# EECS 2001A : Introduction to the Theory of Computation

**Suprakash Datta**

Course page: `http://www.eecs.yorku.ca/course/2001`
Also on Moodle

# Finite Automata

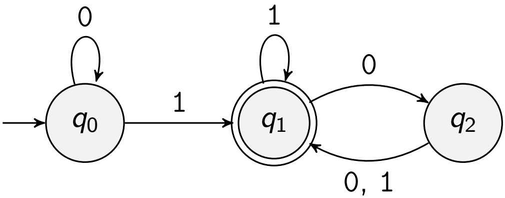
- Simplest machine model
- Design automata for simple problems
- Study languages recognized by finite automata

# Finite Languages

**Recognizing** finite languages

- Just need a lookup table and a search algorithm
- Can be done with very simple hardware (aside: *content addressable memories*)
- Problem – cannot express infinite sets, e.g. odd integers

# Finite Automata: Example




**Components**:

$q_0$: start state,

$q_1$: accept state,

transition rules

# Finite Automata: Determinism vs Non-determinism

- Deterministic: the normal, realizable models
- Non-deterministic: equipped with an unrealizable power, good for studying powers of machine models.
- More on non-determinism later
- Deterministic Finite Automata (DFA) vs Nondeterministic Finite Automata (NFA)

# Finite Automata: Details

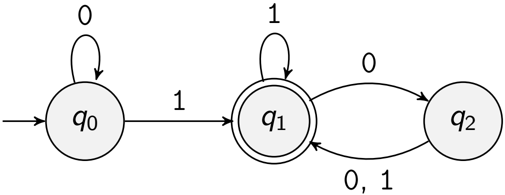
The simplest machine that can recognize an infinite language

- "Read once", "no write" procedure
- Starts at state $q_0$. At each step, consumes the next character of input, moves to a new state as dictated by $\delta$
- At the end it is either in an accept state (the input string is accepted) or is not in an accept state (the input is rejected)
- If a FA accepts all words in a language and rejects every other word, we say that the FA recognizes the language

# Finite Automata: Details

- Useful for describing algorithms also. Used a lot in network protocol description
- Can be implemented very easily in hardware
- Will show: DFA's can accept finite languages as well

# DFA: Tracing inputs




- $\epsilon$: State transitions: $q_0$, Reject
- 0110: State transitions: $q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_2$, Reject
- 011: State transitions: $q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_1$, Accept
- 101: State transitions: $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_1$, Accept
- Argue that 010100100100100 is accepted

# Finite Automata: Examples of Languages

Note: $\Sigma = \{0, 1\}$ in each case

- $L = \{w | w \in \Sigma^*\}$
- $L = \{w | w \in \Sigma^*, w \text{ has no zeroes}\}$
- $L = \{w | w \in \Sigma^*, w \text{ ends with } 1\}$
- $L = \{w | w \in \Sigma^*, w \text{ contains substring } 01\}$
- $L = \{w | w \in \Sigma^*, |w| \text{ is divisible by } 3\}$
- $L = \{w | w \in \Sigma^*, |w| \text{ is odd or w ends with } 1\}$
- $L = \{w | w \in \Sigma^*, |w| \text{ is divisible by } 10^6\}$

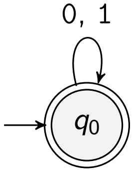
How do we show these?

# DFA Design Example 0

Design DFA for language:

$$L = \{w | w \in \{0, 1\}^*\}$$

One state is enough!




Exercise: Modify the FA above to accept the set of all non-zero length binary strings

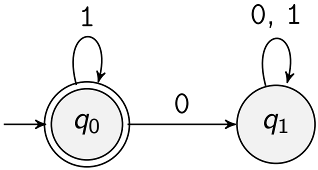
# DFA Design Example 1

Design DFA for language:

$$L = \{w | w \in \{0,1\}^*, w \text{ has no zeroes}\}$$

Two states to remember:

- no symbol so far was a 0 (state $q_0$)
- some symbol was a 0 (state $q_1$)



# DFA Drawing Conventions

- All transitions **must** be present and labelled
- So from each state there should be a transition for each character in the alphabet
- If two transitions vary only in the input (i.e., begin and end in the same nodes) they are drawn as one arrow with multiple labels separated by commas
- If some states or transitions are missing the DFA is **incomplete** and thus undefined

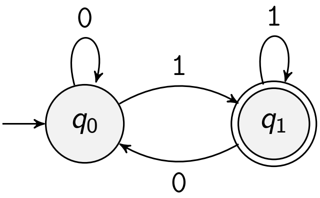
# DFA Design Example 2

Design DFA for language:

$$L = \{w | w \in \{0,1\}^*, w \text{ ends with } 1\}$$

Two states to remember:

- last symbol was not a 1 (state $q_0$)
- last symbol was a 1 (state $q_1$)




Q: What if we made $q_1$ the start state?

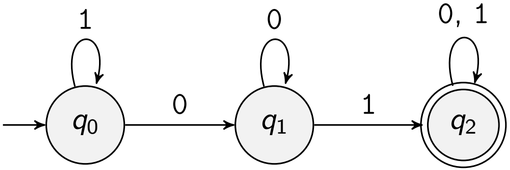
# DFA Design Example 3

Design DFA for language:

$$L = \{w \in 0, 1 * | w \text{ contains substring } 01\}$$

Three states to remember:

- Have seen the substring 01
- Not seen substring 01 and last symbol was 0
- Not seen substring 01 and last symbol was not 0




Q: General principles?

# DFA: Exercises

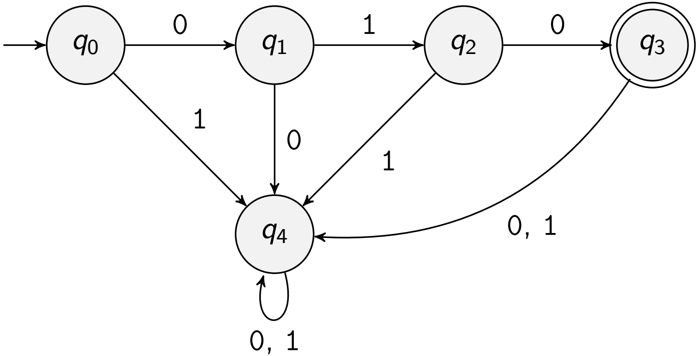
Assuming $\Sigma = \{0, 1\}$ in each case, design DFA's that recognize the following languages

- All words ending with 01
- All words with an odd number of 1's
- All words of length 3 modulo 5
- All words containing both 10 and 01 as subwords

# Recognizing Finite Languages: an Example

Design DFA for language:

$$L = \{010\}$$



# DFA: formal definition

A deterministic finite automaton (DFA) $M$ is defined by a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

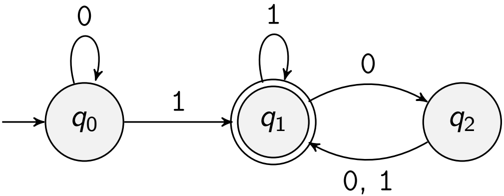
- $Q$: finite set of states
- $\Sigma$: finite alphabet
- $\delta$: transition function $\delta : Q \times \Sigma \to Q$
- $q_0 \in Q$: start state
- $F \subseteq Q$: set of accepting states (could be empty or $Q$)

# Example

$M = (Q, \Sigma, \delta, q_0, F)$

- states $Q = \{q_0, q_1, q_2\}$
- alphabet $\Sigma = \{0, 1\}$
- start state $q_0$
- accept states $F = \{q_1\}$
- transition function $\delta$ :

| | 0 | 1 |
|---|---|---|
| $q_0$ | $q_0$ | $q_1$ |
| $q_1$ | $q_2$ | $q_1$ |
| $q_2$ | $q_1$ | $q_1$ |



# DFA: Recognizing Languages

Recall: a problem can be expressed as a language. Formally:

- A finite automaton $M = (Q, \Sigma, \delta, q, F)$ **accepts** a string/word $w = w_1 \ldots w_n$ if and only if there is a sequence $r_0 \ldots r_n$ of states in Q such that:
  - $r_0 = q_0$
  - $\delta(r_i, w_{i+1}) = r_{i+1}$ for all $i = 0, 1, \ldots, n$–1
  - $r_n \in F$
- Given a language, the DFA **recognizes** it, i.e., it accepts every string in the language, and rejects every string not in the language
- Very commonly forgotten fact: a DFA that recognizes a strict superset of a language does not recognize the language