NFA AND EQUIVALENCE

COMP 4200 – Formal Language

NFAs Review

NFA has more options

DFA only has 1 unique

- 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.*

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. $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function, can have null nowhere to go
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

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".

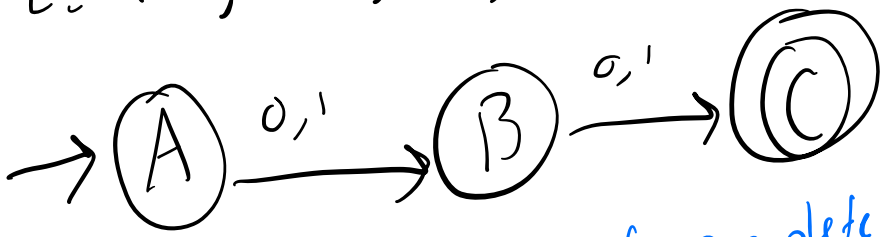
Design a NFA to accept the strings with a's and b's such that string contains either two consecutive a's or two consecutive b's.

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

All DFAs are NFAs

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

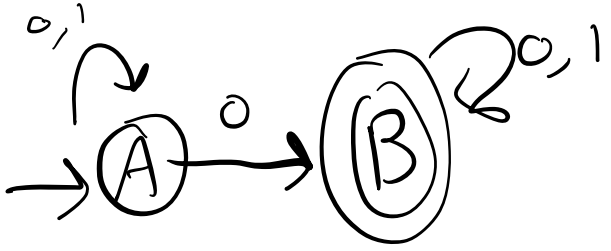
L = {00, 10, 11, 01}



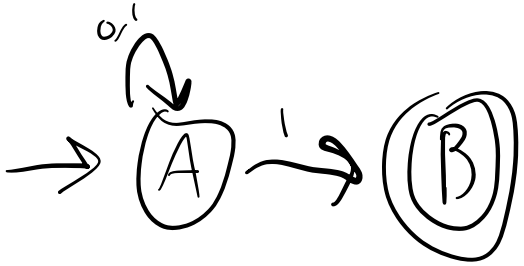
For NFA don't have to complete transitions for every state

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

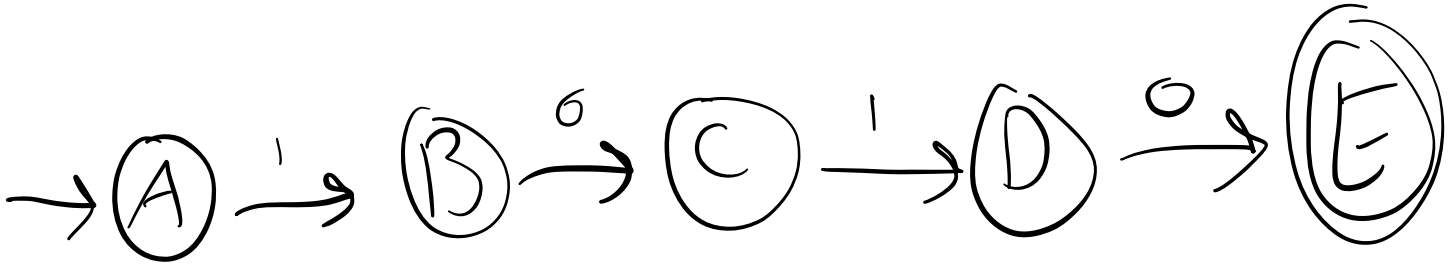
Prof's Answer



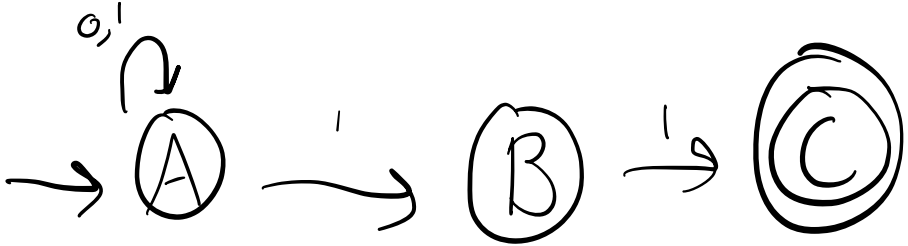
L = {set of all strings over {0,1} that end with 1}



Design a NFA which accept string 1010 only.

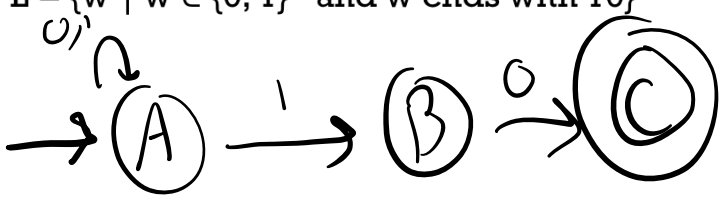


L = {set of all strings that end with 11}

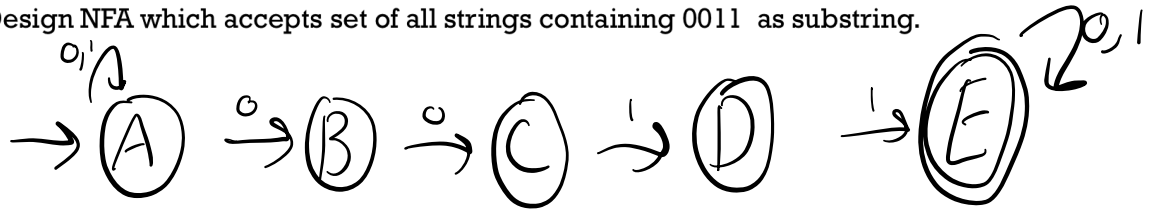


L = {w | w ∈ {0, 1}* and w ends with 00}

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".

ε - no input needed

ε only in NFA not DFA

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.

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:
  - $\delta: Q\ X\ \Sigma \rightarrow 2^Q$

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

NFA A, B, AB, $\emptyset$

DFA A, B

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

Hence, $NFA \cong DFA$

equivalent in nature

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 ′).

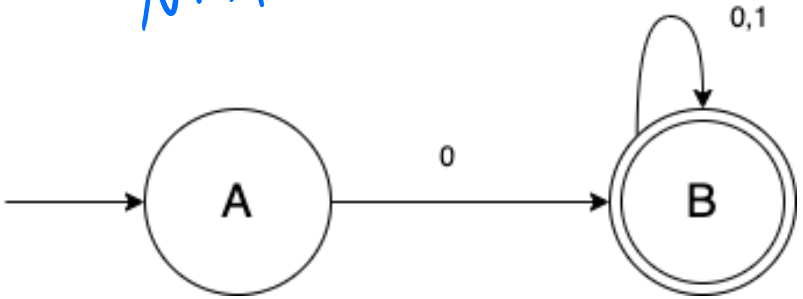
1. Q ′ = $\wp$(Q) Every state of M is a set of states of N.
2. Transition function,
   1. R ∈ Q′ and a ∈ Σ, let δ'(R, a) = {q ∈ Q | q ∈ δ(r, a) for some r ∈ R}.
   2. When M reads symbol a in state R, it shows where a takes each state .
      $\delta'(R, a) = \bigcup_{r \epsilon R} \delta(r, a)$
3. q_0 ‘= $\{q_0\}$. 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}.

Subset Construction

NFA → DFA EXAMPLE

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

NFA

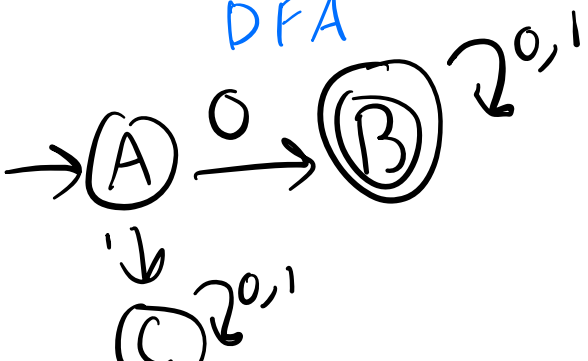


NFA

Transition table for NFA

	0	1
A	B	Ø
B	B	B

DFA



DFA

	0	1
→ A	B	C
C	C	C

new state

EXAMPLES

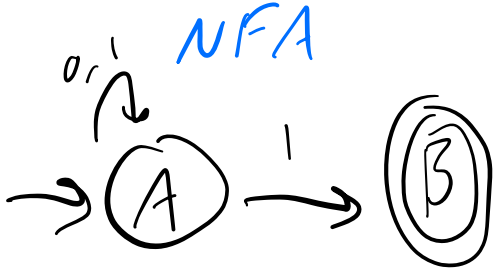
- L = {set of all strings over {0,1} that end with 1}
- L = {set of all strings over {0,1} that end with 01}
- M = {{p,q,r}, (0,1), δ, p, {r}}

	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.

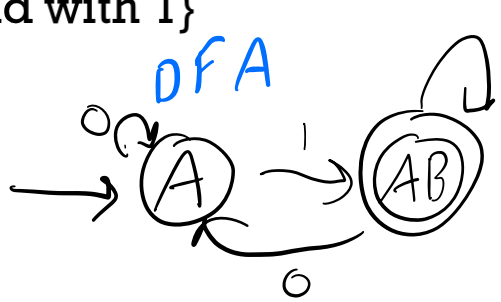
- L = {set of all strings over {0,1} that end with 1}

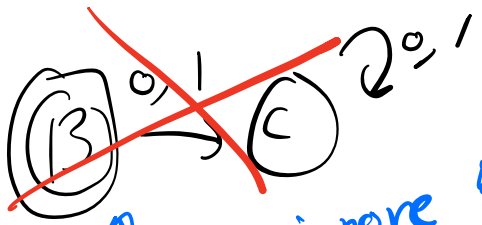
NFA



	0	1
A	A	{A, B}
B	∅	∅

DFA





you can ignore b/c no connection

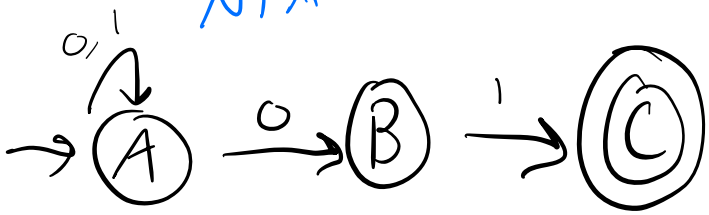
	0	1
→ A	A	AB
~~A~~ AB	A	AB
~~★~~ B	C	C
C	C	C

AB = A∪B

included B

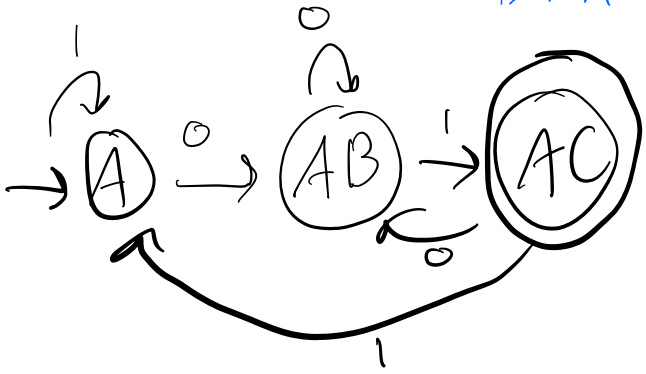
- L = {set of all strings over {0,1} that end with 01}

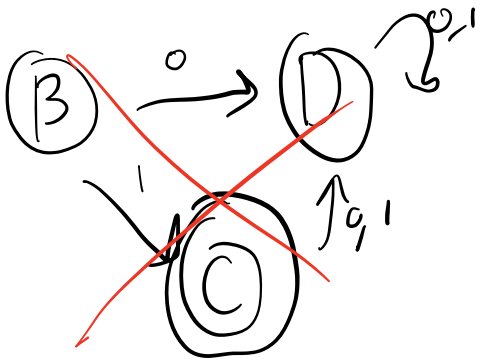
NFA



	0	1
→ A	{A,B}	A
B	∅	C
★ C	∅	∅

DFA



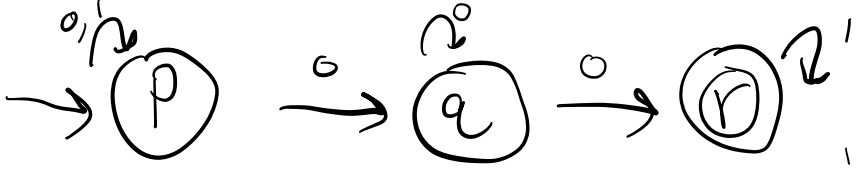


	0	1
→ A	AB	A
AB	AB	AC
★ AC	AB	A
B	D	C
★ C	D	D
D	D	D

- M = {{p,q,r}, (0,1), δ, p, {r}}

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

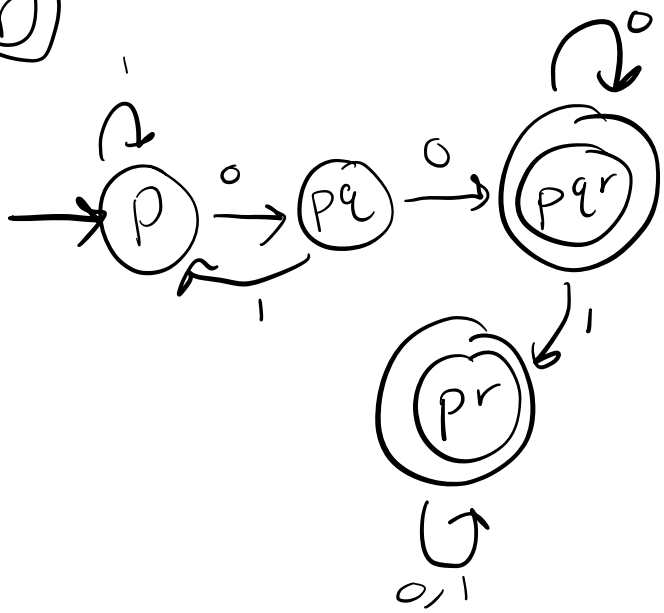
NFA



DFA

	0	1
→ p	pq	p
pq	pqr	p
★ pqr	pqr	pr
★ pr	pq	pr
q	qr	∅
★ r	∅	r
★ qr	qr	r
∅	∅	∅

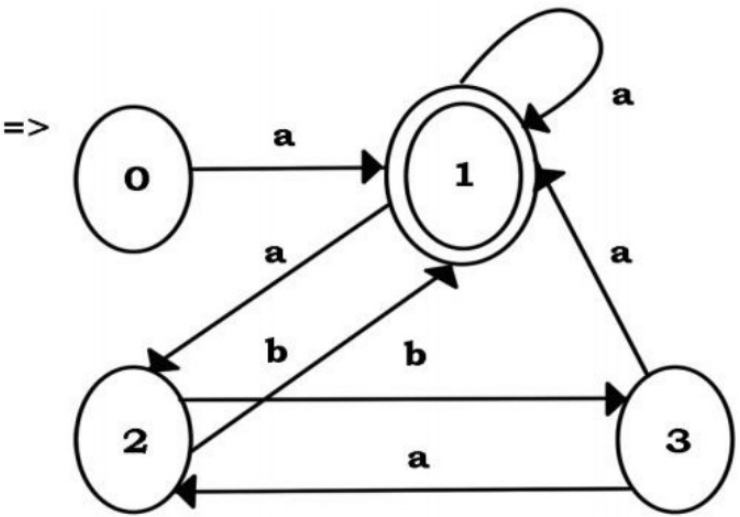
- ignore down

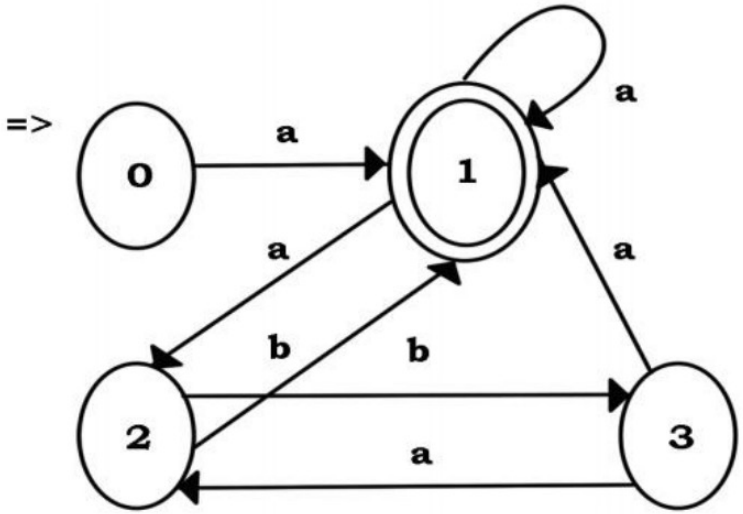


EXAMPLE

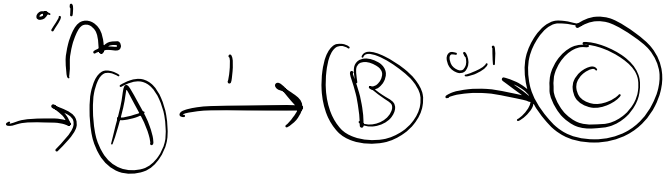
	a	b
→ 0	1	∅
★ 1	12	∅
★ 12	12	13
★ 13	12	∅
∅	∅	∅

	a	b
→ 0	1	∅
★ 1	{1,2}	∅
2	∅	{1,3}
3	{1,2}	∅




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

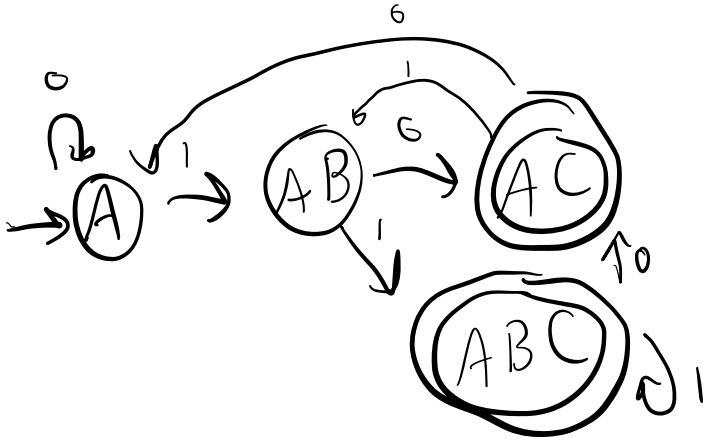
Design NFA for a that accepts strings over {0,1} in which the second last digit is always 1.



	0	1
→ A	A	{AB}
B	C	C
★C	∅	∅

DFA

	0	1
→ A	A	AB
AB	AC	ABC
★AC	A	AB
★ABC	AC	ABC
★C	D	D
D	D	D



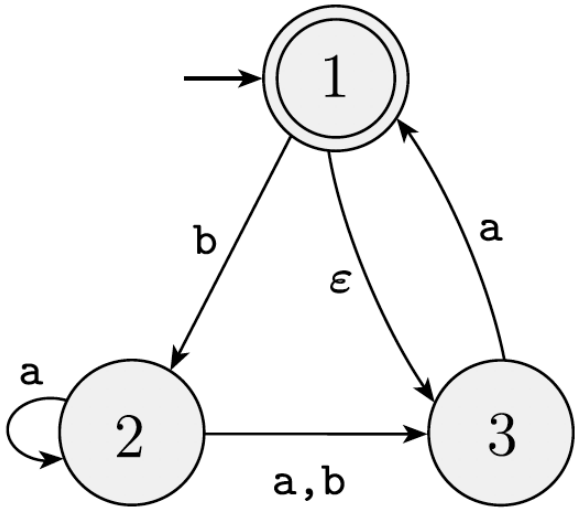
ε-TRANSITIONS

- 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

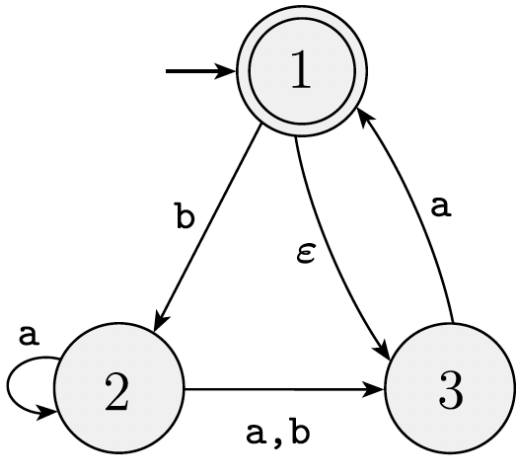
- The formal definition,
  - N4 = (Q, {a,b}, δ, 1, {1}), the set of states Q is {1, 2, 3}
- Construct DFA that is equivalent to N4.
- Steps:
  - Determine number of states in N4, {1,2,3} → so our DFA will have 8 states.
  - D's states are subset of N4

    ✓ {∅, {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

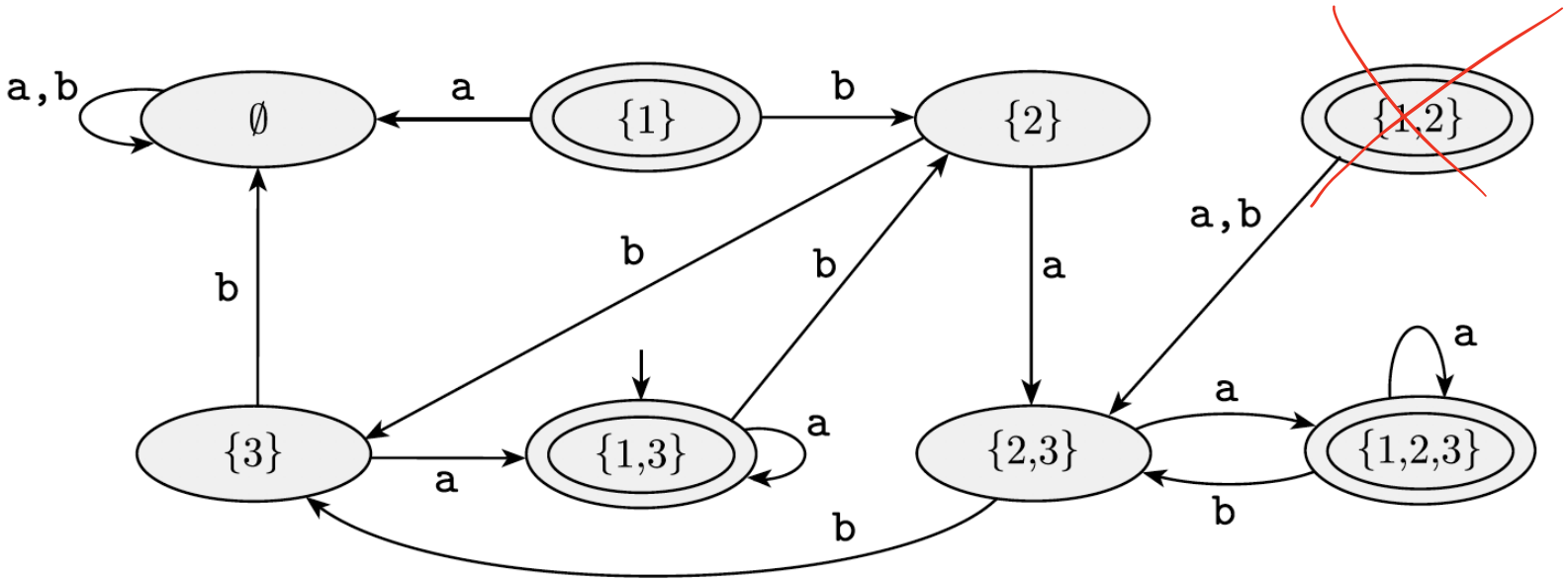


- Transition table

	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

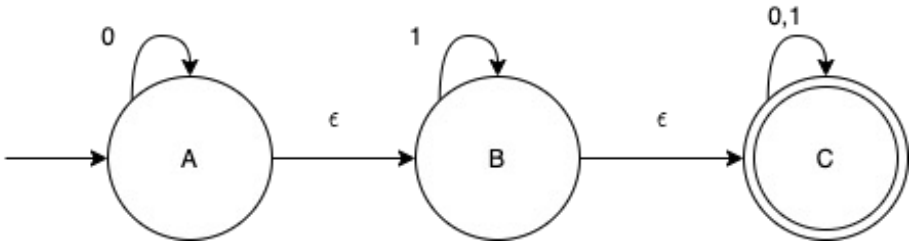
ALTERNATE METHOD

closure

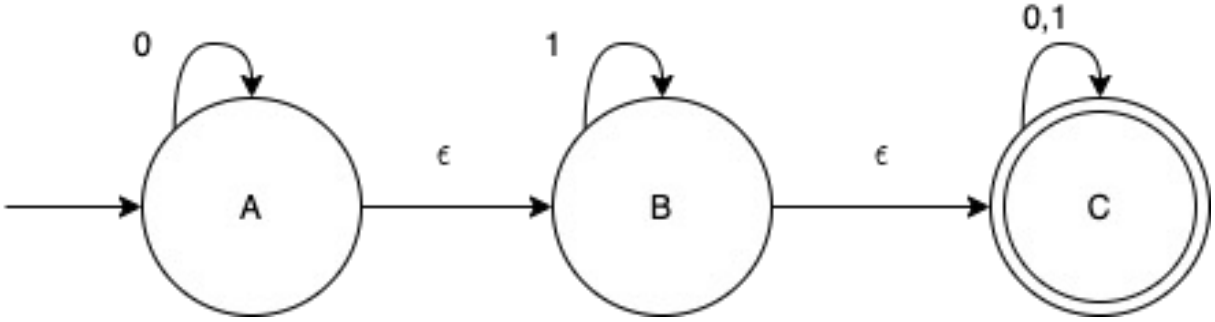
- Steps:
  - Convert the given $\varepsilon - NFA$ into NFA by removing the ε.
  - Convert the regular NFA into equivalent DFA
- Convert the given $\varepsilon - NFA$ into NFA by removing the ε.
  - For each state that we need to check ε^* and where its transition is on getting a particular input
  - 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.

closure

A, B, C



EXAMPLE



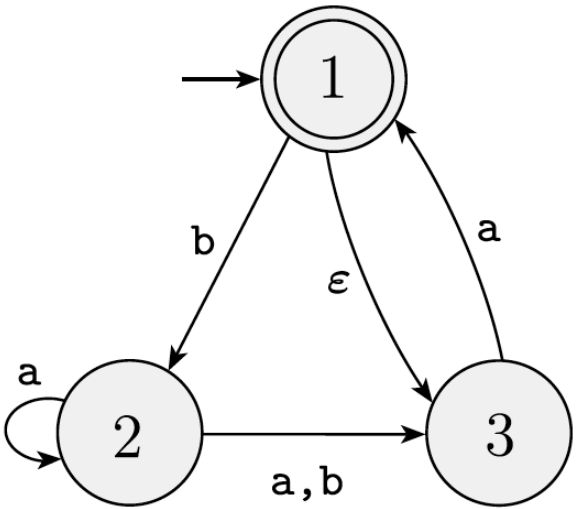
Start state → A
Final/accept state →

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

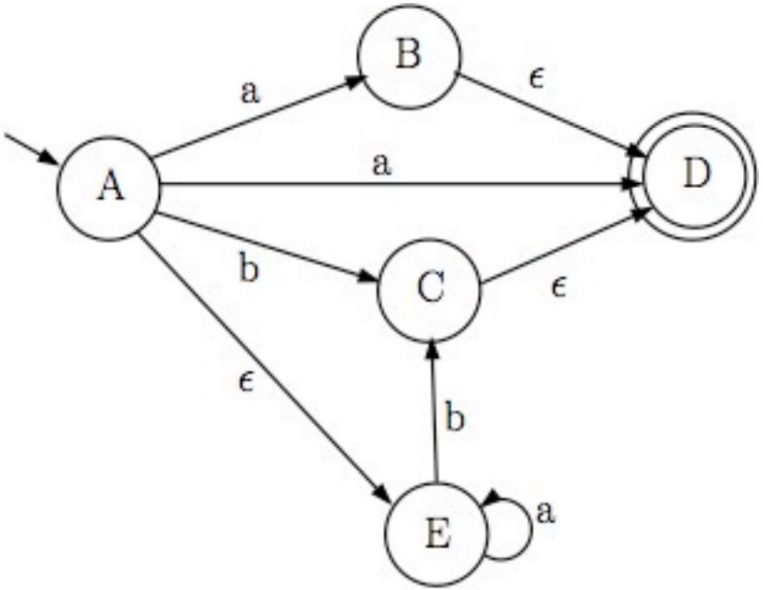
	ε^*	1	ε^*
A	A B C	Ø B C	Ø B,C C
B	B C	B C	B,C C
C	C	C	C

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

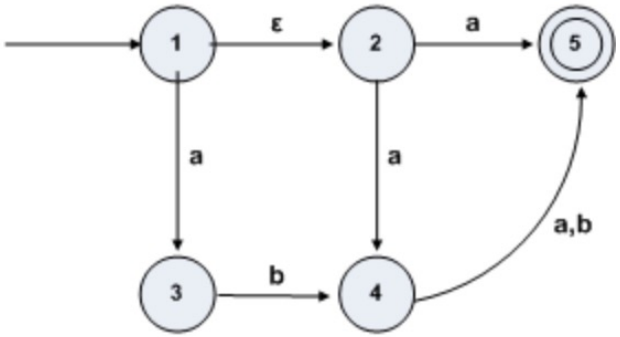
TRY YOURSELF!



EXAMPLE



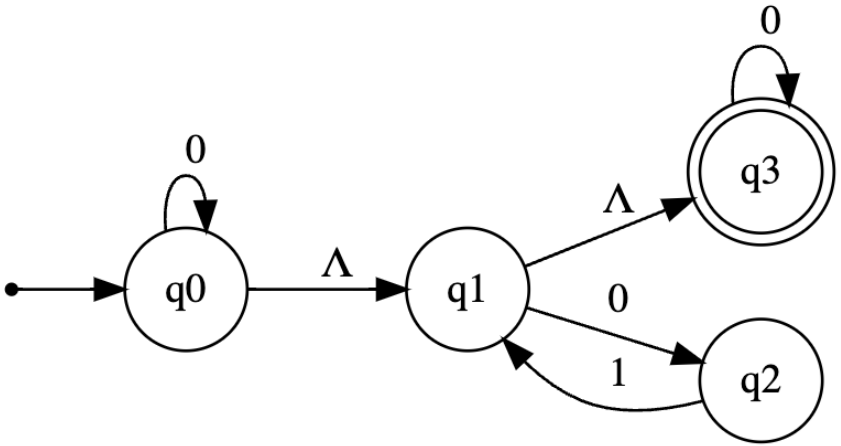
EXAMPLE



EXAMPLE

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

EXAMPLE



EXAMPLE