

# BBM402-Lecture 3: Nondeterminism: NFA definitions, e-transitions, equivalence with DFAs

Lecturer: Lale Özkahya

Resources for the presentation:

<http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-045j-automata-computability-and-complexity-spring-2011/Syllabus/>

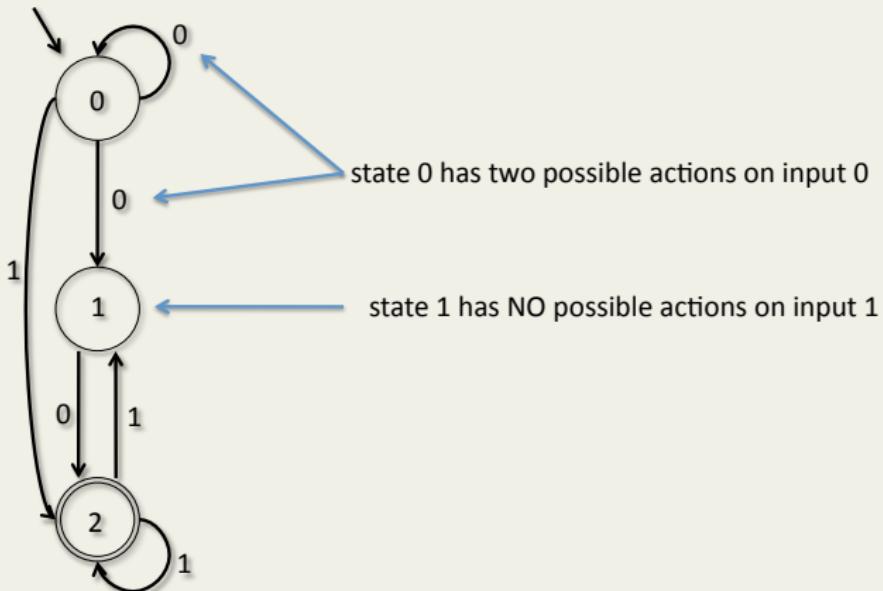
<https://courses.engr.illinois.edu/cs498374/lectures.html>

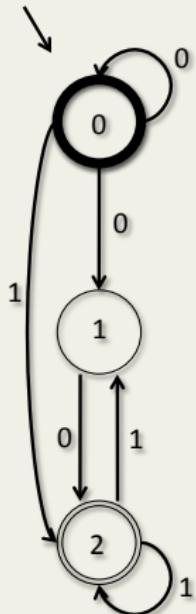
## DFA Reminder

A DFA is a quintuple  $M=(Q,\Sigma,\delta,q_0,F)$ , where:

- $Q$  is a finite set of *states*
- $\Sigma$  is a finite *alphabet* of symbols
- $\delta: Q \times \Sigma \rightarrow Q$  is a *transition function*
- $q_0$  is the *initial state*
- $F \subseteq Q$  is the set of *accepting states*

## NFA: NONDETERMINISTIC FINITE AUTOMATON

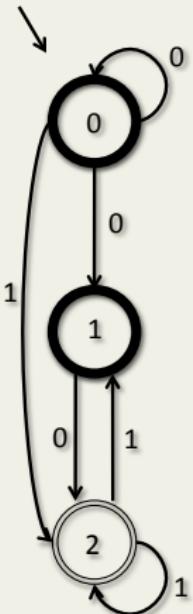




Let's see how a computation proceeds

What is the next state???

INPUT 000110



**Two views:**

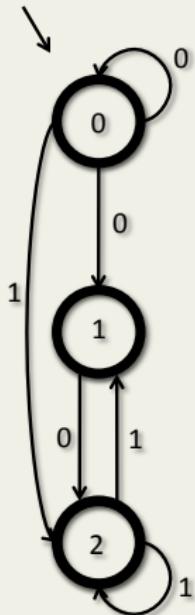
**(a) Possible worlds**

EITHER could be the next state.

**(b) Parallel threads**

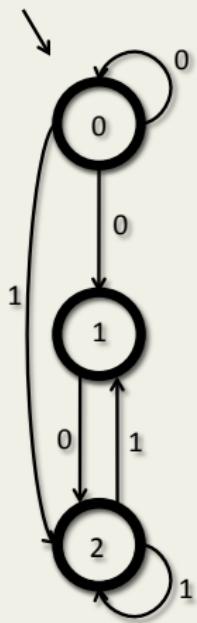
BOTH are “the next state”; the NFA spawns a second thread – it is in two states at the same time.

INPUT    0   0110

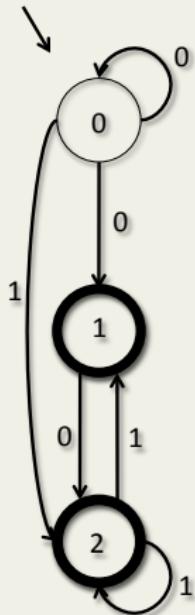


After reading 00, the machine could  
be in ANY of its states!

INPUT    00 $\boxed{0}$ 110

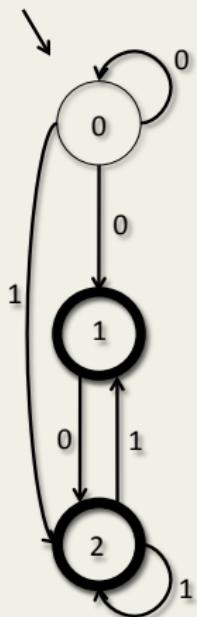


INPUT 000110

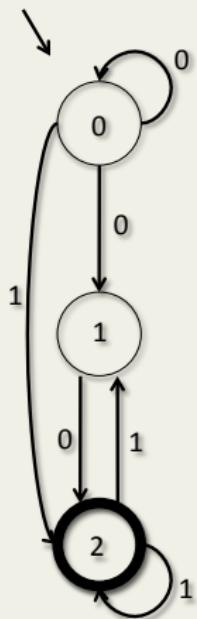


One of the threads has died

INPUT    000110



INPUT    000110



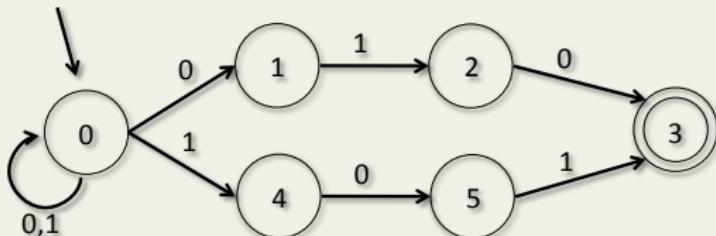
INPUT    000110

## Formalism

- NFA is same as DFA, except transition  $\delta$  returns a *set of possible next states*:
- $\delta: Q \times \Sigma \rightarrow 2^Q$  so that  $\delta(q,a) \subseteq Q$
- We write  $q \xrightarrow{a} p$  if  $p$  is in  $\delta(q,a)$
- Everything else is unchanged. But now for a string  $w$ , there may be many states  $p$  such that  $q \xrightarrow{w} p$ . (Exists a path labeled  $w$  leading from  $q$  to  $p$ )

$N$  accepts  $w$  if *for some*  $f$  in  $F$ ,  $q_0 \xrightarrow{w} f$ .

## *Example NFA N*



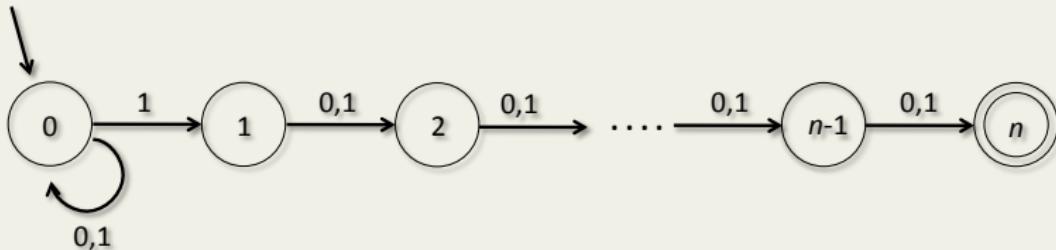
What strings can *possibly* end at state 3?

$$L(N) = \{w \mid w \text{ ends with "010" or with "101"}\}$$

must make sure:

- (1) *every* string with either ending *can* be accepted.
- (2) *every* string without either ending *cannot* possibly be accepted.

## *Example NFA $N$*



What strings can *possibly* end at state  $n$ ?

$$L(N) = \{w : w\text{'s } n^{\text{th}} \text{ from last character is a "1"}\}$$

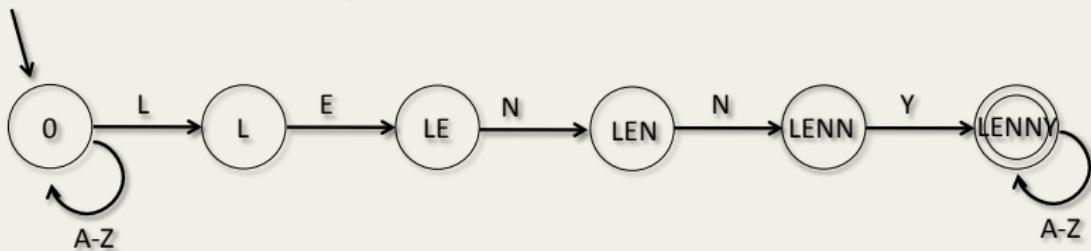
requires  $2^n$  DFA states

must make sure:

- (1) *every* string with 1 in  $n^{\text{th}}$ -from-last position *can* be accepted.
- (2) *every* string with a 0 in  $n^{\text{th}}$ -from-last position *cannot* possibly be accepted.

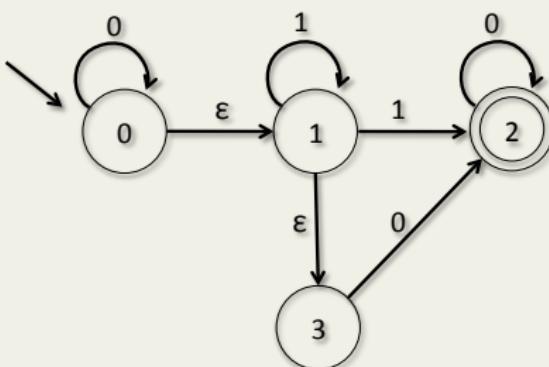
## *Challenge NFA Construction*

Create an NFA that recognizes the set of strings that contain your FIRST NAME.



## $\varepsilon$ -NFAs: NFAs with $\varepsilon$ -edges

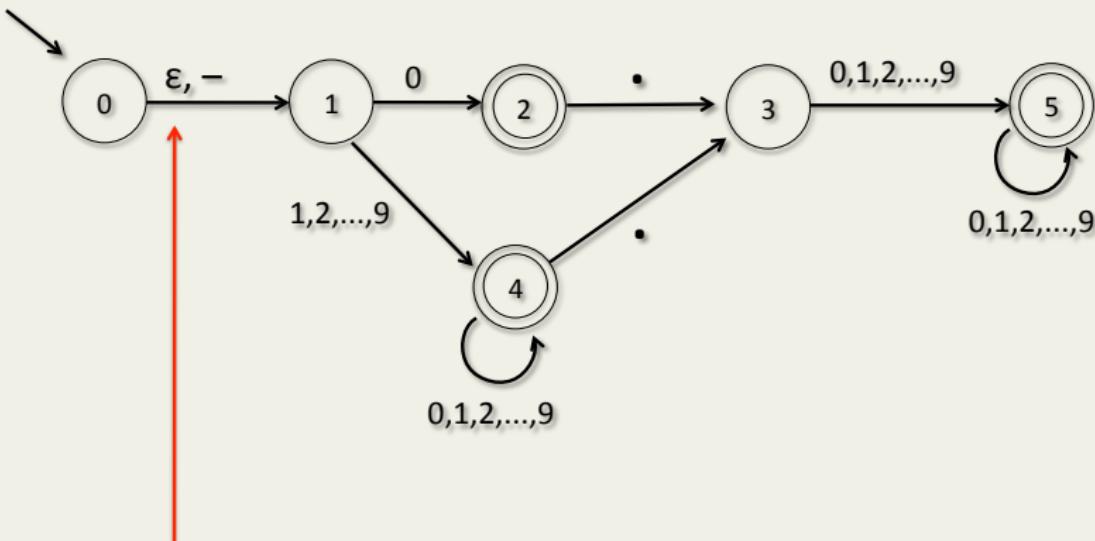
- Allow transition without reading character
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$



On empty input, which states can be reached?

On input 0, which states can be reached?

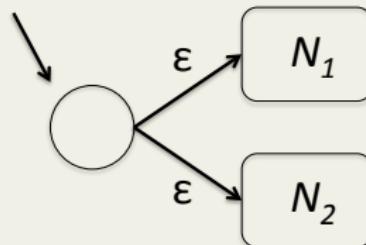
## *Accept decimal numbers*



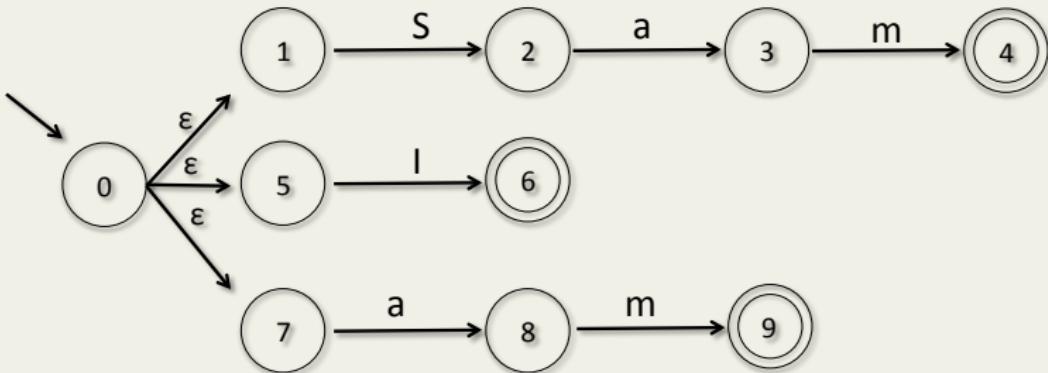
useful when an input symbol is optional

## *Utility of $\epsilon$ -edges*

- nondeterministically choose between two cases
- construct NFA for case 1, NFA for case 2, then add new start state with  $\epsilon$ -edges to choose which NFA to “run”.
- accepts the union of the languages

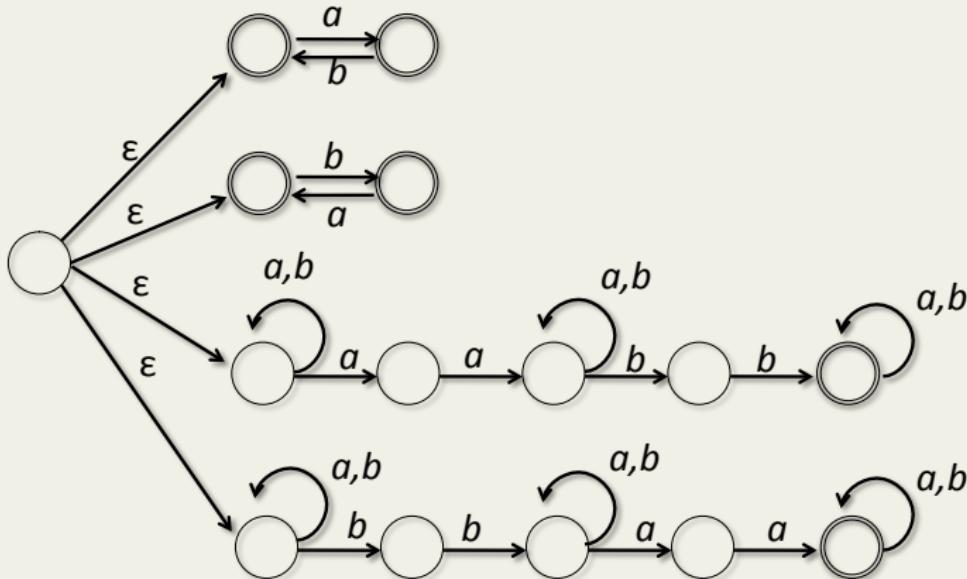


*Example: accept one of several keywords*

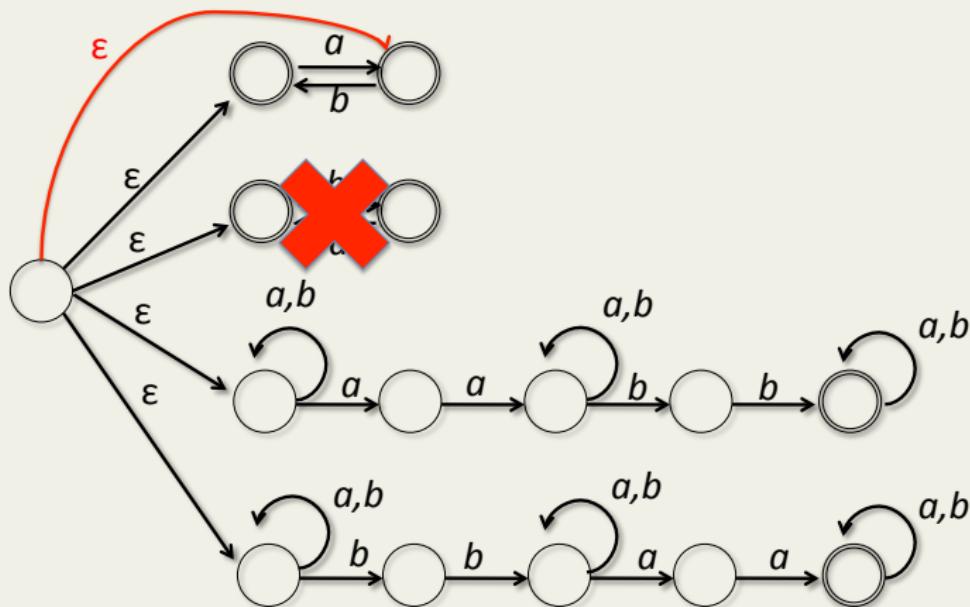


No need to think about how different keywords overlap.  
Can essentially have several starting states.

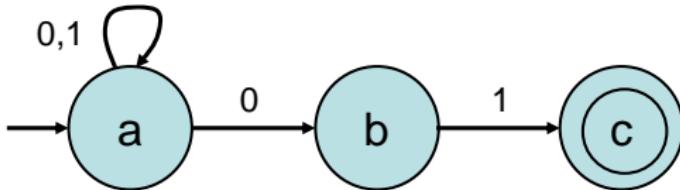
$L(N) = \{w \mid \text{either contains both } aa \text{ and } bb, \text{ or neither}\}$



$$L(N) = \{w \mid \text{contains either both } aa \text{ and } bb, \text{ or neither}\}$$



# NFA Example 1



$$Q = \{ a, b, c \}$$

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

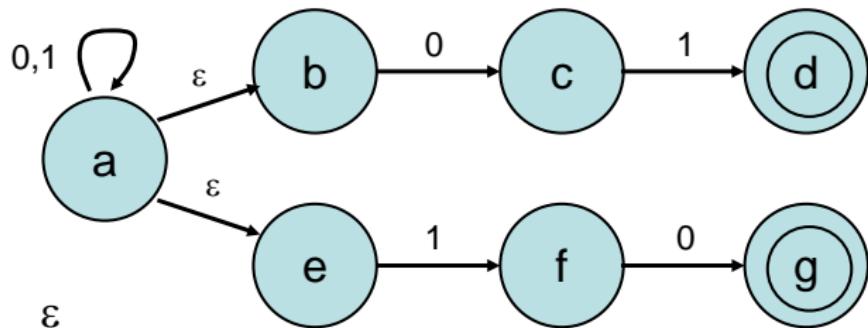
$$q_0 = a$$

$$F = \{ c \}$$

$\delta$ :

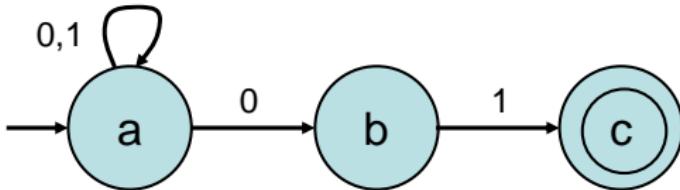
	0	1	$\epsilon$
a	{a,b}	{a}	$\emptyset$
b	$\emptyset$	{c}	$\emptyset$
c	$\emptyset$	$\emptyset$	$\emptyset$

# NFA Example 2



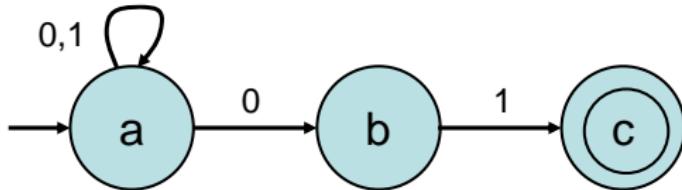
	0	1	$\epsilon$
a	{a}	{a}	{b,c}
b	{c}	$\emptyset$	$\emptyset$
c	$\emptyset$	{d}	$\emptyset$
d	$\emptyset$	$\emptyset$	$\emptyset$
e	$\emptyset$	{f}	$\emptyset$
f	{g}	$\emptyset$	$\emptyset$
g	$\emptyset$	$\emptyset$	$\emptyset$

# Example 1



- $L(M) = \{ w \mid w \text{ ends with } 01 \}$
- M accepts exactly the strings in this set.
- Computations for input word  $w = 101$ :
  - Input word  $w: 1 \ 0 \ 1$
  - States:      a    a    a    a
  - Or:            a    a    b    c
- Since c is an accepting state, M accepts 101

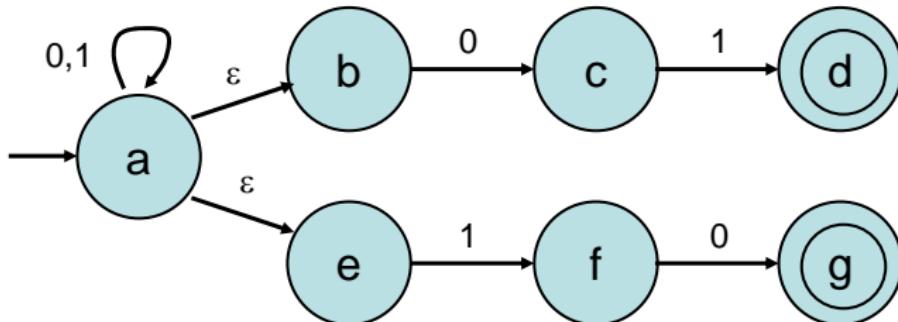
# Example 1



- Computations for input word  $w = 0010$ :
  - Possible states after 0: { a, b }
  - Then after another 0: { a, b }
  - After 1: { a, c }
  - After final 0: { a, b }
- Since neither a nor b is accepting, M does not accept 0010.

$$\{a\} \xrightarrow{0} \{a, b\} \xrightarrow{0} \{a, b\} \xrightarrow{1} \{a, c\} \xrightarrow{0} \{a, b\}$$

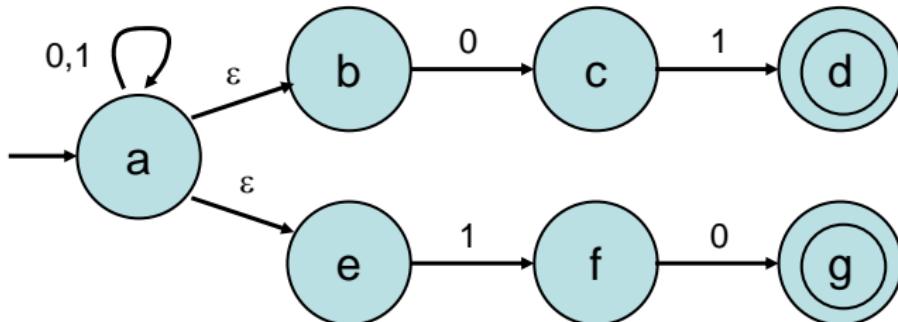
## Example 2



- $L(M) = \{ w \mid w \text{ ends with } 01 \text{ or } 10 \}$
- Computations for  $w = 0010$ :
  - Possible states after no input:  $\{ a, b, e \}$
  - After 0:  $\{ a, b, e, c \}$
  - After 0:  $\{ a, b, e, c \}$
  - After 1:  $\{ a, b, e, d, f \}$
  - After 0:  $\{ a, b, e, c, g \}$
- Since g is accepting, M accepts 0010.

$\{ a, b, e \} \xrightarrow{0} \{ a, b, e, c \} \xrightarrow{0} \{ a, b, e, c \} \xrightarrow{1} \{ a, b, e, d, f \} \xrightarrow{0} \{ a, b, e, c, g \}$

## Example 2



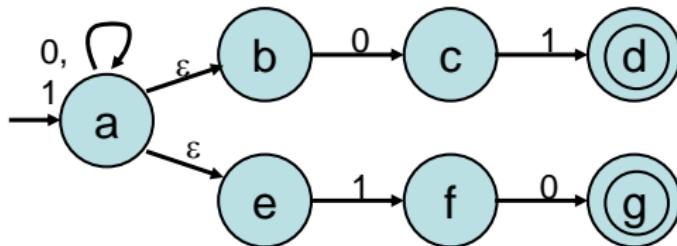
- Computations for  $w = 0010$ :

$$\begin{array}{cc} 0 & 0 \\ \{a, b, e\} \xrightarrow{\quad} & \{a, b, e, c\} \xrightarrow{\quad} \{a, b, e, c\} \\ 1 & 0 \\ \xrightarrow{\quad} & \end{array}$$
$$\{a, b, e, d, f\} \xrightarrow{\quad} \{a, b, e, c, g\}$$

- Path to accepting state:

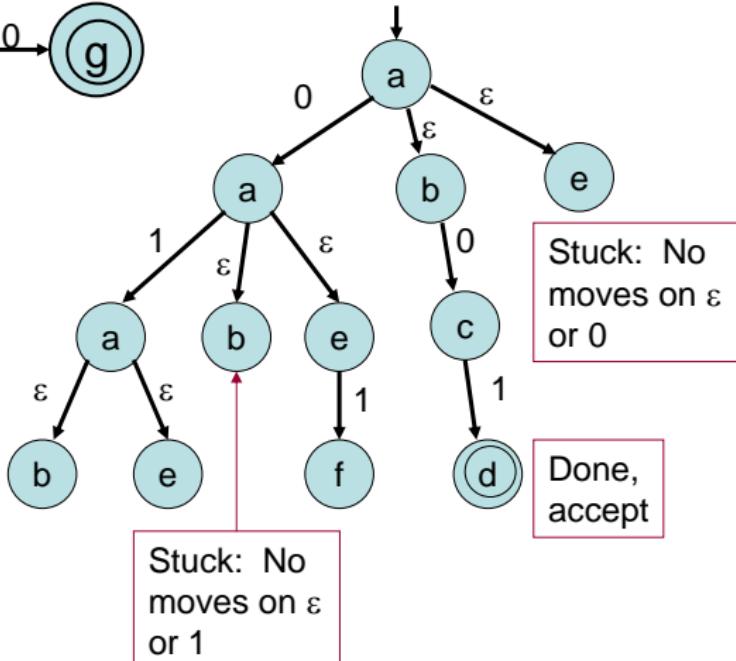
$$\begin{array}{ccccc} 0 & 0 & ε & 1 & 0 \\ a \xrightarrow{\quad} & a \xrightarrow{\quad} & a \xrightarrow{\quad} & e \xrightarrow{\quad} & f \xrightarrow{\quad} g \end{array}$$

# Viewing computations as a tree



Input  $w = 01$

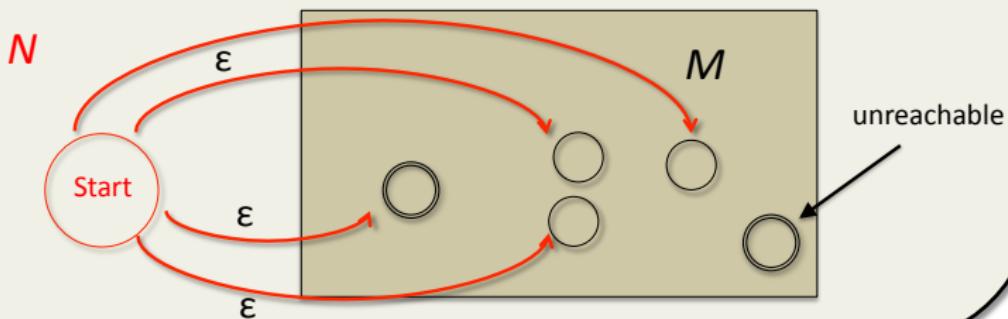
In general, accept if there is a path labeled by the entire input string, possibly interspersed with  $\epsilon$ s, leading to an accepting state.



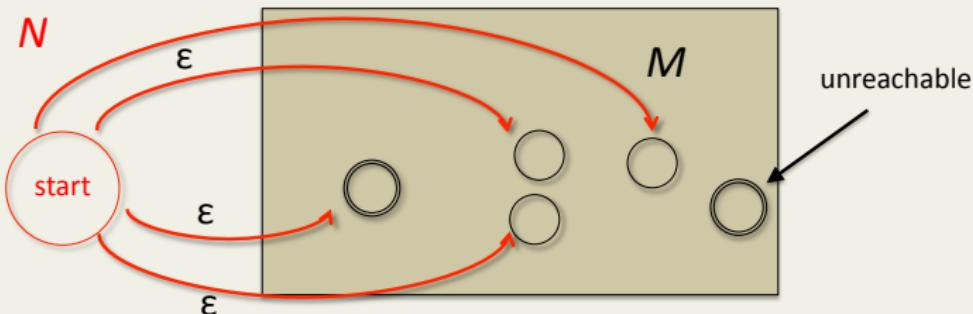
Here, leads to accepting state d.

## Closure under Suffixes

- Show that if  $L$  is accepted by some DFA  $M$ , then there is an NFA  $N$  that accepts  
 $\text{suffixes}(L) = \{w \mid w \text{ is a suffix of some word in } L\}$
- Proof by constructing  $N$  from  $M$ .



## Closure under Suffixes



- Let  $M = (Q, \Sigma, \delta, q_0, F)$ , then  $N = (Q^N, \Sigma^N, \delta^N, q_0^N, F^N)$ 
  - $Q^N = Q \cup \{\text{start}\}$
  - $\Sigma^N = \Sigma$
  - $\delta^N = \delta(q, a)$  for  $q$  in  $Q$ ;  $\delta^N(\text{start}, \epsilon) = \{q \mid q \text{ reachable}\}$
  - $q_0^N = \text{start}$
  - $F_N = F$

## *Informal Definition of acceptance*

$N$  an  $\epsilon$ -NFA accepts a string  $w=a_1a_2\dots a_n$  iff

there is a path (perhaps including  $\epsilon$ -edges) from state 0 to an accepting state, such that the concatenation of symbols along the path =  $a_1a_2\dots a_n$



→ single non- $\epsilon$  edge

→  $\geq 0 \epsilon$ -edges

*$\epsilon$ -edges do not increase power*

### *Theorem*

*If  $L$  is recognized by some NFA  $N$  with  $\epsilon$ -edges, then there is an equivalent NFA  $N'$  without  $\epsilon$ -edges that recognizes  $L$*

Proof is by SIMULATION of  $N$  by an  $N'$

# Construction

Given  $\varepsilon$ -NFA  $N = (Q, \Sigma, \delta, q_0, F)$

Construct NFA  $N' = (Q, \Sigma, \delta', q_0, F')$ :

$F' = F$  unless  $\textcircled{0} \xrightarrow{\varepsilon} \textcircled{\quad}$

in which case  $F' = F \cup \{q_0\}$

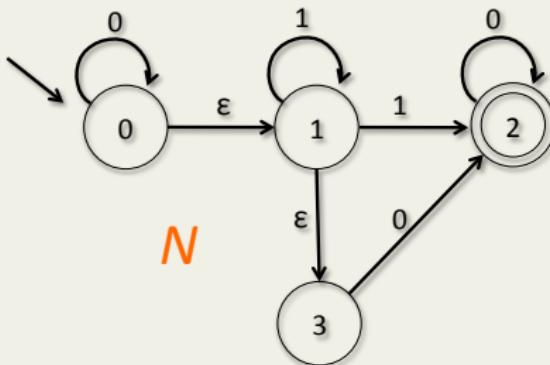
$\delta'$  extends  $\delta$  by adding transitions to compensate for lack of  $\varepsilon$ -edges.

$\delta'$  contains all non- $\varepsilon$ -edges of  $\delta$ , but in addition:  
for every  $p, q$  in  $Q$ , for every  $a$  in  $\Sigma$ , if there is a path



then add  $\textcircled{p} \xrightarrow{a} \textcircled{q}$  to  $N'$

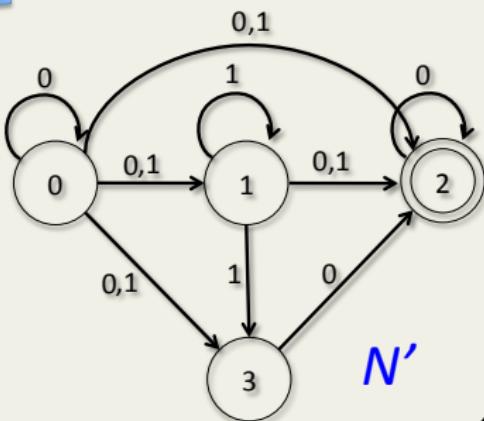
## *Example: eliminating $\epsilon$ -edges*



state 0 on input 0 can reach states 0,1,2,3  
state 0 on input 1 can reach states 1,2,3  
state 1 on input 0 can reach state 2  
state 1 on input 1 can reach states 1,2,3  
state 2 on input 0 can reach state 2  
state 2 on input 1 can reach nothing  
state 3 on input 0 can reach state 2  
state 3 on input 1 can reach nothing

$\epsilon$   
and NOT  $0 \xrightarrow{\epsilon} \bullet$  so  $F' = F$

copy over, omitting  $\epsilon$ -edges



## *Proof that simulation works*

- Prove that  $L(N') = L(N)$
- Use the following

### **Lemma**

For all states  $p, q$ , and for all NONEMPTY strings  $w$

$$p \xrightarrow{w} q \text{ in } N \text{ if and only if } p \xrightarrow{w} q \text{ in } N'$$



perhaps  $\epsilon$ -edges



no  $\epsilon$ -edges

DEF for M:

$$p \xrightarrow{a} q \text{ in } N' \text{ iff}$$

$$p \xrightarrow{a} q \text{ in } N$$

BY INDUCTION:

To show

$$p \xrightarrow{w} q \text{ in } N' \text{ iff}$$

$$p \xrightarrow{w} q \text{ in } N$$

$$\begin{array}{ccccc} p & \xrightarrow{a} & r & \xrightarrow{w} & q \\ & | & & & \\ & u & & & \end{array}$$

ENTIRE ARGUMENT  
WAS IFF

apply definition

$$p \xrightarrow{a} r$$

$$\begin{array}{ccc} & \searrow & \\ & & \end{array}$$

apply inductive hypothesis since  $|u| < |w|$

$$\begin{array}{c} \curvearrowright \\ r \xrightarrow{u} q \end{array}$$

$$r \xrightarrow{u} q$$

$$\begin{array}{ccc} & \swarrow & \\ & & \end{array}$$

$$\begin{array}{ccccc} p & \xrightarrow{a} & r & \xrightarrow{w} & q \\ & | & & & \\ & u & & & \end{array}$$

paste the  
computations back  
together

*Since lemma is true....*

Show for every  $w$ :  $N'$  accepts  $w$  if and only if  $N$  accepts  $w$

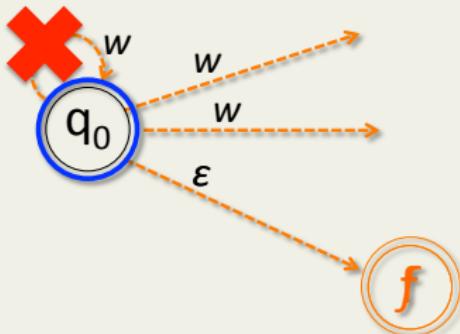
If NOT  $q_0 \xrightarrow{\epsilon} f$  in  $F$  then  $F' = F$  and:

- Neither  $N$  nor  $N'$  accept  $\epsilon$
- If  $|w| > 0$ . By Lemma,  $w$  goes to same states in  $N'$  as it did in  $N$ , and since  $F' = F$ , it is accepted by  $N'$  iff it was accepted by  $N$

*Since lemma is true....*

If  $q_0 \xrightarrow{\varepsilon} f$  in  $F$  then  $F' = F \cup \{q_0\}$

- Case 1:  $w = \varepsilon$ , so is accepted by both  $N$  and  $N'$
- Case 2:  $|w| > 0$ , then
  - Case 2a: NOT  $q_0 \xrightarrow{w} q_0$  in  $N$ .

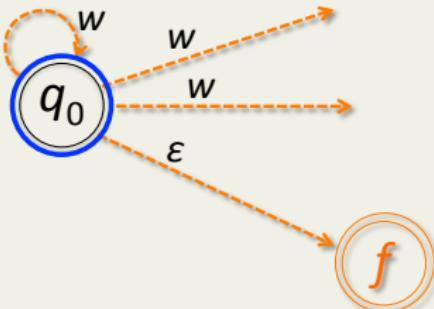


By Lemma,  $w$  doesn't reach  $q_0$  in  $N'$  either  
So adding  $q_0$  to  $F'$  didn't change  $w$ 's acceptance

*Since lemma is true....*

If  $q_0 \xrightarrow{\epsilon} f$  in  $F$  then  $F' = F \cup \{q_0\}$

- Case 1:  $w=\epsilon$ , so is accepted by both  $N$  and  $N'$
- Case 2:  $|w| > 0$ , then
  - Case 2b:  $q_0 \xrightarrow{w} q_0$  in  $N$ .



In  $N'$ ,  $q_0$  was made accepting  
But  $q_0 \xrightarrow{\epsilon} f$  in  $F$  anyway  
So  $w$  is accepted by both

*Eliminating Nondeterminism*

## *NFA → DFA Theorem*

### *Theorem*

*If  $L$  is recognized by some NFA  $N$ , then there is an equivalent DFA that recognizes  $L$  !!*

Nondeterminism doesn't increase the computational power of finite automata

Proof is by **SIMULATION** of an NFA by a DFA

assume **wlog** that NFA has no  $\varepsilon$ -edges

*But first....*

How would you write a program to tell if a word  $w$  was accepted by some DFA  $M$  ?

How is DFA represented?

DFA is table  $\delta$  (2d array)  $Q \times \Sigma$  with constant access time to get  $\delta(q,a)$

$F$  is boolean array indicating accept

state \ input	0	1
0	1	0
1	2	0
2	3	0
3	3	0

*But first....*

How would you write a program to tell if a word  $w$  was accepted by some DFA  $M$  ?

Alg M (*delta*: array;  $w$ : string)

state = 0

for  $i = 1$  to  $|w|$

    state = *delta*(state,  $w_i$ )

output  $F(state)$

state \ input	0	1
0	1	0
1	2	0
2	3	0
3	3	0

## *How about NFAs?*

How would you write a program to tell if a word  $w$  was accepted by some **NFA**  $N$ ?

- What is representation of an NFA?
- NFA has table  $\delta$  (2d array)  $Q \times \Sigma$  where each entry is a pointer to a linked list of possible next states  $\delta(q,a)$
- Time to collect “next states” from  $q, a$  is  $O(n)$

## *Algorithms for NFA computing*

ACCEPTS ( $N, w$ )

    Return ACCEPT?( $q_0, w$ )

ACCEPT? ( $q, w$ )    /\* does  $w$  go to an accepting state from  $q$  \*/

    if  $w = \epsilon$

        return  $F(q)$

    else  $w = au,$

        return OR { ACCEPT? ( $p, u$ ) |  $p$  in  $\delta(q, a)$  }

- Proof that this is correct?                         simple induction on  $|w|$
- Time taken by this algorithm?                        don't ask; don't tell (until later)

## *Algorithms for NFA computing*

ACCEPTS ( $N, w$ )

active, new\_active = sets of states implemented as  
boolean arrays indexed by states

active = [1,0,0,0,0,...,0] (initially, only starting state 0 is active)

for  $i = 1$  to  $|w|$

    new\_active =  $\emptyset$  (zero out the array)

    for each state  $q$  in active

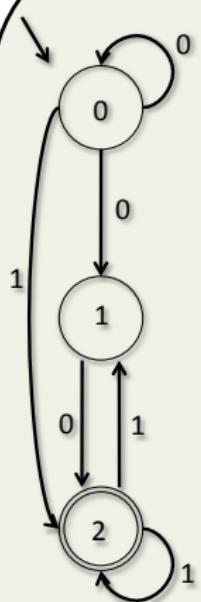
        put each element of  $\delta(q, w_i)$  in new\_active

    active = new\_active

if active contains an accepting state, then return TRUE

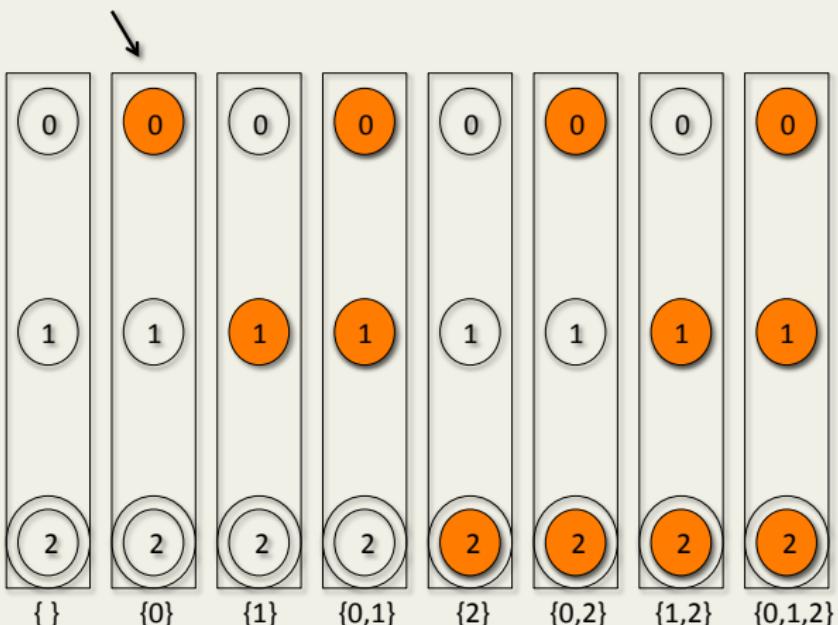
else return FALSE

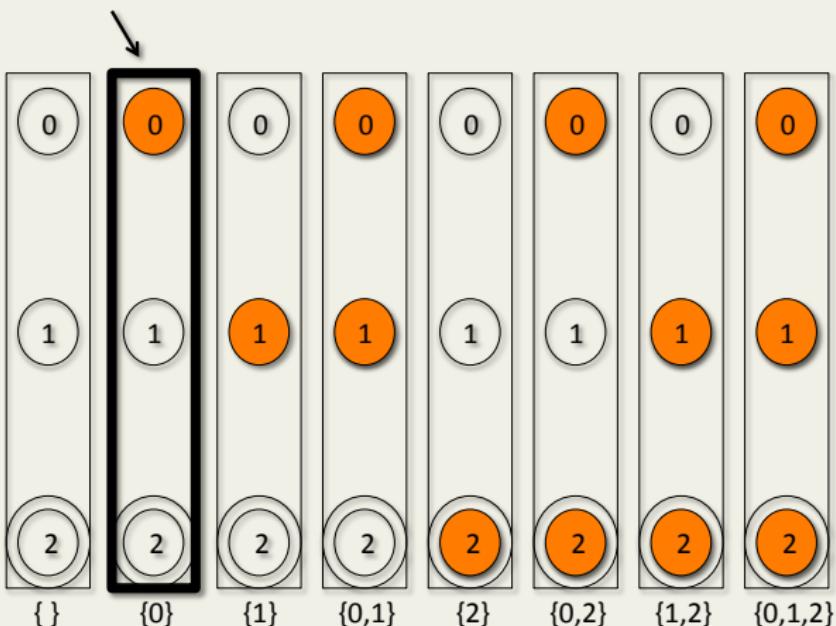
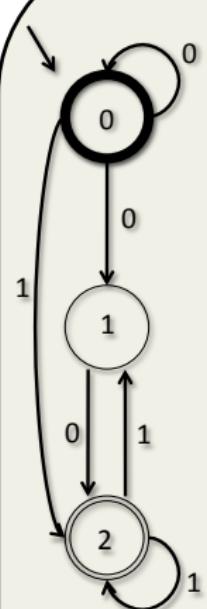
- Proof that this is correct? simple induction on  $|w|$
- Time taken by this algorithm?  $O(n^2w)$



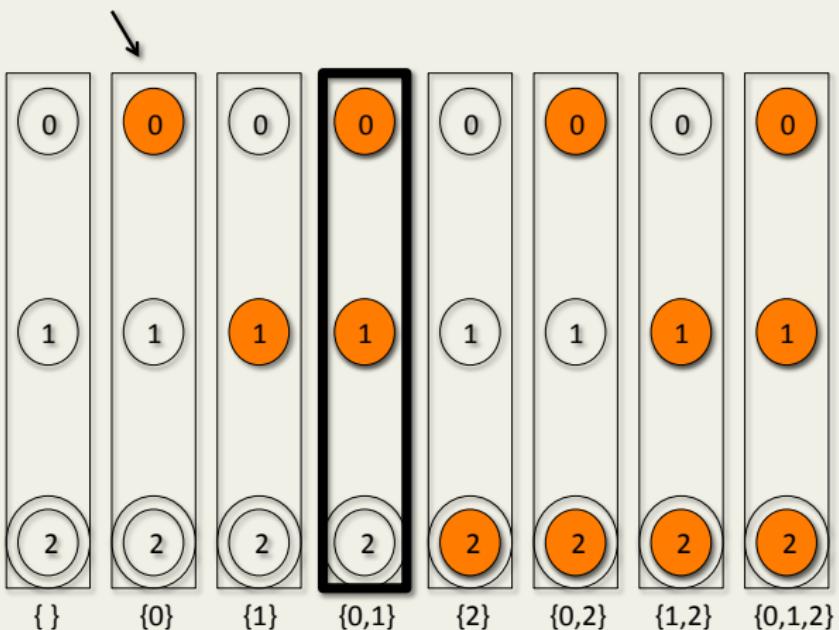
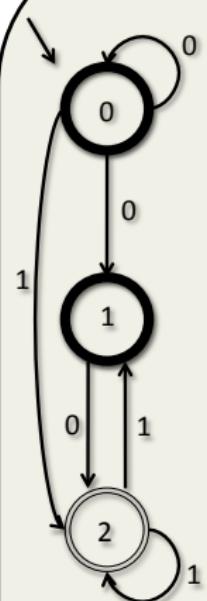
NFA

STATES OF THE EQUIVALENT “POWER-SET” DFA  
*A state corresponds to a subset of states of the NFA, showing which are active threads (a boolean array)*

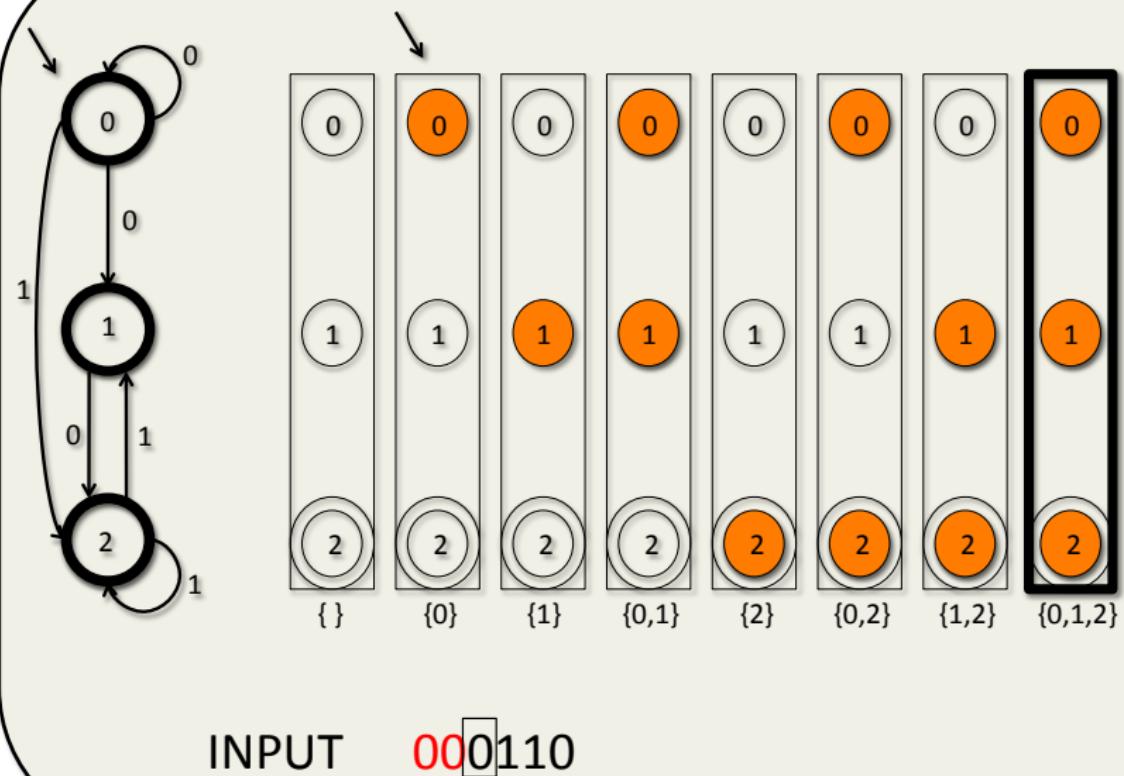


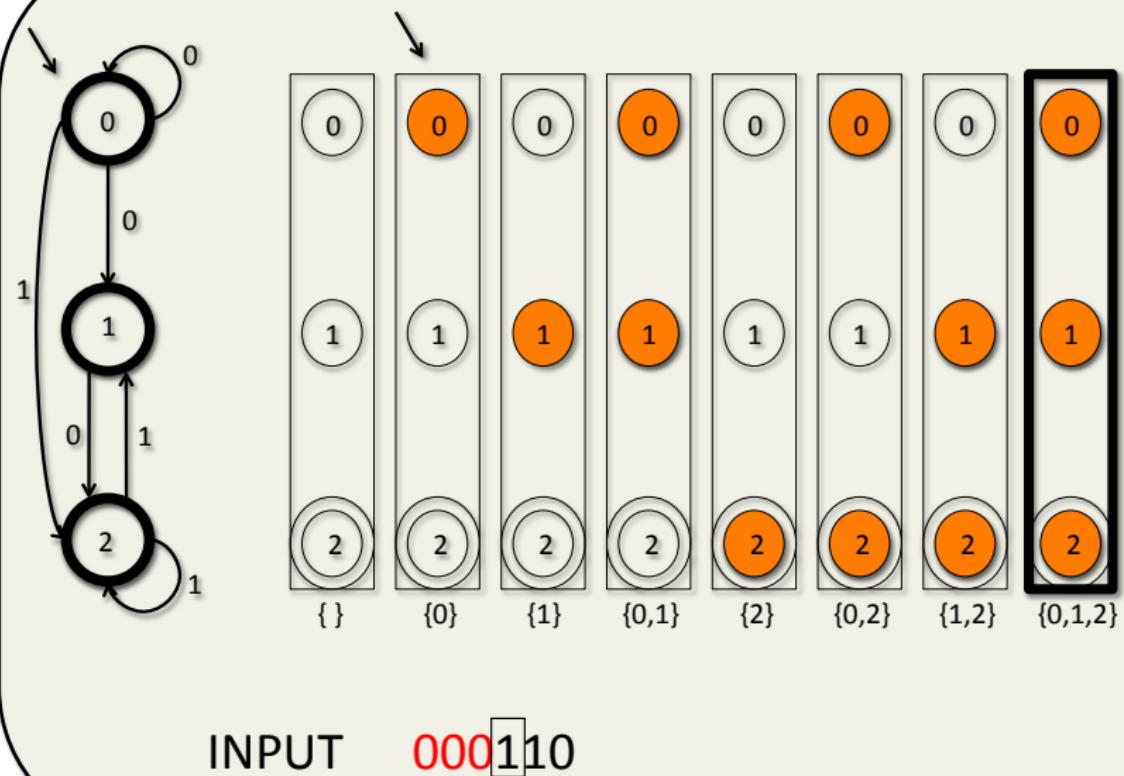


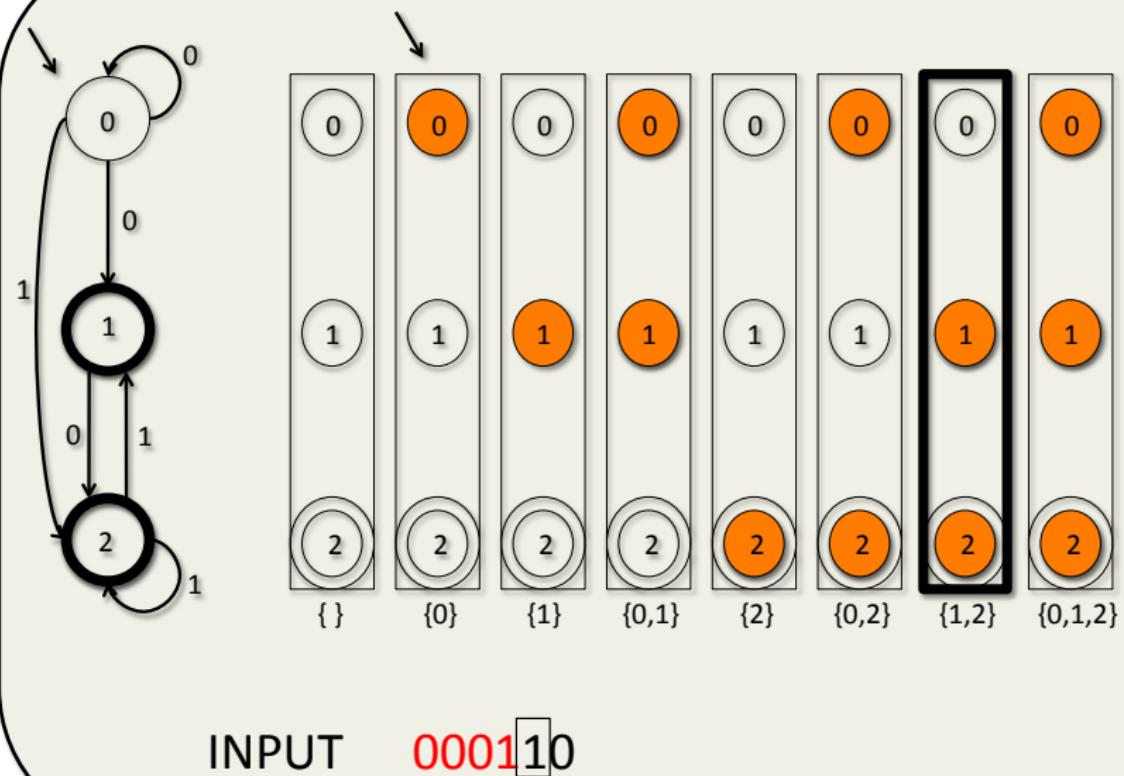
INPUT 000110

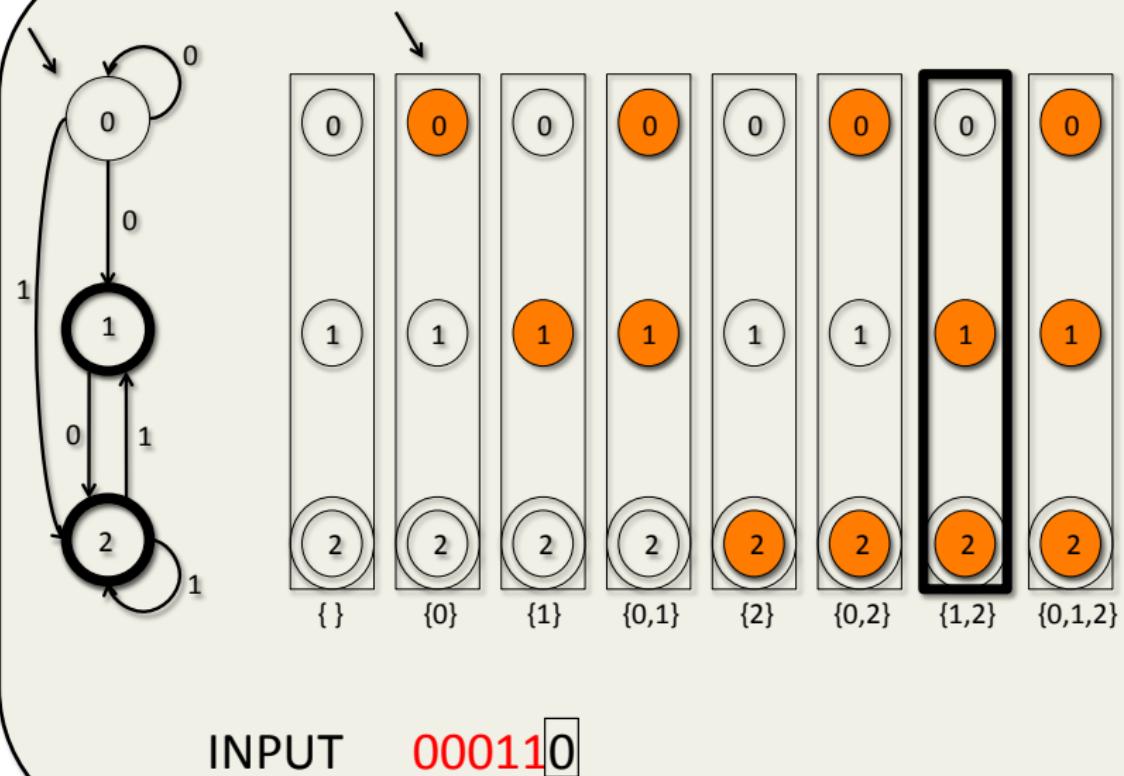


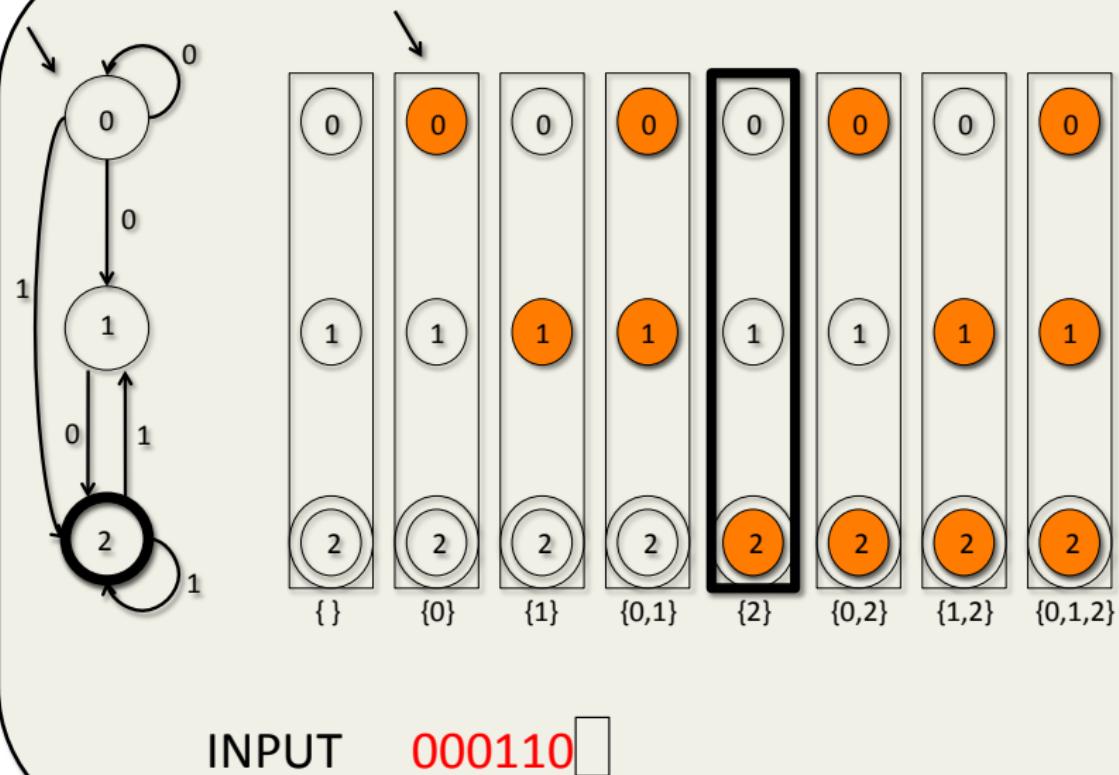
INPUT 000110











## *Formal specification*

- Let  $N = (Q, \Sigma, \delta, q_0, F)$  be an NFA
  - Recall that  $\delta: Q \times \Sigma \rightarrow 2^Q$ , so that  
 $\delta(q, a)$  is a set of possible states.
  - Build  $M = (Q', \Sigma, \delta', q'_0, F')$  that simulates  $N$ :
    - $Q' = 2^Q$  (the power set of  $Q$ )
    - $\Sigma$  is the same
    - $q'_0 = \{q_0\}$
    - $F' = \{S \subseteq Q : S \cap F \neq \emptyset\}$
- if  $N$  could have ended in an accepting state, then  $S$  will contain an element of  $F$
- since  $S$  is in  $Q'$ , it is a subset of  $Q$

## Formal specification

- Let  $N = (Q, \Sigma, \delta, q_0, F)$  be an NFA
- $M = (Q', \Sigma, \delta', q'_0, F')$       What is  $\delta'$  ??

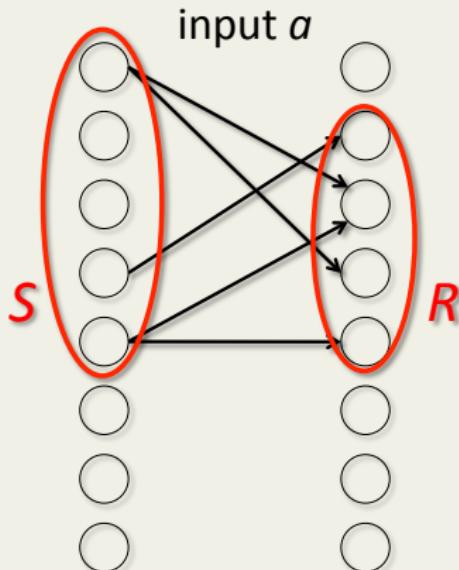
$\delta'(S, a) =$

for every state  $s$  in  $S$  all the states to which  $s$  can go on input  $a$

$S \xrightarrow{a} R$  iff  $R = \{ r : s \xrightarrow{a} r \text{ for some } s \text{ in } S \}$

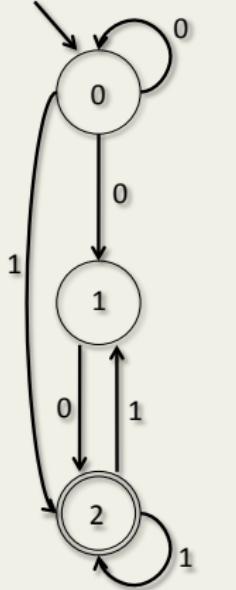
$S \xrightarrow{a}$  exactly the set of states that  $N$  can reach from anything in  $S$  on input  $a$

## *Intuition*

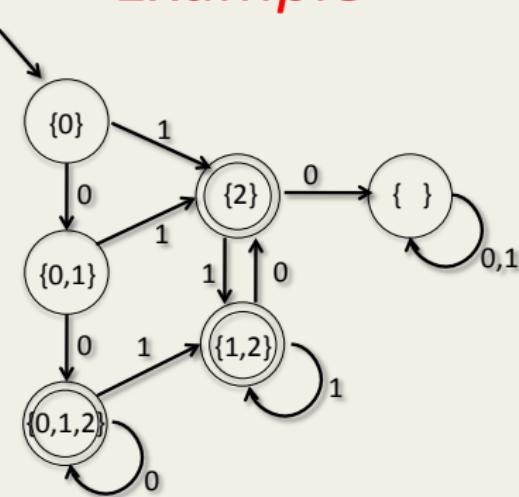


Then set  $S \xrightarrow{a} R$  in  $M$

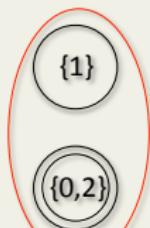
## *Example*



NFA



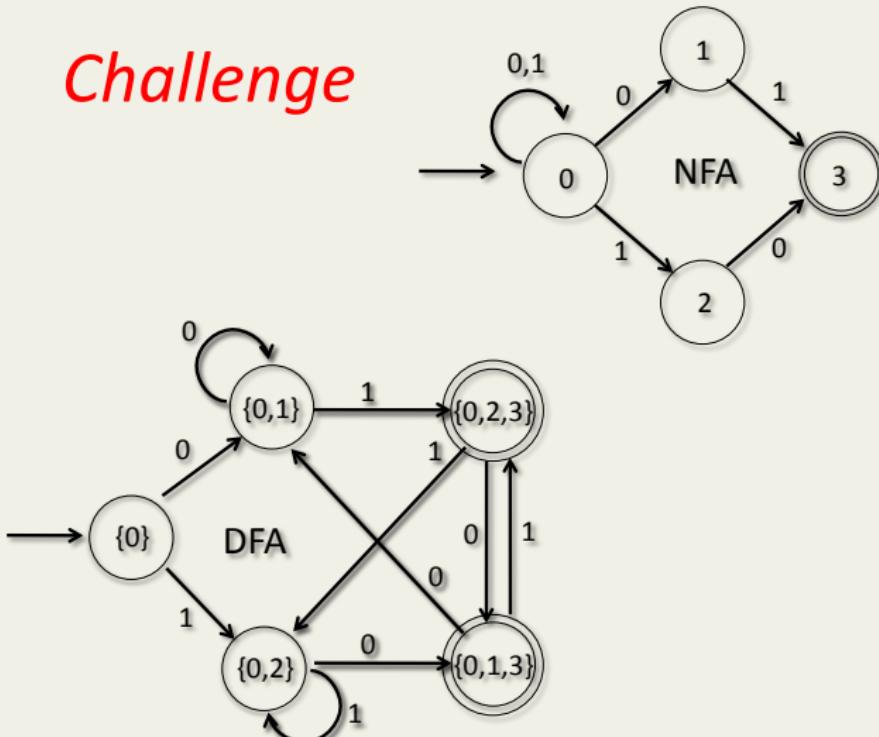
DFA



unreachable

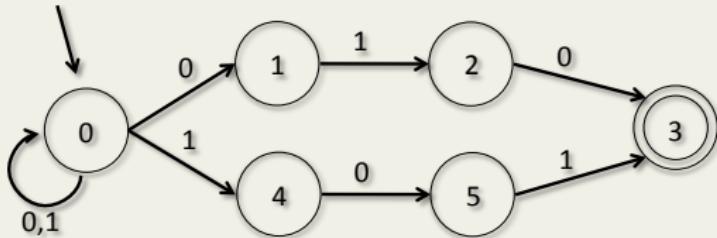
Which are the accepting states?

## *Challenge*



Which are the accepting states?

## *Challenge NFA* *(do at home)*



Be careful – it is easy to get confused

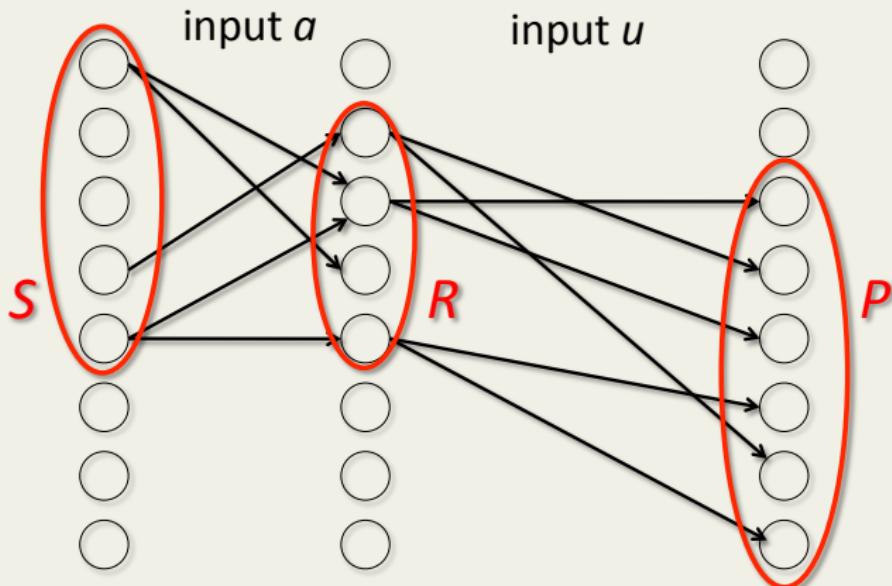
## To show

- “Action” of DFA is correct
- What would that mean?
- $\{q_0\} \xrightarrow{w} P = \{\text{states NFA could be in}\}$
- $= \{p : q_0 \xrightarrow{w} p\}$
- But more generally, for *any* state  $S$  of DFA  
(set of states of NFA)

$$S \xrightarrow{w} P = \{p : \text{for some } s \text{ in } S, s \xrightarrow{w} p\}$$

$S \xrightarrow{w}$  exactly the states that  $N$  could reach starting at a state in  $S$

## *Intuition ( $w = au$ )*



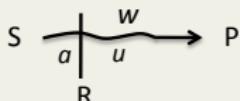
DEF for M:

$S \xrightarrow{a}$  exactly the states that  $N$  can reach from  $S$  on input  $a$

BY INDUCTION:

To show

$S \xrightarrow{w}$  exactly the states that  $N$  can reach from  $S$  on input  $w$



pull apart the computation

$$S \xrightarrow{a} R$$

apply definition

$$R \xrightarrow{u} P$$

apply inductive hypothesis since  $|u| < |w|$

$R$  = the reachable states from  $S$  on  $a$

$P$  = the reachable states from  $R$  on  $u$

$p$  in  $P$  iff  $p$  can be reached via  $u$  from some state  $r$  in  $R$ .

$r$  in  $R$  iff  $r$  can be reached via  $a$  from some state  $s$  in  $S$

paste the computations back together

SO,  $p$  in  $P$  iff  $p$  can be reached via  $au$  from some  $s$  in  $S$

## *Finishing up....*

$S \xrightarrow{w}$  exactly the set of states that  $N$  can reach from  $S$  on input  $w$

$\{q_0\} \xrightarrow{w}$  exactly the set of states that  $N$  can reach from  $q_0$  on input  $w$

$N$  accepts a string  $w$

iff in  $N$ ,  $q_0 \xrightarrow{w} f$  for some  $f$  in  $F$

iff in  $M$ ,  $\{q_0\} \xrightarrow{w} S$  for some  $S$  containing  $f$

iff in  $M$ ,  $\{q_0\} \xrightarrow{w} S$  in  $F'$       (recall  $F' = \{S \subseteq Q : S \cap F \neq \emptyset\}$ )

iff  $M$  accepts  $w$