Deterministic Finite Automata (DFA)

(LECTURE 3)

Introduction

- Finite automata and regular sets
- Definition of deterministic finite automata
- String accepted by DFA

Finite Automata and regular sets (languages)

- States and transitions:

Ex: Consider a counter data structure (system):

- unsigned integer counter: pc; { initially pc = 0}
- operations: inc, dec;

==> The instantaneous *state* of the system can be identified by the value of the counter. Operations called from outside world will cause *transitions* from states to states and hence change the current state of the system.

Problem: how to describe the system :

Mathematical approach: CS = (S, O, T, s, F) where

S = The set of all possible states = N

O = the set of all possible [types of] operations

T = the response of the system on operations at all possible states. (present state, input operation) --> (next state)

Example of a state machine

T can be defined as follows : T: SxO --> S s.t., for all x in S ,

- T(x, inc) = x + 1 and T(x, dec) = x - 1 ; { 0 - 1 $=_{def}$ 0 }

- s = 0 is the initial state of the system
- F $\subseteq$ S is a set of distinguished states, each called a final state.
  (we can use it to, say, determine who can get a prize)
- Graphical representation of CS:
- Note: The system CS is infinite in the sense that S (the set of all possible states) and Transitions (the set of possible transitions) are infinite. A system consists of only finitely many states and transitions is called a *finite-state transition system*. The mathematical tools used to model finite-state transition system are called *finite automata*.
- *examples of state-transition systems: electronic circuits; digital watches, cars, elevators, etc.*

Deterministic Finite automata (the definition)

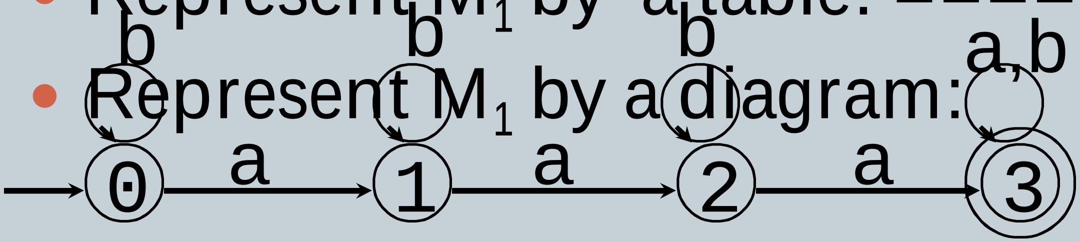
- a DFA is a structure M = (Q,S, d,s,F) where
  - Q is a finite set; elements of S are called states
  - S is a finite set called the input alphabet
  - d:QxS --> Q is the transition function with the intention that if M is in state q and receive an input a, then it will move to state d(q,a).
    - e.g; in CS: d(3, inc) = 4 and d(3, dec) = 2.
  - s in Q is the start state
  - F is a subset of Q; elements of F are called accept or final states.
- To specify a finite automata, we must give all five parts (maybe in some other forms)
- Other possible representations:
  - [state] transition diagram or [state] transition table

Example and other representations

Ex 3.1: M_1 = (Q,S,d,s,F) where

- Q={0,1,2,3}, S={a,b}, s = 0, F = {3} and d is defined by:
- d(0,a) = 1; d(1,a) = 2; d(2,a) = d(3,a) = 3 and
- d(q,b)=q if q = {0,1,2,3}.
- problem: Although precise but tedious and not easy to understand (the behavior of) the machine.

- Represent M_1 by a table: =====>
- Represent M_1 by a diagram:



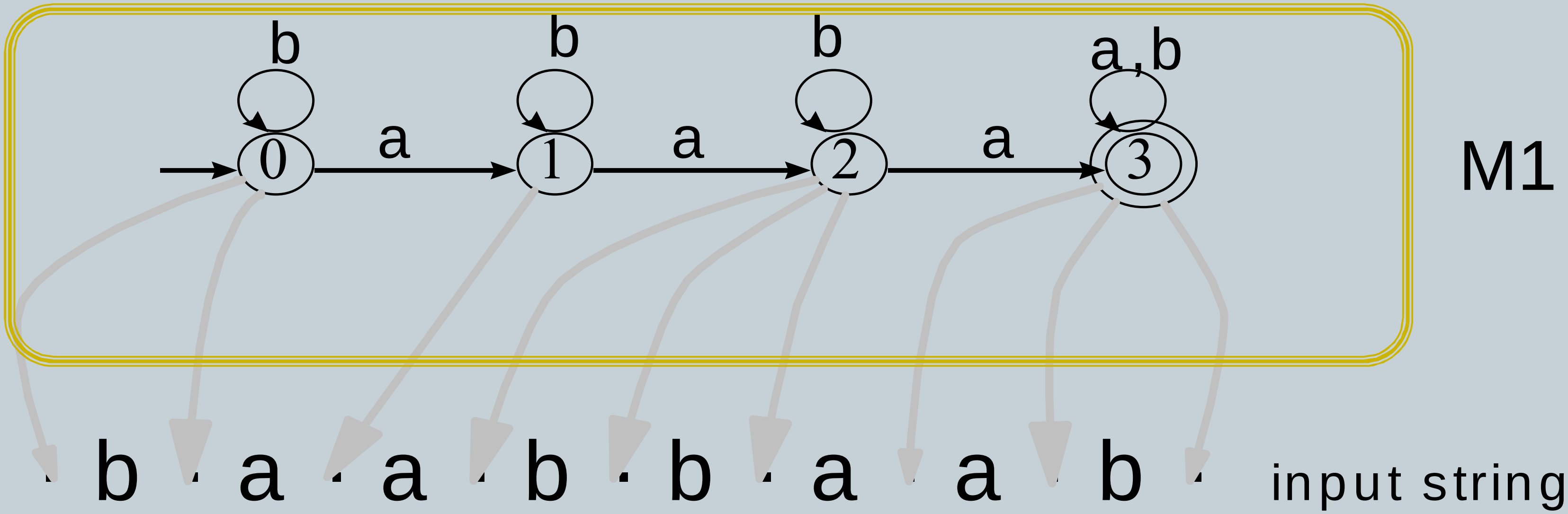
state-transition diagram for M_1

note: the naming of states is not necessary

	a	**b**
>0	**1**	**0**
1	**2**	**1**
2	**3**	**2**
3F	**3**	**3**

Strings accepted by DFAs

- Operations of M_1 on the input 'baabbaab':



0 b 0 a 1 a 2 b 2 b 2 a 3 a 3 b 3 -- execution path

- **Since M_1 can reach a final state (3) after scanning all input symbols starting from initial state, we say the string 'baabbaab' is accepted by M_1.**

Problem: How to formally define the set of all strings accepted by a DFA ?

The extended transition function D

- Meaning of the transition function:

  q1 -- a --> q2 [or d(q1,a) = q2] means

  if M is in state q1 and the currently scanned symbol (of the input strings is a) then

  - 1. Move right one position on the input string (or remove the currently scanned input symbol)
  - 2. go to state q2. [So M will be in state q2 after using up a)

- Now we extend d to a new function D: Q x S* --> Q with the intention that : D(q1,x) =q2 iff

  starting from q1, after using up x the machine will be in state q2. --- D is a multi-step version of d.

Problem: Given a machine M, how to define D [according to d] ?

Note: when string x is a symbol (i.e., |x| = 1) then D(q,x) = d(q,x).

for all state q, so we say D is an extension of d.

The extended transition function D (cont'd)

- D can be defined by induction on |x| as follows:
  - Basis: |x|= 0 (i.e., x = e) ==> D(q, e) = q   --- (3.1)
  - Inductive step: (assume D(q,x) has been defined) then
  - D(q, xa) = d(D(q,x), a)   --- (3.2)
  - --- To reach the state D(q,xa) from q by using up xa, first use up x (and reach D(q,x)) and then go to d((D,qx),a) by using up a.

- Exercise: Show as expected that D(q,a) = d(q,a) for all a in S.

pf: D(q,a) = D(q,ea) = d(D(q,e),a) = d(q,a).

Uniqueness of the extended transition funciton

- Note: D is uniquely defined by M, i.e., for every DFA M, there is exactly one function f:QxS* --> Q satisfying property (3.1) and (3.2.)
  - --- a direct result of the theorem of recursive definition.

pf: Assume $ distinct f1 and f2 satisfy (3.1&3.2).

Now let x be any string with least length s.t. f1(q,x) ≠ f2(q,x) for some state q.

==> 1. x ≠ e (why ?)

2. If x = ya ==> by minimum of |x|, f1(q,y) = f2(q,y), hence

f1(q,ya)=d(f1(q,y),a) = d(f2(q,y),a) = f2(q,ya), a contradiction.

Hence f1 = f2.

Languages accepted by DFAs

- M = (Q,S,d,s,F) : a DFA; x: any string over S;
  D: the extended transition function of M.

1. x is said to be *accepted* by M if D(s,x) $\in$ F
   x is said to be *rejected* by M if D(s,x) $\notin$ F.
2. The set (or language) accepted by M, denoted L(M), is the set of all strings accepted by M. i.e.,
   - L(M) $=_{def}$ {x $\in$ S* | D(s,x) $\in$ F }.
3. A subset A $\subseteq \Sigma^*$ (i.e., a language over S) is said to be *regular* if A is accepted by some finite automaton (i.e., A = L(M) for some DFA M).

Ex: The language accepted by the machine of Ex3.1 is the set

L(M1) = {x $\in$ {a,b}* | x contains at least three a's}

Another example

Ex 3.2: Let A = {xaaay | x,y ∈ {a,b}*}

= {x ∈ {a,b}* | x contains substring aaa }.

Then *baabaaaab* ∈ A and *babbabab* ∉ A.

An Automaton accept A: (diagram form)

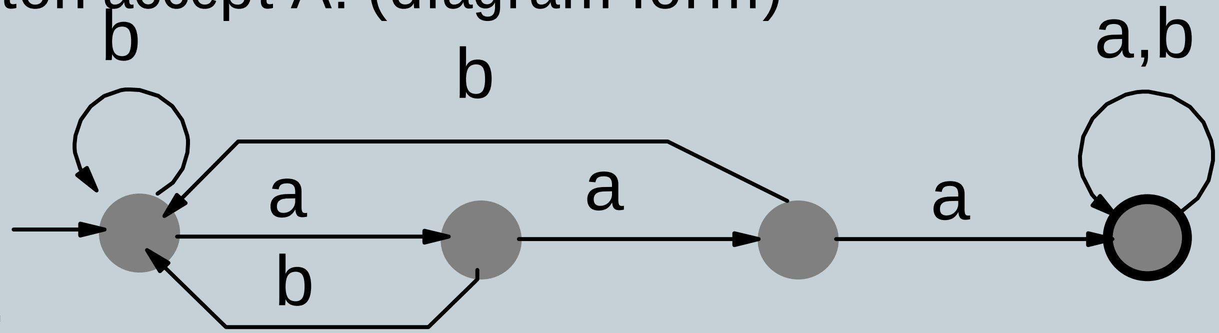


Table form:

	a	b
>0	1	0
1	2	0
2	3	0
3F	3	3

More on regular sets (Lecture 4)

- a little harder example:
  Let A = {x ∈ {0,1}* | x represent a multiple of 3 in binary}.
  - notes: leading 0's permitted; e represents zero.
  - example:
  - e, 0, 00 ==> 0; 011,11,.. ==> 3; 110 ==> 6;
  - 1001 ==> 9; 1100,.. ==> 12; 1111 => 15; ...
- Problem: design a DFA accepting A.

sol: For each bit string x, s(x) = #(x) mod 3, where #(x) is the number represented by x. Note: s: {0,1}* → {0,1,2}

- Ex: s(e) = 0 mod 3 = 0; s(101) = 5 mod 3 = 2;...
- ==> A = { x | s(x) = 0 }
- 1. s(e) = 0;
- s(x0) and s(x1) can be determined from s(x) as follows:

a little harder example

- Since #(x0) = 2 #(x)

==> s(x0) = #(x0) mod 3 = 2(#(x) mod 3) mod 3

= 2s(x) mod 3

==> s(x) can be show as follows:

(note: the DFA M defined by the table is also the automata accepting A)

	0	**1**
>0F	**0**	**1**
1	**2**	**0**
2	**1**	**2**
s(x)	**s(x0)**	**s(x1)**

- Exercise: draw the diagram form of the machine M accepting A.
- Fact: L(M) = A. (i.e., for all bit string x, x in A iff x is accepted by M)

pf: by induction on |x|. Basis: |x| = 0 => x = e in A and is accepted by M.

Ind. step: x = yc where c in {0,1}

=> D(0, yc) = d(D(0,y),c) = d(#(y) mod 3, c)

= (2#(y) mod 3 +c) mod 3 = #(xc) mod 3. QED

Some closure properties of regular sets

Issue: what languages can be accepted by finite automata ?

- Recall the definitions of some language operations:
  - A U B = {x | x ∈ A or x ∈ B}.
  - A ∩ B = {x | x ∈ A /\ x ∈ B}
  - ~A = S* - A = {x ∈ S* | x ∉ A}
  - AB = {xy | x ∈ A /\ y ∈ B}
  - A* = {x_1 x_2 ...x_n | n ≥ 0 /\ x_i ∈ A for 0 ≤ i ≤ n}
  - and more ... ex: A / B = {x | $y ∈ B s.t. xy ∈ A }.
- Problem: If A and B are regular [languages], then which of the above sets are regular as well?

Ans: _______.

The product construction

- $M_1 = (Q_1,S,d_1,s_1,F_1)$, $M_2 = (Q_2,S,d_2,s_2,F_2)$: two DFAs

Define a new machine $M_3 = (Q_3, S, d_3, s_3, F_3)$ where

  - $Q_3 = Q_1 \times Q_2 = \{(q_1,q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$
  - $s_3 = (s_1,s_2)$;
  - $F_3=F_1 x F_2 = \{(q_1,q_2) \mid q_1 \in F_1 \wedge q_2 \in F_2\}$ and
  - $d_3:Q_3 \times S \to Q_3$ is defined to be

    $d_3((q_1,q_2), a) = (d_1 (q_1,a), d_2 (q_2,a))$

    for all $(q_1,q_2) \in Q$, $a \in S$.

- The machine M_3, denoted $M_1 x M_2$, is called the *product* of M_1 and M_2. The behavior of M_3 may be viewed as the parallel execution of M_1 and M_2.
- Lem 4.1: For all $x \in S^*$, $D_3((p,q),x) = (D_1(p,x), D_2(q,x))$.

Pf: By induction on the length $|x|$ of x.

Basis: $|x|= 0$: then $D_3((p,q),e) = (p,q) = (D_1 (p,e), D_2(q,e))$

The product construction (cont'd)

Ind. step: assume the lemma hold for x in S*, we show it holds for xa, where a in S.

$D_3((p,q),xa) = d_3(D_3((p,q),x), a)$ --- definition of D_3

$= d_3((D_1(p,x), D_2 (q,x)), a)$ --- Ind. hyp.

$= (d_1(D_1(p,x),a), d_2(D_2 (q,x),a)$ --- def. of d_3

$= (D_1(p,xa), D_2(p,xa))$ QED --- def of D_1 and D_2.

Theorem 4.2: $L(M_3) = L(M_1) \cap L(M_2)$.

pf: for all $x \in S^*$, $x \in L(M_3)$

iff $D_3(s_3,x) \in F_3$ --- def. of acceptance

iff $D_3((s_1,s_2),x) \in F_3$ --- def. of s_3

iff $(D_1(s_1,x), D_2(s_2,x)) \in F_3 = F_1xF2$ --- def. of F_3

iff $D_1(s_1,x) \in F_1$ and $D_2(s_2,x) \in F_2$ --- def. of set product

iff $x \in L(M_1)$ and $x \in L(M_2)$ --- def. of acceptance

iff $x \in L(M_1) \cap L(M_2)$. QED --- def. of intersection.

Regular languages are closed under U, $\cap$ and ~

Theorem: IF A and B are regular than so are $A \cap B$, ~A and AUB.

pf: (1) A and B are regular

=> \$ DFA M_1 and M_2 s.t. $L(M_1) = A$ and $L(M_2) = B$ -- def. of RL

=> $L(M_1 x M_2) = L(M_1) \cap L(M_2) = A \cap B$ --- Theorem 4.2

==> $A \cap B$ is regular. -- def. of RL.

(2) Let M = (Q,S,d,s,F) be the machine s.t. L(M) = A.

Define M' = (Q,S,d,s,F') where F' = ~F = $\{q \in Q \mid q \notin F\}$.

Now for all x in S*, $x \in L(M')$

<=> $D(s,x) \in F' = {\sim}F$ --- def. of acceptance

<=> $D(s,x) \notin F$ --- def of ~F

<=> $x \notin L(M)$ iff $x \notin A$. -- def. of acceptance

Hence ~A is accepted by L(M') and is regular !

(3). Note that AUB = $\sim(\sim A \cap \sim B)$. Hence the fact that A and B are regular implies ~A, ~B, $(\sim A \cap \sim B)$ and $\sim(\sim A \cap \sim B)$ = AUB are regular too.