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

CSEN1002 Compilers Lab, Spring Term 2023
Task 2: Non-Deterministic Finite Automata to Deterministic Finite Automata

Due: Week starting 04.03.2023

1 Objective

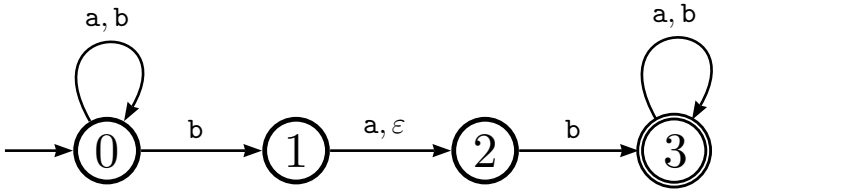
For this task you need to implement the classical algorithm for constructing a deterministic finite automaton (DFA) equivalent to a given non-deterministic finite automaton (NFA). Recall that an NFA is a quintuple $(Q, \Sigma, \delta, q_0, F)$: Q is a non-empty, finite set of states; Σ is non-empty, 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

- We make the following assumptions for simplicity.
  a) The alphabet Σ is always a subset of the Latin alphabet, not including `e`.
  b) The letter "`e`" represents ε.
  c) The set of NFA states Q is always of the form $\{0, \ldots, n\}$, for some $n \in \mathbb{N}$.
- You should implement a class constructor `NfaToDfa` and a method `toString`.
- `NfaToDfa` takes one parameter which is a string description of an NFA and constructs an equivalent DFA. A string describing an NFA is of the form $Q\#A\#T\#I\#F$.
  - Q is a string representation of the set of states; a semicolon-separated sequence of sorted integer literals.
  - A is a string representation of the input alphabet; a semicolon-separated sequence of alphabetically sorted symbols
  - T is a string representation of the transition function. T is a semicolon-separated sequence of triples. Each triple is a string representing a single transition; a comma-separated sequence i, a, j where i is a state of Q, a a symbol of A or `e`, and j a state of Q representing a transition from i to j on input a. These triples are sorted by the source state i, then (if the same state has more than one outgoing transition) by the input a, and then (if multiple triples share the same source state and input, due to non-determinism) by the destination state j.
  - I is an integer literal representing the initial state.
  - F is a string representation of the set of accept states; a semicolon-separated sequence of sorted integer literals.

– For example, the NFA for which the state diagram appears below may have the following string representation.

`0;1;2;3#a;b#0,a,0;0,b,0;0,b,1;1,a,2;1,e,2;2,b,3;3,a,3;3,b,3#0#3`



- `toString` returns a string representation of the constructed DFA. A string representation of a DFA returned by `toString` is similar to that of an NFA—a string of the form $Q\#A\#T\#I\#F$.
  - However, states of such DFA are sets of states of the original NFA. Hence, only the representation of states in the string encoding of DFA is different from that of NFA. A DFA state is represented as a `/`-separated sequence of numerals, with the numerals representing NFA states.
  - These sequences are sorted by their first numerals (assuming the natural order of numerals). Two sequences starting with the same numeral are sorted according to the order of their respective suffixes resulting from dropping the first numeral. The empty sequence precedes any sequence.
  - A DFA state corresponding the empty set of NFA states is represented by `-1`.
  - Thus, following the classical construction, the following is a (split for readability) string representing a DFA equivalent to the NFA in the above figure.

    `0;0/1/2;0/1/2/3;0/2;0/2/3;0/3#a;b#0,a,0;0,b,0/1/2;0/1/2,a,0/2;`
    `0/1/2,b,0/1/2/3;0/1/2/3,a,0/2/3;0/1/2/3,b,0/1/2/3;0/2,a,0;0/2,b,0/1/2/3;`
    `0/2/3,a,0/3;0/2/3,b,0/1/2/3;0/3,a,0/3;0/3,b,0/1/2/3#0#0/1/2/3;0/2/3;0/3`

- Important Details:
  - Your implementation should be done within the template file "`NfaToDfa.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 ten input NFAs.
- You get one point for each correct output of `toString`; hence, a maximum of ten points.

- The evaluation will take place during your lab session of the week starting Saturday March 4.

4 Online Submission

- You should submit your code at the following link.

  `https://forms.gle/PwEuXjdsRXzdRPXy9`

- Submit one Java file (`NfaToDfa.java`) containing executable code.
- **Online submission is due by the end of your lab session.**