

CSEN1002 Compilers Lab, Spring Term 2022  
Task 2: NFA

Due: Week starting 05.03.2022

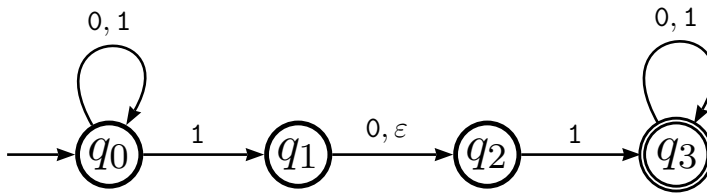
## 1 Objective

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;  $\Sigma$  is non-empty, finite set of symbols (an alphabet);  $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$  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

- We make the following assumptions for simplicity.
  - a) The alphabet  $\Sigma$  is always the binary alphabet  $\{0, 1\}$ .
  - b) The set of states  $Q$  is always of the form  $\{0, \dots, n\}$ , for some  $n \in \mathbb{N}$ .
  - c) The start state is always state 0.
- You should implement a class constructor `NFA` and a method `run`.
- `NFA` takes one parameter which is a string description of an NFA 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  $\varepsilon$ -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  $\delta(i, 0) = j$ , 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.
- Important Details:
  - Your implementation should be done within the template file “`NFA.java`” (uploaded to the CMS).
  - You are not allowed to change package, file, constructor, or method names/signatures.
  - You are allowed to implement as many helper classes/methods within the same file (if needed).
  - Public test cases have been provided on the CMS for you to test your implementation.
  - Please ensure that the public test cases run correctly without modification before coming to the lab to maintain a smooth evaluation process.
  - Private test cases will be uploaded before your session and will have the same structure as the public test cases.

### 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.
- The evaluation will take place during your lab session of the week starting Saturday March 5.

### 4 Online Submission

- You should submit your code at the following link.

<https://forms.gle/zeRR5B47HE96VzYQ9>

- Submit one Java file (`NFA.java`) containing executable code.
- Online submission is due on Thursday, March 17th, by 23:59.