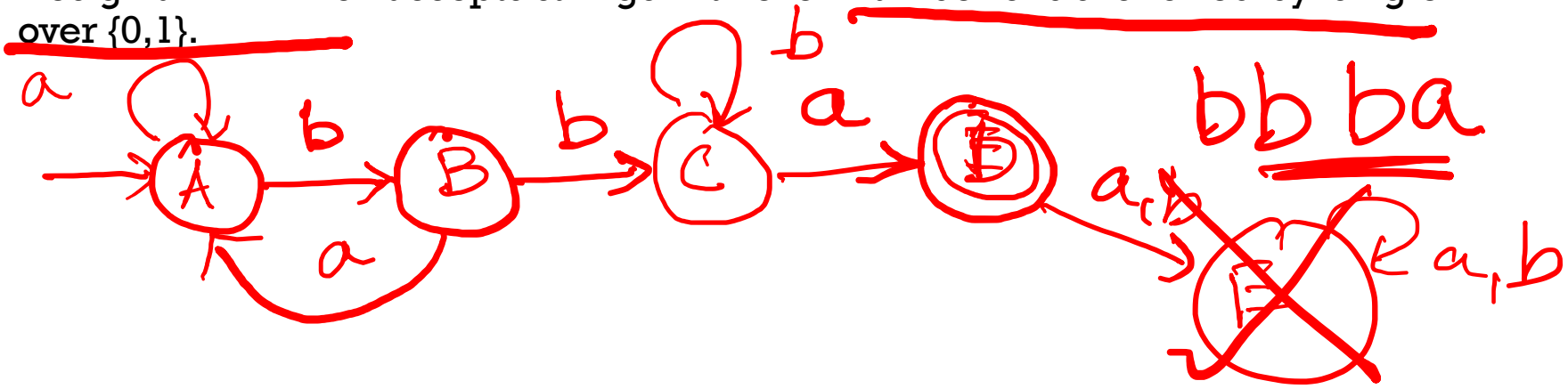
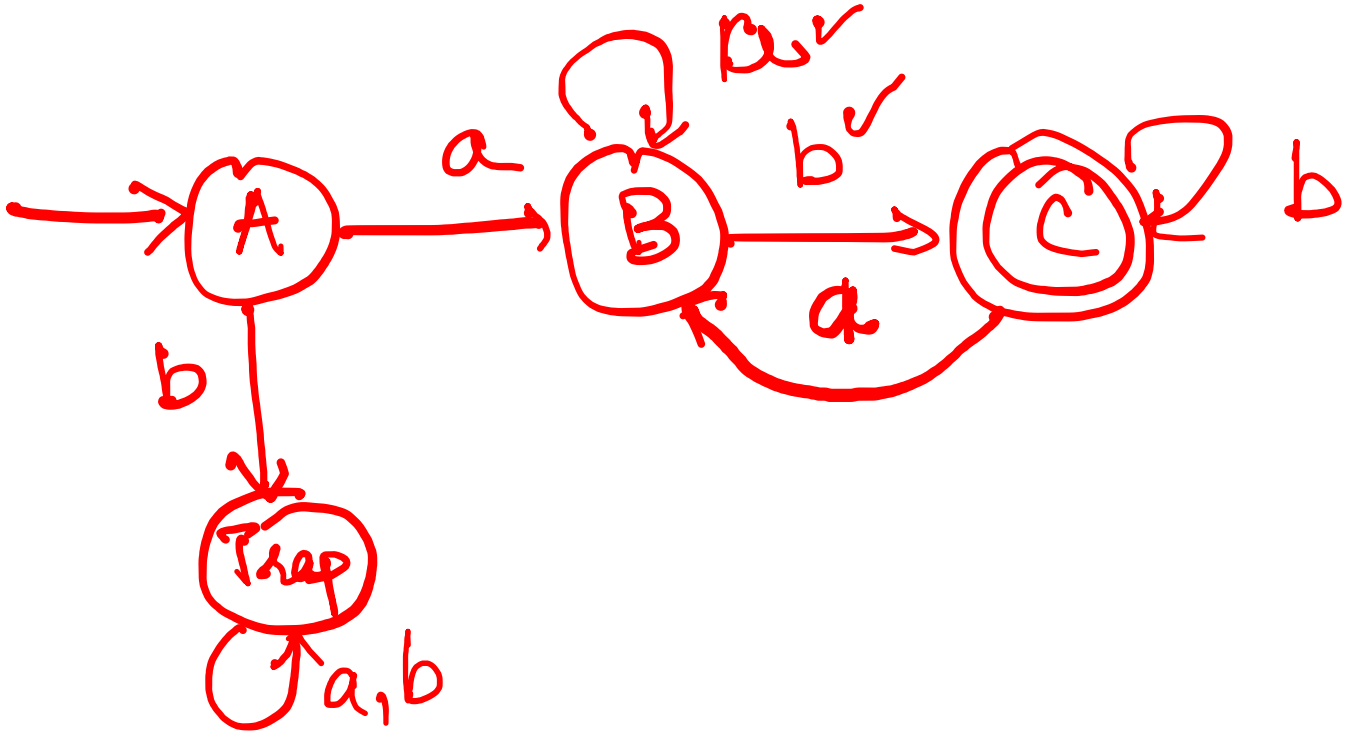
DFA AND NFA

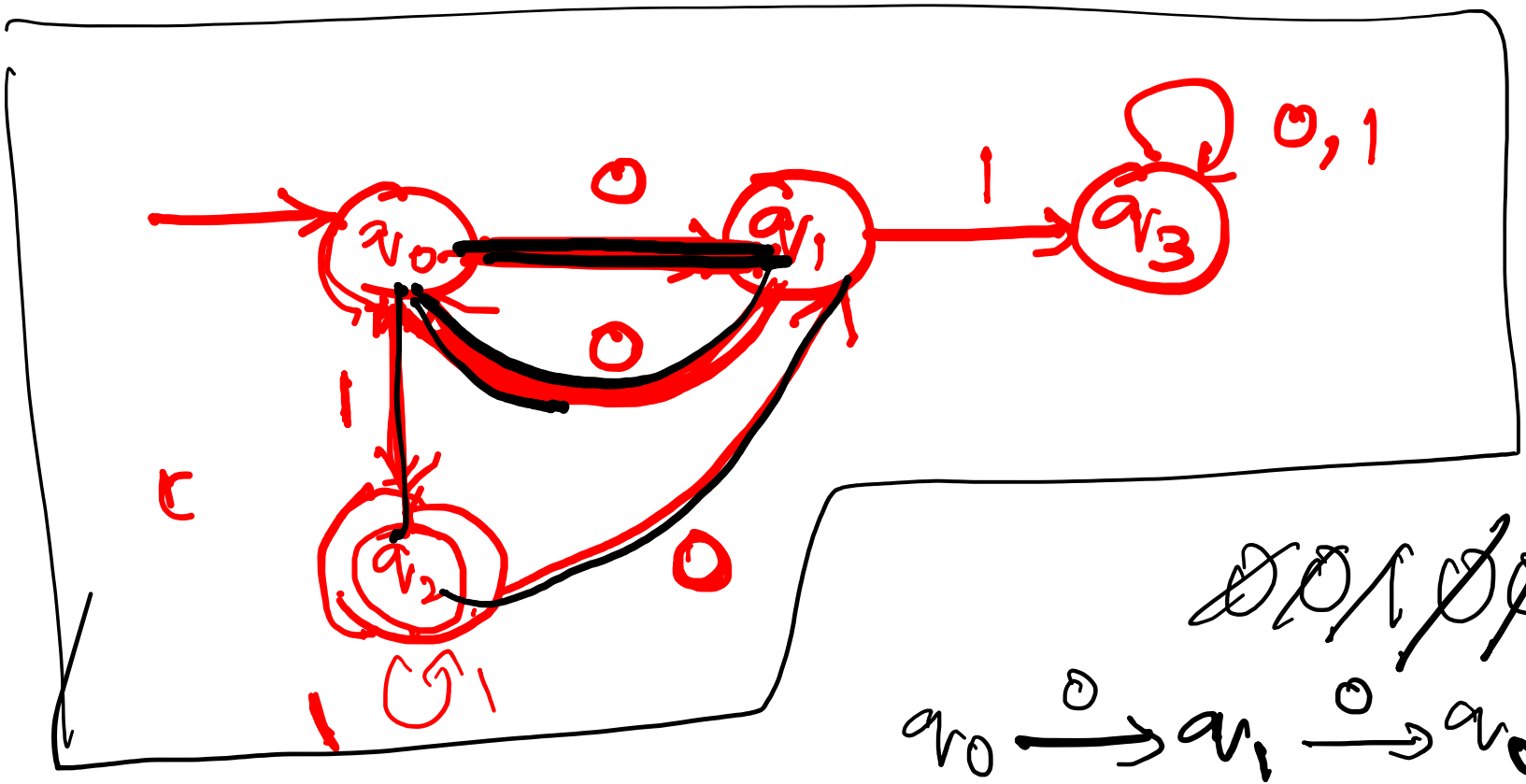
COMP 4200 – Formal Language

EXAMPLES TO TRY!

- Let Σ = {a, b}. Define A = {w $\in$ Σ^* | w ends in "bba"}.
- Let Σ = {a, b}. Define A = {w $\in$ Σ^* | w begins with 'a' and ends with 'b'}.
- Draw a DFA that recognizes the language over the alphabet {0, 1} consisting of all those strings that contain an even number of 1's.
- Design a DFA which accepts strings with even number of 0's followed by single 1 over {0,1}.




A
a
B
b
C
b
a
b
Trap
a, b



$$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_0 \xrightarrow{1} q_2$$

$$q_1 \xleftarrow{0} q_2 \xleftarrow{1} q_2 \xleftarrow{1} q_0 \xleftarrow{0} q_1$$

REGULAR OPERATIONS

- Operations that can be used to construct languages from other languages.
- A language is any set of strings.
- Since languages are sets, we can use the usual set operations:
  - Union → $L_1 \cup L_2$
  - Intersection → $L_1 \cap L_2$
  - Complement → L^c
  - Set difference → $L_1 - L_2$
- We also have new operations defined especially for sets of strings:
  - Concatenation → $L_1 \circ L_2$ or $L_1 L_2$
  - Star → L^*

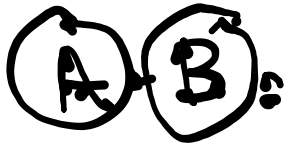
$L = \{a\}$

$\varepsilon, a, aa, aaa, \ldots$

REGULAR OPERATIONS

- Let A and B, be languages
- Out of six, below three are considered as Regular Operation
  - Union → $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$.
  - Concatenation → $A \cdot B = \{ xy \mid x \in A \text{ and } y \in B \}$.
  - Star → $A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$.
    - $A^* = A^0 \cup A^1 \cup A^2 \cup \ldots\ldots$
    - Kleene star is a unary operator on a single language.
    - A^* consists of (possibly empty!) concatenations of strings from A.

EXAMPLES



Let the alphabet Σ be the standard 26 letters {a, b, . . . , z}.

If A = {good, bad} and B = {boy, girl},

- A U B = {good, bad, boy, girl}
- A · B = {goodboy, goodgirl, badboy, badgirl},
- A^* = {∈, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, . . . }

B · A =

boy good
boy bad

EXAMPLES

Let the alphabet Σ be the standard 26 letters {a, b, . . . , z}.

A = {a,b,c} and B = {e,f,g}

- A ∪ B = {a, b, c, e, f, g}
- A · B = {ae, af, ag, be, bf, bg, ce, cf, cg}
- A* = {ε, a, b, c, aa, bb, cc, abc, aabc, . . . }

CLOSURE

- Let N = {1, 2, 3, . . . } be the set of natural numbers.
- When we say that N is closed under multiplication, we mean that for any x and y in N, the product x × y also is in N.
- Is N closed under division?

$x \times y \quad N$

$2 \quad 3 = 6$

$x = 1 \quad y = 2$

$\frac{x}{y} = \frac{1}{2}$

CLOSURE

- Let N = {1, 2, 3, . . . } be the set of natural numbers.
- When we say that N is closed under multiplication, we mean that for any x and y in N, the product x × y also is in N.
- N is not closed under division, as 1 and 2 are in N but 1/2 is not.
- A collection of objects is closed under some operation if applying that operation to members of the collection returns an object still in the collection.
- Simple words: If we start with FA-recognizable languages and apply any of these operations, we get another FA-recognizable language (for a different FA).
- We show that the collection of regular languages is closed under all three of the regular operations.

CARTESIAN PRODUCT

- The Cartesian product $X \times Y$ between two sets X and Y is the set of all possible ordered pairs with first element from X and second element from Y:

$$X \times Y = \{(x,y) : x \in X \text{ and } y \in Y\}.$$

CLOSURE UNDER UNION

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

if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof: Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
   1. This set is the Cartesian product of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$.
2. Σ, the alphabet, is the same as in M_1 and M_2.
3. δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$. Hence δ gets a state of M, together with an input symbol, and returns M's next state.
4. q_0 is the pair (q_1, q_2)
5. F is the set of pairs in which either member is an accept state of M_1 or M_2. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

This expression is the same as $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

SIMPLE EXPLANATION

- Start with FAs M1 and M2 for the same alphabet Σ.
- Get another FA, M3, with $L(M3) = L(M1) \cup L(M2)$.
- Idea: Run M1 and M2 "in parallel" on the same input. If reaches an accepting state, accept.
- Example:
  - L(M1): Contains substring 01.
  - L(M2): Odd number of 1s.
  - L(M3): Contains 01 or has an odd number of 1s.

CLOSURE UNDER UNION

Assume:

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$,

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Define $M = (Q, \Sigma, \delta, q_0, F)$, where

$Q = Q_1 \times Q_2$.

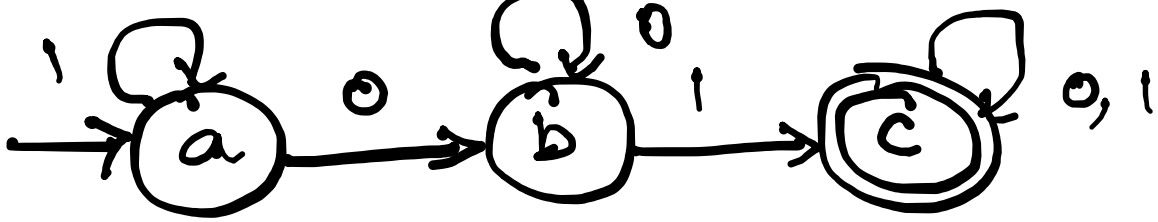
Cartesian product → $Q = \{(q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$

$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$.

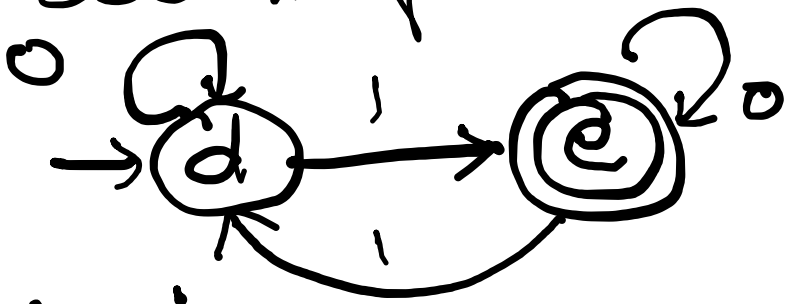
$q_0 = (q_1, q_2)$

$F = \{(q_1, q_2) \mid q_1 \in F_1 \text{ or } q_2 \in F_2\}$

contains substring 01



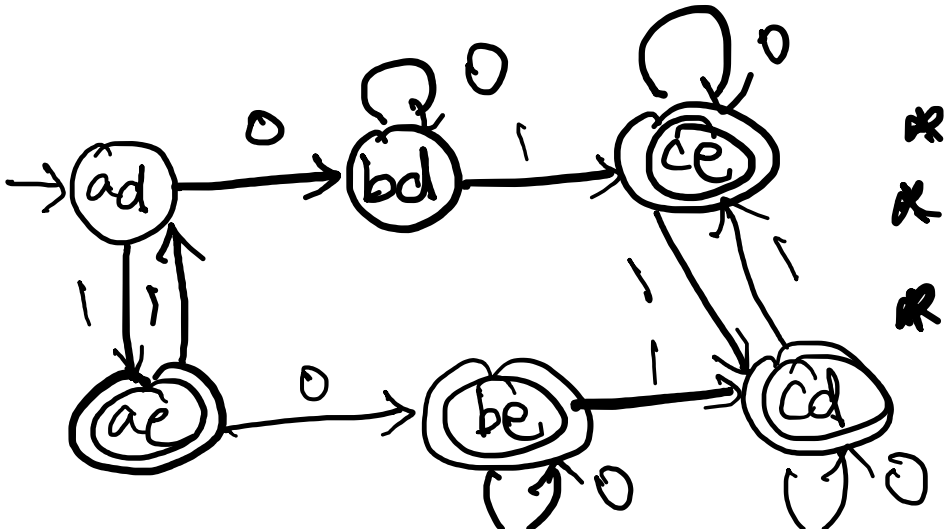
odd # of 1's



	0	1
→ a	b	a
b	b	c
*c	c	c

	0	1
→ d	d	e
*e	e	d

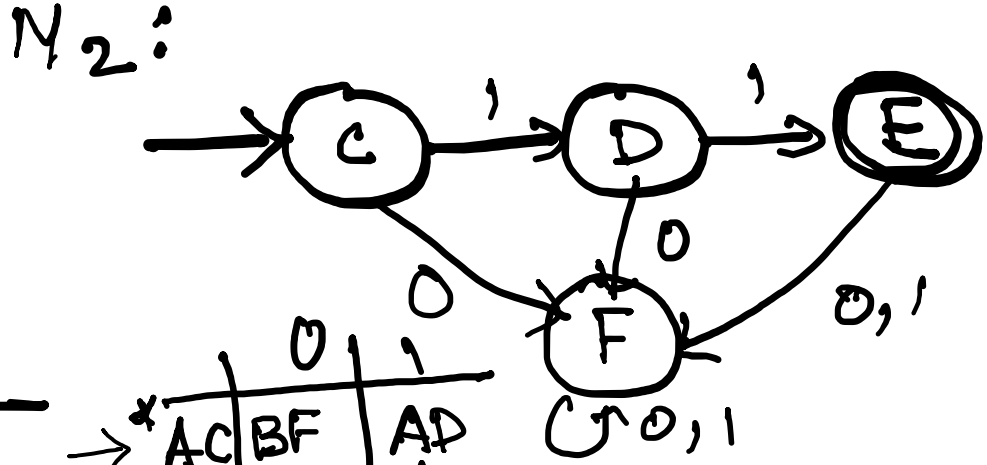
	0	1
→ad	bd	ae
*ae	be	ad
bd	bd	ce
*be	be	cd
*cd	cd	ce
*ce	ce	cd



L(M1): Contains even number of 0's

L(M2): Contains exactly two 1's.

L(M3): Contains even number of 0's or Contains exactly two 1's.



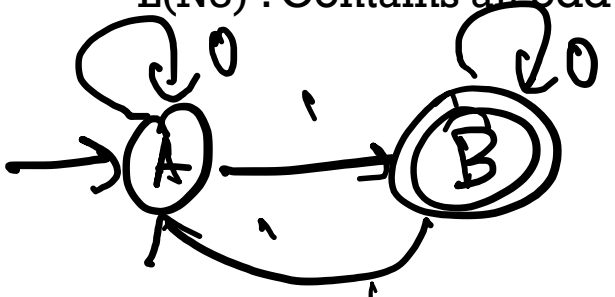
	0	1
→ A	B	A
* B	A	B

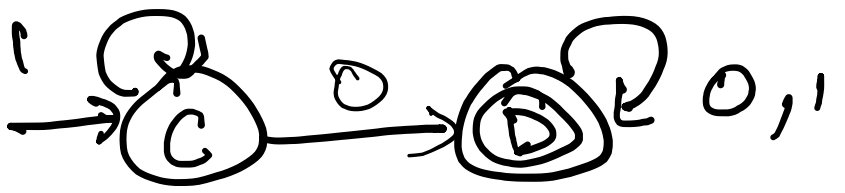
	0	1
→ C	F	D
D	F	E
* E	F	F
F	F	F

	0	1
→ AC	BF	AD
* AD	BF	AE
* AE	BF	AF
* AF	BF	AF
BC	AF	BD
BD	AF	BE
* BE	AF	BF
BF	AF	BF

EXAMPLE

- L(N1) : Contains an odd number of 1s.
- L(N2) : Contains at least one 0.
- L(N3) : Contains an odd number of 1s or contains at least one 0.





	0	1
→ A	A	B
* B	B	A

	0	1
→ C	D	C
* D	D	D

	0	1
→ AC	AD	BC
* AD	AD	BD
* BC	BD	AC
* BD	BD	AD

CLOSURE UNDER CONCATENATION

Theorem: *The class of regular languages is closed under the concatenation operation.*

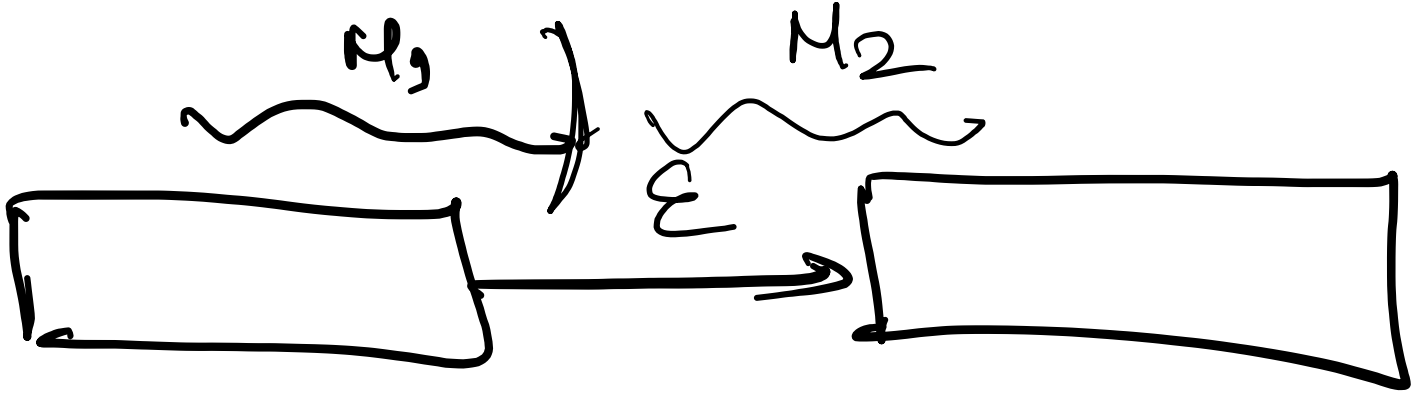
if A_1 and A_2 are regular languages, so is A_1 o A_2.

Proof: Let us start with FA M_1 and M_2 recognizing the regular language A_1 and A_2,

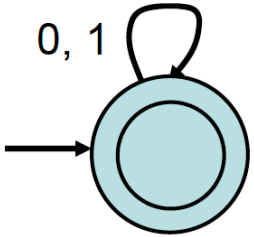
- Instead of constructing automaton M to accept its input if either M_1 or M_2 accept, it must accept if its input can be broken into two pieces, where M_1 accepts the first piece and M_2 accepts the second piece.
- The problem is that M doesn't know where to break its input (i.e., where the first part ends and the second begins).
- To solve this problem, there is a technique called Nondeterminism.

SIMPLE EXPLANATION

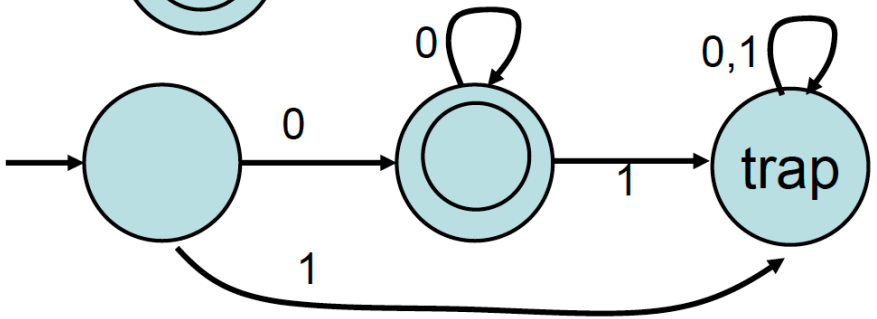
- Start with FAs M_1 and M_2 for the same alphabet Σ.
- Get another FA, M, with $L(M) = L(M_1) \circ L(M_2)$, which is $\{ x_1x_2 \mid x_1 \in L(M_1) \text{ and } x_2 \in L(M_2) \}$
- Idea: ???
  - Attach accepting states of M_1 somehow to the start state of M2.
  - But we have to be careful, since we don't know when we're done with the part of the string in $L(M_1)$---the string could go through accepting states of M_1 several times.



- Theorem 5: FA-recognizable languages are closed under concatenation.
- Example:
  - $\Sigma = \{ 0, 1\}$, $L_1 = \Sigma^*$, $L_2 = \{0\}\{0\}^*$ (just 0s, at least one).
  - $L_1 L_2$ = strings that end with a block of at least one 0
  - M_1:



  - M_2:



  - How to combine?
  - We seem to need to "guess" when to shift to M_2.
  - Leads to our next model, NFAs, which are FAs that can guess.

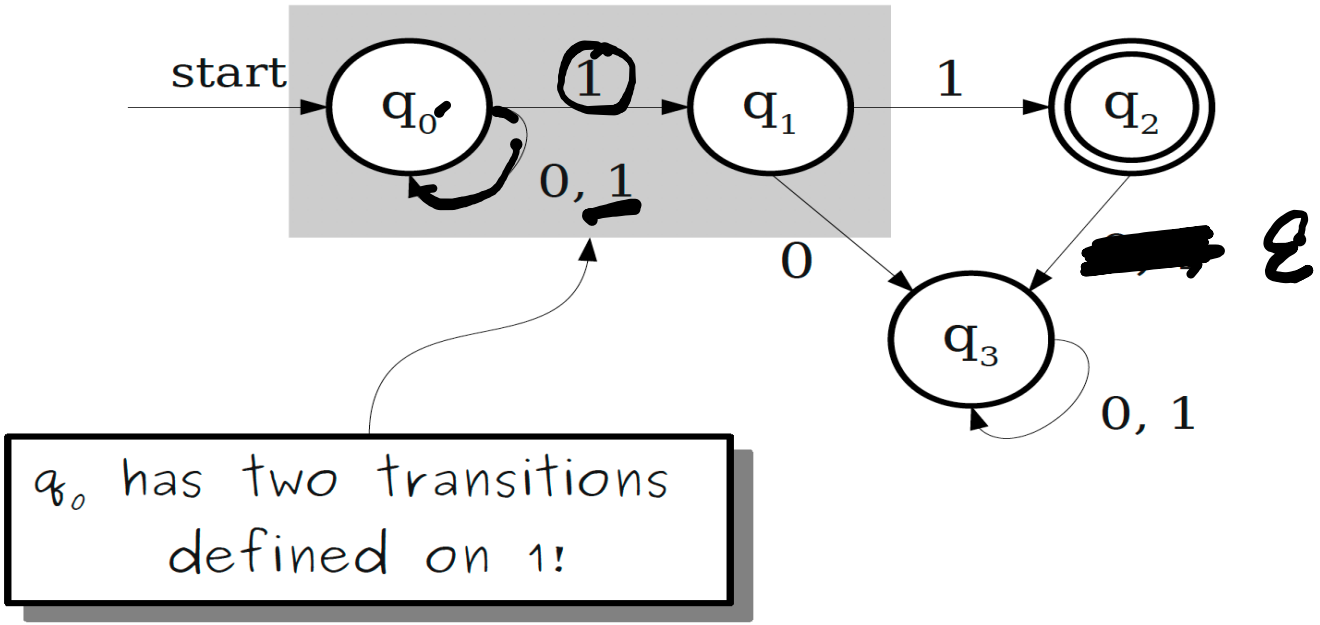
NFAs

- An NFA is a
  - **N**ondeterministic
  - **F**inite
  - **A**utomaton
- Conceptually like a DFA but equipped with the vast power of nondeterminism.
- Given the current state, there can be multiple next states
- The next state may be chosen at random or chosen in parallel
- Nondeterminism is a generalization of determinism, so *every deterministic finite automaton is automatically a nondeterministic finite automaton.*

(NON)DETERMINISM

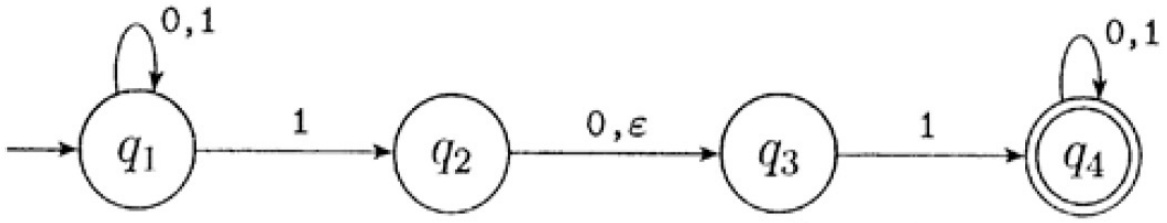
- A model is deterministic if at every point in the computation, there is exactly one choice that can make.
- The machine accepts if that series of choices leads to an accepting state.
- A model is nondeterministic if the computing machine may have multiple decisions that it can make at one point.
- The machine accepts if any series of choices leads to an accepting state.

SIMPLE EXAMPLE OF NFA

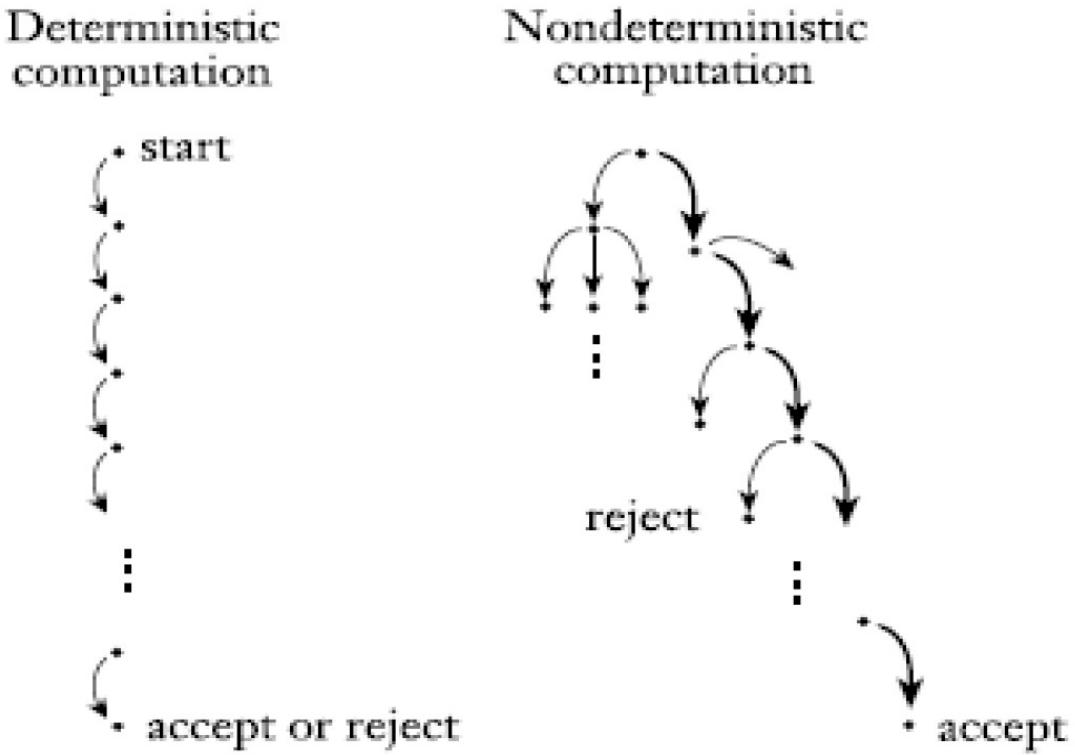


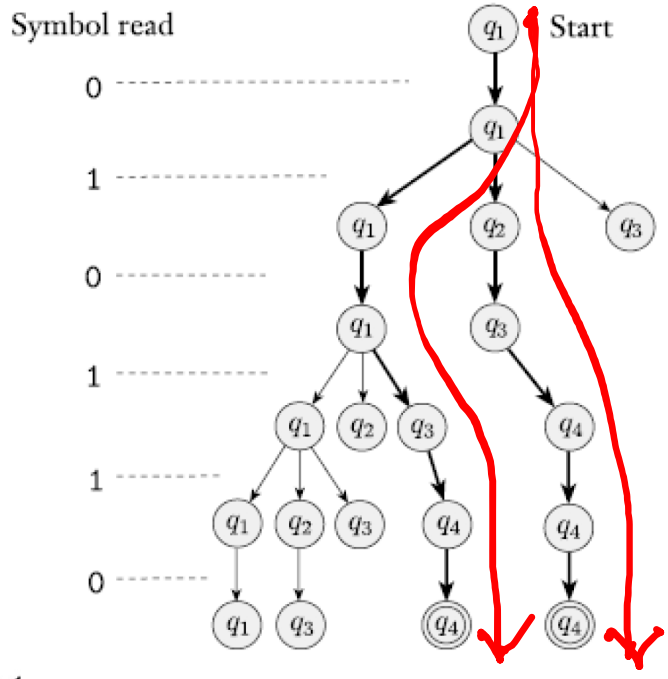
NFA

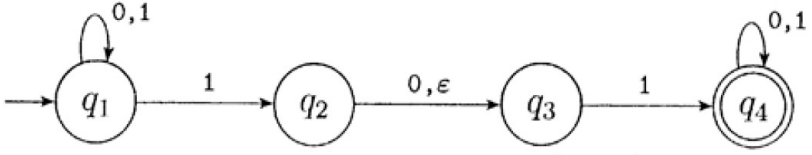
- The machine below violates the rule of unique state.
  - State q1 has one exiting arrow for 0, but it has two for 1;
  - q2 has one arrow for 0, but it has none for 1.
- *In an NFA, a state may have zero, one, or many exiting arrows for each alphabet symbol.*
- An NFA may have arrows labeled with members of the alphabet or ε. Zero, one, or many arrows may exit from each state with the label ε



HOW DOES AN NFA COMPUTE?







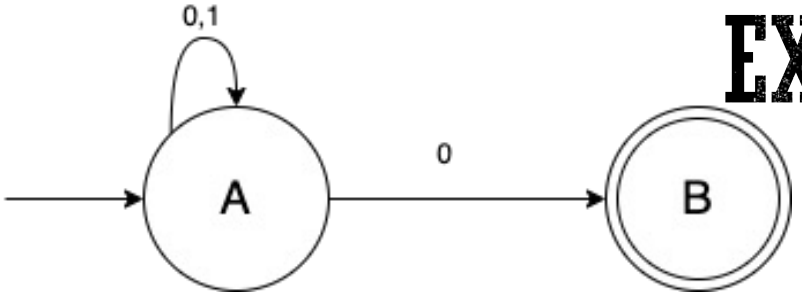
Input → 010110

FORMAL DEFINITION

A nondeterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \longrightarrow P(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

EXAMPLE – 1



A x 0 → A
A x 0 → B
A x 1 → A
B x 0 → ∅
B x 1 → ∅

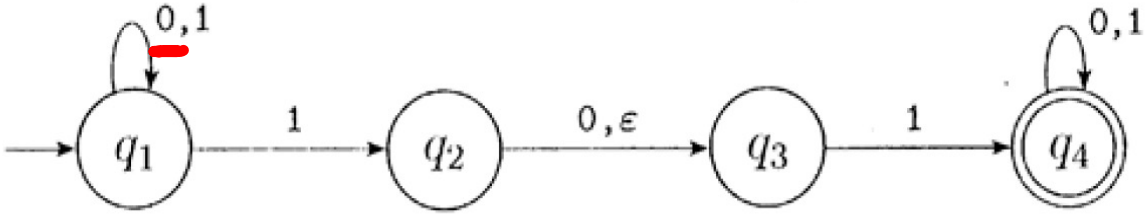
Possibilities for A on input 0:

A → A, B, AB, ∅

L = {set of all strings that end with 0}

- Q → {A, B}
- Σ → {0, 1}
- q_0 → A
- F → {B}
- δ → Q x Σ →

	0	1
→ A	{A, B}	A
B	∅	∅



Q : {q_1, q_2, q_3, q_4}

Σ = {0, 1}

q_0 = q_1

F : {q_4}

	0	1	ε
q_1	q_1	{q_2, q_3}	∅
q_2	q_3	∅	q_3
q_3	∅	q_4	∅
q_4	q_4	q_4	∅

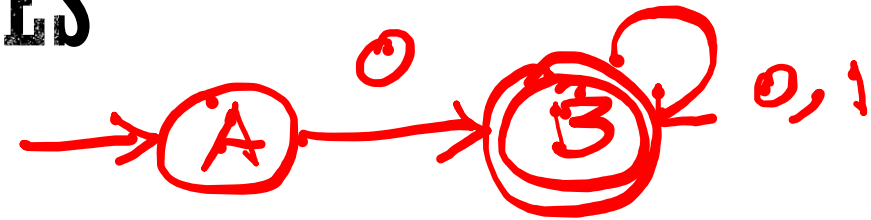
EXAMPLE – 2

FORMAL DEFINITION OF COMPUTATION

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then we say that N accepts w if we can write w as $w = y_1 y_2 \ldots\ldots y_m$, where each y_i is a member of Σ_ε and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \ldots, m - 1$, and
3. $r_m \in F$.

EXAMPLES



L = {set of all strings that start with 0}

L = {set of all strings over {0,1} of length 2}

L = {set of all strings over {0,1} that contain 0}

L = {set of all strings over {0,1} that end with 1}

Design a NFA which accept string 1010 only.

L = {set of all strings that end with 11}

L = {w | w ∈ {0, 1}* and w ends with 00}

L = {w | w ∈ {0, 1}* and w ends with 10}

Design NFA which accepts set of all strings containing 0011 as substring.

Design NFA for the language of all strings over {a, b} that end with one of "abba", "aba", and "aa".

Design a NFA to accept the strings with a's and b's such that string contains either two consecutive a's or two consecutive b's.

Design NFA for the language of all strings over {0, 1, 2} that end with one of 01, 12, and 21.

EQUIVALENCE OF NFAs AND DFAs

- NFAs and DFAs recognize the same class of languages.
  - We say two machines are equivalent if they recognize the same language.
- NFAs have no more power than DFAs:
  - with respect to what can be expressed.
  - But NFAs may make it much easier to describe a given language.
- Every NFA has an equivalent DFA.

PROOF OF EQUIVALENCE OF NFA AND DFA

- Every DFA is an NFA and not vice versa
- There is an equivalent DFA for every NFA.
- DFA:
  - $\delta: Q\ X\ \Sigma\ \rightarrow Q$
- NFA:
  - $\delta: Q\ X\ \Sigma\ \rightarrow\ 2^Q$

From the above transition function, *Q is a part of* 2^Q.

Hence, $NFA\ \cong DFA$

PROOF BY CONSTRUCTION

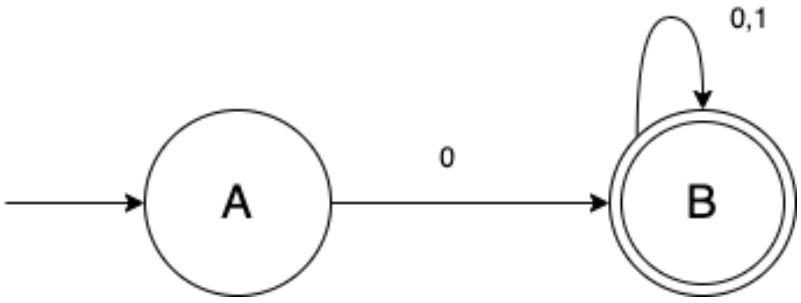
- If k is the number of states of the NFA, it has 2^k subsets of states.
- Find the start state and accept states of the DFA, and what will be its transition function.
- Proof:

Let N = (Q, Σ, δ, q_0, F) be an NFA recognizing language A. Construct a DFA M = (Q′, Σ, δ', q'_0 , F ′).

1. Q ′ = $\wp$(Q) Every state of M is a set of states of N.
2. Transition function,
   1. R ∈ Q′ and a ∈ Σ, let δ'(R, a) = {q ∈ Q | q ∈ δ(r, a) for some r ∈ R}.
   2. When M reads symbol a in state R, it shows where a takes each state .
      $\delta'(R, a) = \bigcup_{r \epsilon R} \delta(r, a)$
3. q_0 ‘= $\{q_0\}$. M starts in the state corresponding to the collection containing just the start state of N.
4. F’= {R ∈ Q’ | R contains an accept state of N}.

NFA → DFA EXAMPLE

- L = {set of all strings over {0,1} that starts with '0'}



Transition table for NFA

	0	1
A	B	∅
B	B	B

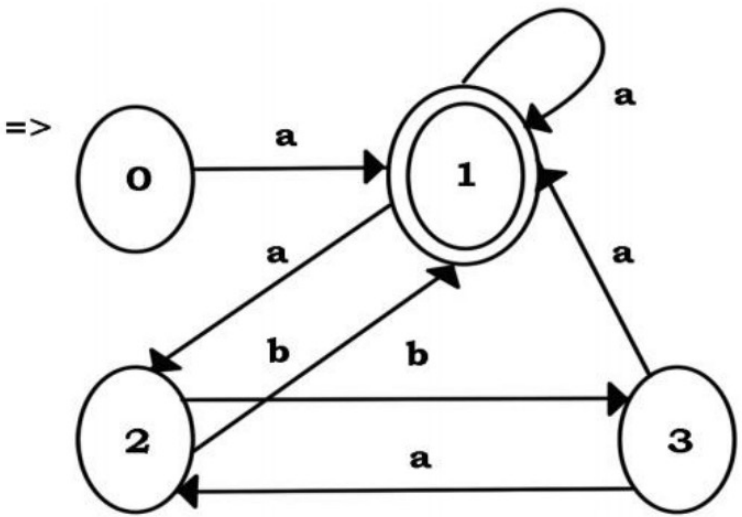
EXAMPLES

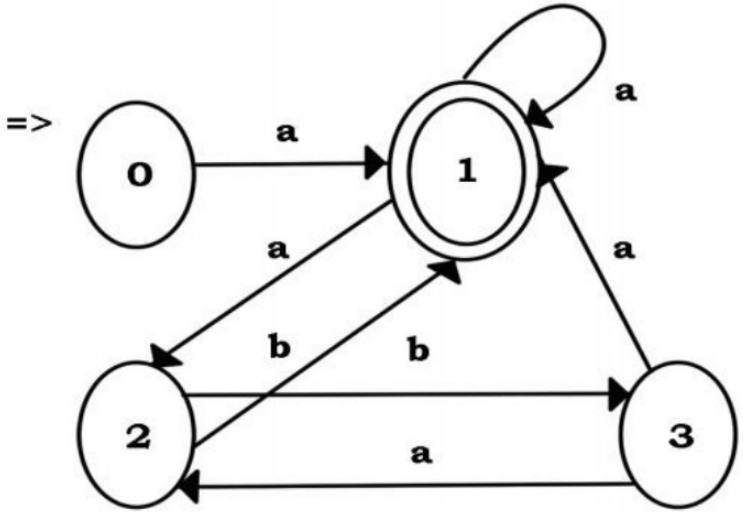
- L = {set of all strings over {0,1} that end with 1}
- L = {set of all strings over {0,1} that end with 01}
- M = {{p,q,r}, (0,1), δ, p, {r}}

	0	1
p	{p,q}	p
q	{q,r}	-
r	-	r

- Design an NFA which accepts set of all binary strings containing the third symbol from the left end is 1 and second symbol from the left end is 0. Convert it to its equivalent DFA.

EXAMPLE




=>
0
1
2
3
a
a
a
a
b
b
a

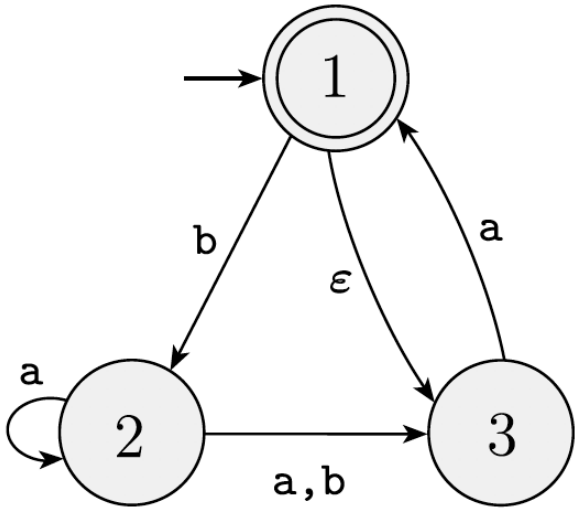
ε-TRANSITIONS

- NFAs have a special type of transition called the ε-transition.
- ε, represents empty symbols
- Every state on ε, goes to itself
- An NFA may follow any number of ε-transitions at any time without consuming any input.

CONVERT ε –NFA TO DFA

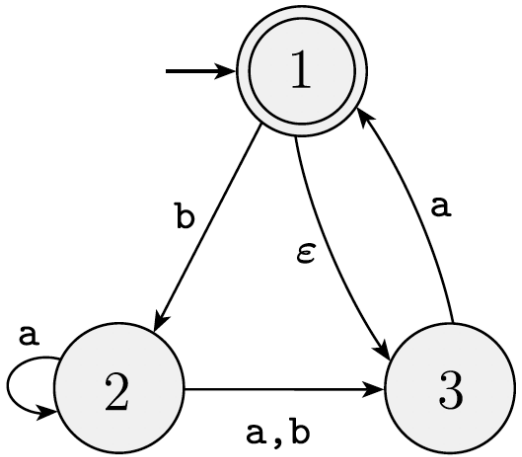
- The formal definition,
  - N4 = (Q, {a,b}, δ, 1, {1}), the set of states Q is {1, 2, 3}
- Construct DFA that is equivalent to N4.
- Steps:
  - Determine number of states in N4, {1,2,3} → so our DFA will have 8 states.
  - D's states are subset of N4

    {∅, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}
  - Determine start and final state of D
    - Start state, q_0 → E({1}) = {1,3}
    - Final state, F → {1}, {1,2}, {1,3}, {1,2,3}
  - Determine transition function

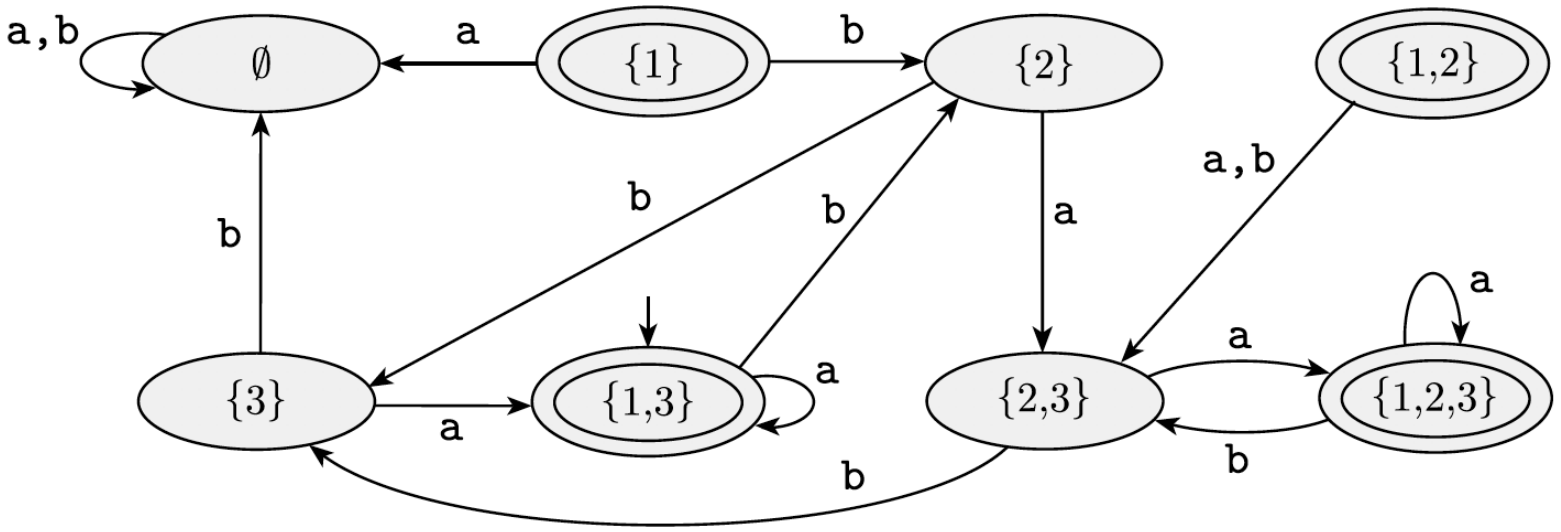


- Transition table

	a	b
∅	∅	∅
{1}	∅	{2}
{2}	{2,3}	{3}
{3}	{1,3}	∅
{1,2}	{2,3}	{2,3}
{1,3}	{1,3}	{2}
{2,3}	{1,2,3}	{3}
{1,2,3}	{1,2,3}	{2,3}

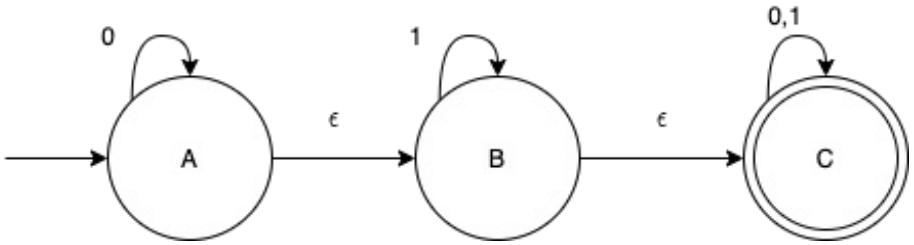


- What are the states in DFA?
  - {∅, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}
- What is the start state?
  - The start states of the NFA, including those reachable by ε-transitions
  - {1,3}
- What are the accept/final states?
  - {1}, {1,2}, {1,3}, {1,2,3}

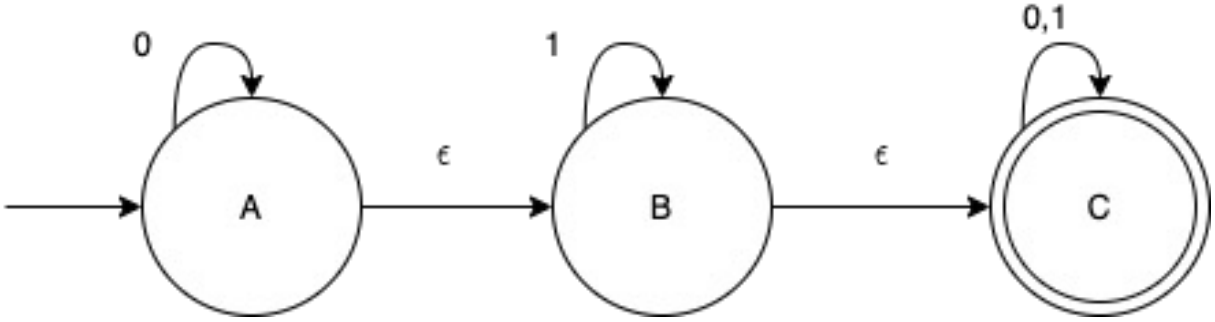

a,b
∅
a
{1}
b
{2}
{1,2}
a,b
b
b
b
a
{3}
a
{1,3}
a
{2,3}
a
{1,2,3}
a
b
b

ALTERNATE METHOD

- Steps:
  - Convert the given $\varepsilon - NFA$ into NFA by removing the ε.
  - Convert the regular NFA into equivalent DFA
- Convert the given $\varepsilon - NFA$ into NFA by removing the ε.
  - For each state that we need to check ε^* and where its transition is on getting a particular input
  - Check on the set of states that we get with ε^* again.
  - What is ε^*?
    - All the states that can be reached from a particular state only by seeing the ε symbol.



EXAMPLE



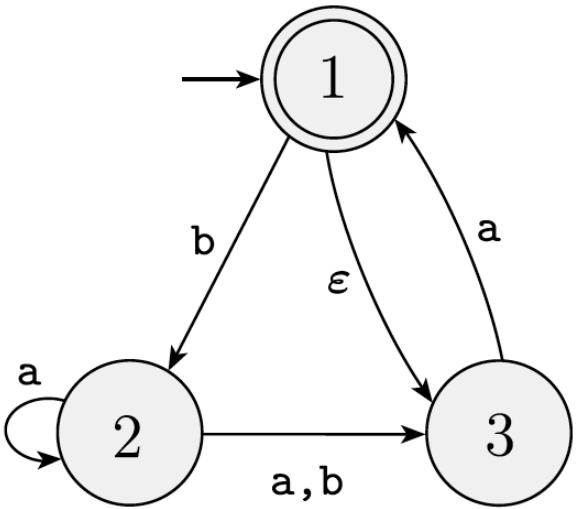
Start state → A
Final/accept state →

	ε^*	0	ε^*
A	A B C	A ∅ C	A, B, C ∅ C
B	B C	∅ C	∅ C
C	C	C	C

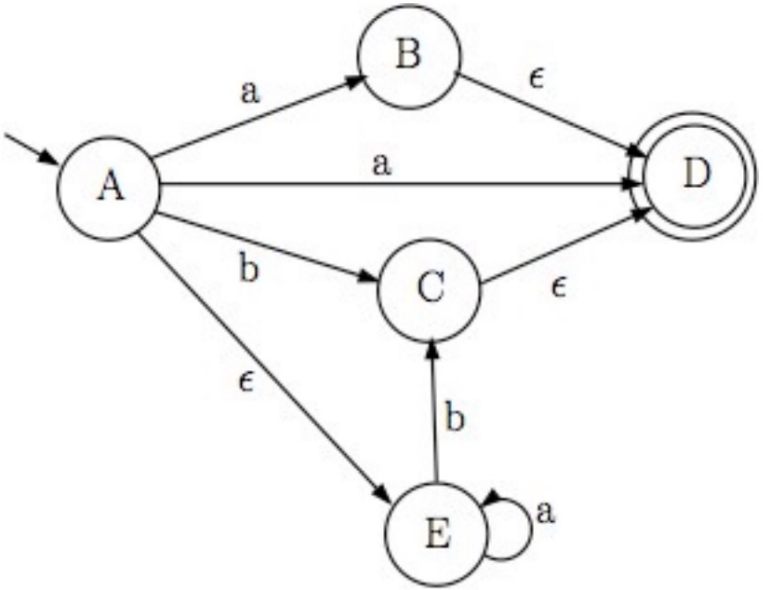
	ε^*	1	ε^*
A	A B C	∅ B C	∅ B,C C
B	B C	B C	B,C C
C	C	C	C

	0	1
A	{A,B,C}	{B,C}
B	{C}	{B,C}
C	{C}	{C}

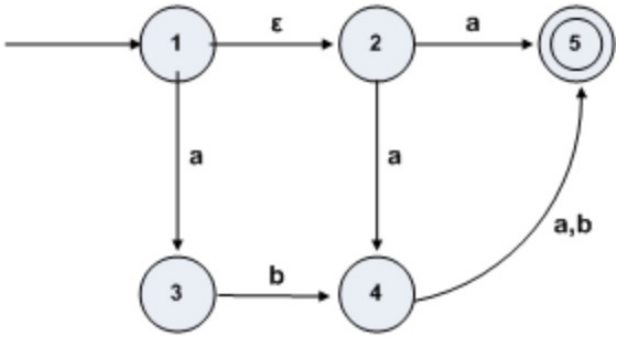
TRY YOURSELF!



EXAMPLE



EXAMPLE



EXAMPLE

STATES	INPUTS			
	ε	a	b	c
=>p	Φ	{p}	{q}	{r}
q	{p}	{q}	{r}	Φ
*r	{q}	{r}	Φ	{p}