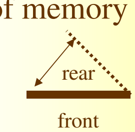
CHAPTER 1
Regular Languages

Contents

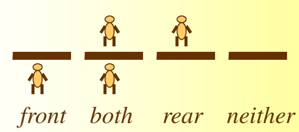
- **Finite Automata (FA or DFA)**
  - **definitions, examples, designing, regular operations**
- Non-deterministic Finite Automata (NFA)
  - definitions, equivalence of NFAs and DFAs, closure under regular operations
- Regular expressions
  - definitions, equivalence with finite automata
- Non-regular Languages
  - the pumping lemma for regular languages

Finite Automata

- A simplest computational model
- Models computers with very small amount of memory
  - various electromechanical devices
- Example: automatic door controller
  - i.e., entries, exits in supermarkets



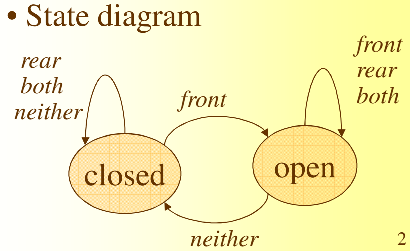
- Two states: *open, closed*
- 4 possible inputs (from two sensors)



- State transition table

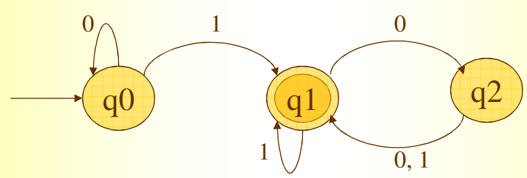
	Input signal			
State	*neither*	*front*	*rear*	*both*
closed	*closed*	*open*	*closed*	*closed*
open	*closed*	*open*	*open*	*open*

- State diagram



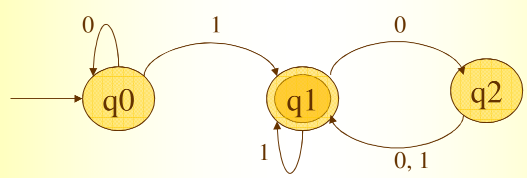
Finite Automata

- Other examples:
  - elevator controller, dish washers, digital watches, calculators …
- Next we will give
  - a definition of FA from mathematical prospective
  - terminology for describing and manipulating FA
  - theoretic results describing their power and limitation
- A finite automaton M_1 that has three states



(Deterministic) Finite Automata: Definition

- An important way to describe certain simple but highly useful languages called 'regular languages'. It is
  - A graph with a finite number of nodes, called *states*.
  - Arcs are labeled with one or more *symbols* from some *alphabet*.
  - One state is chosen as the *start state* or *initial state*.
  - Some states are *final states* or *accepting states*.
  - The *language of the FA* is the set of strings that label paths that go from the start state to some final state.
- Example: a finite automaton M_1 that has three states

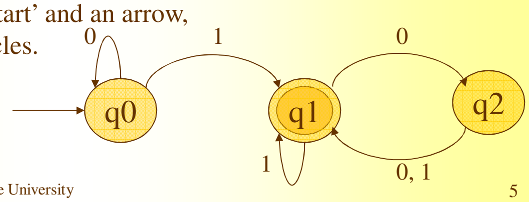


Formal Definition of DFAs

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

Q	is a finite set of states,	$Q = \{q0, q1, q2\}$
Σ	is a finite alphabet,	$\Sigma = \{0,1\}$
$\delta : Q \times \Sigma \to Q$	is the transition function,	$\delta(q0,1) = q1,\ \delta(q2,0) = q1,\ \ldots$
$q_0 \in Q$	is the initial state,	$q_0 = q0$
$F \subseteq Q$	is the set of final states.	$F = \{q1\}$

- A DFA is usually represented by a directed labeled graph (so called *transition diagram*)
  - nodes = states,
  - arc from q to p is labeled by the set of input symbols a such that $\delta(q, a) = p$
  - no arc if no such a,
  - start state indicated by word 'start' and an arrow,
  - accepting states get double circles.
- For DFA M_1 we have

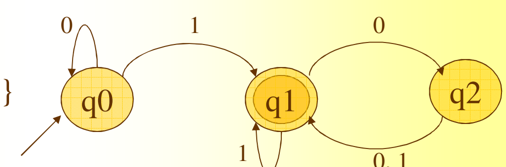


Acceptance of Strings and the Language of DFA

- Let $M=$ $(Q, \Sigma, \delta, q_0, F)$ be a finite automaton
- Let $w = w_1, w_2, \ldots, w_n$ be a string over Σ
- *M* accepts *w* if a sequence of states $r_0, r_1, r_2, \ldots, r_n$ exists in *Q* with the following 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. $r_n \in F$

- If *L* is a set of strings that *M* accepts, we say that *L* is the *language of machine M* and write *L=L(M).*
- We say *M recognizes L* or *M accepts L.*
- In our example, L(M_1)=L, where L={w: w contains at least one 1 and an even number of 0s follow the last 1}

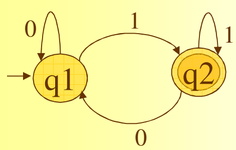


Regular Languages

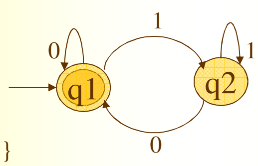
- A language is called *regular language* if some finite automaton recognizes it.
- Main questions is:

  Given a language L, construct, if possible, a FA M which recognizes L, i.e., $L(M)=L$. (Is L a regular language?)
- Another question is

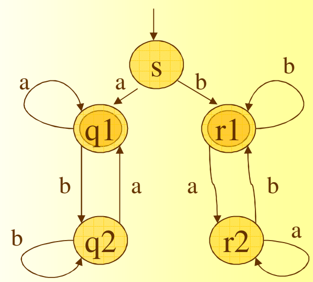
  Given a FA M, describe its language L(M) as a set of strings over its alphabet.
- First consider some examples:



$L(M_2)=\{$w: w ends in a 1$\}$



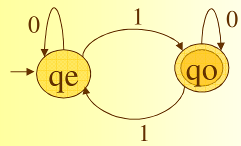
$L(M_3)=\{$w: w is ε or ends in a 0$\}$



$L(M_4)=\{$w: w starts and ends with the same symbol$\}$

Design of DFAs

- Given a language, design a finite automaton that recognizes it.
- This is a creative process, is art.
- No simple universal methods.
- One intuitive way: put yourself in the place of the machine you are trying to design and see how you would go about performing the machine's task.
  - we need to receive a string and to determine whether it is a member of the language.
    - we get to see the symbols in the string one by one.
    - after each symbol we must decide whether the string seen so far is in the language (we don't know when the end of the string is coming, so we must always be ready with the answer).
    - we have a finite memory, so we have to figure out what we need to remember about the string as we are reading it (we cannot remember all we have seen). In this way we will find out how many states our automaton will have (and what are the states).
- Example: the language consists of all strings over alphabet {0,1} with odd number of 1s.



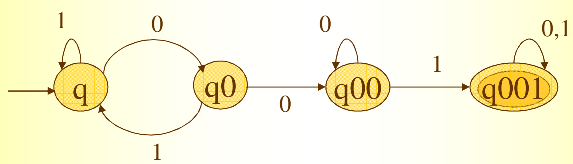
Two possibilities:
1. Even so far,
2. Odd so far.

- Hence two states.

L={strings with odd number of 1s}

Design of a FA for L={strings containing 001}

- We scan our string. There are four possibilities:
  1. have not just seen any symbol of the pattern,
  2. have just seen a 0,
  3. have just seen 00, or
  4. have seen the entire pattern 001.
- So, four states q, q0, q00, q001
- Therefore,



Regular Operations

- Let L1 and L2 be languages. We define the regular operations ***union, intersection, concatenation,*** and ***star*** as follows.
  - ***Union:*** $L1 \cup L2 = \{w: \ w \in L1 \quad or \quad w \in L2\}.$
  - ***Intersection:*** $L1 \cap L2 = \{w: \ w \in L1 \quad and \quad w \in L2\}.$
  - ***Concatenation:*** $L1 \circ L2 = \{wv: \ w \in L1 \quad and \quad v \in L2\}.$
  - ***Star:*** $L1^* = \{w_1 w_2 ... w_k : \ k \geq 0 \quad and \quad each \quad w_i \in L1\}.$
- Example: Let the alphabet Σ be the standard 26 letters {a,b,…,z}.
  - If L1={good, bad} and L2= {boy, girl}, then

$L1 \cup L2 =$ {good, bad, boy, girl}.

$L1 \cap L2 = \varnothing$

$L1 \circ L2 =$ {goodboy, badboy, goodgirl, badgirl}.

$L1^* =$ { ε , good, bad, goodgood, badgood, badbad, goodbad, goodgoodgood, goodgoodbad, goodbadbad, …}

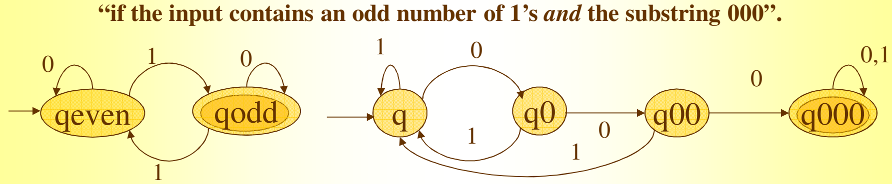
Product construction

- It would be useful if we can construct FAs for $L1 \cup L2$ and $L1 \cap L2$ from FAs for *L1* and *L2,* no matter what *L1* and *L2*.
- This would give a systematic way of constructing from the automata M1 and M2 an automaton that checks

  "if the input contains an odd number of 1's *or* the substring 000"

  as well as

  "if the input contains an odd number of 1's *and* the substring 000".



L(M1)={w: w contains an odd number of 1's} L(M2)={w: w contains substring 000}

- we can run both automata 'in parallel', by remembering the states of both automata while reading the input.
  - This is achieved by the *product construction*

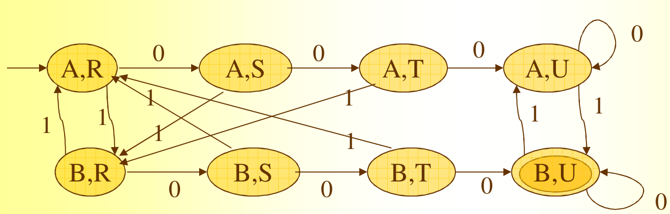
Product construction

- Let M1= $(Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and M2= $(Q_2, \Sigma, \delta_2, q_{02}, F_2)$ be two FA with the same alphabet.
- We define a finite automaton $M_\cup$ such that $L(M_\cup) = L(M1) \cup L(M2)$ and $M_\cap$ such that $L(M_\cap) = L(M1) \cap L(M2)$ as follows.

$$M_\cup = (Q, \Sigma, \delta, q_0, F_\cup)$$
$$M_\cap = (Q, \Sigma, \delta, q_0, F_\cap)$$

where

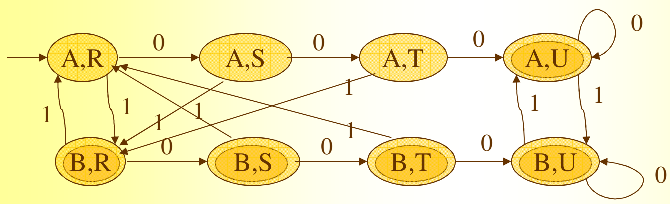
$$Q = Q_1 \times Q_2,$$
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),$$
$$q_0 = (q_{01}, q_{02}),$$
$$(q_1, q_2) \in F_\cap \;\; iff \;\; q_1 \in F_1 \;\; and \;\; q_2 \in F_2,$$
$$(q_1, q_2) \in F_\cup \;\; iff \;\; q_1 \in F_1 \;\; or \;\; q_2 \in F_2.$$



Here,
A=qood
B=qeven
R=q
S=q0
T=q00
U=q000

$L(M_\cap) = L(M1) \cap L(M2)$ ={w: w contains an odd number of 1's *and* the substring 000}

Product construction



$L(M_{\cup}) = L(M1) \cup L(M2) =$ {w: w contains an odd number of 1's *or* the substring 000}

We have proved by construction that

Theorem 1. The class of regular languages is closed under the *union* operation.

Theorem 2. The class of regular languages is closed under the *intersection* operation.

Later we will show that they are closed under *concatenation* and *star* operations.