SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

FOUNDATIONS OF COMPUTER SCIENCE
LECTURE 1: Languages and Automata

Prof. Daniele Gorla

Alphabets, Strings and Languages

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- An ***alphabet*** Σ is any non-empty finite set, whose elements are called ***symbols*** of the alphabet (or ***characters***)

  EXAMPLE:

  $\Sigma_1 = \{0,1\}$
  $\Sigma_2 = \{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$

- A ***string*** w over an alphabet Σ is a finite sequence of symbols from Σ, usually written one after the other and not separated by any other symbol.

- If w is a string over Σ, the ***length*** of w, written $|w|$, is the number of symbols that it contains. The string of length zero is called the ***empty string*** and is written ε.

- The ***reverse*** of w, written w^R, is the string obtained by writing w in the opposite order

- String z is a ***substring*** of w if z appears consecutively within w.

  EXAMPLE: 01001 is a string over Σ_1 , its length is 5, its reverse is 10010 and some of its substrings are ε , 100 , 01

Alphabets, Strings and Languages (cont'd)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- If we have strings x and y, their ***concatenation***, written xy, is the string obtained by appending y to the end of x; clearly, $|xy| = |x| + |y|$
- Notation x^k denotes the concatenation of x with itself k times (for $k \in \mathbf{N}$)

  → so, in particular: $x^0 = \varepsilon$, $x^1 = x$, $x^2 = xx$, …

- A string x is a ***prefix*** of string y if there exists a string z such that $xz = y$; x is a ***proper prefix*** of y if, in addition, $x \neq y$ (or, if you prefer, $z \neq \varepsilon$)
- A string x is a ***suffix*** of string y if there exists a string z such that $zx = y$; x is a ***proper suffix*** of y if, in addition, $x \neq y$ (or, if you prefer, $z \neq \varepsilon$)

- A ***language*** is a set of strings; it is ***prefix-free*** if no member is a proper prefix of another member.

- The ***lexicographic order*** of strings is the same as the familiar dictionary order.
- The ***shortlex order*** or simply ***string order***, is identical to lexicographic order, except that shorter strings precede longer strings.

  EXAMPLE: the string ordering of the language of binary strings is

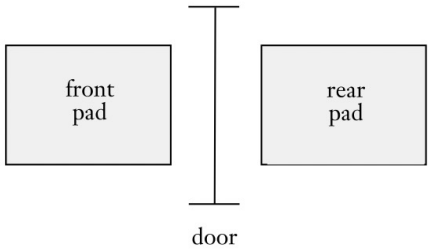
  ε, 0, 1, 00, 01, 10, 11, 000, . . .

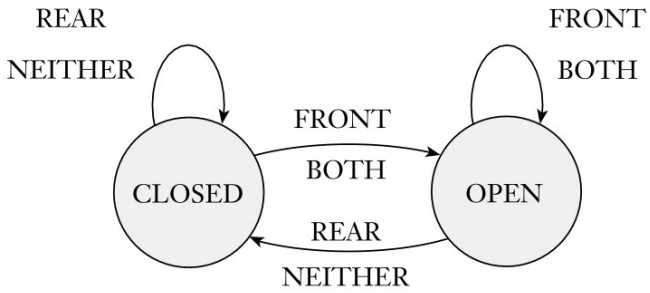
Automata

Finite automata are good models for computers with a finite amount of memory.

EXAMPLE:

- The controller for the automatic door of a supermarket opens the door when a person is approaching and closes it after the passage.
- The door has a front pad (to detect the presence of a person about to walk through the doorway) and a rear pad (to hold the door open long enough for the person to pass all the way through)





- The controller can be in two states: “OPEN” or “CLOSED”
- There are four possible input conditions (that make state change):
  - “FRONT” (a person is standing on the pad in front of the doorway),
  - “REAR” (a person is standing on the pad to the rear of the doorway),
  - “BOTH” (people are standing on both pads), and
  - “NEITHER” (no one is standing on either pad).

Finite Automaton

DEFINITION

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

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

   Accept states sometimes are called ***final states***.

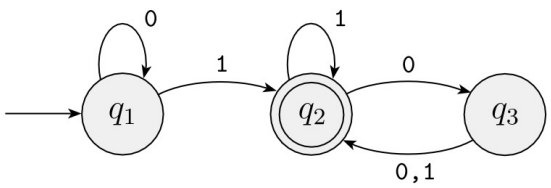
EXAMPLE: the 5-tuple:

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state,
5. $F = \{q_2\}$.

can be graphically represented as:



Language of an Automaton

- When an automaton receives an input string, it moves from the initial state according to
  - the input characters read (one by one), and
  - the transition function
- This leads the automaton to an arrival state.
- If such a state belongs to F (accept/final states), then we say that the automaton ***accepts*** the input; otherwise, it ***rejects*** the input.

- The ***language of an automaton M,*** written $L(M)$, is the set of all the input strings that lead the automaton from its starting state to a final one. Formally:

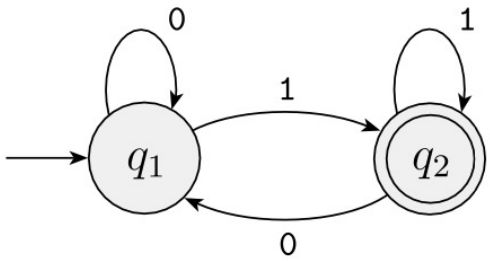
Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$, and
3. $r_n \in F$.

We say that M ***recognizes language*** A if $A = \{w | M \text{ accepts } w\}$.

Examples (1)

Consider the following automaton:



that corresponds to the 5-tuple $(\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$

where δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2.

The strings it accepts (leading from q_1 – the initial state – to q_2 – the only final state) are:

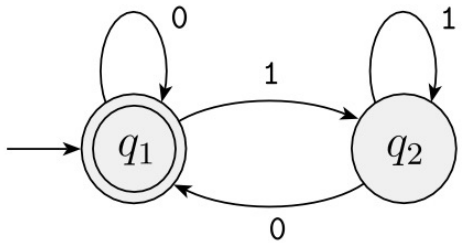
1 , 01 , 11 , 001 , 011 , 101 , 111 , …

By looking at the picture, all arrows that enter into q_2 are labeled with 1; hence, we can conclude that the language accepted by this automaton is

$\{w \mid w$ ends with a 1$\}$

Examples (2)

If we now change the final state of the previous automaton, we obtain



and now the initial state and the final one do coincide.

By reasoning like before, we have that

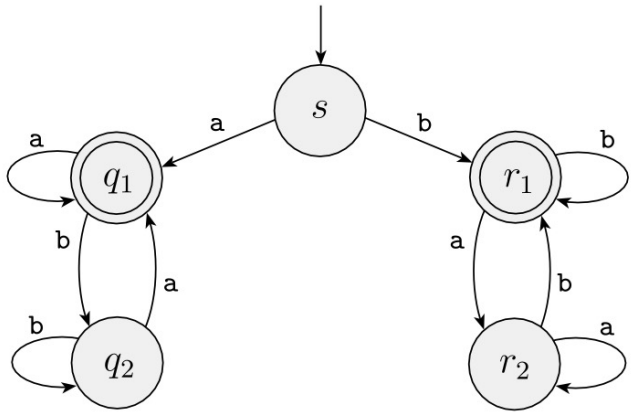
$L(M)$ = {w | w does NOT end with a 1}
= {w | w is ε or it ends with a 0}

EXERCISE: Draw an automaton that accepts all binary sequences starting and ending with a 1.

Examples (3)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Consider the automaton:



All input sequences that arrive to q_1 start with an a (arrow from s to q_1) and terminate with an a (all arrows entering into q_1 are labeled with a)

→ q_1 accepts all strings starting and ending with an a

All input sequences that arrive to r_1 start with a b (arrow from s to r_1) and terminate with a b (all arrows entering into r_1 are labeled with b)

→ r_1 accepts all strings starting and ending with a b

So,

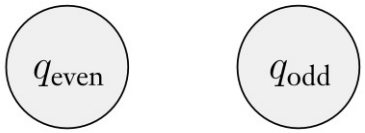
$L(M) = \{w \mid w$ starts and ends with an a or with a $b\}$

Regular Languages

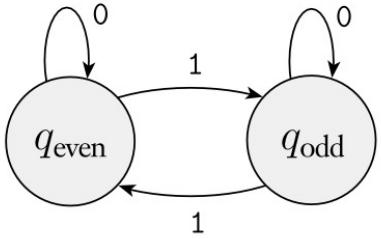
Def.: A language L is said ***regular*** if there exists a finite automaton that accepts L.

EXAMPLE: The language of binary strings with an odd number of 1s is regular?

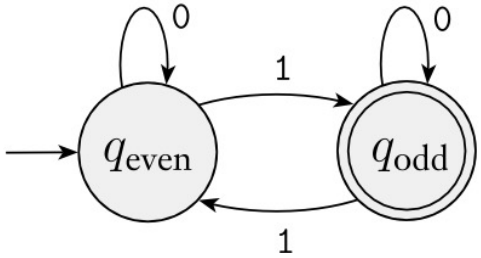
- You need just 2 states, recording whether the 1s received so far are odd or even:



- 0s don't change the parity of 1s received so far, whereas 1s do:



- At the beninning no 1 has been received yet (so #1s = 0, that is an even number) and we accept whenever we have seen an odd number of 1s:



Example

The language of all binary strings that contain the string 001 as a substring is regular?

Idea:

- initially skip all 1s
- If a 0 arrives, this CAN BE the first one in the pattern 001 you are seeking.
- If at this point you see a 1, there were too few 0s, so you go back to the initial state
- But if you see a 0 at that point, you should remember that you have just seen two symbols of the pattern.
- Now you simply need to continue scanning until you see a 1.
- If you find it, remember that you succeeded in finding the pattern and continue reading the input string until you get to the end.

This requires 4 states:

1. haven't just seen any symbols of the pattern

2. have just seen a 0

3. have just seen 00

4. have seen all 001