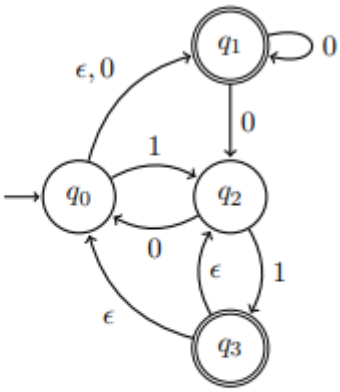
NFA to DFA converter – Explanation of the program

This program illustrates all the steps from converting 'non-deterministic finite automaton' (NFA) to 'deterministic finite automaton (DFA). There are four main functions that shows how this conversion process occur. It includes calculating 'epsilon closures' from the given NFA, constructing an equivalent epsilon free NFA from given epsilon closures, constructing an equivalent DFA from given epsilon free NFA and deciding if the strings are in the language by looking at the DFA.

Understanding the format of input/output

The example files include specific format to test the selected function with given parameter it requires.

```
epsilon-closure
q0,q1,q2,q3
0,1
q0
q1,q3
q0,0,q1
q0,1,q2
q0,,q1
q1,0,q1
q1,0,q2
q2,0,q0
q2,1,q3
q3,,q0
q3,,q2
end
```



This is an example of the input and illustration of it for the program.

The index refers to the line number:

1. Selecting what kind of function to run
    - There are 4 types of functions:
        - epsilon-closure
            - Compute epsilon closures from given NFA
        - nfa-to-enfa
            - Construct ENFA from given epsilon closures
        - enfa-to-dfa
            - Construct DFA from given ENFA
        - compute-dfa
            - Decide if the given string are in the language of the DFA
2. List of states that exists in the automaton
3. List of alphabets in the language
4. Starting State
5. Accept State

The remaining lines shows the transitions “δ(q, Σ) = qs”

Where:

- δ: Transition
- q: State
- Σ: Alphabet
- qs: All states that can be reached from a state by the given alphabet

OR

Shows epsilon closures q: qs

Where:

- q: State
- qs: All states that can be reached from a state by the epsilon transitions.

For “compute-dfa” it also have lists of language to check if it’s in the DFA or not. 1 = True, 0 = False.

Then it indicates the end of the input by “end”.

Testing/Running the program

Simply download the ‘NFAtoDFA’ folder and run the ‘main.py’ from a console using the command:

- python3 main.py < tests/filename.in

There are premade input examples under the test folder that can be used.

Additional Information

This program was not developed for commercial use/professional use which means incorrect input cannot be handled by the program. The time complexity of each functions isn’t optimised and where recursion could be used for better optimisation hasn’t been used.