

# Today - Finite State Automata

3/28/07 ①

## Definitions

if  $A = \{1, 2\}$ ,  $B = \{x, y, z\}$

cartesian product (cross product)  
 $A \times B = \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\}$

$$|A \times B| = |A| \cdot |B|$$

Power Set:  $2^A$  the set of all the subsets of  $A$

$$A = \{x, y, z\}$$

$$2^A = \{\emptyset, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}$$

$$|2^A| = 2^{|A|} \Rightarrow 2^3 = 8$$

also written  $\mathcal{P}(A)$

Alphabet  $\Sigma$ : set of symbols we will use to put strings together

$$\Sigma_1 = \{0, 1\}$$

$$\Sigma_2 = \{a, b, c, \dots, y, z\}$$

A string over an alphabet  $\Sigma$  is a finite sequence of symbols from that alphabet. eg.  $w = 010110111101$  is a string over  $\Sigma = \{0, 1\}$

string concatenation:  $x = abc$      $xy = abcdef$   
 $y = def$

Language = set of strings made up of symbols from alphabet  $\Sigma$ .

Complement of Language  $L \Rightarrow \bar{L} = \Sigma^* - L$

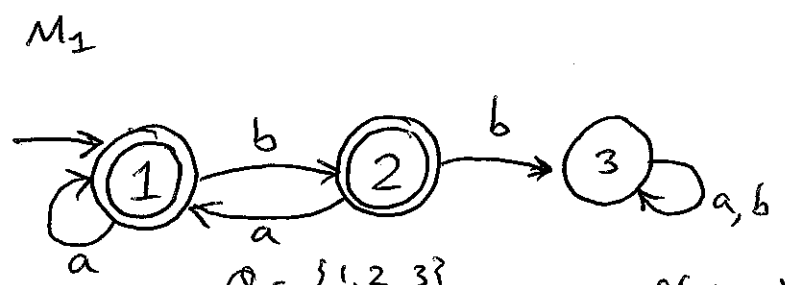
concatenation of languages  $L, L_2 = \{x, x_2 \mid x \in L, \text{ and } x_2 \in L_2\}$

# Finite State Automaton

# DFA - Deterministic Finite-state Automaton

1.  $Q$ : finite ~~set~~ set of states
2.  $\Sigma$ : finite set of symbols called alphabet
3.  $\delta: Q \times \Sigma \rightarrow Q$  transition functions
4.  $q_0 \in Q$  is the start state
5.  $F \subseteq Q$  set of accept states

EX:



$Q = \{1, 2, 3\}$   
 $q_0 = 1$   
 $\Sigma = \{a, b\}$   
 $F = \{1, 2\}$

$\delta(1, a) = 1$   
 $\delta(1, b) = 2$   
 $\delta(2, a) = 1$   
 $\delta(2, b) = 3$   
 $\delta(3, x) = 3$

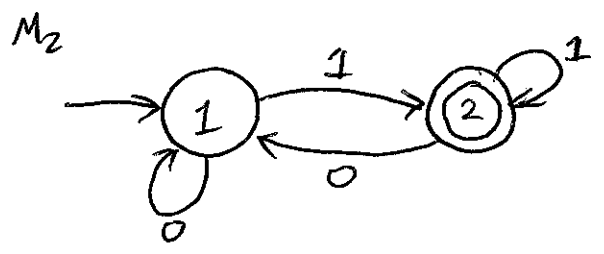
? 0 or more  
 \* 0 or more  
 + 1 or more  
 regular expressions

$L(M_1) = \{w \mid w \text{ does not have 2 b's in a row}\}$

A language is a "regular language" iff there <sup>exists</sup> a DFA that ~~recognizes~~ <sup>recognizes</sup> it

M decides L iff it returns "accept" iff  $w \in L$  and returns "reject" iff  $w \notin L$ .

EX:



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

1. Is the language  $L$ , that contains strings with odd #'s of b's regular?

$L = \{b, ab, abbb, \dots\}$

2. Language  $n$  a's followed by  $k$  b's?

Is the complement of  $\bar{L}$  regular?

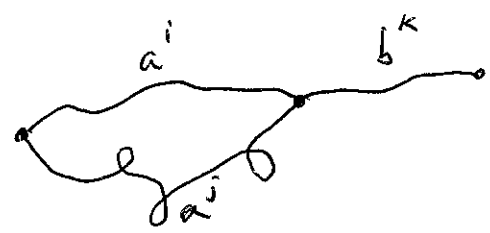
$\bar{L}^c = \{ \text{set of strings } w, w \notin L \}$

Yes! Just switch accepting, rejecting states

So regular languages are closed under complement.

What about  $L = \{ a^n b^n \mid n \geq 0 \}$ ? No.

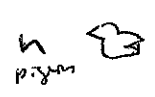
Suppose that reading  $a^i$  and  $a^j$  puts you in the same state:



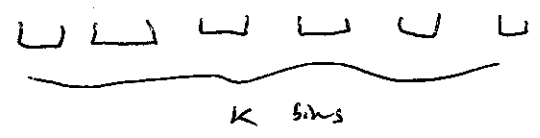
$\left. \begin{matrix} a^i b^k \\ a^j b^k \end{matrix} \right\}$  these two will give you the same

$\therefore$  each  $n$  requires it's own state  
unique  
by the pigeon hole principle, we need an  $\infty$   
number of states because  $n$  can be arbitrarily large.

Pigeon Hole Principle



if  $n > k$  then some holes will have  $\geq 2$  pigeons!

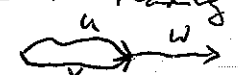


Equivalent

let  $u, v \in \{a, b\}^*$   
we say  $u \sim v$   $\swarrow$  equivalent

$\forall w, uw \in L \text{ iff } vw \in L$

if  $M$  recognizes  $L$ , ~~and~~ if reading  $u, v$  puts  $M$  in the same state then  $u \sim v$



if there are  $k$  equivalence classes,  
then you need at least  $k$  states.

Therefore, if you have  $\infty$  # of equivalence classes, then  
it isn't regular! Also show that  $a^n b^n$  is not  
regular!

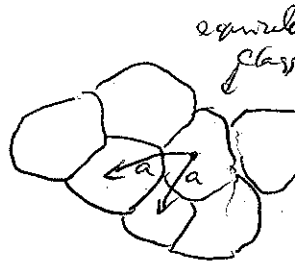
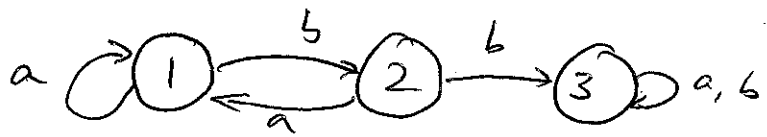
Let's look at example from earlier:

$$L = \{w \mid w \text{ does not contain 2 b's in a row}\}$$

what are the equivalence classes?

- 1) strings that don't end in a b (and do not have bb in them)
- 2) strings that end in a b (and ~~do~~ don't have bb)
- 3) strings that have 2b's in a row.

map to the 3 states of the machine



Lemma: Let  $u, v \in \Sigma^*$ , let  $a \in \Sigma$   
if  $u \sim v$ , then  $ua \sim va$

↓  
⇒ minimal machine has 1 state for each equivalence class

Myhill-Nerode Theorem

(DFA)

every regular language has a minimal finite state automata associated with it  
which has the same number of states as # of equivalence classes. Equivalent  
DFA could have the same structure. also called the index of the language

# NFA

non-deterministic Finite-state Automata is a DFA except that it can have multiple transitions with the same symbol.

$$\delta(q_i, a) = \text{list of possible new states}$$

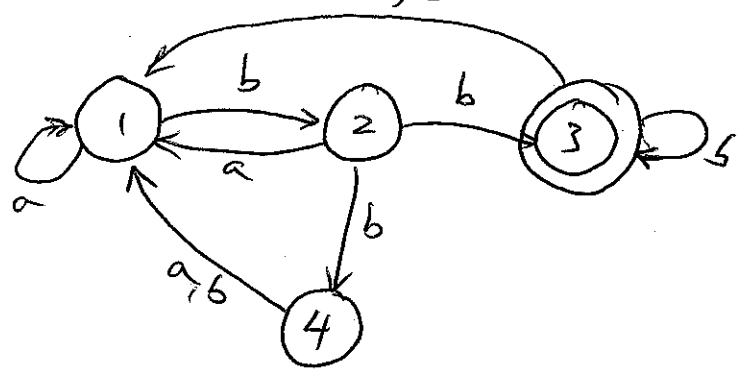
$$\delta: Q \times \Sigma \rightarrow Q \quad (\text{DFA})$$

$$\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q) \quad (\text{NFA})$$

power set

$2^Q$  ← all the possible subsets of Q.

An NFA accepts a string if any of the transition choices leads it to an accepting state.  $\exists$  an accepting state



## NFA

1.  $Q$  finite set of states
2.  $\Sigma$  is a finite alphabet
3.  $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$  transition function
4.  $q_0 \in Q$  start state
5.  $F \subseteq Q$  set of accept states

DFA vs. NFA  
which one is <sup>more</sup> powerful?