German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail

CSEN1002 Compilers Lab, Spring Term 2020
Task 2: NFA

Due: Week starting 15.02.2020

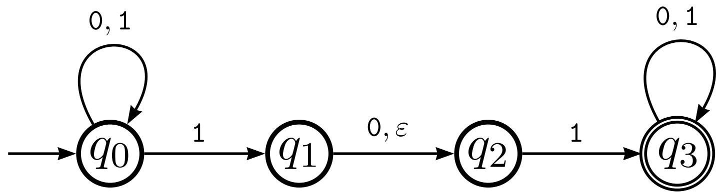
1 Objective

For this task you need to implement the classical algorithm for constructing a deterministic finite automaton (DFA) equivalent to a non-deterministic finite automaton (NFA). Recall that an NFA is a quintuple $(Q, \Sigma, \delta, q_0, F)$: Q is a non-empty, finite set of states; Σ is non-empty, finite set of symbols (an alphabet); $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow \mathcal{P}(Q)$ is the transition function; $q_0 \in Q$ is the start state; and $F \subseteq Q$ is the set of accept states. Given a description of an NFA, you need to construct an equivalent DFA.

2 Requirements

- You may use the programming language of your choice.
- We make the following assumptions for simplicity.
  a) The alphabet Σ is always the binary alphabet $\{\texttt{0}, \texttt{1}\}$.
  b) The set of states Q is always of the form $\{0, \ldots, n\}$, for some $n \in \mathbb{N}$.
  c) The start state is always state 0.
- You should implement two functions: `dfa` and `run`.
- `dfa` (which could be a class constructor) takes one parameter which is a string description of an NFA (not a DFA!) and returns (or constructs) an equivalent DFA.
- A string describing an NFA is of the form $Z\#O\#E\#F$, where Z, O, and E, respectively, represent the `0`-transitions, the `1`-transitions, and the ε-transitions. F represents the set of accept state.
- Z, O, and E are semicolon-separated sequences of pairs of states; each pair is a comma-separated sequence of two states. A pair i, j represents a transition from state i to state j; for Z this means that $\delta(i, \texttt{0}) = j$, similarly for O and E.
- F is a comma-separated sequence of states.
- For example, the NFA for which the state diagram appears below may have the following string representation.

```
0,0;1,2;3,3#0,0;0,1;2,3;3,3#1,2#3
```



- `run` simulates the operation of the constructed DFA on a given binary string. It returns `true` if the string is accepted by the DFA and `false` otherwise.

3 Evaluation

- Your implementation will be tested by constructing the DFA equivalent to two NFA and running each on five strings.
- You get one point for each correct output of `run`; hence, a maximum of ten points.