Finite Automata

- Deterministic Finite Automata
- Languages decided by a DFA – Regular Languages
- Closure Properties of regular languages
- Non-Deterministic Finite Automata, DFA= NFA
- Regular Expression: Computation as Description
- DFA=NFA=RegExp, Generalized NFA
- Non-Regular Languages, The Pumping Lemma
- Minimizing DFA

Imdad ullah Khan

Non-Deterministic Finite Automata

Are Regular Languages closed under "Reversal"?

Reverse $L^R = \{w^R : w \in L\}$

$L = \{\epsilon, a, ab, aab, aaab, aaaab\} \implies L^R = \{\epsilon, a, ba, baa, baaa, baaaa\}$

If L is regular, then is L^R also regular?

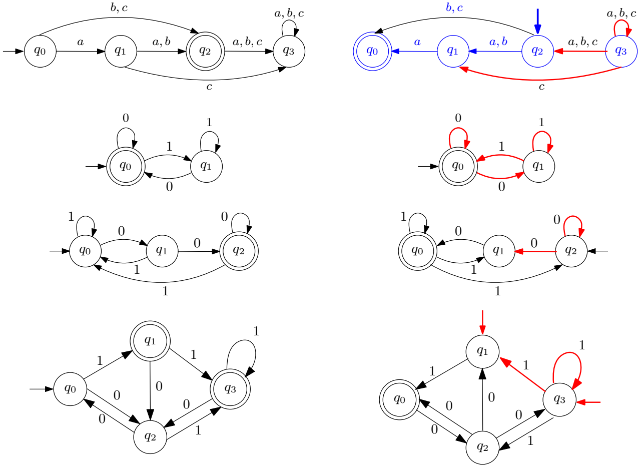
$M = (Q, \Sigma, q_0, \delta, F)$: DFA recognizing L

M accepts $w \implies w$ describes a directed "walk" from start to final state

What if we make M^R by reversing all edge directions, making start state a final state and turning final states into start state(s)

▷ Essentially, a "reverse DFA" that reads strings from right to left

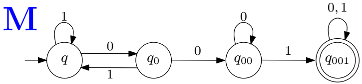
What if we make M^R by reversing all edge directions, making start state a final state and turning final states into start state(s)

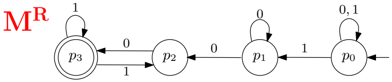


- multiple start states
- states with ≥ 1 transitions for σ
- states with 0 transitions for σ

Issues with "Reverse" DFA

What if we make M^R by reversing all edge directions, making start state a final state and turning final states into start state(s)





$L(M) = \{x001y : x, y \in \{0,1\}^*\}$

Ignoring the issues with M^R

Run M_R on $100 \in L^R$ - Move/stay choices at p_0 and p_1 on input 1 and 0

There is a walk (with a set of choices) that "accepts" string $100 = 001^R$

There is a walk (with a set of choices) that "accepts" $010011 = 110010^R$

There is a walk (with a set of choices) that "accepts" 000100

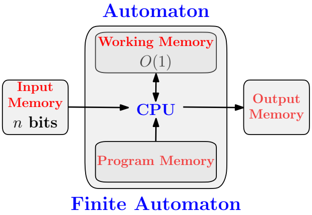
There is no walk (with any set of choices) that "accepts" 001111

There is no walk (with any set of choices) that "accepts" 1111

We got a creature that somewhat recognizes reverse of a regular language

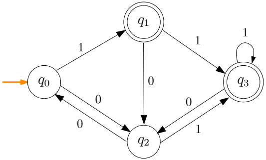
Non-Deterministic Finite Automata (NDFA)

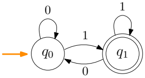
A Non Deterministic Finite Automata has constant working memory and perfect guessing capability



A nondeterministic finite automata or finite state machine is a little creature

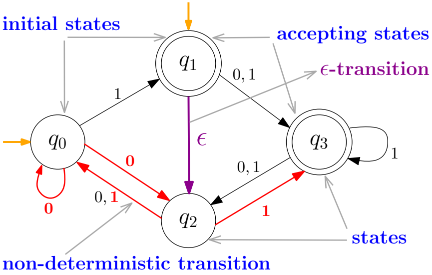
- it has tiny eyes that can see only one symbol
- it changes its state of mind according to the symbol it sees or without it
- it can only remember its current state of mind
- it can make choice to change its state



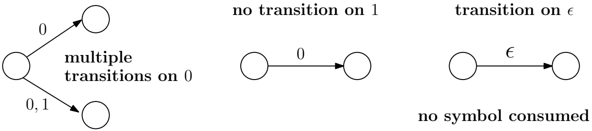


Anatomy of NFA

NFA over Σ depicted as digraph with self-loops



- Multiple start states allowed
- Any number of outgoing arrows allowed from a state for a symbol σ
- Transitions on an empty string ϵ are also allowed



NFA accepts string w if there is some walk from a start to accept state

NFA: Formal Definition

An NFA N is a 5-tuple $N = (Q, \Sigma, Q_0, \delta, F)$

- Q: A finite set of states
- Σ: Alphabet ▷ A finite set of characters
- $Q_0 \subseteq Q$ A set of start or initial states
- $\delta : Q \times \big(\Sigma \cup \{\epsilon\}\big) \mapsto \mathcal{P}\big(Q\big)$ Non-deterministic transition function
- $F \subseteq Q$ Set of accept/final states

N accepts $w \in \Sigma^*$ if there is a sequence $r_0, r_1, \ldots, r_n$ and w can be written as $w_1, w_2, \ldots, w_n$ with $w_i \in \Sigma \cup \{\epsilon\}$ such that

- $r_0 \in Q_0$
- $r_{i+1} \in \delta(r_i, w_{i+1})$
- $r_n \in F$

$L(N)$: the language reccognized by N = set of strings N accepts

Languages of NFA

$$N = (Q, \Sigma, Q_0, \delta, F)$$

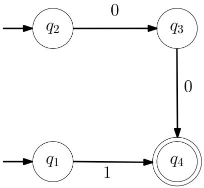
$$Q = \{q_1, q_2, q_3, q_4\}$$

$$\Sigma = \{0, 1\}$$

$$Q_0 = \{q_1, q_2\}$$

$$\delta(q_1, 1) = \{q_4\},\ \delta(q_2, 0) = \{q_3\},$$
$$\delta(q_3, 0) = \{q_4\}$$

$$F = \{q_4\}$$



$L(N) = \{00, 1\}$?

Languages of NFA

$N = (Q, \Sigma, Q_0, \delta, F)$

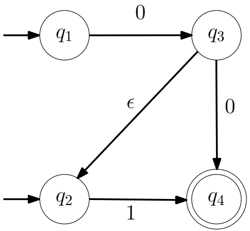
$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$Q_0 = \{q_1, q_2\}$

$\delta(q_1, 0) = \{q_3\}$, $\delta(q_2, 1) = \{q_4\}$,
$\delta(q_3, 0) = \{q_4\}$, $\delta(q_3, \epsilon) = \{q_2\}$

$F = \{q_4\}$



$0 \in L(N)$? $1 \in L(N)$? $00 \in L(N)$? $01 \in L(N)$? $10 \in L(N)$?
$11 \in L(N)$?

Languages of NFA

$N = (Q, \Sigma, Q_0, \delta, F)$

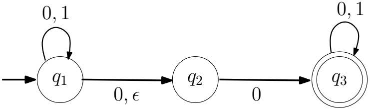
$Q = \{q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$Q_0 = \{q_1\}$

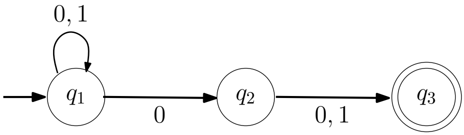
$\delta(q_1, 0) = \{q_1, q_2\}$, $\delta(q_1, 1) = \{q_1\}$,
$\delta(q_1, \epsilon) = \{q_2\}$, $\delta(q_2, 0) = \{q_3\}$,
$\delta(q_3, 0) = \{q_3\}$, $\delta(q_3, 1) = \{q_3\}$

$F = \{q_3\}$



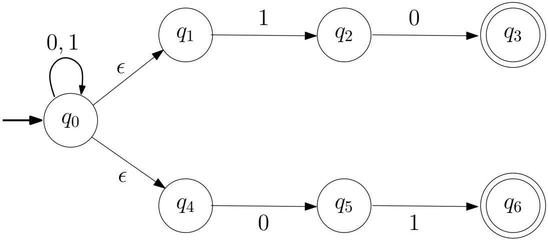
$\mathbf{L(N)} = \{\mathbf{w} : \mathbf{w} \text{ contains a } \mathbf{0}\}$

Languages of NFA



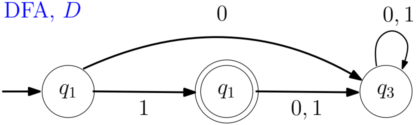
$$\mathbf{L}(\mathbf{N}) = \{\mathbf{w} : \mathbf{w} \text{ ends with } \mathbf{00} \text{ or } \mathbf{01}\}$$

Languages of NFA

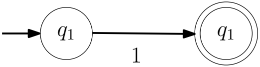


NFA are generally significantly simpler than DFA

DFA, D



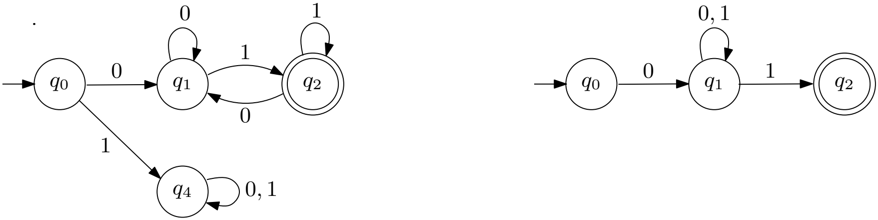
NFA, N



$$\mathbf{L(D) = L(N) = \{1\}}$$

NFA vs DFA

NFA are generally significantly simpler than DFA



$$\mathbf{L(D) = L(N) = \{w : w \text{ begins with } 0 \text{ and ends with } 1\}}$$

NFA vs DFA

Multiple start states is not an issue

Some authors require even an NFA to have exactly one start state

Any NFA with multiple start states can be converted to one with one start state as follows