Lecture 2

Automata : our 1st computation model

Input to our Automata (Strings) $\leftarrow \Sigma^*$

$\Sigma \leftarrow$ finite . (26 a b c - - -) $\Sigma_k = \{ \omega_1 \ldots \omega_k \mid \omega_i \in \Sigma \}$

$\leftarrow \{0, 1\}$ binary alphabet $\qquad \bigcup_{k=0}^{\infty} \Sigma_k$

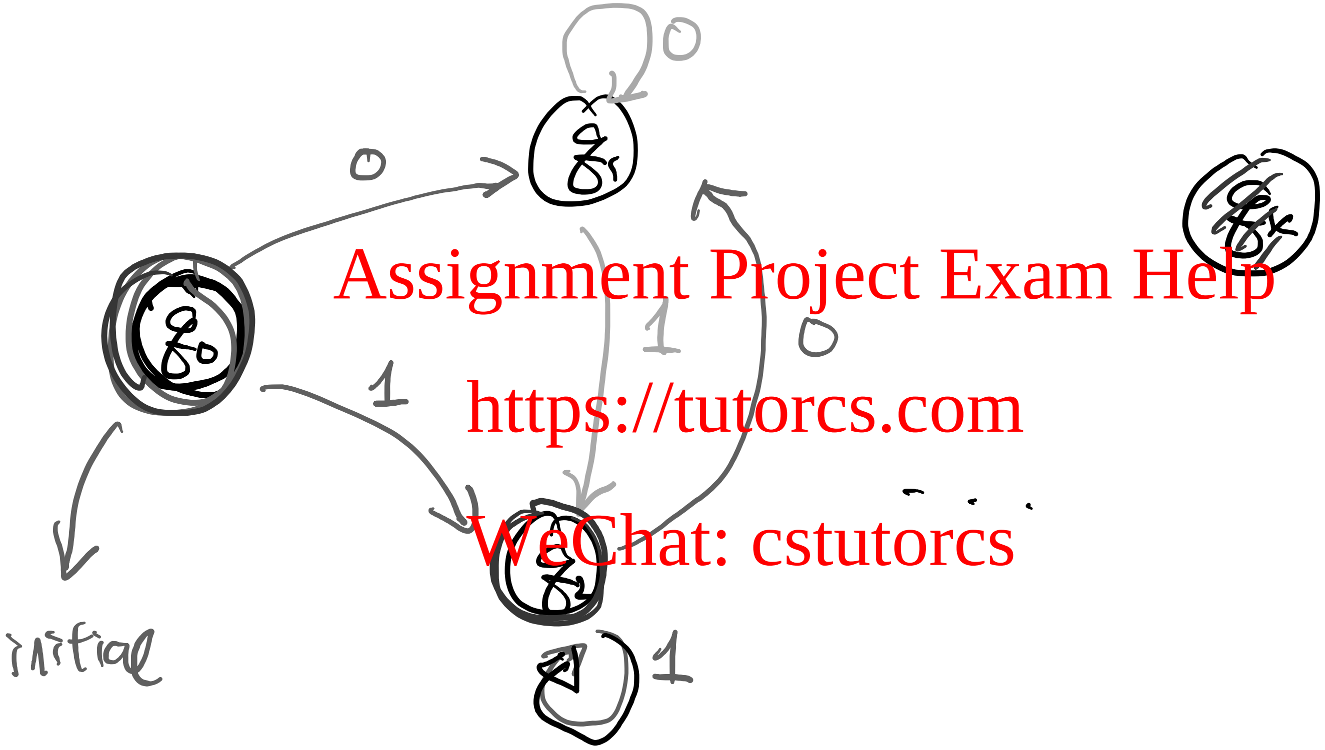
ex 011

011110

111000 . . .

strings formed via the alphabet $\Sigma = \{0, 1\}$.

(Finitely Many) States



Starting State

Transition Function

Accepting State

Formal Definition

FA (DFA)

Deterministic Finite Automata

Q := set of states

$|Q|$ must be finite

$q_0 \in Q$ → starting / initial state.

$S \subseteq Q$ → accepting states.

Σ → alphabet

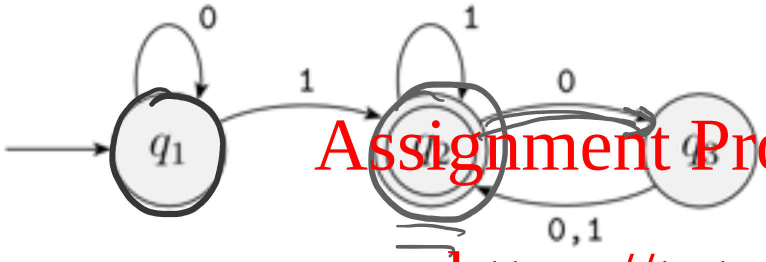
$\delta : \underline{\underline{\Sigma}} \times \underline{\underline{Q}} \longrightarrow Q$

$\delta(\underline{\underline{0}}, \underline{\underline{q_2}}) = q_1$

input, current state

Example Automata (automatic door)

Example from text



$S = \{q_2\}$

FIGURE 1.4
A finite automaton called M_1 that has three states

accepting strings

1
01
001
0001
⋮

$\underbrace{1 \ldots 1}_{1\text{'s}}$
10
110
1101
$\boxed{1100}111$
1
0

(doesnot end with single zero).

Resources used ?

input of size n.

- Time ? Space ?

n

...(?)

Language of Automata

$L(M) :=$ set of strings accepted by machine (FA) M.

$L(M) \subseteq \Sigma^*$ → all possible strings with alphabet Σ

Regular Language

$\forall$

L is regular if $\exists$ Finite Automata M s.t.

$L \subseteq \Sigma^*$ $L = L(M)$

in other words

L is regular iff

$\exists$ Finite Automata M. such that

M accepts w if and only if $w \in L$

M accepts w if $w \in L$

only if

Example of Regular Language (Odd/Even)

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

$0 \to 0$

$11 \to 3$

$1 \to 1$

$10 \to 2$

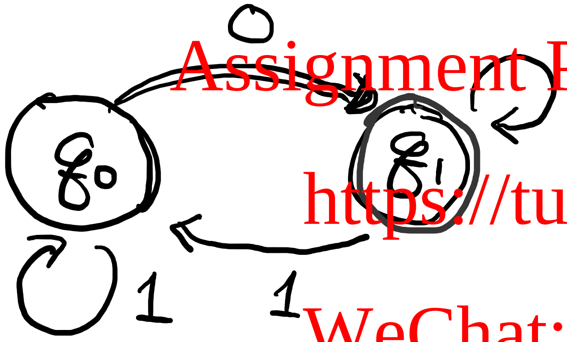
Σ $L_{even} = \{ w \in \{0,1\}^* \mid$ binary representation of w is even $\}$.

- M_{even} s.t. M_{even} accepts if and only if w is from L_{even}.

Example of Regular Language (Odd/Even)

$L_{even} := \{ \omega \mid \text{last digit of } \omega \text{ is } 0 \}.$

M_{even} :



$Q = \{q_0, q_1\}$

$F = \{q_1\}$

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

$q_0 = q_0$

$$\begin{cases} \delta\left(0, \left[\begin{array}{l} q_0 \\ q_1 \end{array}\right.\right) = q_1 \\ \delta(1, *) = q_0 \end{cases}$$

Another Example

via this machine we know that

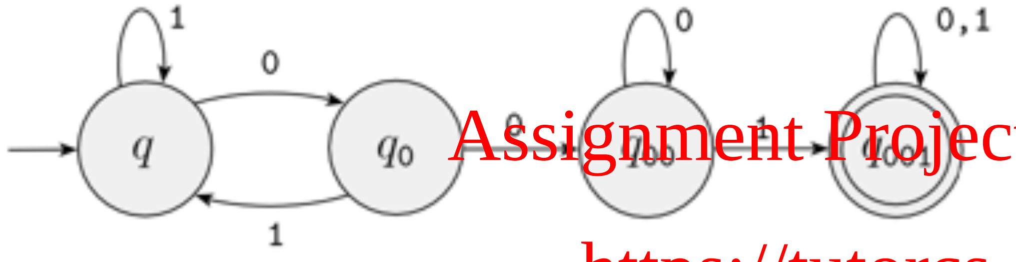


FIGURE 1.22
Accepts strings containing 001

$L_{001} = \{ w \mid w \text{ contain } 001 \}$

is regular.

Recipe for showing that a language is regular

$L \longrightarrow$ design a corresponding machine M.

such that M accepts if and only if

w is from L $(w \in L)$.

ex) L_{even}, L_{001}

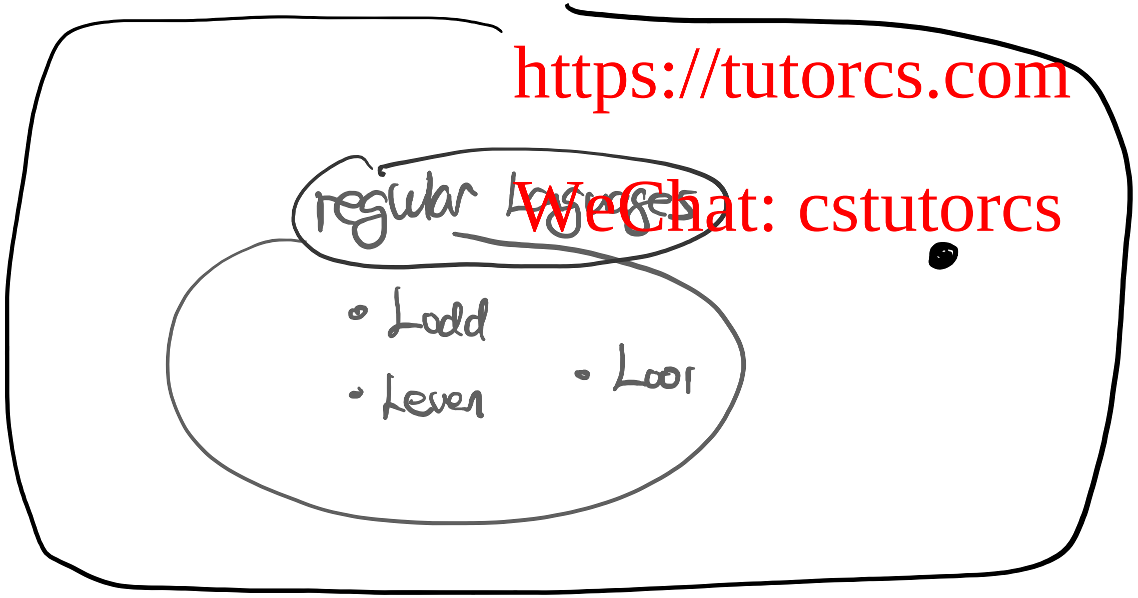
How to show that a language is not regular ?

much harder question

- Do they even exist?

$L \subseteq \{0,1\}^*$

↳ regular if $\exists$ corresponding machine.



Set of all possible languages

Creating more regular languages

- Union, Concatenation, Star , Complement $\rightarrow$ L is regular, L^c is regular

- if L_1 is regular and L_2 is regular than $L_1 \cup L_2$ is also regular.

- L_1 " L_2 "

$L_1 \circ L_2 = \{ w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$. is also regular

- L is regular, L^* is regular $L^k = \underbrace{L \circ \cdots L}_{k \text{ times}}$.

$L^* := \bigcup_{k=0} L^k$

Union (Idea)

Complement :

$M_L = (Q, q_0, S, \Sigma, \delta)$

$Q - S$

$L(M_L') = L^c$

$L_1 \quad L_2$

$\downarrow \quad \downarrow$

$M_1 \quad M_2$

$\downarrow \quad \downarrow$

$Q_1 \quad Q_2$

$Q = Q_1 \times Q_2$

$(\; , \;)$

Proof for Concatenation / Star ?

- What is the difficulty ?