CS310 Automata Theory – 2016-2017

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Lecture 2: Finite state automata
January 03, 2017

Last class

Course outline

- Regular languages, DFA/NFA, related topics.
- Pushdown automata, context-free languages, other models of computation.
- Turing machines and computability.
- Effective computation, NP vs. P, one-way functions.

Finite state automata

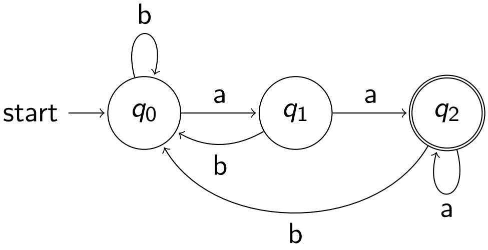
- Examples of finite state automata
- Definition of DFA
- Definition of acceptance by a DFA.

Finite state automata

Example

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

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

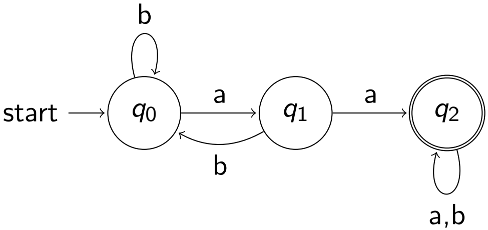


Finite state automata

Example

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

Check: does the file contain the string 'aa'

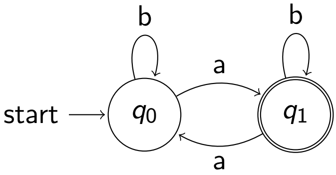


Finite state automata

Example

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

Check: does w have odd number of as? i.e. is $\#_a(w) \equiv 1 \ (mod 2)$?

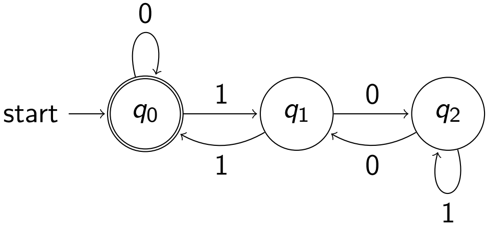


Finite state automata

Example

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?



Definition of finite state automata

Definition (DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, q_0, F, \delta)$, 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. $\delta \subseteq Q \times \Sigma \times Q$ such that $\forall q \in Q, \forall a \in \Sigma,\ |\delta(q, a)| \leq 1$.

Acceptance by DFA

Definition (Acceptance by DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, \delta, q_0, q_f)$, 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.

$p_0 = q_0$,

$p_n \in F$,

$\delta(p_i, w_{i+1}) = p_{i+1}$ for all $0 \leq i \leq n$.

Regular languages

Definition

A language $L \subseteq \Sigma^*$ is a said to be recognized by a DFA A if
$L = \{w \mid w \text{ is accepted by } A\}$.

Definition (REG)

A language is said to be a regular language if it is recognized by some DFA.

Examples

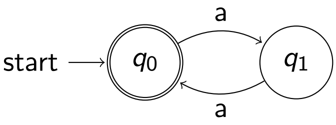
$$
\begin{aligned}
L &= \{w \in \{a,b\}^* \mid w \text{ ends with aa}\} \\
L' &= \{w \in \{a,b\}^* \mid w \text{ contains aa}\} \\
L_{odd} &= \{w \in \{a,b\}^* \mid w \text{ contains odd number of a}\} \\
L_3 &= \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}
\end{aligned}
$$

Closure properties of regular languages

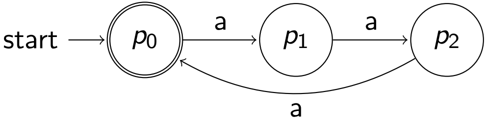
Example

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

Let $L_1 = \{w \mid |w| \equiv 0 \ (mod\ 2)\}$



Let $L_2 = \{w \mid |w| \equiv 0 \ (mod\ 3)\}$



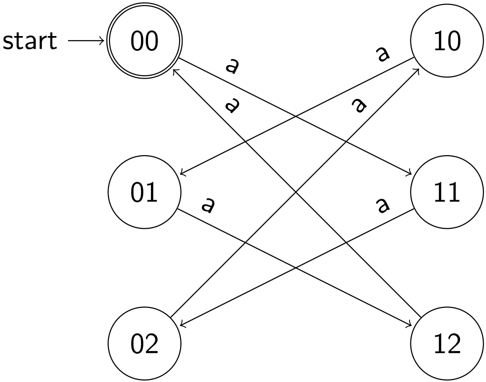
What is $L_1 \cap L_2$?

$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \ (mod\ 6)\}$

Closure properties of regular languages

Example continued

$$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \ (mod\ 6)\}$$



Closure properties of regular languages

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

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

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

Correctness

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

□

Closure properties of regular languages

Lemma

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

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \circ L_2$ is also a regular language, where $L_1 \circ L_2 = \{w \cdot w' \mid w \in L_1, w' \in L_2\}$ and $\cdot$ represents the concatenation operation.

Non-deterministic finite state automata

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

Definition (NFA)

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

Q is a set of states,

$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ 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. $\delta \subseteq Q \times \Sigma_\epsilon \times 2^Q$

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

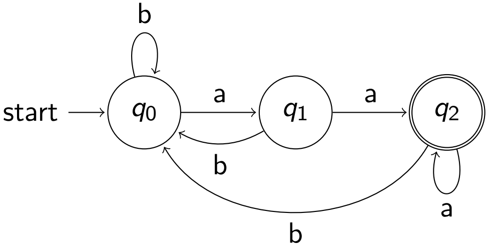
$\forall q \in Q, \forall a \in \Sigma_\epsilon$, $\delta(q, a) \subseteq Q$.

Non-deterministic finite state automata

Example

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

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

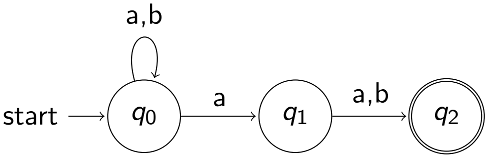


Non-deterministic finite state automata

Example

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

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



Non-deterministic finite state automata

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

Definition (NFA)

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

Q is a set of states,

$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ 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. $\delta \subseteq Q \times \Sigma_\epsilon \times 2^Q$

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

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