COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

January 11, 2023

Lecture 4: Closure properties of regular languages, nondeterminism

Recap

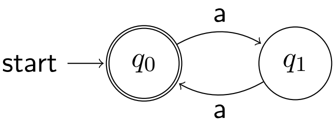
- Languages, Decision problems.
- Finite State Automata - devices with finite memory.
- Deterministic Finite State Automata (DFA)
  - From one state, reading an action we move to exactly one other state.
  - So, for each input word, there is exactly one run!
- Regular languages
  - L is regular if there exists some DFA A such that $L = L(A)$.
  - Closed under Union, Intersection, Complement.
  - Other operations of languages: Concatenation ($L \circ L'$), Kleene star (L^*)

Building complicated DFAs from simple ones

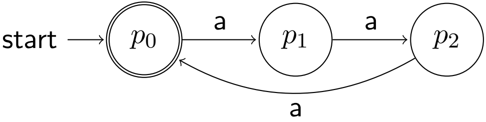
Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$



Let $L_2 = \{w \mid |w| \equiv 0 \pmod 3\}$



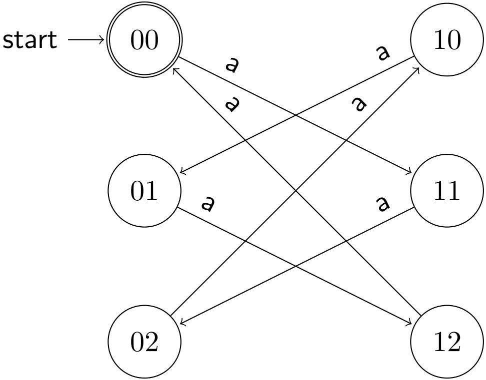
What is $L_1 \cap L_2$?

$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \pmod 6\}$

Building complicated DFAs from simple ones

Example

$$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \pmod 6\}$$



Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$
\begin{array}{rcl}
Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\
q_0 & = & (q_0^1, q_0^2) \\
F & = & \{(q, q') \mid q \in F_1, \text{ and } q' \in F_2\} = F_1 \times F_2 \\
\delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a))
\end{array}
$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{ccc} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \\ F & = & \{(q, q') \mid q \in F_1, \text{ and } \; q' \in F_2\} = F_1 \times F_2 \\ \delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a)) \end{array}$$

Correctness

$\forall w \in \Sigma^*$, w is accepted by A iff w is accepted by both A_1 and A_2.

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a), \delta_2(q',a))$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a),\delta_2(q',a))$

Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q') \\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a),\delta_2(q',a))$
Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q') \\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p,q),x) = (\hat{\delta}_1(p,x),\hat{\delta}_2(q,x))$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a),\delta_2(q',a))$
Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q') \\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p,q),x) = (\hat{\delta}_1(p,x),\hat{\delta}_2(q,x))$

Claim: $L(A) = L(A_1) \cap L(A_2)$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a),\delta_2(q',a))$
Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q')\\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p,q),x) = (\hat{\delta}_1(p,x),\hat{\delta}_2(q,x))$

Claim: $L(A) = L(A_1) \cap L(A_2)$

$$x \in L(A) \iff \hat{\delta}(q_0,x) \in F$$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a),\delta_2(q',a))$
Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q') \\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p,q),x) = (\hat{\delta}_1(p,x),\hat{\delta}_2(q,x))$

Claim: $L(A) = L(A_1) \cap L(A_2)$

$$\begin{aligned}x \in L(A) &\iff \hat{\delta}(q_0,x) \in F \\ &\iff \hat{\delta}((q_0^1,q_0^2),x) \in (F_1 \times F_2)\end{aligned}$$

Proof of Correctness

Proof.

Recall, $\delta((q, q'), a) := (\delta_1(q, a), \delta_2(q', a))$
Inductively, we have

$$\begin{aligned} \hat{\delta}((q, q'), \varepsilon) &= (q, q') \\ \hat{\delta}((q, q'), xa) &= \delta(\hat{\delta}((q, q'), x), a) \end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p, q), x) = (\hat{\delta}_1(p, x), \hat{\delta}_2(q, x))$

Claim: $L(A) = L(A_1) \cap L(A_2)$

$$\begin{aligned} x \in L(A) &\iff \hat{\delta}(q_0, x) \in F \\ &\iff \hat{\delta}((q_0^1, q_0^2), x) \in (F_1 \times F_2) \\ &\iff (\hat{\delta}_1(q_0^1, x), \hat{\delta}_2(q_0^2, x)) \in (F_1 \times F_2) \end{aligned}$$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a), \delta_2(q',a))$
Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q')\\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p,q),x) = (\hat{\delta}_1(p,x), \hat{\delta}_2(q,x))$

Claim: $L(A) = L(A_1) \cap L(A_2)$

$$\begin{aligned}x \in L(A) &\iff \hat{\delta}(q_0,x) \in F\\ &\iff \hat{\delta}((q_0^1,q_0^2),x) \in (F_1 \times F_2)\\ &\iff (\hat{\delta}_1(q_0^1,x), \hat{\delta}_2(q_0^2,x)) \in (F_1 \times F_2)\\ &\iff \hat{\delta}_1(q_0^1,x) \in F_1 \text{ and } \hat{\delta}_2(q_0^2,x) \in F_2\end{aligned}$$

Proof of Correctness

Proof.

Recall, $\delta((q,q'),a) := (\delta_1(q,a), \delta_2(q',a))$
Inductively, we have

$$\begin{aligned}\hat{\delta}((q,q'),\varepsilon) &= (q,q') \\ \hat{\delta}((q,q'),xa) &= \delta(\hat{\delta}((q,q'),x),a)\end{aligned}$$

Claim: $\forall x \in \Sigma^*$, $\hat{\delta}((p,q),x) = (\hat{\delta}_1(p,x), \hat{\delta}_2(q,x))$

Claim: $L(A) = L(A_1) \cap L(A_2)$

$$\begin{aligned}x \in L(A) &\iff \hat{\delta}(q_0,x) \in F \\ &\iff \hat{\delta}((q_0^1,q_0^2),x) \in (F_1 \times F_2) \\ &\iff (\hat{\delta}_1(q_0^1,x), \hat{\delta}_2(q_0^2,x)) \in (F_1 \times F_2) \\ &\iff \hat{\delta}_1(q_0^1,x) \in F_1 \text{ and } \hat{\delta}_2(q_0^2,x) \in F_2 \\ &\iff x \in L(A_1) \cap L(A_2)\end{aligned}$$

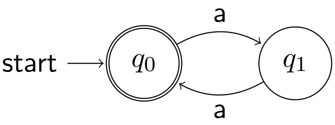
□

Building complicated DFAs from simple ones

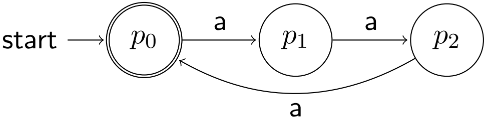
Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$



Let $L_2 = \{w \mid |w| \equiv 0 \pmod 3\}$



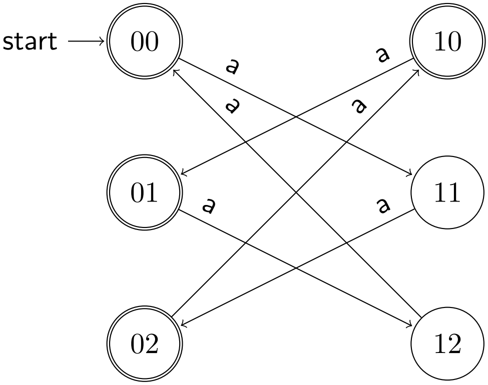
What is $L_1 \cap L_2$?

$L_1 \cup L_2 = \{w \mid |w| \equiv 0 \pmod 2 \text{ or } |w| \equiv 0 \pmod 3\}$

Building complicated DFAs from simple ones

Example

$$L_1 \cup L_2 = \{w \mid |w| \equiv 0 \pmod 2 \text{ or } |w| \equiv 0 \pmod 3\}$$



Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$Q = \{(q, q') \mid q \in Q_1, q' \in Q_2\}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$
\begin{array}{lcl}
Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\
q_0 & = & (q_0^1, q_0^2)
\end{array}
$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$
\begin{array}{lcl}
Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\
q_0 & = & (q_0^1, q_0^2) \\
F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\}
\end{array}
$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{rcl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \\ F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\} \\ \delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a)) \end{array}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$
\begin{array}{rcl}
Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\
q_0 & = & (q_0^1, q_0^2) \\
F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\} \\
\delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a))
\end{array}
$$

Correctness

$\forall w \in \Sigma^*$, w is accepted by A iff w is accepted by either A_1 or A_2.

Complementation

Let $L = \{w \mid |w| \equiv 0 \pmod 3\}$

Complementation

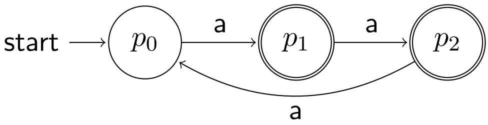
Let $L = \{w \mid |w| \equiv 0 \pmod 3\}$

$\overline{L} = \{w \mid |w| \not\equiv 0 \pmod 3\}$

Complementation

Let $L = \{w \mid |w| \equiv 0 \pmod 3\}$

$\overline{L} = \{w \mid |w| \not\equiv 0 \pmod 3\}$



Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q_0', F', \delta')$ s.t.

$$\begin{array}{ccc} Q' & = & Q \\ q_0' & = & q_0 \\ F' & = & \{q \in Q \mid q \notin F\} \\ \delta' & = & \delta \end{array}$$

Correctness

$\forall w \in \Sigma^*$, w is accepted by A' iff w is not accepted by A.

$\square$

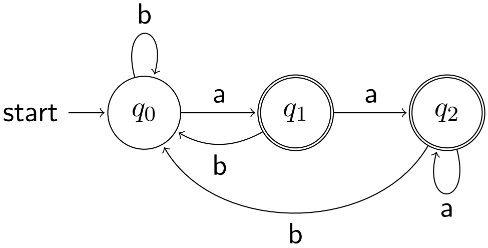
Concatenation and Kleene star

Let $L_1, L_2, L \subseteq \Sigma^*$

$$L_1 \circ L_2 := \{xy \mid x \in L_1, y \in L_2\}$$

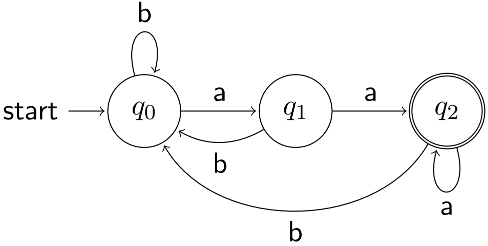
$$\begin{aligned} L^k &:= \{x_1 x_2 \dots x_k \mid x_i \in L\} \\ L^* &:= \bigcup_{k \geq 0} L^k \end{aligned}$$

Example



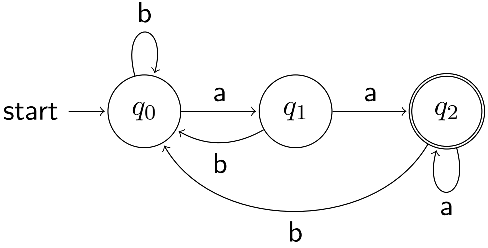
Non-deterministic finite state automata

Example



Non-deterministic finite state automata

Example

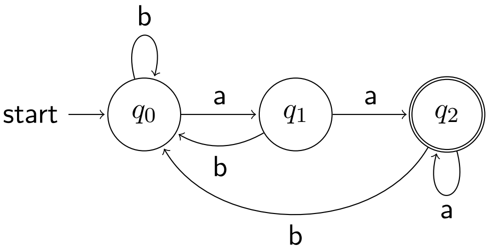


Non-deterministic finite state automata

Example

Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string 'aa'

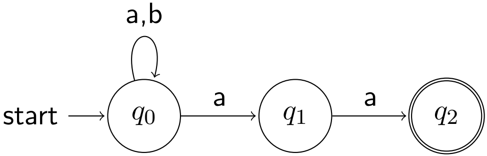


Non-deterministic finite state automata

Example

Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string ‘aa’



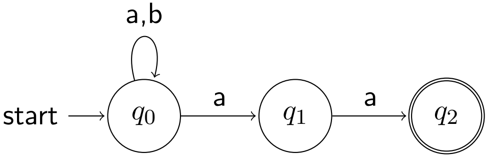
Note that: $\delta(q_0, a) = \{q_0, q_1\}$

Non-deterministic finite state automata

Example

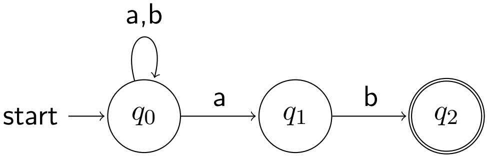
Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string 'aa'

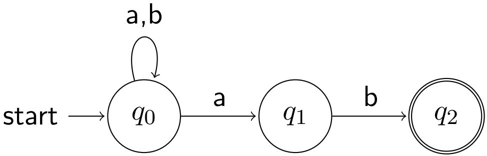


Note that: $\delta(q_0, a) = \{q_0, q_1\}$

Runs of a NFA

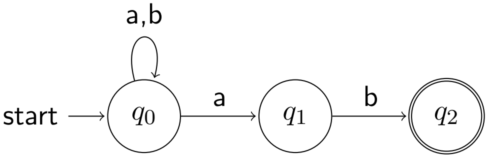


Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

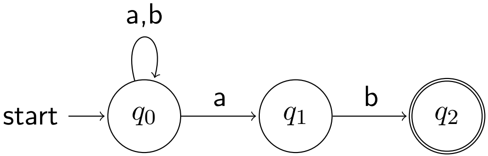
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b}$

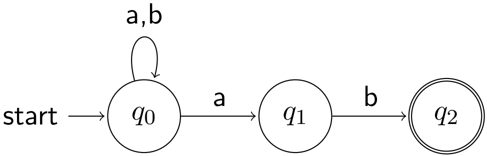
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a}$

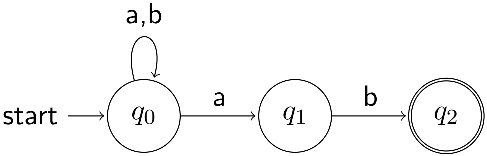
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b}$

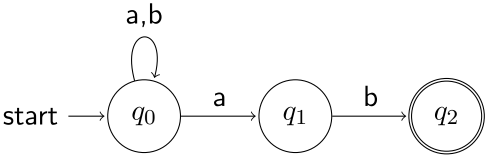
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$

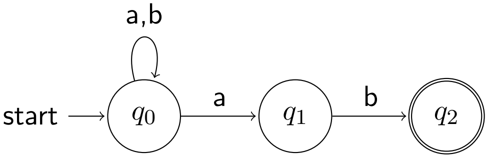
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.

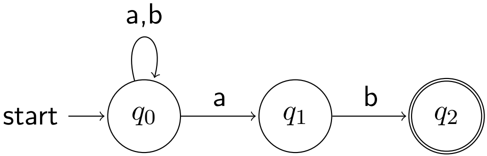
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b}$

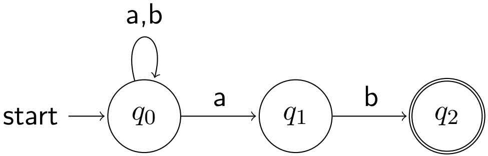
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a}$

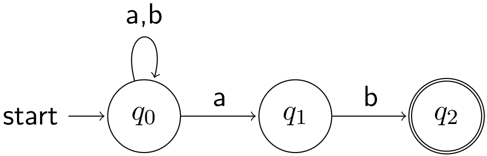
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad$ $\ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b}$

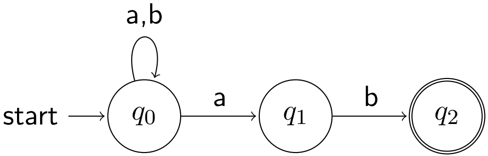
Runs of a NFA



Runs on $b \;\; a \;\; b \;\; a \;\; b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 : \xrightarrow{a}$

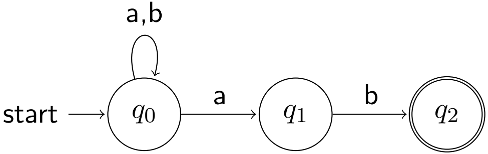
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad$ $\ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_0 \xrightarrow{b}$

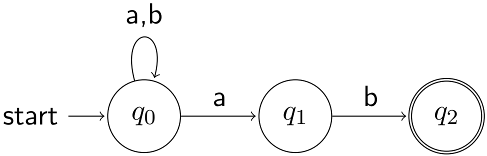
Runs of a NFA



Runs on $b \;\; a \;\; b \;\; a \;\; b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad$ $*$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 : \xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad$ $*$ run not accepted.

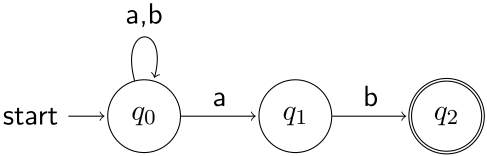
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_0 \xrightarrow{b} q_0$. $\ast$ run not accepted.
- $q_0 \xrightarrow{b}$

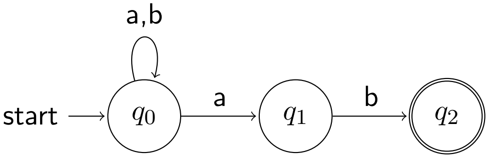
Runs of a NFA



Runs on $b \;\; a \;\; b \;\; a \;\; b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad * $ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad *$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a}$

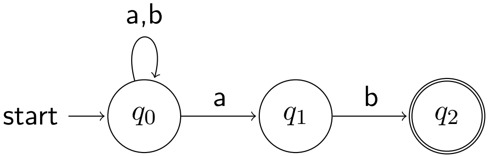
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad \ast$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b}$

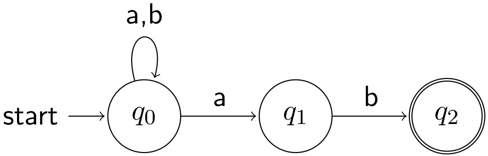
Runs of a NFA



Runs on $b \;\; a \;\; b \;\; a \;\; b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad \ast$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a}$

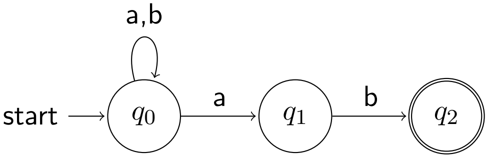
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \,:\!\xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad \ast$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \,:\!\xrightarrow{a} q_1 \xrightarrow{b}$

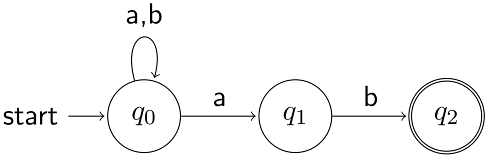
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad \ast$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad \ast$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_1 \xrightarrow{b} q_2.$ $\quad \ast$ accepted!
- Much more concise than a DFA.

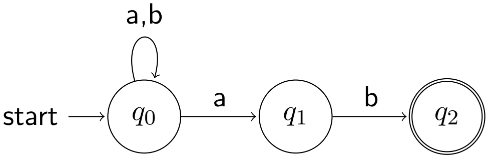
Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad$ $*$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad$ $*$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 :\xrightarrow{a} q_1 \xrightarrow{b} q_2.$ $\quad$ $*$ accepted!
- Much more concise than a DFA.
- Exercise: List all runs of $bbaba$.

Runs of a NFA



Runs on $b \quad a \quad b \quad a \quad b$

- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a}$: stuck! $\quad *$ unfinished runs are not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 : \xrightarrow{a} q_0 \xrightarrow{b} q_0.$ $\quad *$ run not accepted.
- $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 : \xrightarrow{a} q_1 \xrightarrow{b} q_2.$ $\quad *$ accepted!
- Much more concise than a DFA.
- Exercise: List all runs of $bbaba$.
- Nondeterminism can be thought of as a guess!

Non-deterministic finite state automata

Example

Input: $w \in \{a, b\}^*$

Non-deterministic finite state automata

Example

Input: $w \in \{a, b\}^*$

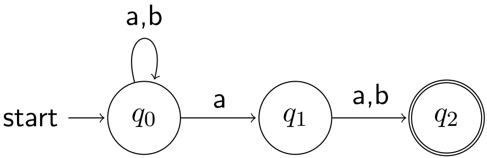
Check: Is a the second-last letter of w?

Non-deterministic finite state automata

Example

Input: $w \in \{a, b\}^*$

Check: Is a the second-last letter of w?

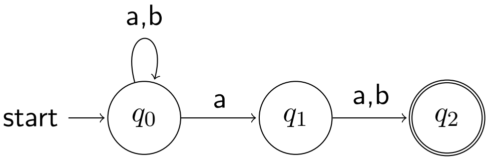


Non-deterministic finite state automata

Example

Input: $w \in \{a, b\}^*$

Check: Is a the second-last letter of w?



Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

- Q is a set of states

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

- Q is a set of states,
- Σ is the input alphabet

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

- Q is a set of states,
- Σ is the input alphabet
- q_0 is the initial state

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

Q is a set of states,

Σ is the input alphabet

q_0 is the initial state,

$F \subseteq Q$ is the set of final states

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

- Q is a set of states,
- Σ is the input alphabet
- q_0 is the initial state,
- $F \subseteq Q$ is the set of final states,
- δ is a set of transitions, i.e. $\delta \subseteq Q \times \Sigma \times Q$

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

Q is a set of states,

Σ is the input alphabet

q_0 is the initial state,

$F \subseteq Q$ is the set of final states,

δ is a set of transitions, i.e. $\delta \subseteq Q \times \Sigma \times Q$

$\cancel{\forall q \in Q, \forall a \in \Sigma, |\delta(q, a)| \leq 1}$.

Non-deterministic finite state automata

Informal description: A finite state automaton which can branch out to different states on the same letter.

NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

Q is a set of states,

Σ is the input alphabet

q_0 is the initial state,

$F \subseteq Q$ is the set of final states,

δ is a set of transitions, i.e. $\delta \subseteq Q \times \Sigma \times Q$

$\cancel{\forall q \in Q, \forall a \in \Sigma, |\delta(q, a)| \leq 1}$.

$\forall q \in Q, \forall a \in \Sigma, \delta(q, a) \subseteq Q$.

Acceptance by NFA

Acceptance by NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, is said to accept a word $w \in \Sigma^*$, where $w = w_1 w_2 \dots w_n$ if

> there exists a sequence of states $p_0, p_1, \dots p_n$ s.t.

Acceptance by NFA

Acceptance by NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, is said to accept a word $w \in \Sigma^*$, where $w = w_1 w_2 \ldots w_n$ if

there exists a sequence of states $p_0, p_1, \ldots p_n$ s.t.

$p_0 = q_0$

Acceptance by NFA

Acceptance by NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, is said to accept a word $w \in \Sigma^*$, where $w = w_1 w_2 \ldots w_n$ if

there exists a sequence of states $p_0, p_1, \ldots p_n$ s.t.

$p_0 = q_0$,

$p_n \in F$

Acceptance by NFA

Acceptance by NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, is said to accept a word $w \in \Sigma^*$, where $w = w_1 w_2 \dots w_n$ if

there exists a sequence of states $p_0, p_1, \dots p_n$ s.t.

$p_0 = q_0$,

$p_n \in F$,

$p_{i+1} \in \delta(p_i, x_{i+1})$ for all $0 \leq i \leq n-1$.

Acceptance by NFA

Acceptance by NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, is said to accept a word $w \in \Sigma^*$, where $w = w_1 w_2 \ldots w_n$ if

there exists a sequence of states $p_0, p_1, \ldots p_n$ s.t.

$p_0 = q_0$,

$p_n \in F$,

$p_{i+1} \in \delta(p_i, x_{i+1})$ for all $0 \leq i \leq n-1$.

Extended Transition Function Let $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ be defined as:

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q \\ \hat{\delta}(q, wa) &= \delta(\hat{\delta}(q, w), a) \end{aligned}$$

Acceptance by NFA

Acceptance by NFA

A non-deterministic finite state automaton (NFA) $A = (Q, \Sigma, q_0, F, \delta)$, is said to accept a word $w \in \Sigma^*$, where $w = w_1 w_2 \dots w_n$ if

there exists a sequence of states $p_0, p_1, \dots p_n$ s.t.

$p_0 = q_0$,

$p_n \in F$,

$p_{i+1} \in \delta(p_i, x_{i+1})$ for all $0 \le i \le n-1$.

Extended Transition Function Let $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ be defined as:

$$\begin{array}{rcl} \hat{\delta}(q, \varepsilon) & = & q \\ \hat{\delta}(q, wa) & = & \delta(\hat{\delta}(q, w), a) \end{array}$$

$$L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in 2^F\}$$

An NFA A is said to accept a language L if $L = \{w \mid A \text{ accepts } w\}$.

Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language.

Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

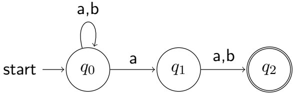
We will work it out for an example.

Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

We will work it out for an example.

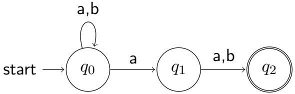


Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

We will work it out for an example.

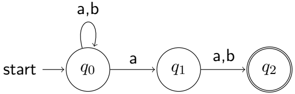


Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

We will work it out for an example.



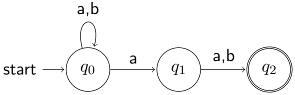
$\varnothing \quad \{0\} \quad \{1\} \quad \{2\} \quad \{0,1\} \quad \{0,2\} \quad \{1,2\} \quad \{0,1,2\}$

Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

We will work it out for an example.



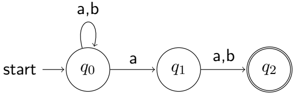
	$\varnothing$	$\{0\}$	$\{1\}$	$\{2\}$	$\{0,1\}$	$\{0,2\}$	$\{1,2\}$	$\{0,1,2\}$
a	$\varnothing$	$\{0,1\}$	$\{2\}$	$\varnothing$	$\{0,1,2\}$	$\{0,1\}$	$\{2\}$	$\{0,1,2\}$

Power of NFAs

Lemma

Let A be an NFA. Then $L(A)$ is a regular language. That is, NFA and DFA accept the same set of languages.

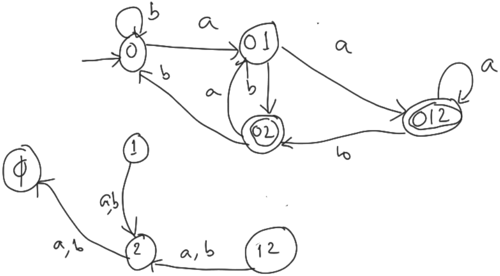
We will work it out for an example.



	$\varnothing$	$\{0\}$	$\{1\}$	$\{2\}$	$\{0,1\}$	$\{0,2\}$	$\{1,2\}$	$\{0,1,2\}$
a	$\varnothing$	$\{0,1\}$	$\{2\}$	$\varnothing$	$\{0,1,2\}$	$\{0,1\}$	$\{2\}$	$\{0,1,2\}$
b	$\varnothing$	$\{0\}$	$\{2\}$	$\varnothing$	$\{0,2\}$	$\{0\}$	$\{2\}$	$\{0,2\}$

Equivalent DFA

Equivalent DFA



Example

Example

$$L = \{x \in \{a\}^* \mid |x| \text{ is divisible by } 3 \text{ or } 5\}$$

Example

$$L = \{x \in \{a\}^* \mid |x| \text{ is divisible by } 3 \text{ or } 5\}$$