



Concordia University

# **Engineering and Computer Science**

**COMP 5361**

**Discrete Structures and Formal Languages**

Programming Assignment-3 Report

Submitted To: Prof. Gösta Grahne

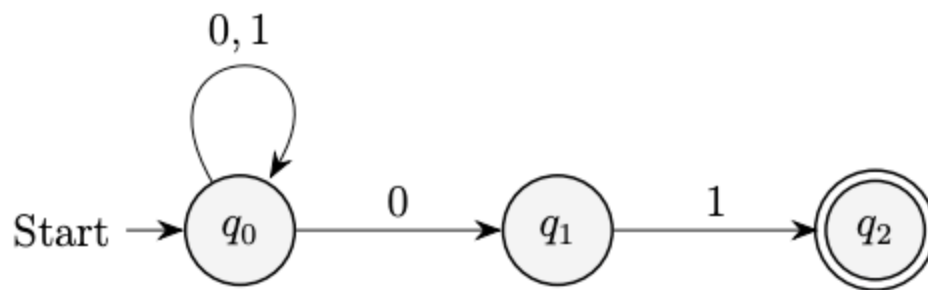
Yash Jayeshkumar Pandya (40119272)

## Problem statement:

1. Use the PySimpleAutomata-software to write a program that takes as input a transition table for a DFA or NFA A, and outputs the transition diagram for A. For example, the input could be the automaton  $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$  that has transition table

$\delta$	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
$q_1$	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$

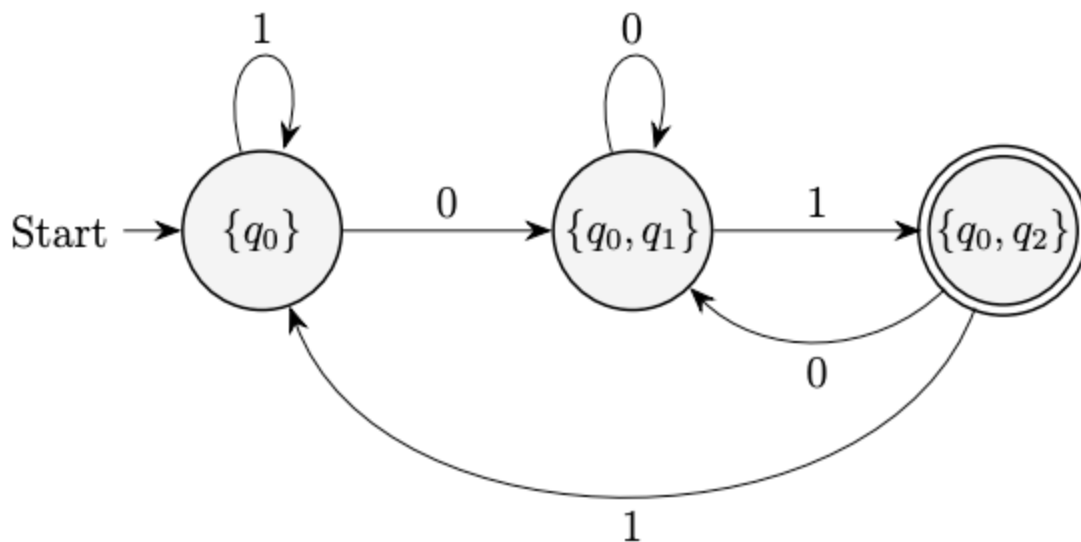
The automaton A as a transition diagram:



2. Write a Python program that takes as input the transition table for an NFA A (no  $\rightarrow$ -transitions), such as the one above, and outputs the transition table and diagram for a DFA B, such that  $L(A) = L(B)$ . For example, the NFA above should result in output

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\star \{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\star \{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\star \{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

and



### Program Logic:

The program takes user input of either the user wants to generate NFA/DFA diagram by providing the transition table or the user wants to generate Equivalent DFA by providing the NFA transition table. Once the user select his/her choice of operation, they are prompt with some interactive input console on where the user needs to enter alphabets, start/initial state(s), accepting/final state(s) and multiple transition from one state to another state(s) based on certain provided alphabet.

The program stores the data into a json file based on either selection of question 1 or question 2 and the selection of either NFA or DFA for question 1. The output generates .dot file and .svg image file and based on selection and stores into the current directory. The program also generates some console output to display transitions and equivalent tables.

The program works in an OOP manner. One class named COMP5361 maintains all variables and methods that the program uses. Each of the variables stores the value from user input and the names of the variables define its use.

```
initial_states,    accepting_states,    transitions,    alphabet,    states,
dfa_initial_states, dfa_transitions, dfa_result_states, dfa_accepting_states
```

There are multiple getter and setter methods which are useful to set and get the values of above mentioned variables from the user input.

Additionally, there are certain methods which help to prompt the user with input request and stores the data in appropriate variables, these methods are

```
read_and_store_alphabets, read_and_store_states, read_and_store_initial_states,  
read_and_store_accepting_states, read_and_store_nfa_transitions,  
read_and_store_dfa_transitions
```

The program uses list and dictionary data structures to store sequential and key value paired data.

The variable `data` contains the json/dictionary that can be sent to `pysimple-automata` library instance which converts json data to `.dot` file and generates the output.

For the second question, method named `nfa_to_dfa_conversion` converts the NFA data ie. states/transition and maps the equivalent DFA data before sending it to `pysimple-automata` for generating DFA result image. Based on NFA initial state, the method starts storing states into DFA states which can be followed by a separation rule if multiple outgoing edges are present in NFA for the same alphabet. It adds new state ie. if state names are A and B then it creates a new state named AB.

Assuming state A if the coming alphabet is 0 and next states are A and B then the program creates a new state named AB. This new state has all outgoing edges which A and B states have. Based on the accepting/final state of NFA. The same method maps accepting states of equivalent DFA and stores in `dfa_accepting_states`.

Basic I/O operation related to file writing is implemented. Along with this, below methods are used from the `automata_IO` library to generate images and `.dot` files from json data.

```
dfa_json_importer, dfa_to_dot, nfa_json_importer, nfa_to_dot
```

## How to Run:

- **Run Program ( Command: `python asgn3.py` )**

When the program runs user is displayed with following menu in the console

```
COMP-5361 Assignment-3 Menu  
-----  
1. Produce Transition Diagram from Transition table  
2. Produce DFA Transition Diagram and Transition table from NFA Transition  
   table  
3. Exit  
  
Select: 1
```

The user need to select any one of the available option at a time, otherwise it will raise an error for invalid choice.

After selecting choice 1 user need to select another choice on what user wants go generate a NFA or a DFA.

```
Transition Table Generation Menu
-----
1. NFA
2. DFA
3. Exit

Select: 1

After selecting selection, interactive console input begins where users need to give some
input as shown below.

Enter alphabet(s), if multiple then seperate by comma : 0,1

===== Alphabet(s) ===== : 0, 1

Enter state(s), if multiple then seperate by comma : q0,q1,q2

===== State(s) ===== : q0, q1, q2

Enter number of initial state(s) : 1

Available state(s) : q0, q1, q2
Select the initial state 1 : q0

===== Initial state(s) ===== : q0

Enter number of accepting state(s) : 1

Available state(s) : q0, q1, q2
Select the accepting state 1 : q2

===== Accepting state(s) ===== : q2

Available state(s) : q0, q1, q2
Select next state(s) == Note: if multiple then seperate by comma /
hit enter for leaving it empty

q0 => 0 => (?) : q0,q1

q0 => 1 => (?) : q0
```

q1 => 0 => (?) :

q1 => 1 => (?) : q2

q2 => 0 => (?) :

q2 => 1 => (?) :

==== Transition(s) =====

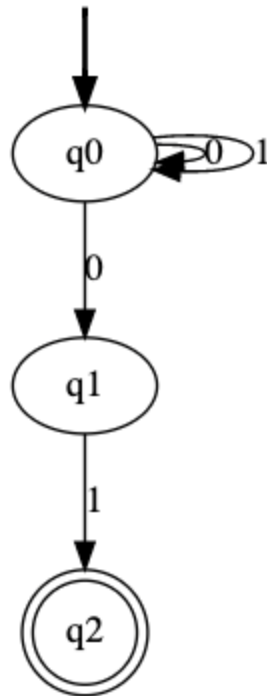
q0 => 0 => q0

q0 => 0 => q1

q0 => 1 => q0

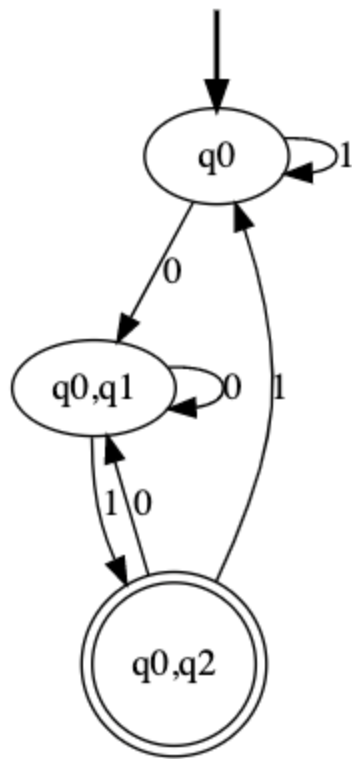
q1 => 1 => q2

above is the interactive console result for selection **choice 1**: Produce Transition Diagram from Transition table It also produces either NFA or DFA based on selected choice and save it into the same directory with file name "nfa" or "dfa".



**choice 2:** Produce DFA Transition Diagram and Transition table from NFA Transition table

Above selection also takes the same input as selection choice 1. However, It produces the equivalent DFA transition table and displays on the console.



NFA To DFA Transition Table

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
{q0}	{q0,q1}	{q0}
{q1}	$\emptyset$	{q2}
{q2}	$\emptyset$	$\emptyset$
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	{q0}
{q1,q2}	$\emptyset$	{q2}
{q0,q1,q2}	{q0,q1}	{q0,q2}

It also generates, equivalent DFA image within the same directory with name "dfa\_Part2".

## Test Cases:

Finder File Edit View Go Window Help

Terminal: Local +

(tutorial) ypandya@Yashs-MacBook-Pro COMP-5361-Assignment-3 % python asgn3.py

COMP-5361 Assignment-3 Menu

-----

1. Produce Transition Diagram from Transition table  
2. Produce DFA Transition Diagram and Transition table from NFA Transition table  
3. Exit

Select: 1

-----

Transition Table Generation Menu

-----

1. NFA  
2. DFA  
3. Exit

Select: 1

Enter alphabet(s), if multiple then separate by comma : 0,1

==== Alphabet(s) ===== : 0, 1

Enter state(s), if multiple then separate by comma : A,B

==== State(s) ===== : A, B

Enter number of Initial state(s) : 1

Available state(s) : A, B  
Select the initial state 1 : A

==== Initial state(s) ===== : A

Enter number of accepting state(s) : 1

Available state(s) : A, B  
Select the accepting state 1 : B

==== Accepting state(s) ===== : B

Available state(s) : A, B  
Select next state(s) == Note: if multiple then separate by comma / hit enter for leaving it empty

A => 0 => (?) : A

A => 1 => (?) : A,B

B => 0 => (?) :

B => 1 => (?) :

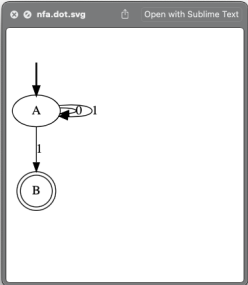
==== Transition(s) =====

A => 0 => A

A => 1 => A

A => 1 => B

(tutorial) ypandya@Yashs-MacBook-Pro COMP-5361-Assignment-3 %



Test case 1(a) - NFA

Finder File Edit View Go Window Help

Terminal: Local +

COMP-5361 Assignment-3 Menu

-----

1. Produce Transition Diagram from Transition table  
2. Produce DFA Transition Diagram and Transition table from NFA Transition table  
3. Exit

Select: 2

Enter alphabet(s), if multiple then separate by comma : 0,1

==== Alphabet(s) ===== : 0, 1

Enter state(s), if multiple then separate by comma : A,B

==== State(s) ===== : A, B

Enter number of Initial state(s) : 1

Available state(s) : A, B  
Select the initial state 1 : A

==== Initial state(s) ===== : A

Enter number of accepting state(s) : 1

Available state(s) : A, B  
Select the accepting state 1 : B

==== Accepting state(s) ===== : B

Available state(s) : A, B  
Select next state(s) == Note: if multiple then separate by comma / hit enter for leaving it empty

A => 0 => (?) : A

A => 1 => (?) : A,B

B => 0 => (?) :

B => 1 => (?) :

==== Transition(s) =====

A => 0 => A

A => 1 => A

A => 1 => B

=====

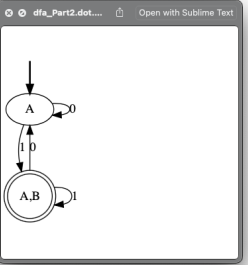
NFA To DFA Transition Table

=====

$\sigma$	a	a
{A}	{A}	{A,B}
{B}	a	a
{A,B}	{A}	{A,B}

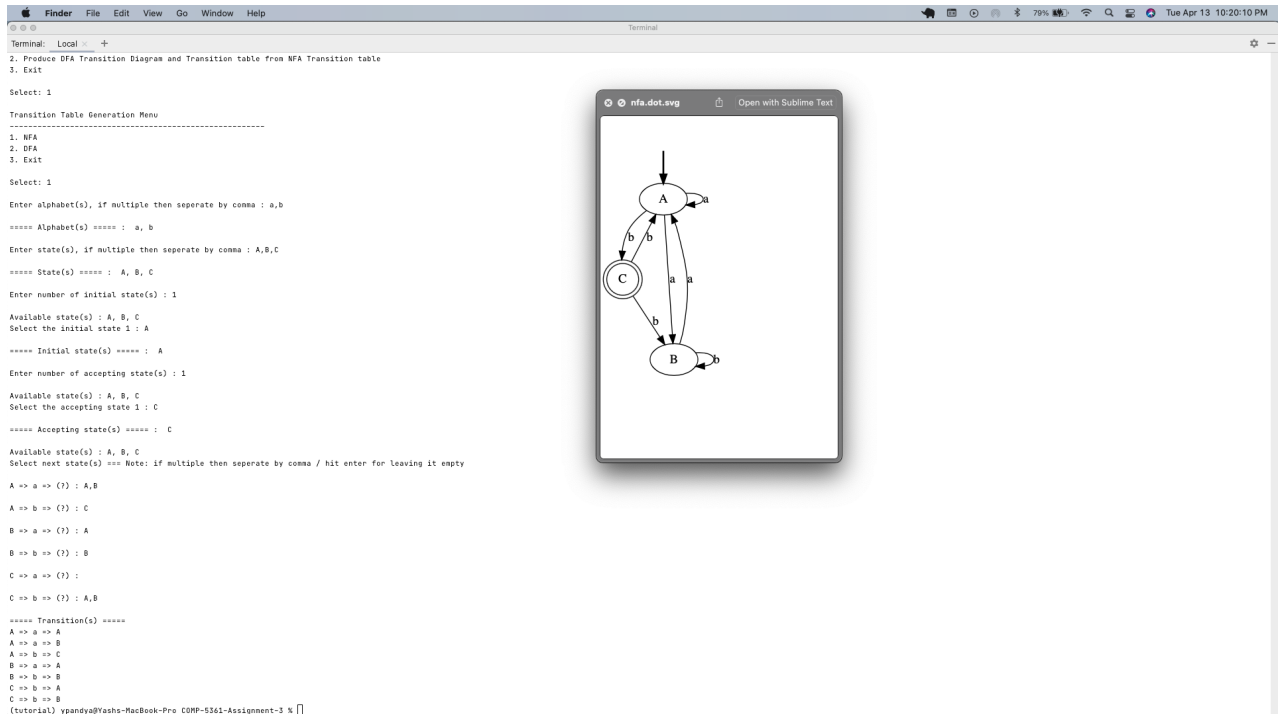
=====

(tutorial) ypandya@Yashs-MacBook-Pro COMP-5361-Assignment-3 %

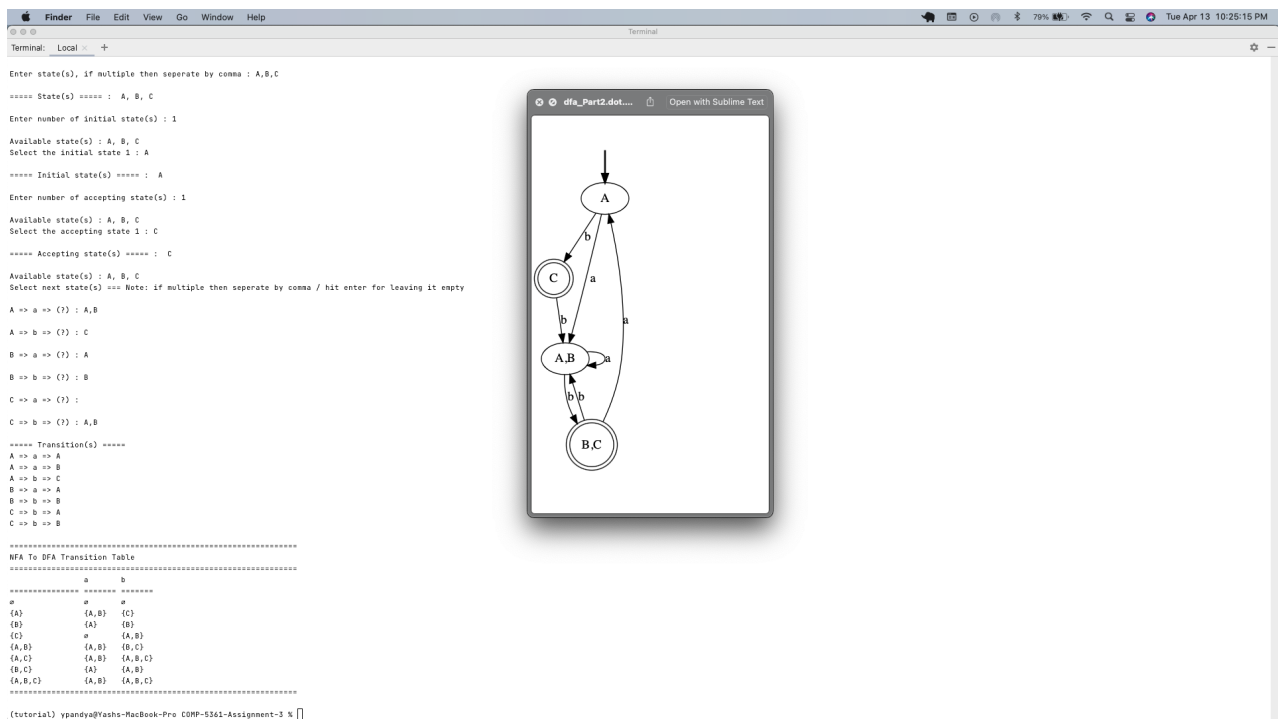


Test case 1(b) - Equivalent DFA

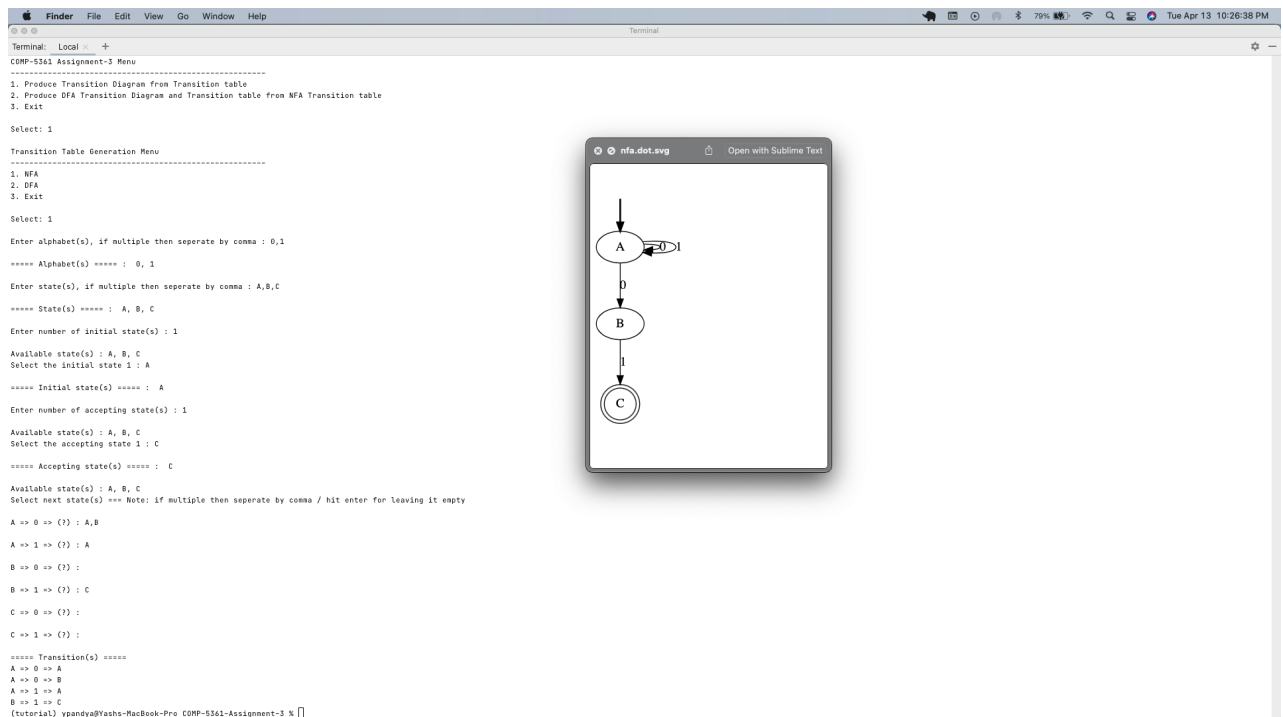




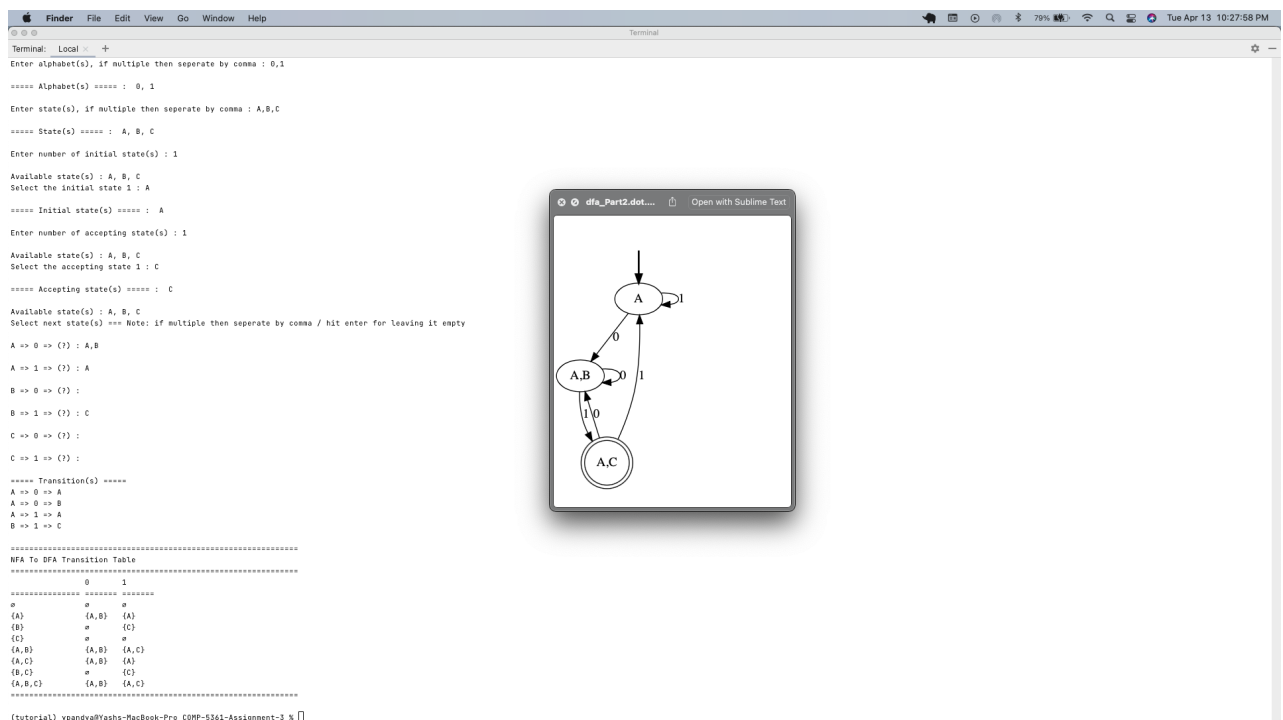
Test case 2(a) - NFA



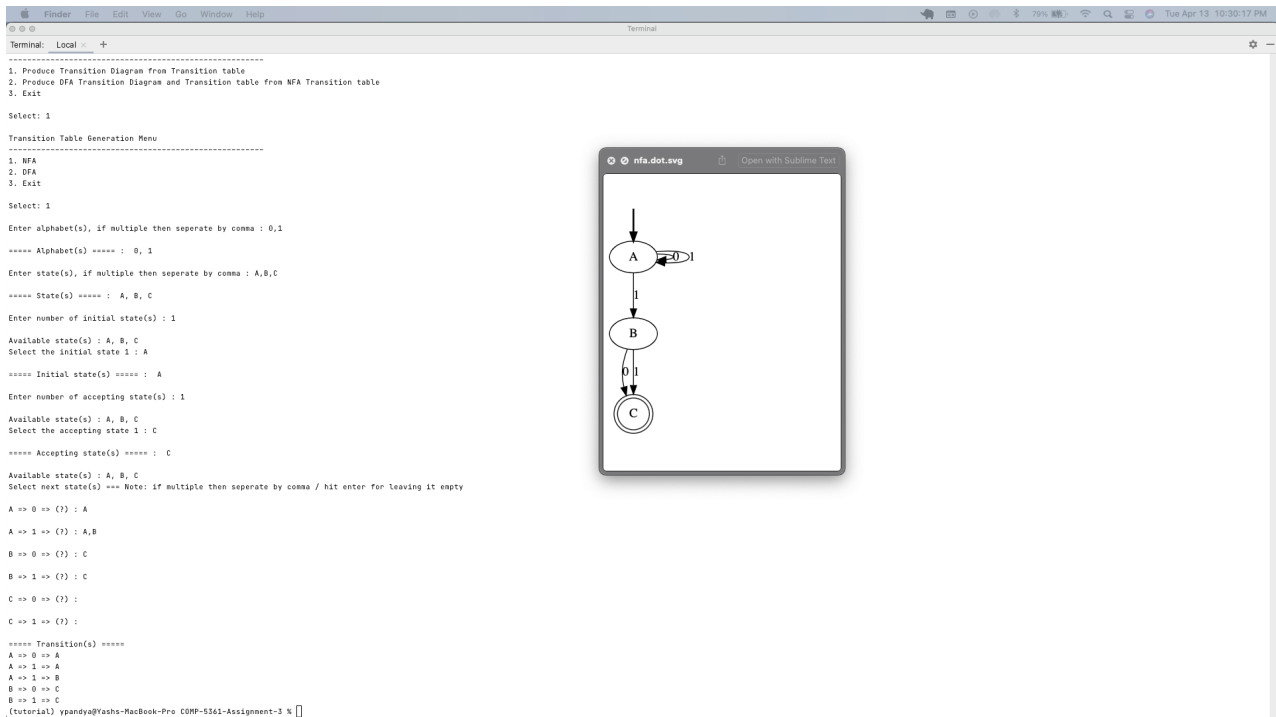
Test case 2(b) - Equivalent DFA



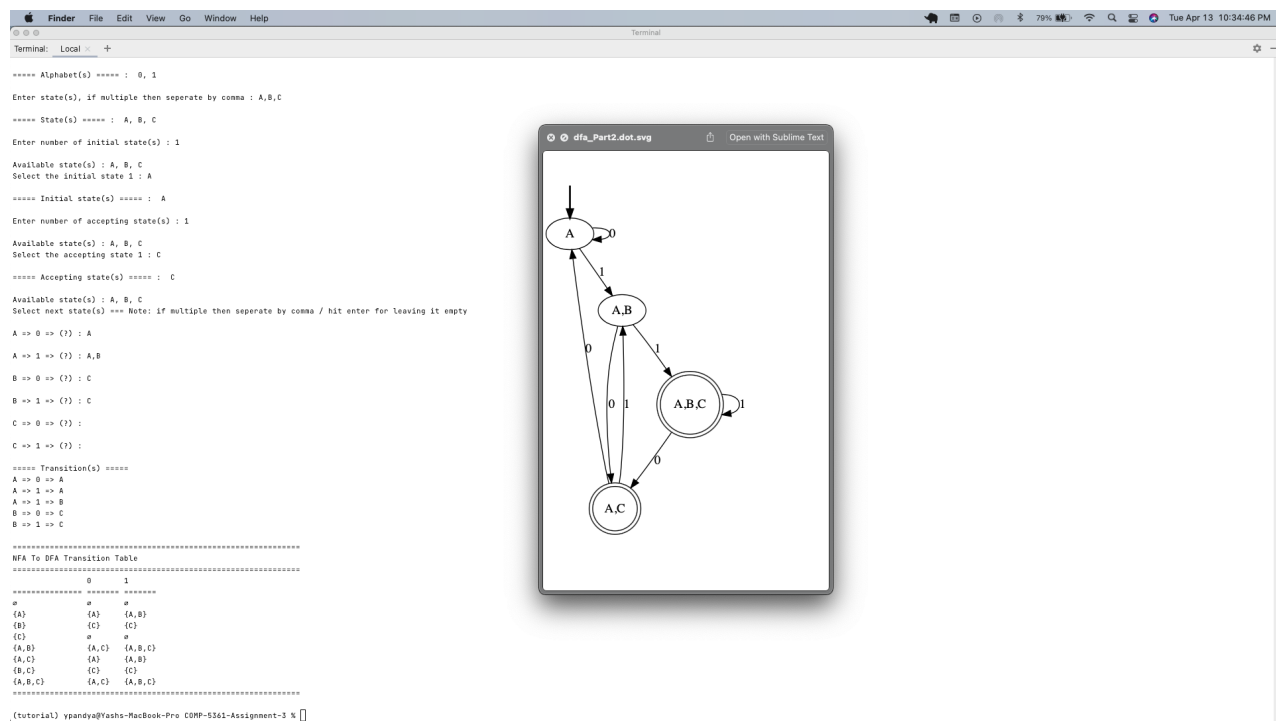
Test case 3(a) - NFA



Test case 3(b) - Equivalent DFA



Test case 4(a) - NFA



Test case 4(b) - Equivalent DFA

## References:

- <https://stackoverflow.com/questions/53526207/how-do-i-add-a-row-of-dashes-between-the-first-two-print-lines-in-python>
- <https://pysimpleautomata.readthedocs.io/en/latest/tutorial.html>
- Test case 1: <https://www.youtube.com/watch?v=ponyXglXpKnc>
- Test case 2: <https://www.youtube.com/watch?v=i-fk9o46oVY>
- Test case 3: <https://www.youtube.com/watch?v=dY1bCC6syLI>
- Test case 4: <https://www.youtube.com/watch?v=Y92dtMnarAU>