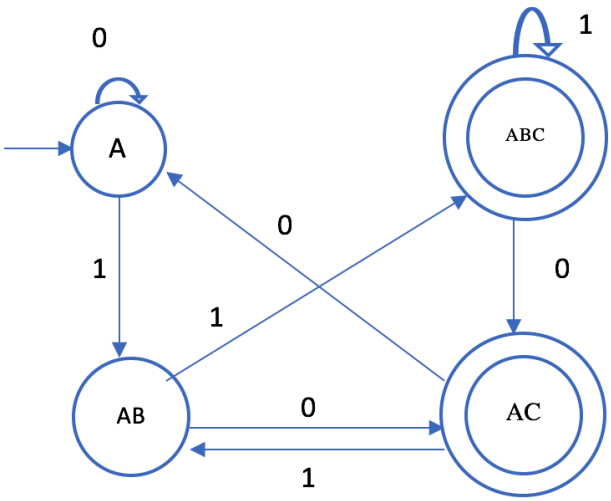
	0	1
→ A	{A}	{A,B}
B	{C}	{C}
* C	∅	∅



DFA

State Transition Diagram for DFA

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



- To do it, we need to first convert the following NFA to DFA. We will use the table as guidance. And we will use python code as the process of converting NFA to DFA.
- 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.

Input:

```
import pandas as pd #To help create the state visualization table
# all of the bottom code is to help create the state visualization table
for Conversion of NFA to DFA.
nf = {} #Indicator of NFA
```

```
o = int(input("The number of states for NFA")) #code to tell the state
number
p = int(input("Number of transitions in NFA")) #code to tell the
transition number
for i in range(o):
    sta = input("The name of the state: ")  #The part for labeling the
state name like 2,3,e,f,and so on
    nf[sta] = {}
    for j in range(p):
        pa = input("What are the paths ") #This is where you lay out the
two paths in the state transition table
        print("What are the final states from first state  {} that will
travel throught the path {} : ".format(sta, pa))
        approach_sta = [w for w in input().split()]  #The part where The
state of NFA is about to stop.
        nf[sta][pa] = approach_sta #Of course when the state stops, it has
to be shown
print("\nNFA :- \n") #Yes, we need results after the process.
print(nf)
print("\nThe NFA Table :- ") # This is where we can see this table
masterpiece.

nf_tab = pd.DataFrame(nf)
print(nf_tab.transpose())

print("The very last NFA")
the_fin_sta_for_nf = [w for w in input().split()]
 #Of course, There has to be a limit when making the Nfa Table.

new_sta_order = []  #This code will be responsible for handling new and
made DFA
df= {} #This symbol will help determine to correct DFA
ke_list = list(list(nf.keys())[0])  #states created in dfa are also
appended further
pa_list = list(nf[ke_list[0]].keys()) #list of all the paths eg: [a,b] or
[0,1]


df[ke_list[0]] = {}   #creating a nested dictionary in dfa
for y in range(p):
```

```
    var = "".join(nf[ke_list[0]][pa_list[y]])#creating a single string
from all the elements of the list which is a new state
    df[ke_list[0]][pa_list[y]] = var #assigning the state in DFA table
    if var not in ke_list:   #if the state is newly created
        new_sta_order.append(var)      #then append it to the
new_sta_order
        ke_list.append(var)             #as well as to the ke_list which
contains all the states
```

Output:

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

What are the paths b
```

```
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 & 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. Aloof this two need like to almost need to be similar to how I create the code for making the NFA table.

Conclusion And Recommendation

Overall, the code still needs 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. To be able to convert from NFA to DFA, there are two things that needs to be done which is display the DFA table and the result after the table of DFA.

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)

Question and answer

Q:

A:

Q:

A:

Q:

A:

Q:

A:

Q:

A: