Lecture 3

Deterministic Finite Accepters

COT 4420

Theory of Computation

Section 2.1

Review Question

- What's the number of non-empty languages that contain only strings of a's and b's of length n?
    a. $2^n - 1$
    b. 2^n
    c. 2^{2^n}
    d. $2^{2^n} - 1$

Answer: d. There are 2^n strings of a's and b's of length n. Each of these strings can show up or not show up.

Finite Automaton

- Finite Automaton is a mathematical model that remembers only a finite amount of information.
- States
- States changes in response to inputs
- Rules that tell how the states change are called transitions.

Finite Automaton

- Used in design and verification of communication protocols.

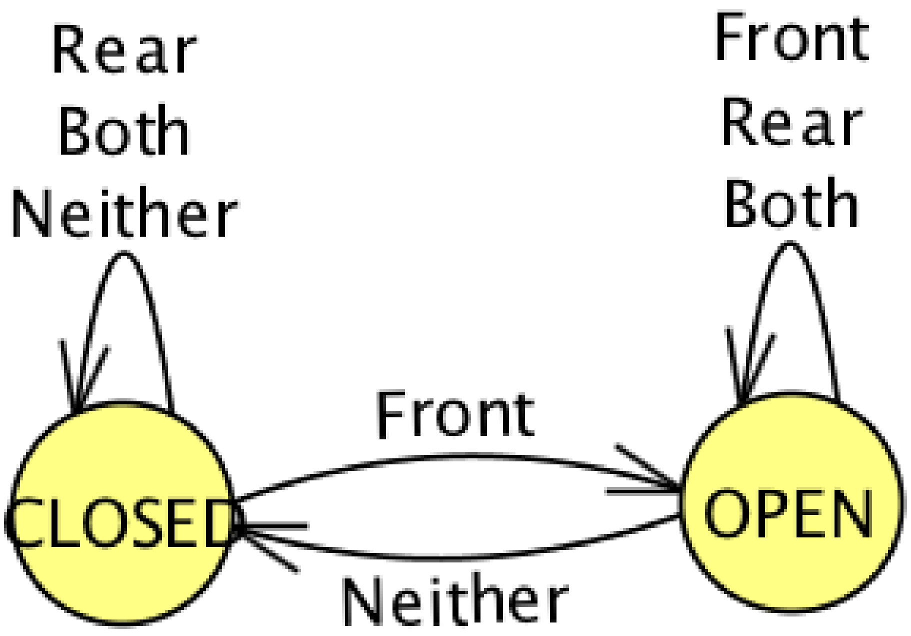
- Used for text processing and in text searching algorithms

- Used in programming languages compilers for lexical analyzing and parsing.

Simple Example
Automatic door

- The controller is in either of two states: OPEN, CLOSED
- There are four input possibilities: Front, Rear, Both, Neither
- The controller moves from state to state depending on the input it receives

Simple Example
Automatic door



Deterministic Finite Accepter (DFA)

DFA is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

- Q: a finite set of states
- Σ: a finite set of symbols called input alphabet
- δ: transition function ($Q \times \Sigma \rightarrow Q$)
- q_0: the start state ($q_0 \in Q$)
- F: a set of final/accepting states ($F \subseteq Q$)

The way it works

- It starts in the start state, and with the leftmost symbol of the input.
- Each move consumes one input symbol, and based on the transition functions moves to a different state.
- When the end of the input string is reached, the string is accepted if the automaton is in one of the final states, otherwise it is rejected.

The transition function

- Takes a state (q) and an input symbol (a) and returns a state (q')

$$\delta(q, a) = q'$$

This means that if the automaton is in state q, and the current input symbol is a, the DFA will go into state q'.

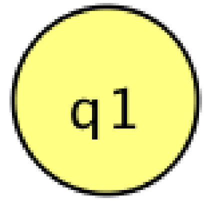
Example

$M = < \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\} >$

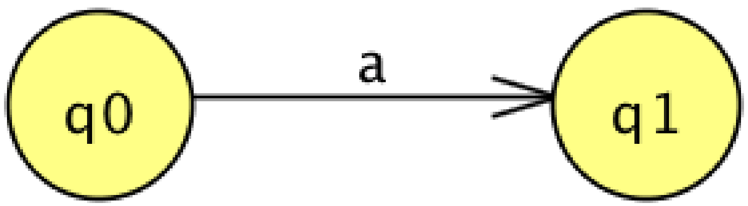
$\delta(q_0, 0) = q_0$	$\delta(q_1, 0) = q_0$	$\delta(q_2, 0) = q_2$
$\delta(q_0, 1) = q_1$	$\delta(q_1, 1) = q_2$	$\delta(q_2, 1) = q_2$

Graph representation

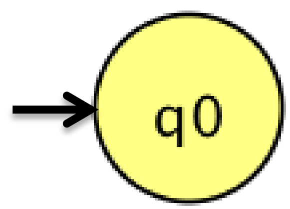
- States = nodes



- Transition function = arc $\delta(q_0, a) = q_1$

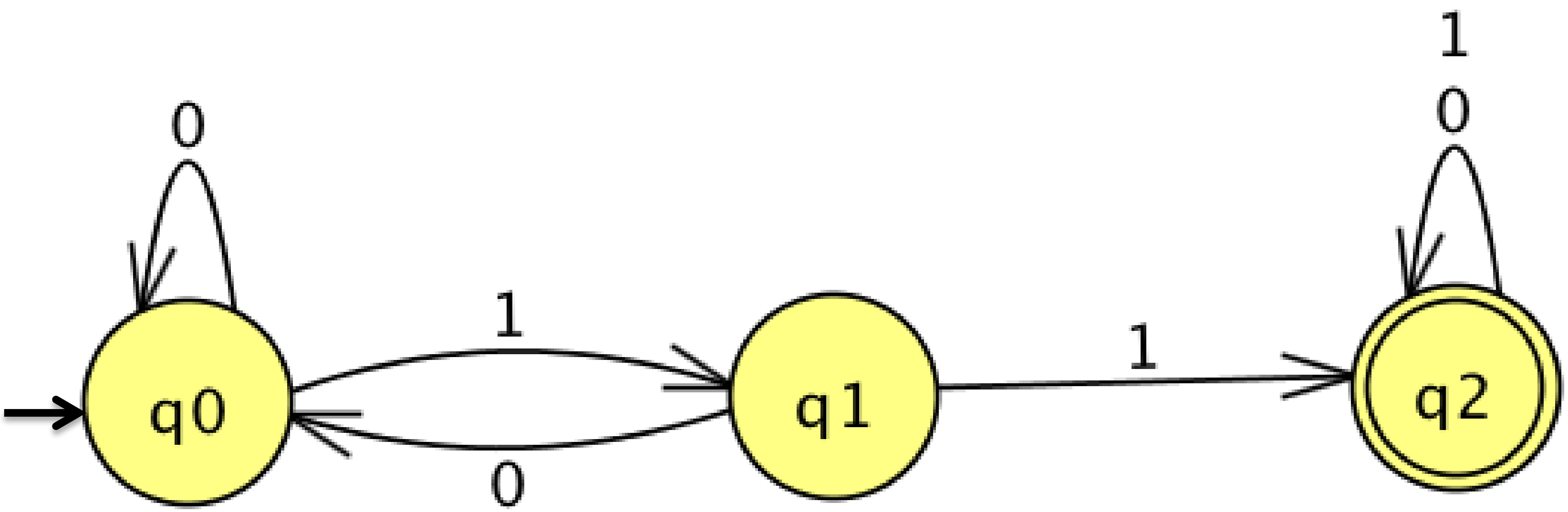


- Start symbol = arrow



- Final state = double circle

Example: String with 11



$M = < \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\} >$

$\delta(q_0, 0) = q_0$ $\delta(q_1, 0) = q_0$ $\delta(q_2, 0) = q_2$

$\delta(q_0, 1) = q_1$ $\delta(q_1, 1) = q_2$ $\delta(q_2, 1) = q_2$

Alternative Representation: Transition Table

state \ input	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_2

- Columns: current input symbol
- Rows: current state
- Entries: next state

Deterministic Finite Accepter (DFA)

- The transition function δ needs to be a total function. It needs to be defined for every input value in Σ.

- At each step, a unique move is defined for every input symbol. So in every state, upon reading the input symbol, the automaton jumps deterministically to another state.

Extended Transition Function

$\delta^*: Q \times \Sigma^* \rightarrow Q$

Example: $w = ab$

$\delta(q_0, a)=q_1 \quad , \quad \delta(q_1, b)=q_2$

$\delta^* (q_0, ab) = q_2$

Formally δ^* is defined recursively by:

$\delta^*(q, \lambda)=q$

$\delta^*(q, wa)= \delta(\delta^*(q, w),a) \qquad w \in \Sigma^* , a \in \Sigma$

Extended Transition Function

	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_2

$$\delta^*(q, \lambda)=q$$

$$\delta^*(q, wa)= \delta(\delta^*(q, w),a)$$

$$w \in \Sigma^* \text{ , } a \in \Sigma$$

$$\delta^*(q_1, 011) = \delta(\delta^*(q_1,01), 1) = \delta(\delta(\delta^*(q_1, 0),1),1)$$

$$=\delta(\delta(\delta(\underbrace{\delta^*(q_1, \lambda)}_{q_1},0),1),1) = \delta(\delta(\underbrace{\delta(q_1,0)}_{q_0},1),1)$$

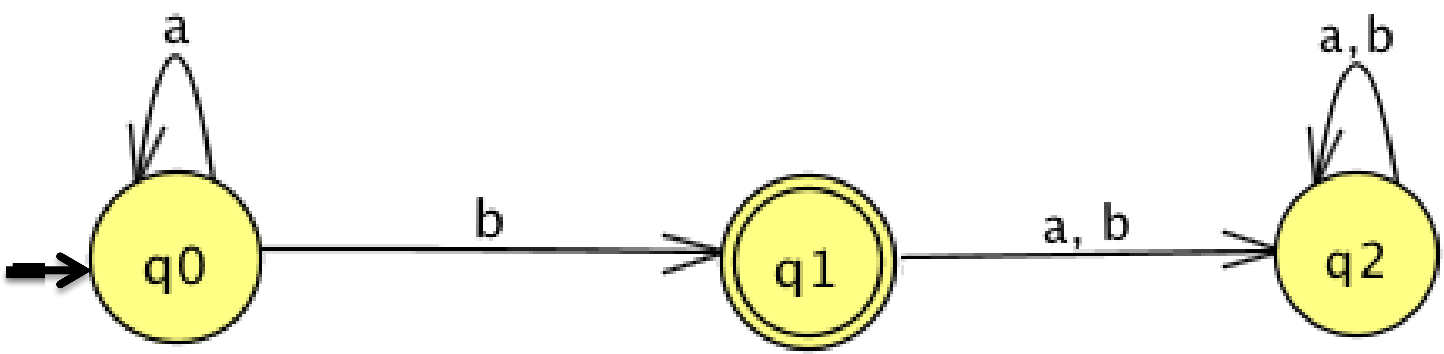
$$\delta(\underbrace{\delta(q_0,1)}_{q_1},1) = \delta(q_1,1) = q_2$$

Language of a DFA

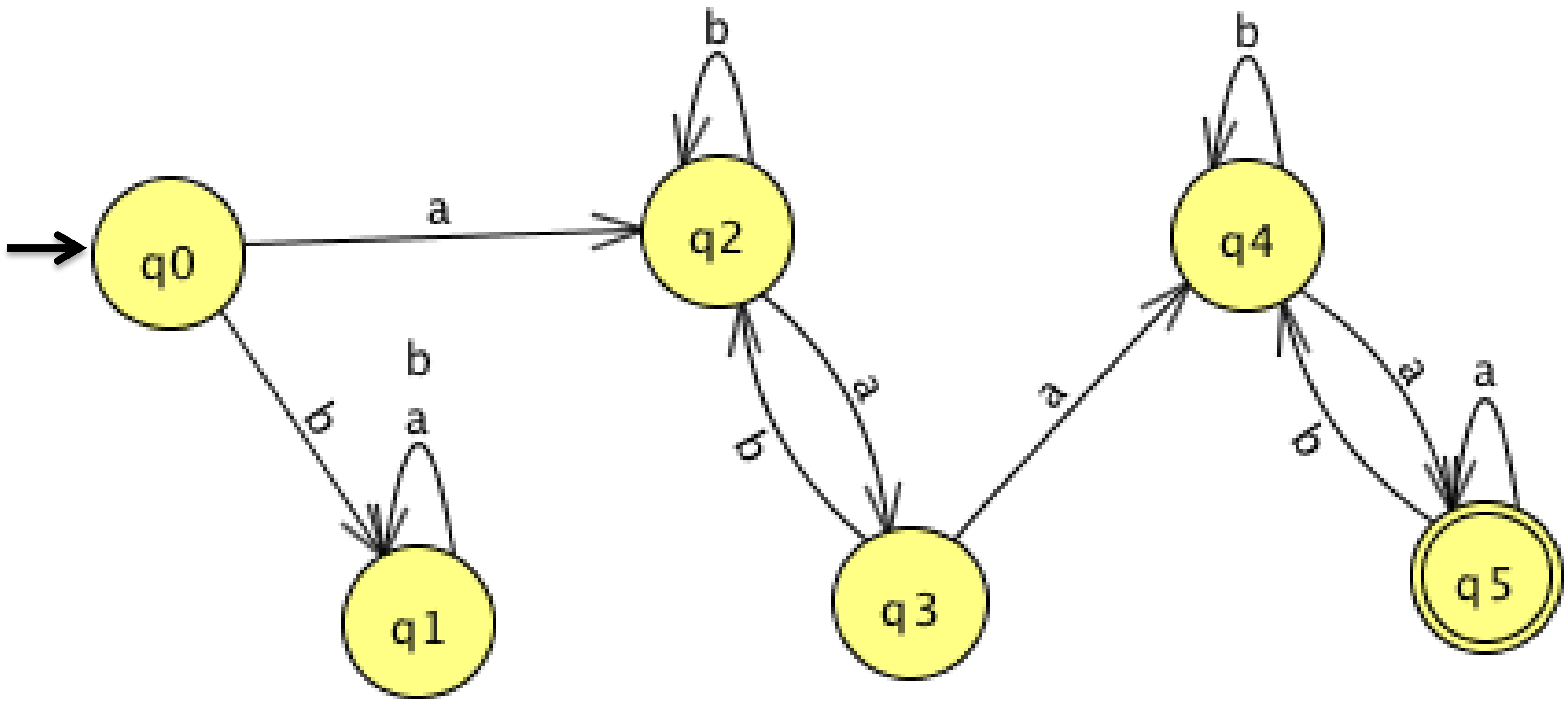
The language recognized by a dfa $M = (Q, \Sigma, \delta, q_0, F)$ is the set of all strings accepted by M.

$$L(M) = \{ w \in \Sigma^* : \; \delta^*(q_0, w) \in F \}$$

Find dfa for $L = \{ a^n b : n \geq 0 \}$



Example



aaababbbaaaa ✓

baabbaaaba ✗

abbabbaab ✗

abbaabba ✓

Theorem

Theorem: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and G_M be its associated transition graph. For every $q_i, q_j \in Q$ and $w \in \Sigma^+$, $\delta^*(q_i,w) = q_j$ iff there is a walk with label w from q_i to q_j in G_M.

Induction on the length w

Base case: $|w| = 1$ $\delta^*(q_i,w) = q_j$ obviously there is an edge (q_i, q_j) with label w in G_M.

Induction: Assume it is true for all strings v with $|v| \leq n$

We want to show it for a *w* with length n+1: $w = va$

Theorem (Cont'd)

Suppose now $\delta^*(q_i, v) = q_k$ since $|v|=n$ there must be a walk in G_M labeled v from q_i to q_k. If $\delta^*(q_i, va) = q_j$ then M must have a transition $\delta(q_k, a) = q_j$ so by construction G_M has an edge (q_k,q_j) with label a.

Regular Languages

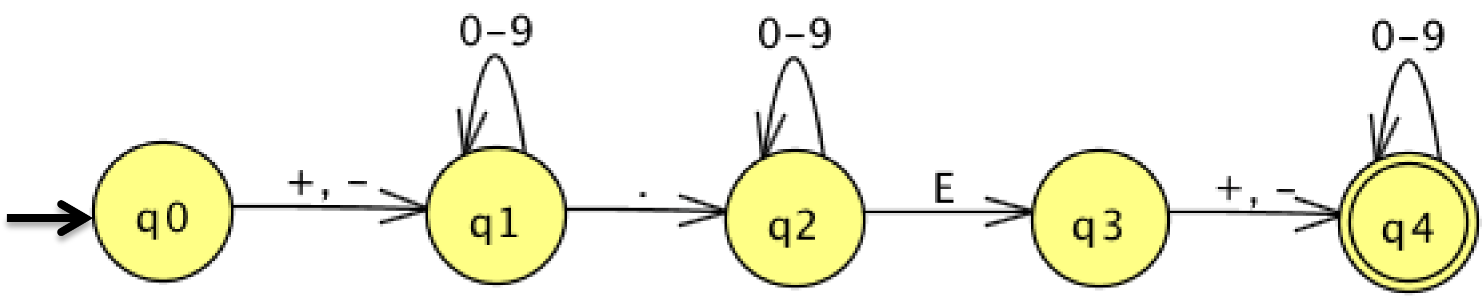
- A language L is called regular if and only if there exists some deterministic finite accepter M such that

$$L = L(M)$$

So in order to show that a language is *regular* we can find a dfa for it. (Note: soon we will see other ways to describe the regular languages such as regular expressions and nondeterministic automata)

Regular Languages

- Regular languages are common and appear in many context.
- Example: the set of strings that represent some floating-point number is a regular language.



Non-regular Languages

- Example: $L = \{ 0^n1^n : n \geq 1 \}$

  L = { 01, 0011, 000111, … }

- Example: L = { w | w in {(,)}* and w is balanced}

  L = { (), (()), ()()(()()), … }