CSE 211: Non-deterministic Finite Automaton

Md. Shaifur Rahman

Dept. of CSE
Bangladesh University of Engineering & Technology

Class 4

Outline

Outline

Outline

What is NFA

Non-Deterministic Finite Automaton (NFA)

- For a single input symbol, transition from the current state to one or more states
- Transition from current state to one or more states without consuming any input symbol (in other word, transition consuming ϵ)

What is NFA

Non-Deterministic Finite Automaton (NFA)

- For a single input symbol, transition from the current state to one or more states
- Transition from current state to one or more states without consuming any input symbol (in other word, transition consuming ϵ)

Advantage

- Easier and simpler state diagram
- More intuitive

What is NFA

Non-Deterministic Finite Automaton (NFA)

- For a single input symbol, transition from the current state to one or more states
- Transition from current state to one or more states without consuming any input symbol (in other word, transition consuming ϵ)

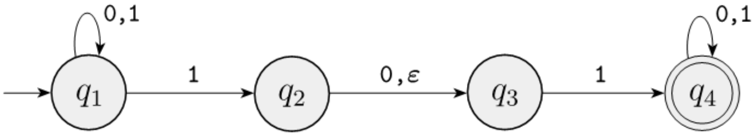
Advantage

- Easier and simpler state diagram
- More intuitive

Disadvantage

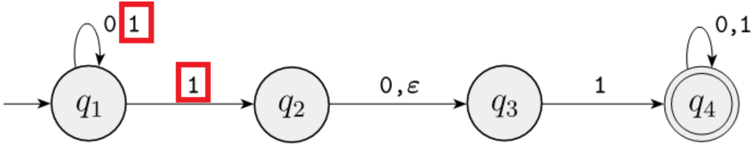
- Harder computation for decision-making

NFA Example



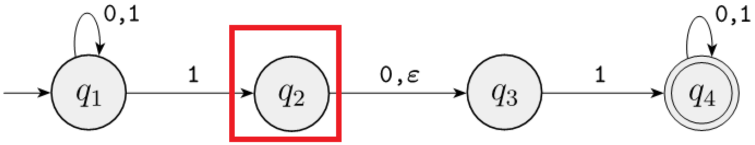
- NFA N_1 has four states

NFA Example



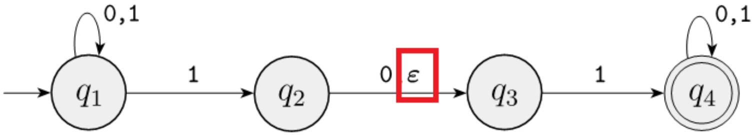
- NFA N_1 has four states
- For input 1, $q_1 \rightarrow q_1$ OR $q_1 \rightarrow q_2$

NFA Example



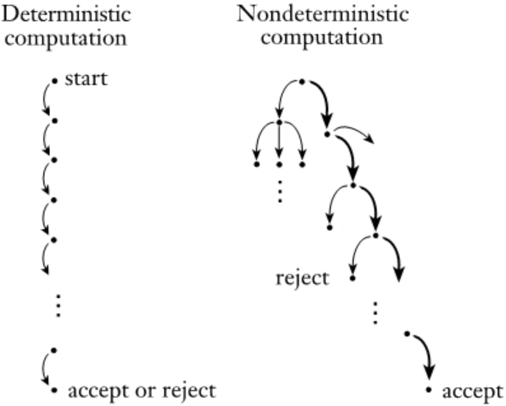
- NFA N_1 has four states
- For input 1, $q_1 \rightarrow q_1$ OR $q_1 \rightarrow q_2$
- State q_2 has only transition for input 0 BUT not for 1

NFA Example



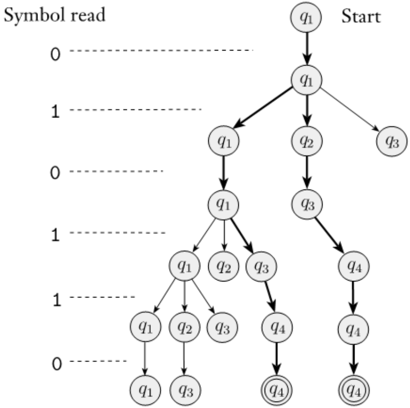
- NFA N_1 has four states
- For input 1, $q_1 \rightarrow q_1$ OR $q_1 \rightarrow q_2$
- State q_2 has only transition for input 0 BUT not for 1
- N_1 can transit $q_2 \rightarrow q_3$ without any input!

Examples of NFA (Contd.)

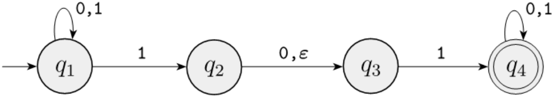


NFA Decision Tree

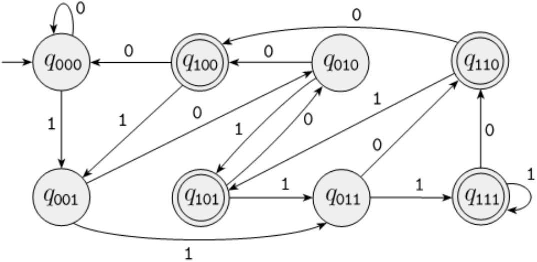
Examples of NFA (Contd.)



NFA Decision Tree for N_1 for computation of 010110



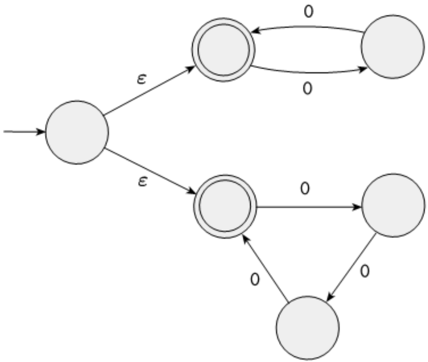
NFA N_1



Equivalent DFA D_1

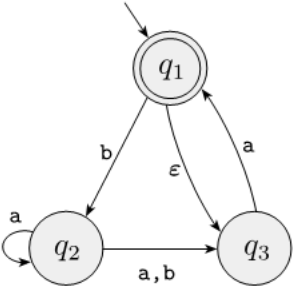
Examples of NFA (Contd.)

An NFA which all strings of the form 0^k where k is a multiple of 2 or 3



Examples of NFA (Contd.)

An NFA which all strings of the form $\epsilon, a, baba, baa, \ldots$
But does not accept $b, bb, babba$ etc.



Formal Defintion of NFA

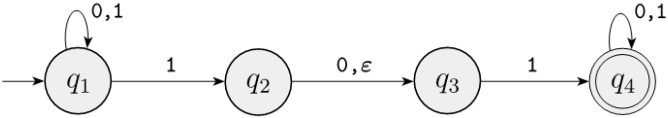
NFA

a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set of states
2. σ is a finite alphabet
3. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$
4. $q_0 \in Q$ is the start-state
5. $F \subseteq Q$ is the accept-state set

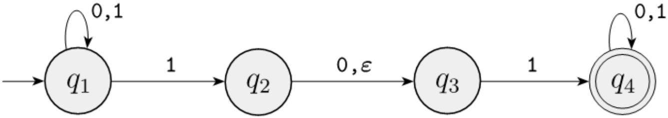
Examples of NFA (Contd.)

Give the formal definition of for the following NFA:



Examples of NFA (Contd.)

Give the formal definition of for the following NFA:



- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- start-state, $q_0 = q_1$
- accept-states, $F = \{q_4\}$

δ is:

	0	1	ϵ
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
q_2	$\{q_3\}$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$	$\varnothing$

Computation by NFA

Formal Definition of Computation by NFA

$N = (Q, \Sigma, \delta, q_0, F)$ is an NFA, w is a string of Σ
N accepts w if $w = y_1 \, y_2 \ldots y_m$ where each $y_i \in \Sigma_\epsilon$ and sequence of states $r_0, r_1, \ldots, r_m$ each exists in Q such that:

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $m = 0, 1, \ldots, m-1$ and
3. $r_m \in F$

Equivalence of NFA and DFA

Equivalence of Machines

Two machines are equivalent if they recognize the same languages

Theorem 1.39

Every NFA has an equivalent DFA

Equivalence of NFA and DFA

Equivalence of Machines

Two machines are equivalent if they recognize the same languages

Theorem 1.39

Every NFA has an equivalent DFA

Proof Idea

Convert the NFA into an equivalent DFA that simulates it. Issues are:

- How will you simulate the NFA by the DFA?
- How will you keep track of the input and branches of computation as the input is processed?
- For a k-state NFA, the DFA may have to remember 2^k states!
- What will be the start-state, accept-state and transition function of the DFA?

NFA has an equivalent DFA ... proof contd.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA
and $M = (Q', \Sigma, \delta', q_0', F')$ be the equivalent DFA. Both N and M recognizes the same language A.

Case I: There is no ϵ arrow in the NFA N.

1. $Q' = \mathcal{P}(Q)$, because every state of M is a set of states of N
2. $q_0' = \{q_0\}$
3. $F' = \{R \in Q' |\ R$ contains an accept-state of $N\}$
4. For $R \in Q'$ and $a \in Q'$
   $\delta'(R, a) = \{q \in Q |\ q \in \delta(r, a)$ for some $r \in R\}$
   $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$ set of all states reachable from all states of R for input a according to the δ of N

NFA=DFA (Contd.)

NFA has an equivalent DFA . . . proof contd.

Case II: There are ϵ arrow in the NFA N.

- For any state R of M, $E(M)$: collection of states reachable from any state of set R going along the ϵ-arrows, $R \subseteq Q$
- Formally, $E(R) = \{q |\, q$ is reachable from R by traveling along 0 or more ϵ arrows $\}$
- Replacing $\delta(r, a)$ by $E(\delta(r, a))$
  $\delta'(R, a) = \{q \in Q |\, q \in E(\delta(r, a))$ for some $r \in R\}$
- $q_0' = E(\{q_0\})$

Does the construction of M works correctly?

NFA=DFA (Contd.)

NFA has an equivalent DFA ... proof contd.

Case II: There are ϵ arrow in the NFA N.

- For any state R of M, $E(M)$: collection of states reachable from any state of set R going along the ϵ-arrows, $R \subseteq Q$
- Formally, $E(R) = \{q | \, q$ is reachable from R by traveling along 0 or more ϵ arrows $\}$
- Replacing $\delta(r, a)$ by $E(\delta(r, a))$
  $\delta'(R, a) = \{q \in Q | \, q \in E(\delta(r, a))$ for some $r \in R\}$
- $q'_0 = E(\{q_0\})$

Does the construction of M works correctly?
At every step of computation on an input, M enters a state that corresponds to the subset of states that N could be in at that point.

NFA=DFA (Contd.)

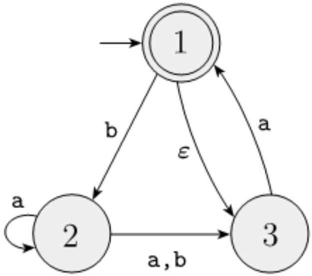
Corollary 1.40

A language is regular if and only if some non-deterministic finite automaton recognizes it

Proof

- A language is regular if some NFA recognizes it
  If some NFA recognizes the language, so does an equivalent DFA. Hence, the language is regular.
- A language is regular only if some NFA recognizes it
  If a language is regular, a DFA recognizes it. A DFA is also an NFA. So, an NFA also recognizes the regular language.

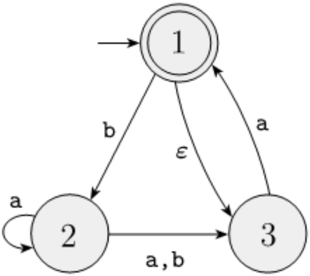
Examples of Conversion from NFA to DFA



$DFA = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \}$
- $\Sigma = \{a, b\}$
- $q_0 = E(\{1\}) = \{1, 3\}$
- $F = \{$ All states of Q that have a final state of $N\}$
  $=\{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$

Examples of Conversion from NFA to DFA (Contd.)

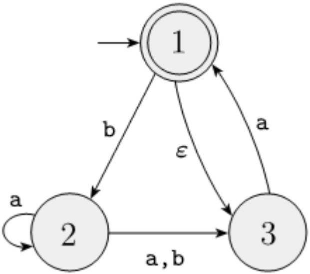


Transition Function δ=?

- On input a: $\{2\} \rightarrow \{2,3\}$, on input b: $\{2\} \rightarrow \{3\}$
- On input a: $\{1\} \rightarrow \varnothing$, on input b: $\{1\} \rightarrow \{2\}$
- On input a: $\{3\} \rightarrow \{1,3\}$, on input b: $\{3\} \rightarrow \varnothing$
- On input a: $\{1,2\} \rightarrow \{2,3\}$, on input b: $\{1,2\} \rightarrow \{2,3\}$
- Similarly, for rest of the states ...

Correction to class lecture

attention: follow ϵ after applying $\delta()$, not before!



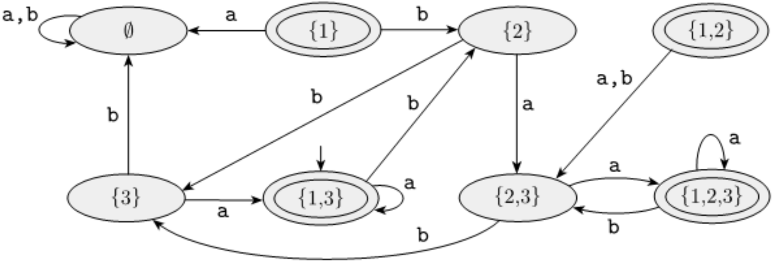
$\delta'(\{1\}, a) = \varnothing$

INCORRECT!, $\delta'(E(\{1\}), a) = \delta'(\{1, 3\}, a) = \{1\}$

But, $\delta'(\{3\}, a) = E(\{1\}) = \{1, 3\}$

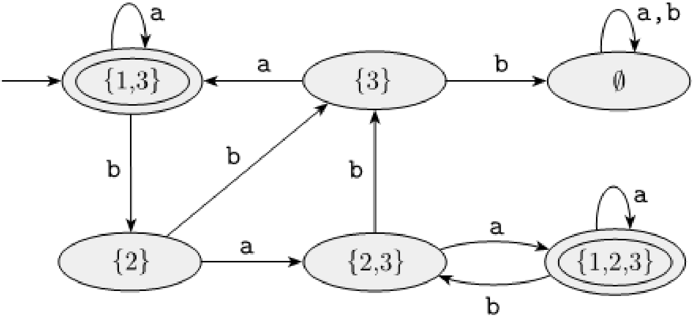
Examples of Conversion from NFA to DFA (Contd.)

After conversion . . .



Examples of Conversion from NFA to DFA (Contd.)

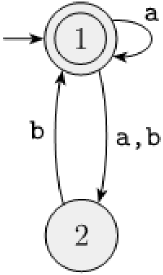
After conversion and simplification . . .



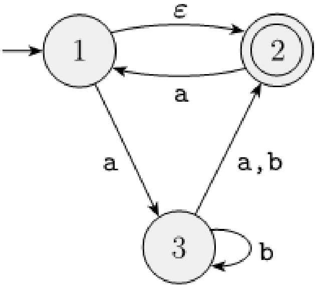
Dropping the states from which there is no outgoing arrow

Try Yourself

Convert the following NFA into the equivalent DFA



(a)



(b)

Question?