Finite Automata

LECTURE 5

Finite Automaton (FA)

- Informally, a state diagram that comprehensively captures all possible states and transitions that a machine can take while responding to a stream or sequence of input symbols
- Recognizer for "Regular Languages"

- Deterministic Finite Automata (DFA)
  – The machine can exist in only one state at any given time
- Non-deterministic Finite Automata (NFA)
  – The machine can exist in multiple states at the same time

Deterministic Finite Automata

Definition: A deterministic finite automaton (DFA) consists of

1. a finite set of *states* (often denoted Q)
2. a finite set Σ of *symbols* (alphabet)
3. a *transition function* that takes as argument a state and a symbol and returns a state (often denoted δ)
4. a *start state* often denoted q_0
5. a set of *final* or *accepting* states (often denoted F)

We have $q_0 \in Q$ and $F \subseteq Q$

Deterministic Finite Automata - Definition

- A Deterministic Finite Automaton (DFA) consists of:
  - Q ==> a finite set of states
  - Σ ==> a finite set of input symbols (alphabet)
  - q_o ==> a start state
  - F ==> set of accepting states
  - δ ==> a transition function, which is a mapping between $Q \times \Sigma$ ==> Q
- A DFA is defined by the 5-tuple:
  - $\{Q, \Sigma, q_o, F, \delta\}$

What does a DFA do on reading an input string?

- <u>Input:</u> a word w in $\sum^*$
- <u>Question:</u> Is w acceptable by the DFA?
- <u>Steps:</u>
  - Start at the "start state" q_o
  - For every input symbol in the sequence w do
    - Compute the next state from the current state, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed, the current state is one of the accepting states (F) then *accept w;*
  - Otherwise, *reject w*.

Deterministic Finite Automata

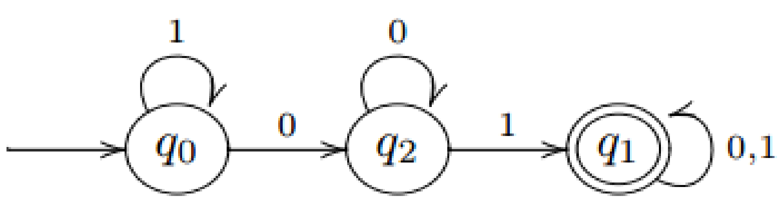
How to present a DFA? With a *transition table*

	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

The $\rightarrow$ indicates the *start* state: here q_0

The $*$ indicates the final state(s) (here only one final state q_1)

This defines the following *transition diagram*

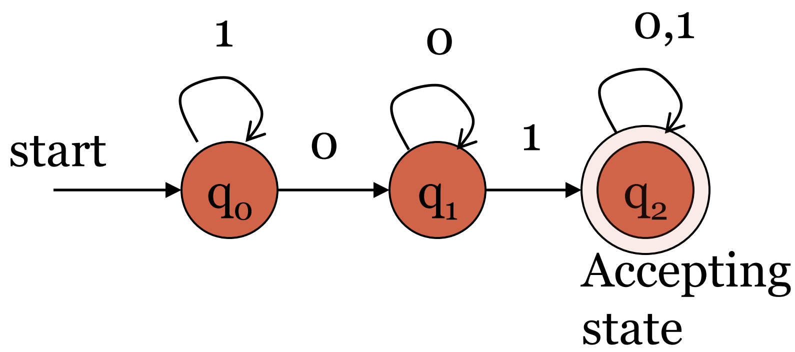


Example #1

- Build a DFA for the following language:
  - L = {w | w is a binary string that contains 01 as a substring}
- Steps for building a DFA to recognize L:
  - ∑ = {0,1}
  - Decide on the states: Q
  - Designate start state and final state(s)
  - δ: Decide on the transitions:
- "Final" states == same as "accepting states"
- Other states == same as "non-accepting states"

Regular expression: (0+1)*01(0+1)*

DFA for strings containing 01



- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

symbols

δ	**0**	**1**
→ **q_0**	q_1	q_0
q_1	q_1	q_2
***q_2**	q_2	q_2

states

δ is a *function* from $Q \times \Sigma$ to Q

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

$\delta(q_0, 1) = q_0$

$\delta(q_0, 0) = q_2$

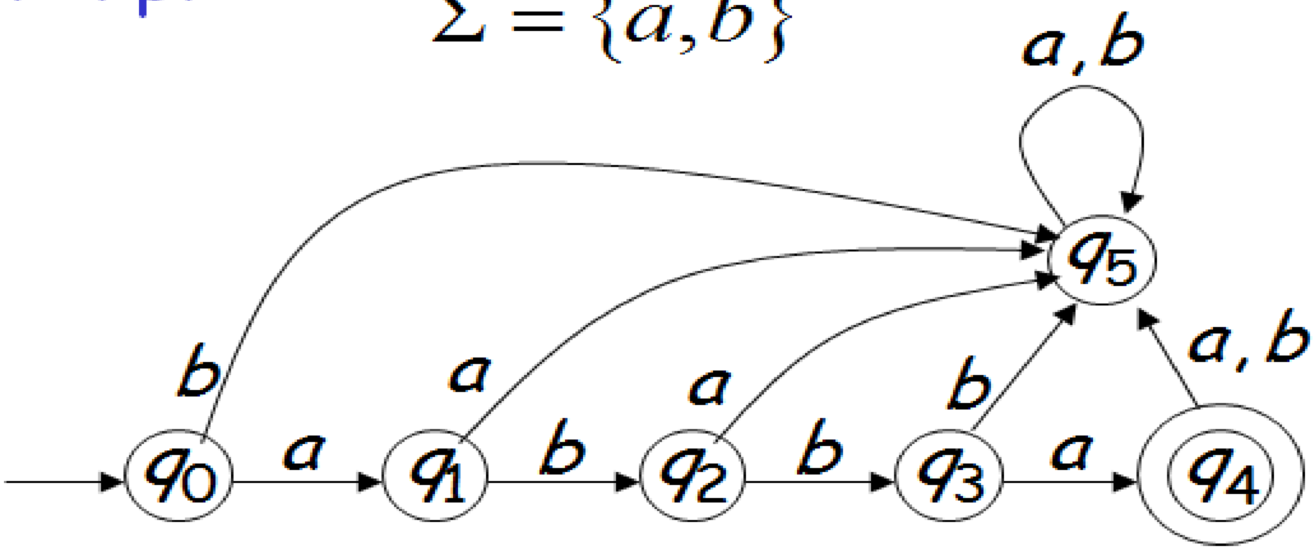
Example 2

Input Alphabet Σ

$\lambda \notin \Sigma$:the input alphabet never contains λ

Example

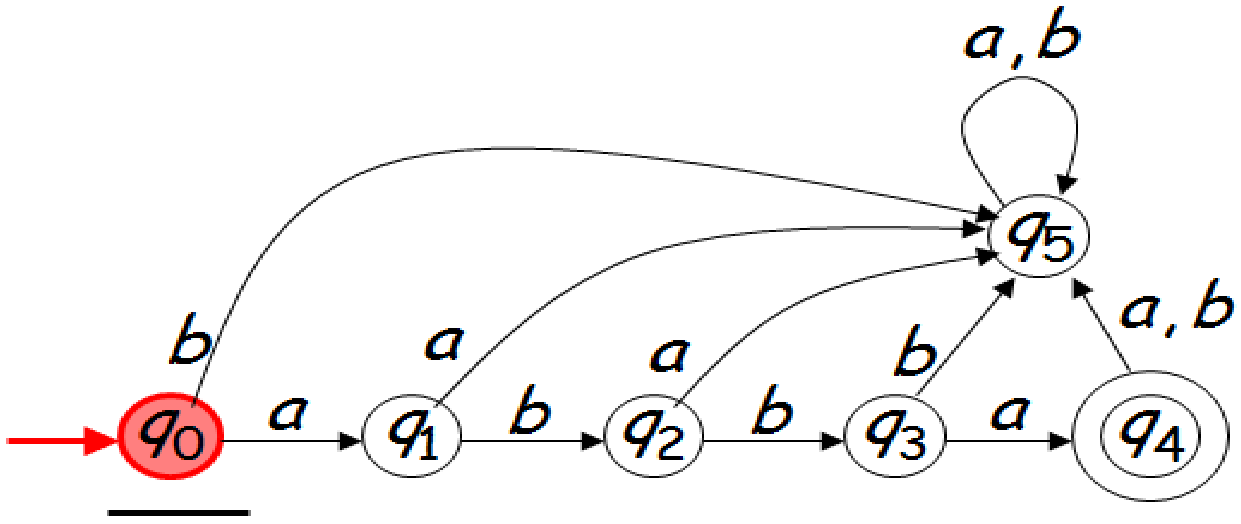
$$\Sigma = \{a, b\}$$



Example 2

Initial State q_0

Example

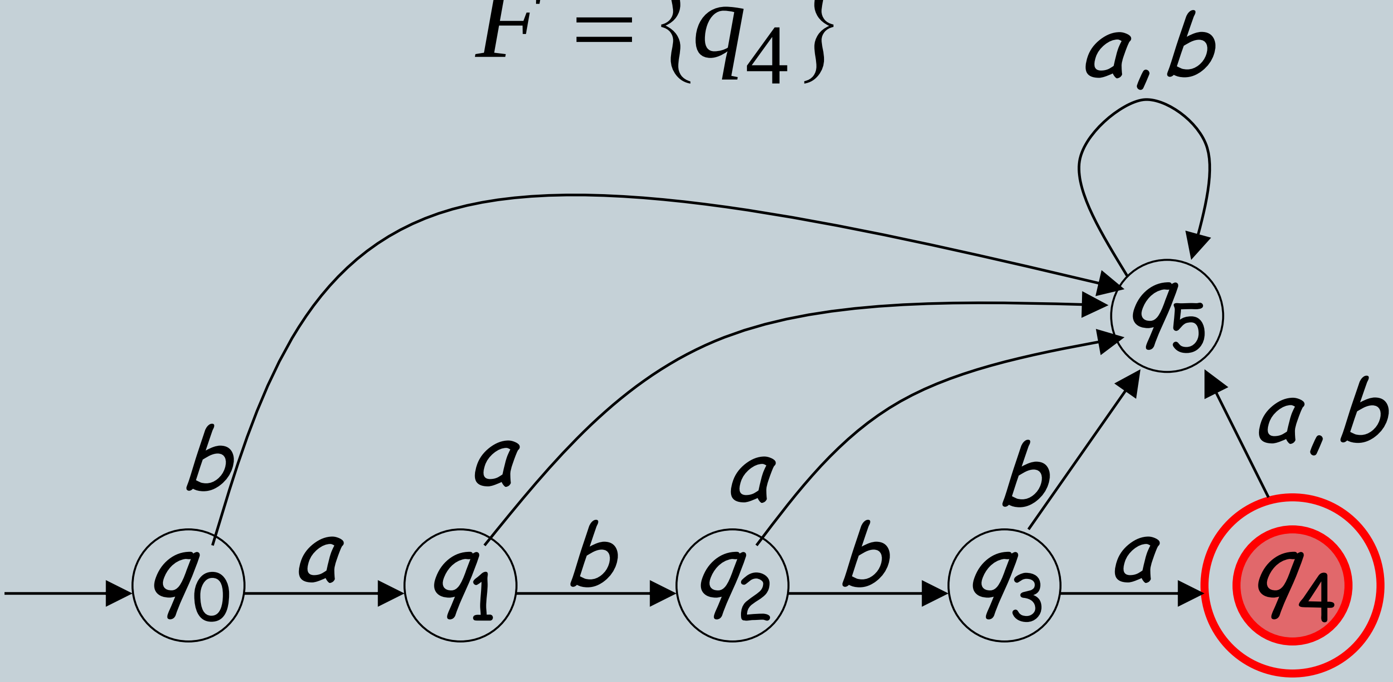


Set of Accepting States

$F \subseteq Q$

- Example

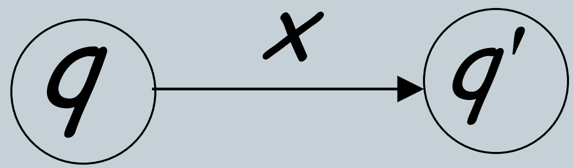
$$F = \{q_4\}$$



Transition Function

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

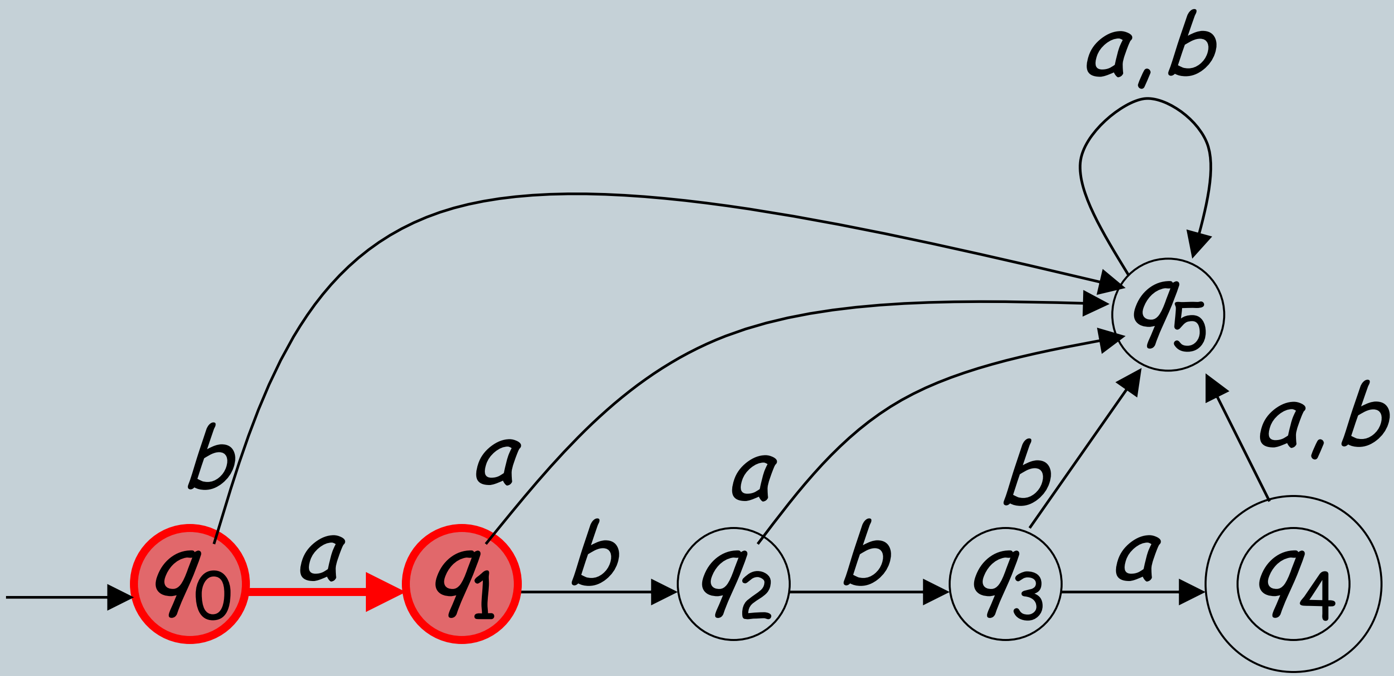
- $$\delta(q, x) = q'$$



Describes the result of a transition from state q with symbol x

Transition function Example : 2

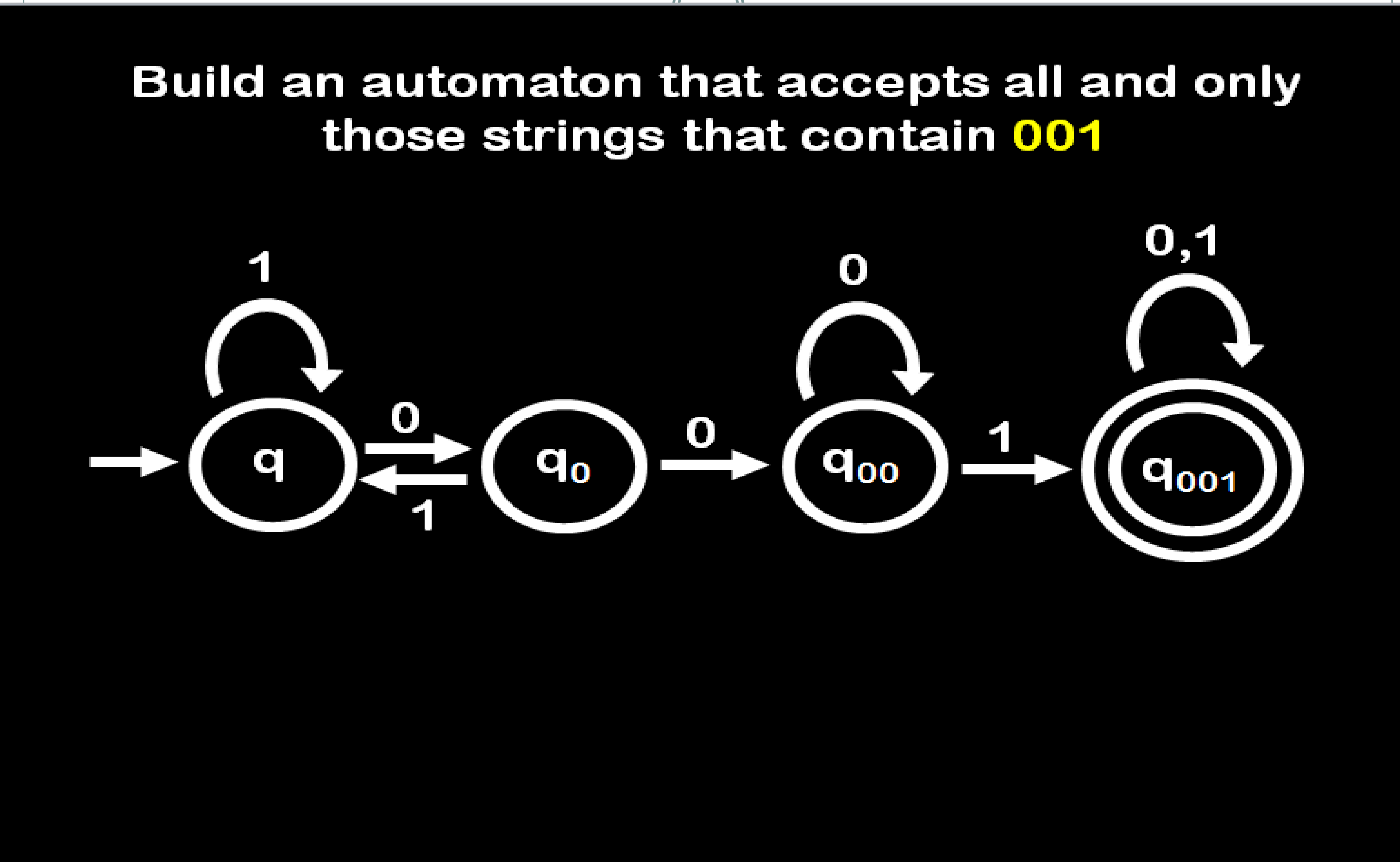
- $$\delta(q_0, a) = q_1$$



Example 2

- Complete All tuples states
- Transition table
- Transition function

Example 3


Build an automaton that accepts all and only those strings that contain 001
1
0
0,1
0
0
1
1
q
q0
q00
q001

Automata

- Deterministic automata – each move is uniquely determined by the current configuration
  - Single path
- Nondeterministic automata – multiple moves possible
  - Can't determine next move accurately
  - May have multiple next moves or paths

Automata

- An automaton whose output response is limited to yes or no is an acceptor
  - Accepts input string and either accepts or rejects it
- Measures of complexity
  - Running time
  - Amount of memory used

Automata

- Finite automaton
  - Uses a limited, constant amount of memory
  - Easy to model
  - Limited application

Finite Automata

Given an automaton $A = (Q,\Sigma,\delta,q_o,F)$, and a string $w \in \Sigma^*$:

- w is **accepted** by A if the configuration (q_o,w) yields the configuration $(F, \in)$, where F is an accepting state
- the **language accepted by** A, written L(A), is defined by:

$$L(A) = \{w \in \Sigma^* \textbf{ : } w \text{ is accepted by } A\}$$