

Computational Models - Lecture 1¹

Handout Mode

Iftach Haitner.

Tel Aviv University.

October 31, 2016

¹Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- ▶ Languages, words and alphabets
- ▶ Finite automata and regular languages
- ▶ Regular operations

- ▶ Sipser's book, chapter [1.1](#)

Part I

Languages, words and alphabets

Languages, words and alphabets

Definition 1

An **alphabet** Σ is a **finite** set of letters.

- ▶ $\Sigma = \{a, b, c, \dots, z\}$ – the English alphabet.
- ▶ $\Sigma = \{\alpha, \beta, \gamma, \dots, \zeta\}$ – the Greek alphabet.
- ▶ $\Sigma = \{0, 1\}$ – the binary alphabet.
- ▶ $\Sigma = \{0, 1, \dots, 9\}$ – the digital alphabet.

Definition 2

A **word** (i.e., string) over Σ , is a **finite** sequence of letters from Σ .

The collection of all strings over Σ is denoted by Σ^* .

For $\{0, 1\}$, the binary alphabet, ϵ , 1 , 0 , 00000000 , 1111111000 are all members of Σ^* .

Definition 3

A **language** over Σ is a (possibly infinite) subset of Σ^* .

Language Examples

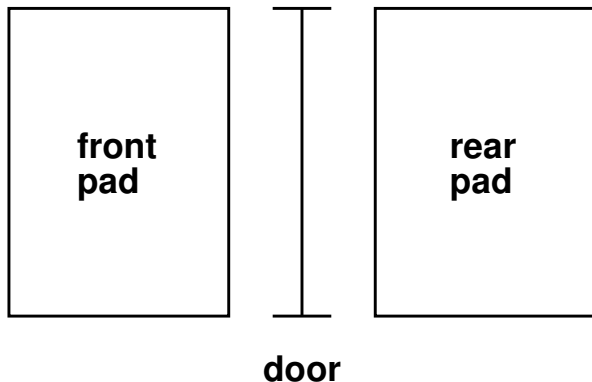
- ▶ Modern English.
- ▶ Ancient Greek.
- ▶ All prime numbers, written using digits.
- ▶ $\mathcal{A} = \{w \in \{0, 1\}^* : w \text{ has at most seventeen } 0\text{'s}\}$.
- ▶ $\mathcal{B} = \{0^n 1^n : n \geq 0\}$.
- ▶ $\mathcal{C} = \{w \in \{0, 1\}^* : w \text{ has an equal number of } 0\text{'s and } 1\text{'s}\}$.

Make sure you understand what the above notions stand for...

Part II

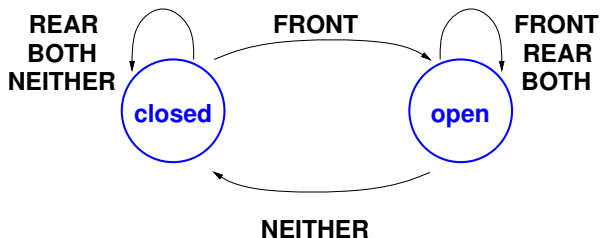
Finite Automata

Example: A One-Way Automatic Door



- ▶ open when person approaches
- ▶ hold open until person clears
- ▶ don't open when someone standing behind door

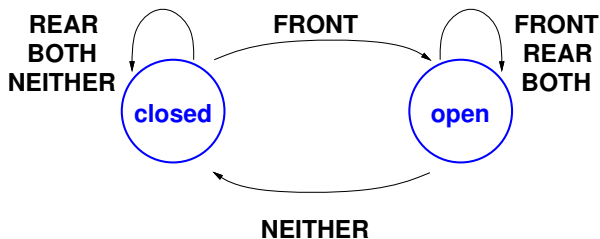
The Automatic Door as DFA



- ▶ States:
 - ▶ OPEN
 - ▶ CLOSED
- ▶ Sensor:
 - ▶ FRONT: someone on front pad
 - ▶ REAR: someone on rear pad
 - ▶ BOTH: someone(s) on both pads
 - ▶ NEITHER no one on either pad.

The Automatic Door as DFA

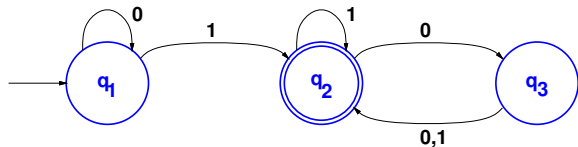
A DFA is Deterministic Finite Automata



	neither	front	rear	both
closed	closed	open	closed	closed
open	closed	open	open	open

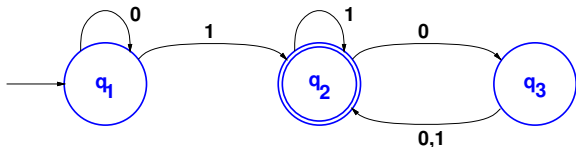
DFA: Informal definition

The machine M_1 :



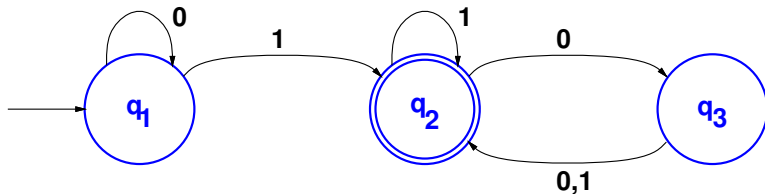
- ▶ **States:** q_1 , q_2 , and q_3 .
- ▶ **Start state:** q_1 (arrow from “outside”).
- ▶ **Accept state:** q_2 (double circle).
- ▶ **State transitions:** arrows tagged with letters.

DFA: Informal definition (cont.)



- ▶ On an input string
 - ▶ DFA begins in start state q_1
 - ▶ after reading each symbol, DFA makes **state transition** with matching label.
- ▶ After reading last symbol, DFA “produces” output:
 - ▶ **accept** if DFA is an accepting state.
 - ▶ **reject** otherwise.

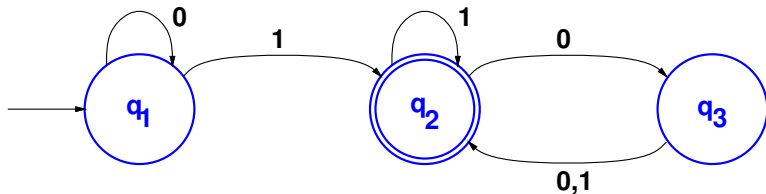
DFA: Informal definition (cont..)



What happens on the following input strings:

- ▶ 1101
- ▶ 0010
- ▶ 01100
- ▶ In general?!

DFA: Informal definition (cont...)



This DFA **accepts**

- ▶ All input strings that end with a **1**
- ▶ All input strings that contain at least one **1**, and end with an even number of **0**'s
- ▶ No other strings

Proof: ?

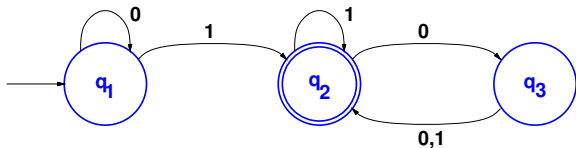
DFA - Formal Definition

Definition 4

A **deterministic finite automaton** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- ▶ Q is a finite set called the **states**
- ▶ Σ is a finite set called the **alphabet**
- ▶ $\delta : Q \times \Sigma \rightarrow Q$ is the **transition function**
- ▶ $q_0 \in Q$ is the **start state**
- ▶ $F \subseteq Q$ is the set of **accept states**

Back to M_1



$M_1 = (Q, \Sigma, \delta, q_1, F)$ where

▶ $Q = \{q_1, q_2, q_3\}, \Sigma = \{0, 1\},$

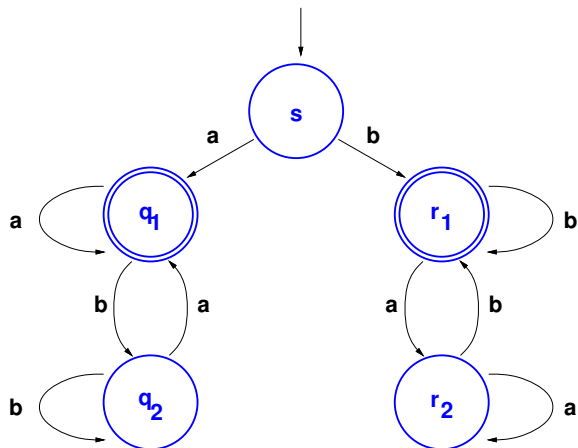
▶ the transition function δ is

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

▶ q_1 is the start state

▶ $F = \{q_2\}.$

Another Example



Formal Model of Computation

Definition 5

$M = (Q, \Sigma, \delta, q_0, F)$ accepts $w \in \Sigma^*$ if $\hat{\delta}_M(q_0, w) \in F$.

Definition 6 ($\hat{\delta}$)

$\hat{\delta}_M: Q \times \Sigma^* \mapsto Q$ is defined by

$$\hat{\delta}_M(q, w) = \begin{cases} \delta(\hat{\delta}(q, w_1, \dots, w_{n-1}), w_n), & n = |w| \geq 1 \\ q, & w = \varepsilon. \end{cases}$$

- ▶ w_1, \dots, w_k stands for the word w_1, \dots, w_k — the k -letter prefix of w
($w_1, \dots, w_0 = w_\emptyset = \varepsilon$)
- ▶ Note that $\hat{\delta}_M(q, \sigma) = \delta(q, \sigma)$ for $\sigma \in \Sigma$.
- ▶ We write $\hat{\delta}$ when M is clear from the context.

An equivalent definition

Definition 7 (Equivalent definition)

$M = (Q, \Sigma, \delta, q_0, F)$ **accepts** $w = w_1 w_2 \dots w_n$, if $\exists r_0, \dots, r_n \in Q$ s.t.,

- ▶ $r_0 = q_0$.
- ▶ $\delta(r_i, w_{i+1}) = r_{i+1}$, for all $0 \leq i < n$.
- ▶ $r_n \in F$.

The language of a DFA

Definition 8

$\mathcal{L}(M)$, the language of a DFA M , is the set of strings that M accepts.

We sometime say that M accepts the language $\mathcal{L}(M)$.

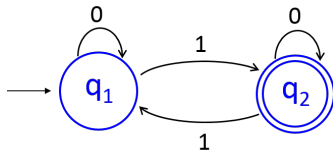
- ▶ M may accept many strings
- ▶ M accepts only one language.

What language does M accept if it accepts no strings?

Definition 9

A language is called regular, if some deterministic finite automaton accepts it.

Example: DFA M_2



- ▶ $Q = \{q_1, q_2\}$, $\Sigma = \{0, 1\}$, $F = \{q_2\}$, $\delta = ?$
- ▶ What is $\mathcal{L}(M_2) := \{w \in \{0, 1\}^* : \widehat{\delta}(q_1, w) = q_2\}$?

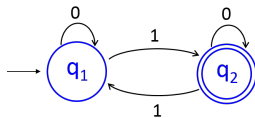
Theorem 10

$\mathcal{L}(M_2) = \{w \in \{0, 1\}^* : \#_1(w) \text{ is odd}\}$. $\#_1(w)$ — number of ones in w .

- ▶ Proof by induction on the word length
- ▶ What does it mean?

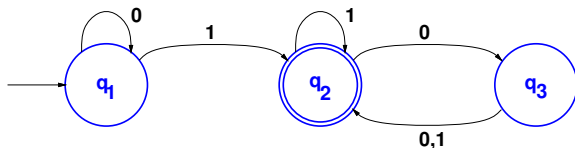
The j 'th assumption: $w \in \{0, 1\}^j$ is in $\mathcal{L}(M)$ iff $\#_1(w)$ is odd

Proving $\mathcal{L}(M_2) = \mathcal{L}' := \{w \in \Sigma^* : \#_1(w) \text{ is odd}\}$



- ▶ *Basis (length 0):* $\varepsilon \notin \mathcal{L}'$ and $\widehat{\delta}(q_1, \varepsilon) = q_1 \implies \varepsilon \notin \mathcal{L}(M_2)$.
- ▶ *Step:* assume hypothesis holds for words of length $j \geq 0$.
- ▶ Let $x = y\sigma \in \{0, 1\}^{j+1}$ for $\sigma \in \{0, 1\}$ (hence, $|y| = j$).
- ▶ If $\#_1(y)$ is **even**. By assumption $\widehat{\delta}(q_1, y) = q_1$.
 - ▶ $\sigma = 1 \implies$
 1. $\#_1(x)$ is **odd** $\implies x \in \mathcal{L}'$
 2. $\widehat{\delta}(q_1, x) := \delta(\widehat{\delta}(q_1, y), \sigma) = \delta(q_1, 1) = q_2 \implies x \in \mathcal{L}(M_2)$.
 - ▶ $\sigma = 0 \implies$
 1. $\#_1(x)$ is **even** $\implies x \notin \mathcal{L}'$
 2. $\widehat{\delta}(q_1, x) := \delta(\widehat{\delta}(q_1, y), \sigma) = \delta(q_1, 0) = q_1 \implies x \notin \mathcal{L}(M_2)$.
- ▶ If $\#_1(y)$ is **odd**...
- ▶ We proved for $x \in \{0, 1\}^{j+1}$: $x \in \mathcal{L}(M_2) \iff x \in \mathcal{L}'$

The language of M_1



Theorem 11

$$\mathcal{L}(M_1) = \{w10^{2k} : k \geq 0, w \in \{0, 1\}^*\}$$

Proof:

Claim 12 (implies the theorem)

Let $\mathcal{L}'_i = \{x \in \{0, 1\}^* : \widehat{\delta}(q_1, x) = q_i\}$ and let

- ▶ $\mathcal{L}_1 = \{0^k : k \geq 0\}$
- ▶ $\mathcal{L}_2 = \{w10^{2k} : k \geq 0, w \in \{0, 1\}^*\}$
- ▶ $\mathcal{L}_3 = \{w10^{2k+1} : k \geq 0, w \in \{0, 1\}^*\}$

Then, $\mathcal{L}'_i = \mathcal{L}_i$ for every $i \in \{1, 2, 3\}$

Proving Claim 12

We need to prove that $\forall i \in \{1, 2, 3\}$:

$$x \in \mathcal{L}_i \iff x \in \mathcal{L}'_i.$$

Recall that

$$x \in \mathcal{L}'_i \iff \widehat{\delta}(x) = q_i$$

- ▶ Proof by induction on word length.
- ▶ Induction basis: Easy to see that hypothesis holds for ϵ .
- ▶ Induction step: Assume hypothesis holds for words of length $j \geq 0$.
- ▶ Let $x = y\sigma \in \{0, 1\}^{j+1}$ for $\sigma \in \{0, 1\}$.
- ▶ We prove the hypothesis for x , separately for each $i \in \{1, 2, 3\}$

$$\mathcal{L}_1 = \{0^k : k \geq 0\}$$

Recall $x = y\sigma \in \{0, 1\}^{j+1}$ for $\sigma \in \{0, 1\}$.

Proving: $x \in \mathcal{L}_1 \implies \widehat{\delta}(q_1, x) = q_1 \quad (\implies x \in \mathcal{L}'_1).$

- ▶ $x = 0^{j+1}$, $y = 0^j$ and $\sigma = 0$.
- ▶ Since $y \in \mathcal{L}_1$, by i.h. $\widehat{\delta}(q_1, y) = q_1$
- ▶ Therefore, $\widehat{\delta}(q_1, x) = \delta(\widehat{\delta}(q_1, y), \sigma) = \delta(q_1, 0) = q_1$. \square

Proving: $\widehat{\delta}(q_1, x) = q_1 \implies x \in \mathcal{L}_1$.

- ▶ Let $q_y = \widehat{\delta}(q_1, y)$ (hence, $\widehat{\delta}(q_1, x) = \delta(q_y, \sigma) = q_1$)
- ▶ $q_y = q_1$ and $\sigma = 0$. (?)
- ▶ By i.h. $y = 0^j$.
- ▶ Hence, $x = y\sigma = 0^j 0 = 0^{j+1} \in \mathcal{L}_1$. \square

$$\mathcal{L}_2 = \{w10^{2k} : k \geq 0, w \in \{0, 1\}^*\}$$

Recall $x = y\sigma \in \{0, 1\}^{j+1}$ for $\sigma \in \{0, 1\}$.

Proving $x \in \mathcal{L}_2 \implies \widehat{\delta}(q_1, x) = q_2$.

▶ Assume $\sigma = 1$

▶ Since $\delta(q_i, 1) = q_2$ for any $i \implies \widehat{\delta}(q_1, x) = q_2$.

▶ Assume $x = w10^{2k}$ for $k > 0$ ($\implies y = w10^{2k-1}$ and $\sigma = 0$)

▶ Hence, $y \in \mathcal{L}_3$.

▶ By i.h. $\widehat{\delta}(q_1, y) = q_3$

▶ Thus, $\widehat{\delta}(q_1, x) = \delta(q_3, 0) = q_2$.

Proving $\widehat{\delta}(q_1, x) = q_2 \implies x \in \mathcal{L}_2$.

▶ Assume $\sigma = 1 \implies x \in \mathcal{L}_2$. (?)

▶ Assume $\sigma = 0$

▶ $q_y := \widehat{\delta}(q_1, y) = q_3$

▶ By i.h. $y = w10^{2k+1}$ for some $k \geq 0$

▶ Therefore $x = y\sigma = w10^{2k+1}0 \in \mathcal{L}_2$.

$$\mathcal{L}_3 = \{w10^{2k+1} : k \geq 0, w \in \{0, 1\}^*\}$$

Recall $x = y\sigma \in \{0, 1\}^{j+1}$ for $\sigma \in \{0, 1\}$.

Proving $x \in \mathcal{L}_3 \implies \widehat{\delta}(q_1, x) = q_3$.

- ▶ $x = w10^{2k+1}$, $y = w10^{2k}$ and $\sigma = 0$
- ▶ $y \in \mathcal{L}_2$
- ▶ By i.h. $\widehat{\delta}(q_1, y) = q_2$.
- ▶ Therefore, $\widehat{\delta}(q_1, x) = \delta(q_2, 0) = q_3$. \square

Proving $\widehat{\delta}(q_1, x) = q_3 \implies x \in \mathcal{L}_3$.

- ▶ Let $q_y = \widehat{\delta}(q_1, y)$
- ▶ Hence, $q_y = q_2$ and $\sigma = 0$ (?)
- ▶ By i.h. $y = w10^{2k}$
- ▶ Therefore, $x = y\sigma = w10^{2k}0 \in \mathcal{L}_3$. \square

Part III

Regular Operations

Additional examples of regular languages

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

- ▶ $\{w \in \{0, 1\}^* : \#_1(w) \equiv 0 \pmod{7}\}$.
- ▶ Sequence of 0 followed by sequence of 1, i.e., $\{0^m 1^n : m, n \geq 0\}$.
- ▶ Any **finite** language.

All the above languages are regular

Is there a simple proof?

The regular operations

Let \mathcal{A} and \mathcal{B} be languages.

The **union** operation:

$$\mathcal{A} \cup \mathcal{B} = \{x : x \in \mathcal{A} \vee x \in \mathcal{B}\}$$

The **concatenation** operation:

$$\mathcal{A} \parallel \mathcal{B} = \{xy : x \in \mathcal{A} \wedge y \in \mathcal{B}\}$$

The **star** operation:

$$\mathcal{A}^* = \{x_1 x_2 \dots x_k : k \geq 0 \text{ and each } x_i \in \mathcal{A}\}$$

The regular operations – Examples

Let $\mathcal{A} = \{\text{good, bad}\}$ and $\mathcal{B} = \{\text{boy, girl}\}$.

Union

$$\mathcal{A} \cup \mathcal{B} = \{\text{good, bad, boy, girl}\}$$

Concatenation

$$\mathcal{A} \parallel \mathcal{B} = \{\text{goodboy, goodgirl, badboy, badgirl}\}$$

Star

$$\mathcal{A}^* = \{\varepsilon, \text{good, bad, goodgood, goodbad, badbad, badgood, \dots}\}$$

Closure under union

Theorem 13

If \mathcal{L}_1 and \mathcal{L}_2 are regular languages, then so is $\mathcal{L}_1 \cup \mathcal{L}_2$.

Approach to Proof:

- ▶ Some DFA M_1 accepts \mathcal{L}_1
- ▶ Some DFA M_2 accepts \mathcal{L}_2
- ▶ Construct DFA M that accepts $\mathcal{L}_1 \cup \mathcal{L}_2$.

Attempted Proof Idea:

- ▶ first emulate M_1 , and
- ▶ if M_1 doesn't accept, then emulate M_2 .

What's **wrong** with this?


Fix: Emulate both machines **simultaneously**.

Closure Under Union: Correct Proof

Suppose

- ▶ $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ accepts \mathcal{L}_1 ,
- ▶ $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ accepts \mathcal{L}_2 .

Define M as follows (M will accept $\mathcal{L}_1 \cup \mathcal{L}_2$):

- ▶ $Q = Q_1 \times Q_2$.
- ▶ Σ is the same.
- ▶ For each $(r_1, r_2) \in Q$ and $a \in \Sigma$,
 $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- ▶ $q_0 = (q_1, q_2)$
- ▶ $F = \{(r_1, r_2) : r_1 \in F_1 \text{ or } r_2 \in F_2\}$.
- ▶ Formal proof (next slide) 

(hey, why not choose $F = F_1 \times F_2$?)

Correctness of the construction

Claim 14

$$\mathcal{L}(M) = \mathcal{L}(M_1) \cup \mathcal{L}(M_2).$$

Follows by the next claim.

Claim 15

$$\widehat{\delta}_M((q_1, q_2), x) = (\widehat{\delta}_1(q_1, x), \widehat{\delta}_2(q_2, x)).$$

Proof: By induction on word length. DIY ... ♣

Proving Claim 14:

- ▶ $x \in \mathcal{L}(M_1) \implies \widehat{\delta}_1(q_1, x) = r_1 \in F_1$. (similar if $x \in \mathcal{L}(M_2)$.)
- ▶ By Claim 15, $\widehat{\delta}_M((q_1, q_2), x) = (r_1, \cdot) \in F \implies x \in \mathcal{L}(M)$.
- ▶ $x \in \mathcal{L}(M) \implies \widehat{\delta}_M((q_1, q_2), x) = (r_1, r_2) \in F$.
- ▶ By Claim 15, $(r_1, r_2) = (\widehat{\delta}_1(q_1, x), \widehat{\delta}_2(q_2, x))$.
- ▶ Hence, either $r_1 \in F_1$ or $r_2 \in F_2 \implies x \in \mathcal{L}(M_1) \cup \mathcal{L}(M_2)$.

What about concatenation?

Theorem 16

If $\mathcal{L}_1, \mathcal{L}_2$ are regular languages, then so is $\mathcal{L}_1\|\mathcal{L}_2$.

Example: $\mathcal{L}_1 = \{\text{good, bad}\}$ and $\mathcal{L}_2 = \{\text{boy, girl}\}$.

$$\mathcal{L}_1\|\mathcal{L}_2 = \{\text{goodboy, goodgirl, badboy, badgirl}\}$$

This is much harder to prove.

Idea: Simulate M_1 for a while, then **switch** to M_2 .

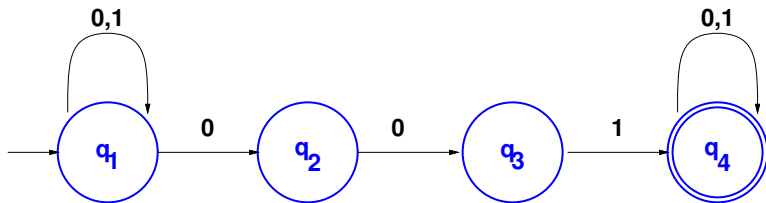
Problem: But **when** do you switch?

This leads us into **non-determinism**, wait for next class...

Part IV

Non-deterministic Finite Automata

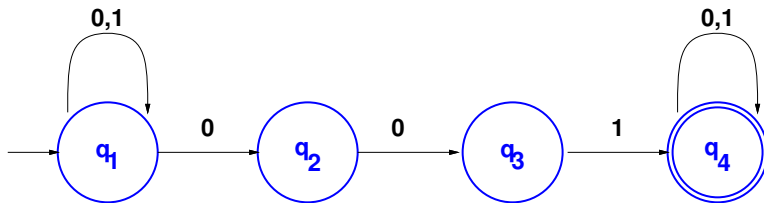
NFA — non-deterministic Finite Automata



- ▶ May have **more than one transition** labeled with the same symbol,
- ▶ May have **no transitions** labeled with a certain symbol,
- ▶ May have transitions labeled with ϵ , the symbol of the **empty string**. **Will deal with this latter**

Every **DFA** “is” also an NFA.

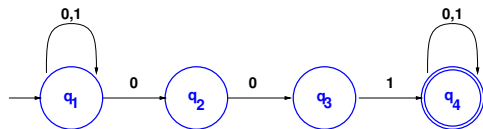
Non-deterministic computation



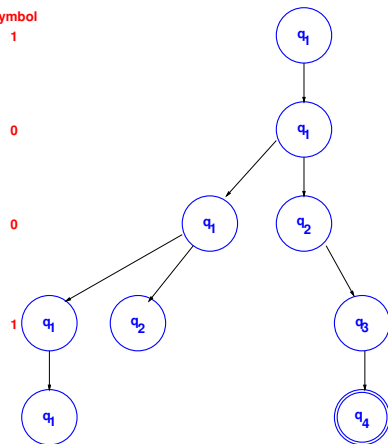
What happens when more than one transition is possible?

- ▶ The machine “splits” into **multiple copies**
- ▶ Each branch follows one possibility
- ▶ Together, branches follow **all** possibilities.
- ▶ If the input doesn't appear, that branch “dies”.
- ▶ Automaton accepts if **some** branch accepts.

Computation on 1001



symbol
1



Why non-determinism?

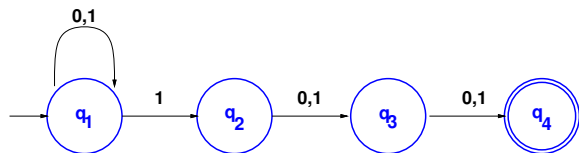
Theorem 17 (Informal, to be proved soon)

*Deterministic and non-deterministic finite automata, **accept** exactly the **same set of languages**.*

Q.: So **why** do we need NFA's?

Design a finite automaton for the language \mathcal{L} — all binary strings with a **1** in their **third-to-the-last** position?

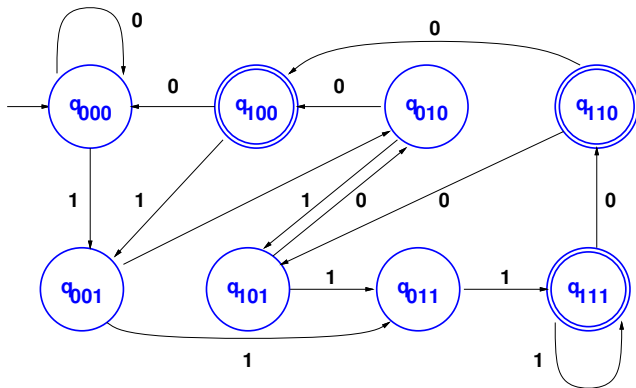
NFA for \mathcal{L}



- ▶ “Guesses” which symbol is third from the last, and
- ▶ checks that indeed it is a **1**.
- ▶ If guess is premature, that branch “dies”, and no harm occurs.

DFA for \mathcal{L}

- ▶ Have 8 states, encoding the last three observed letters.
- ▶ A state for each string in $\{0, 1\}^3$.
- ▶ Add transitions on modifying the suffix, give the new letter.
- ▶ Mark as accepting, the strings $1**$



DFA has few bugs...