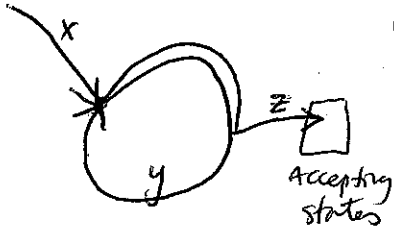


# Pumping Lemma

8/30/07 (1)

if  $|w| > \#$  of states you need to loop somewhere in the DFA



$$w = xyz \in L$$

$$xyyz \in L$$

$$xyyyz \in L$$

etc.

constant  $k$  s.t.

Proof

regular language  $\Rightarrow$  DFA

DFA  $\Rightarrow$  finite # states

at some  $k > \#$  states we have a repetition loop

if  $L$  is a regular language there exists a constant  $k$  s.t.  
if  $|w| > k$  (number of states) and  $w \in L$  then  $w$  can be written

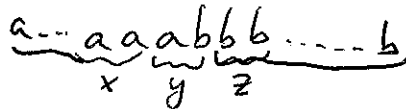
$$w = xyz \text{ st. } \forall j \geq 0, xy^jz \in L$$

$$|y| \geq 1, |x| < k$$

$$|y| < k$$

$$|xz| < k$$

Show that  $L = \{a^n b^n \mid n > 0\}$  is not regular using the pumping lemma



You can't use it to show a language is regular

Reg  $\Rightarrow$  Pumping

but Pumping  $\not\Rightarrow$  Regular

But you can use it to show something is not regular!

## Back to NFA's

Defn:

1.  $Q$  set of states
2.  $\Sigma, \epsilon$  alphabet + "empty" character
3.  $\delta: Q \times \Sigma \cup \{\epsilon\} \rightarrow \mathcal{P}(Q)$  transition function
4.  $q_0 \in Q$  start state
5.  $F \subseteq Q$  accept states

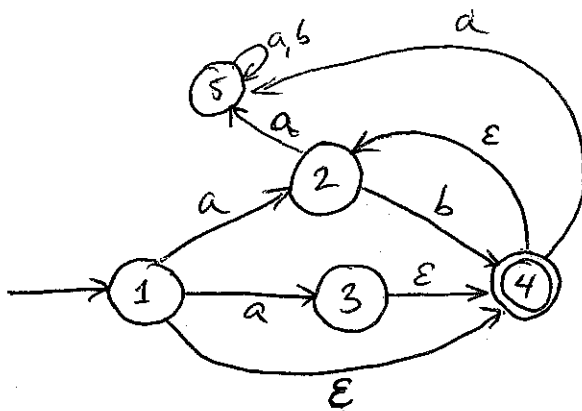
We "accept" the string when ~~any~~ it terminates in any accept state. Just 1 needs to be accept for the string to be accepted.

# Converting an NFA to a DFA

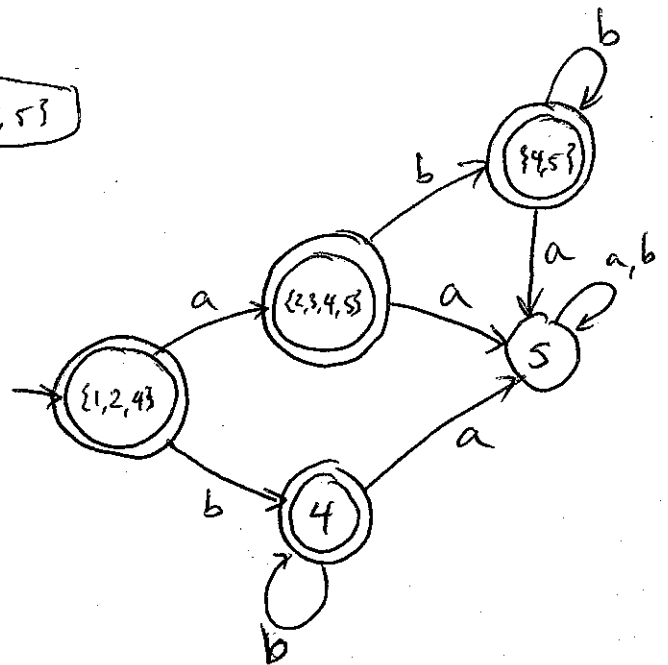
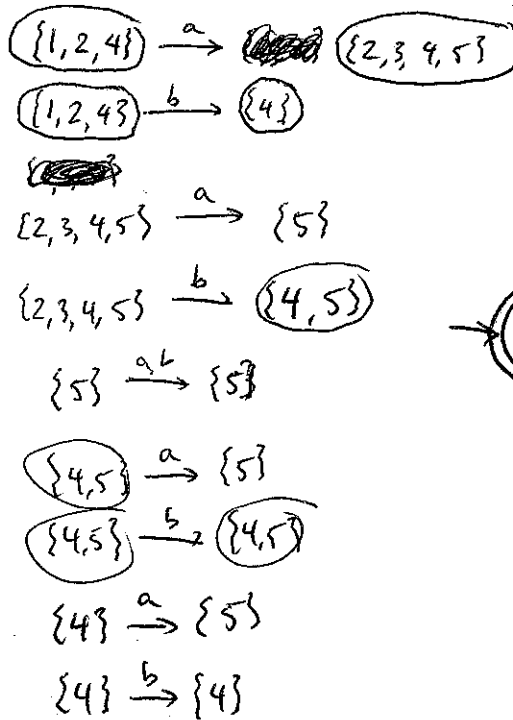
Can always be done using "powerset construction"

- Observations:
- 1) At any point in time we will be "in" a subset of states of  $Q$ .
  - 2) As we transition, we are going to go from this subset to another subset, in a deterministic manner.
  - 3) There are a finite number of subsets of  $Q$ :  $(2^{|Q|})$
  - 4) Therefore, it is possible to construct a DFA where each of the subsets is its own state and draw transitions in between them!

EX:



$(a/\epsilon)b^*$



# DFA = NFA

$Q' = \mathcal{P}(Q)$	$Q$ states	} — Note the exponential increase of states! $ Q  \rightarrow 2^{ Q }$
$F' = \{S \subseteq Q' \mid S \cap F \neq \emptyset\}$	$F \subseteq Q$ accepting states	
$\Sigma$	$\Sigma$ alphabet	
$q_0' = \{q_0\}$	$q_0 \in Q$ initial state	
$\delta' : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$	$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$	
$Q' \times \Sigma \rightarrow Q'$		

Let  $DFA(k)$  be the set of regular languages that can be recognized by a DFA with  $k$  states.

$$DFA(k) \neq NFA(k)$$

$$\bigcup_{k \geq 0} DFA(k) = \bigcup_{k \geq 0} NFA(k)$$

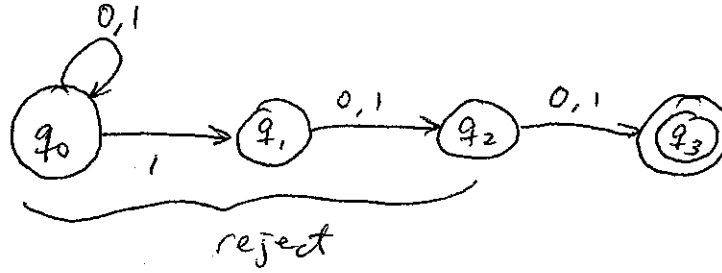
↓

$$DFA = NFA$$

Is there a family of languages  $L_k$  such that  $L_k$  can be recognized by an NFA with  $k$  states but the minimal DFA has  $\Omega(2^k)$  states?

Ex: let  $L$  be the set of strings with  $\Sigma = \{0,1\}$  s.t. the third to the last symbol is a "1".

NFA



How many states does the equivalent DFA need?

→ How many equivalence classes does the DFA have?

1	~~~~~	100	} accept
2	~~~~~	101	
3	~~~~~	110	
4	~~~~~	111	

5	~~~~~	000	} Fail
6	~~~~~	001	
7	~~~~~	010	
8	~~~~~	011	

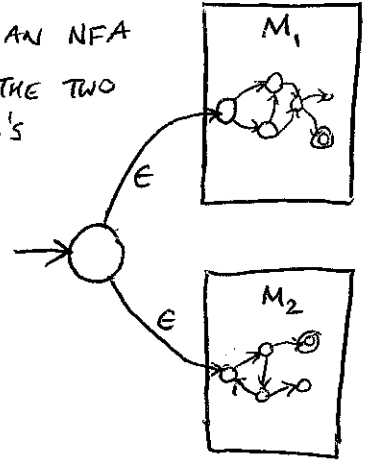
8 states required

Are regular languages closed under union? YES!

$L = \{L_1 \cup L_2\}$  eg.  $w \in L$  iff  $w \in L_1$  or  $w \in L_2$ ,  $L_1$  and  $L_2$  are regular

Assume that there is a DFA  $M_1$  that recognizes  $L_1$   
there is a DFA  $M_2$  that recognizes  $L_2$

CREATE AN NFA FROM THE TWO DFA'S



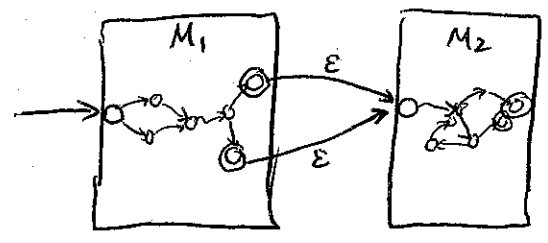
This NFA can be converted into a DFA that recognizes  $L$ .

Hence regular languages are closed under union.

Are regular languages closed under concatenation? YES!

$L = \{L_1 L_2\}$  eg.  $w = uv \in L$  if  $u \in L_1$  and  $v \in L_2$

again  $M_1$  recognizes  $L_1$   
 $M_2$  recognizes  $L_2$

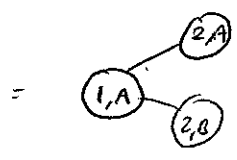
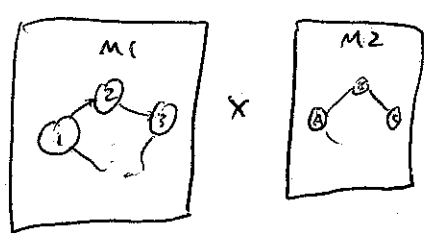


CREATE an NFA that is made up of the two DFA's chained together

Regular languages closed under intersection? YES!

$L = \{L_1 \cap L_2\}$

create a new machine  $M_3$  with states  $Q = Q_1 \times Q_2$  ← pairs of all the



$F = F_1 \times F_2$  ← both states would be "accept"

Another way of showing this:  $A \cap B = \overline{\overline{A} \cup \overline{B}}$  ← DeMorgan's laws  
reg. lang. closed under complement  
reg. lang. closed under union

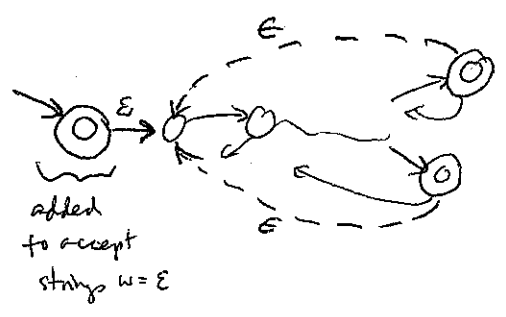
Ex: Show that  $L = \{w \mid \#_a(w) = \#_b(w)\}$  is not regular

$$L \cap \underbrace{a^*b^*}_{\text{regular}} = \underbrace{\{a^n b^n\}}_{\text{not regular}}$$

$\therefore L$  must not be regular!  
if it were, then  $a^n b^n$  would be regular.

Regular languages closed under  $*$ ?  $L^*$ ? YES!

$\epsilon$   
 $L$   
 $LL$   
 $LLL$   
Build a machine ~~DFA~~ NFA such that all of the accept states of the DFA  $M$ .  
So back to the beginning on  $\epsilon$



Regular expressions

Any language that can be expressed with strings is regular.

concatenation  
 $\downarrow$   
 $\circ, \cup, *$  starting with finite

ex:  $(a \cup bcb^*)^* \cup (cc^*a)^*$

proof by induction on the smaller regular expressions

$$\text{reg expressions} \subseteq \text{DFA} = \text{NFA}$$

# Generalized Non deterministic Finite Automata (GFNA)

(7)

Def.

1.  $Q$  finite set of states

2.  $\Sigma$  input alphabet

~~3.  $Q \subseteq \Sigma$~~

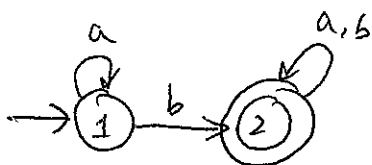
3.  $q_0$  start state

4.  $q_{accept}$  accept state

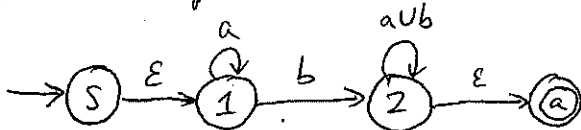
5.  $\delta: (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow R$

collection of all regular expressions over alphabet  $\Sigma$

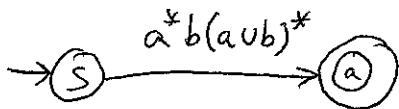
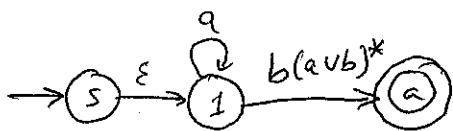
Ex:



step 1: add explicit start and end state



step 2: collapse states



ex:

aaaba

abbbbbaabbbabab

reg. exp = DFA = NFA  $\subset$  P