Theory of Computation

Borhan (ASH2101008M)

What is TOC?

- A branch of theoretical CS
- Whether and how efficiently a problem can be solved on computational model, using an algorithm

TOC has major 3 branches.

- Automata theory
- Computability theory
- Computational Complexity theory

Model of Computation: mathematical abstraction of Computer

Definition: describe a object and notations.

Theorem: mathematical statement basis on previously stablished statement.

Proof: convincing logical argument that statement is true.

Lemma: A minor result (theorem, it’s the lemma) to prove another theorem.

Corollaries: A result in which the proof relies heavily.

THEOREM

If m and n are any two whole numbers and

- $a = m^2 - n^2$
- $b = 2mn$
- $c = m^2 + n^2$

then $a^2 + b^2 = c^2$

Proof:

$$\begin{aligned} \mathbf{a^2 + b^2} &= (m^2 - n^2)^2 + (2mn)^2 \\ &= m^4 - 2m^2n^2 + n^4 + 4m^2n^2 \\ &= m^4 + 2m^2n^2 + n^4 \\ &= (m^2 + n^2)^2 \\ &\mathbf{= c^2} \end{aligned}$$

Deductive Proof: If H (hypothesis), then C (conclusion)

- Sequence of statement
- -hypothesis to conclusion

Contradiction Proof:

- Assume the theorem is false, this assumption leads to false
  Example: sqrt (2) is irrational.

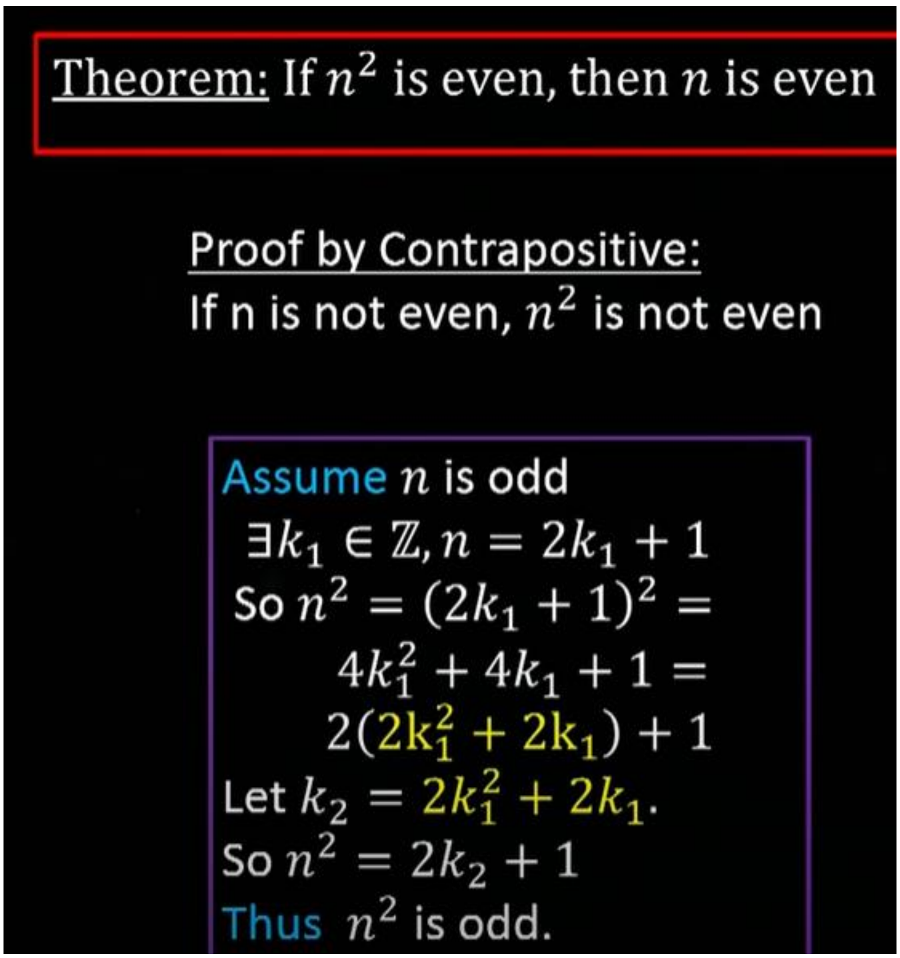
Induction proof:

- All elements of infinite set have a specific property
- 2 Steps
  - Base case

- Induction Step (Assume S(k), then S(k+1))
- Example: Σn = n(n+1)/2

Contraposition Proof:

- P=>Q is equivalent to ¬Q => ¬P
- Assume ¬Q is true, prove ¬P is true
- Example : If n is even, n^2 is even



Counter Example proof:

- Show an example to disprove the claim

Automata theory

- Study of abstract machines and computational problems that can be solved by these machines

Automata: abstract machine

Consists of

- States: Circle
- Transition: Arrow, input, one state to another

Basic Definition:

1. Symbols
    a. Symbols are indivisible objects or entity that cannot be defined.

2. Alphabets
    a. Finite set of symbols
    b. Σ

3. String
    a. Finite sequence of symbols
    b. Denoted by w,z,y,z

4. Empty String
    a. Denoted by ε
    b. The length of the empty string is 0

5. Length of a string
    a. Denoted by |W|

6. Power of Alphabets
    a. Set of string length k, Σ^k
        i. $\Sigma^0 = \{\varepsilon\}$

b. Set of all string including empty, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \ldots \Sigma^n$
c. Set of all String w/o empty string, $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \ldots \cup \Sigma^n$

7. Concatenation of a string
    a. x = 01, y=10 concatenation of x and y, xy = 0110, yx = 1001
8. Language
    a. A language over an alphabet is a set of strings over that alphabet.
    b. Set of all Σ^*
    c. Empty language Φ

3 requirements of automata:

- Taking input
- Producing output
- May have Temporary storage
- Control unit: can change state according to transition function

Finite automata

- No temporary storage
- Used to recognize pattern
- Accept or reject input depending on pattern

2 types

- DFA (Deterministic Finite Automata)
- NFA (Non-Deterministic Finite Automata)

DFA	NFA
one state transition in DFA	May have more than one
Cannot ε	Can use ε
Understand as one machine	Multiple machine
have max. one possible next state	May have multiple next possible states
Difficult to construct	Easier
Time less	Executing time more
All DFA = NFA	All NFA != DFA
$\delta: Q\times\Sigma \rightarrow Q$	$\delta: Q\times\Sigma \rightarrow 2^Q$

State:

- Description of the Status of system waiting to execute a transition
- Denoted by Circle/Vertex

Transition:

- set of actions to execute when a condition is fulfilled or an event received.
- Denoted by Arrow/Edge

DFA

`Deterministic finite automata (or DFA) are finite state machines that accept or reject strings of characters by parsing them through a sequence that is` **`uniquely determined by each string.`**

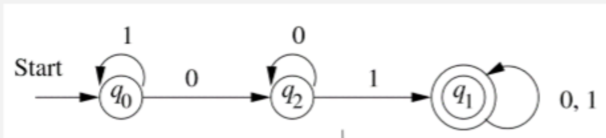
- A formalism for defining languages, consisting of:
  1. A finite set of *states* (Q, typically).
  2. A finite set of *input symbols* (Σ, typically).
  3. A *transition function* (δ, typically).
  4. A *start state* (q_0, one of the states in Q, typically).
     1. Only one start state
  5. A set of *final states* ($F \subseteq Q$, typically).
     - “Final” and “accepting” are synonyms.
     - May have multiple final states
- So, A DFA is a *five-tuple* notation:

  $A=(Q, \Sigma, \delta, q_0, F)$

where ***A*** is the name of the DFA.

Lecture 3:

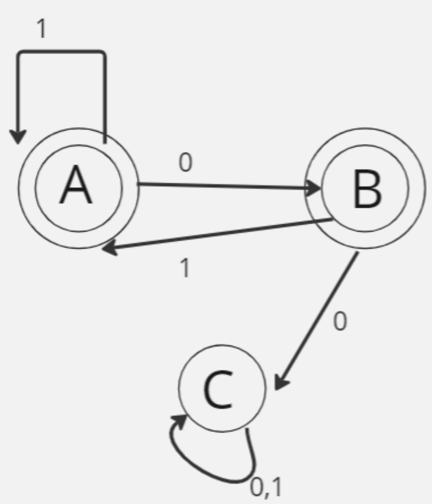
1. Show a transition table for the following diagram/automata:



→ q0	q2	q1
q2	q2	q1
*q1	q1	q1

2. Draw a transition diagram from the following table:

	0	1
→ [illegible]	[illegible]	[illegible]
$*B$	C	A
C	C	C



Extended Transition Functions

- Denoted by $\hat{\delta}$
- Takes state q and string w (where Transition function usually takes only alphabet)

A recursive algorithm is used to reach the final state, which is as follows:

1. Base condition:

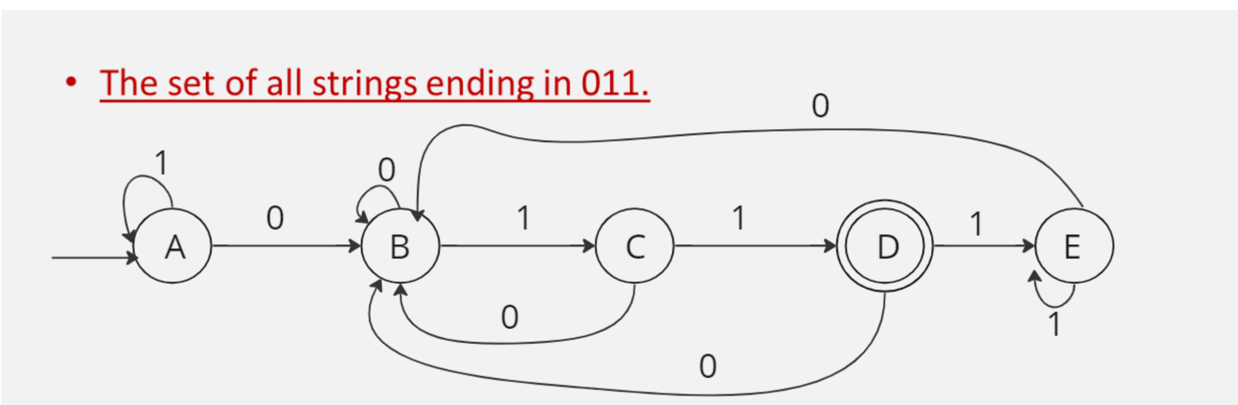
$$\hat{\delta}(q, \epsilon) \rightarrow q$$

2. Recursion rule:

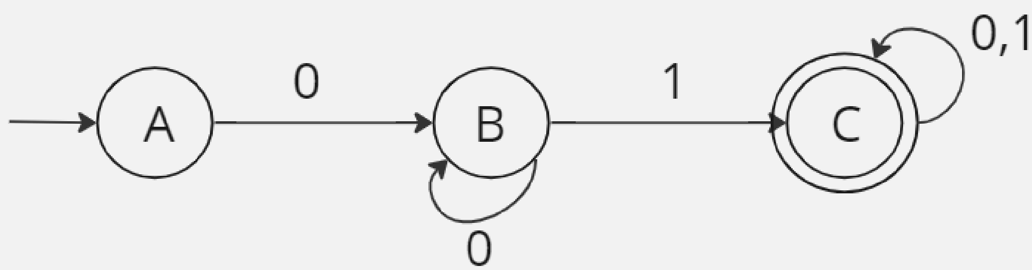
$$\hat{\delta}(q, xa) \rightarrow \delta(\hat{\delta}(q, x), a)$$

Here, $x \in \Sigma^* \; and \; a \in \Sigma$. Also,$x$ is a string of characters belonging to the set of the input symbols and a is a single character.

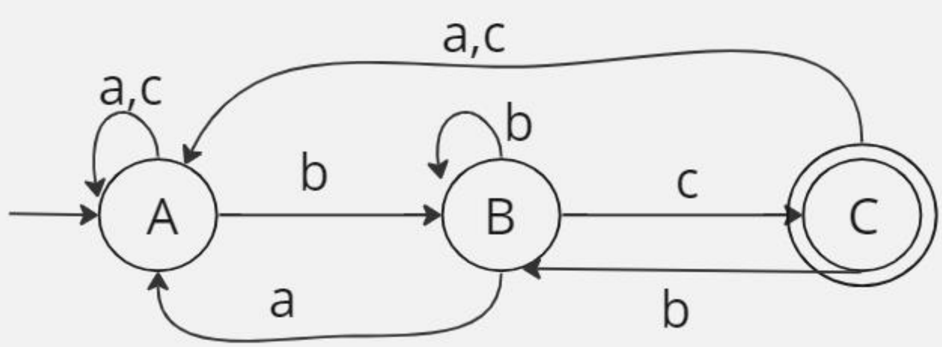
Lecture 4:



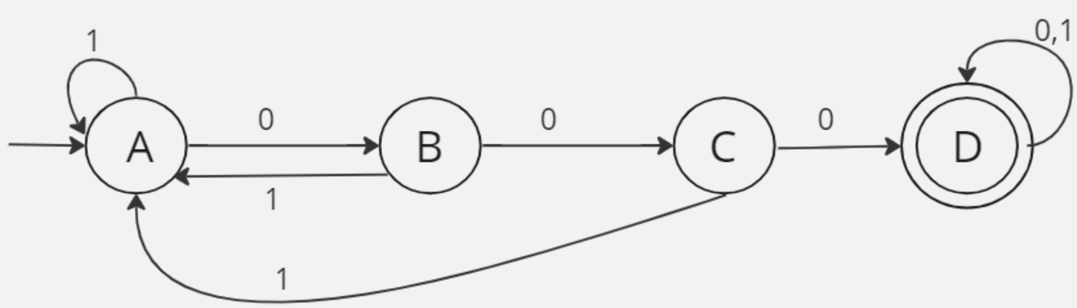
- The set of all string with 01 as a substring.



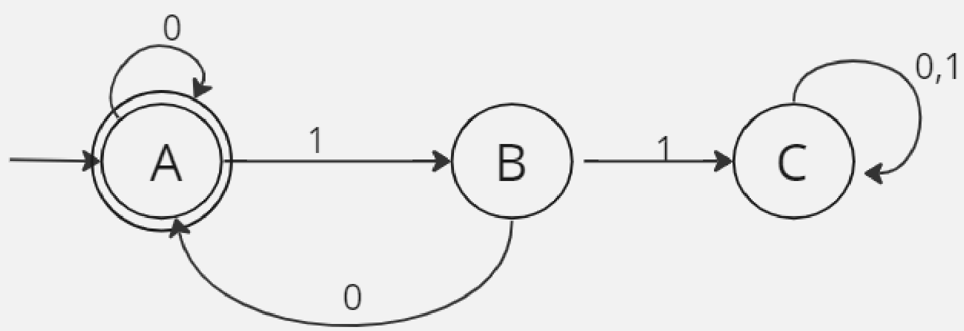
- Draw a DFA for the following language over $\Sigma = \{a,b,c\}$
  - The set of all strings ending in bc.



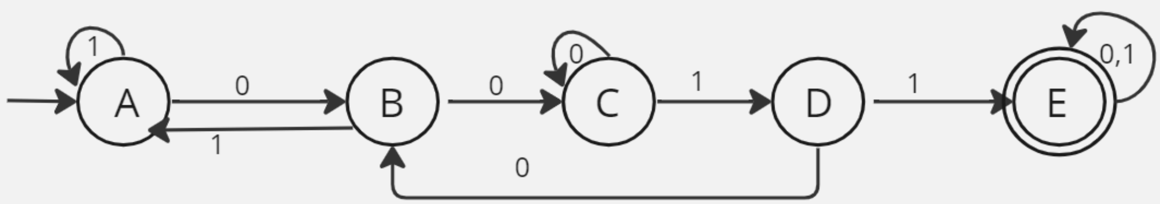
- The set of all strings with three consecutive 0's (not necessarily at the end).



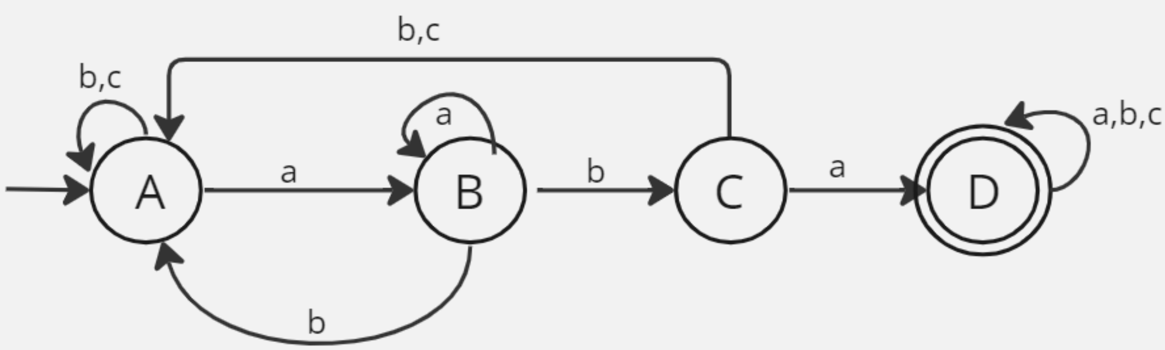
- <u>The set of all strings with not having two consecutive 1’s (not necessarily at end).</u>



- <u>The set of strings containing two consecutive zero’s followed by two consecutive ones.</u>



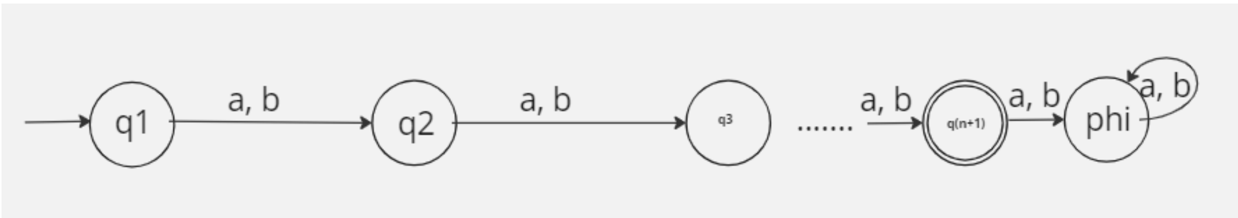
- <u>The set of all strings with aba is a substring.</u>



Lecture : 5

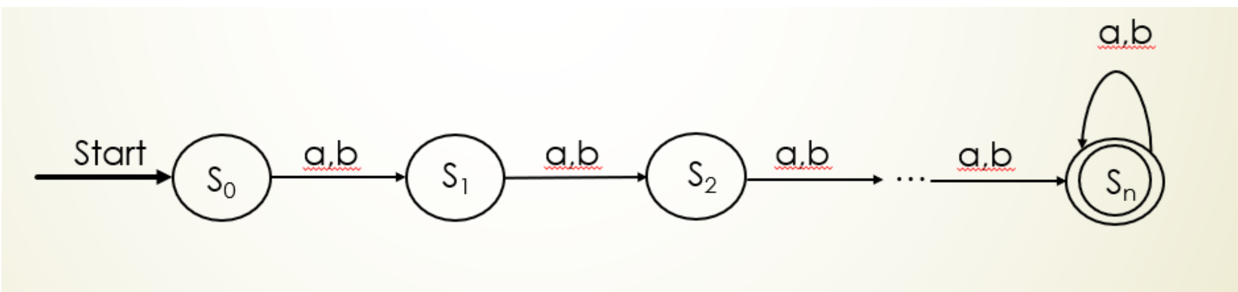
DFA with exactly n alphabets

i. Number of states: n+2



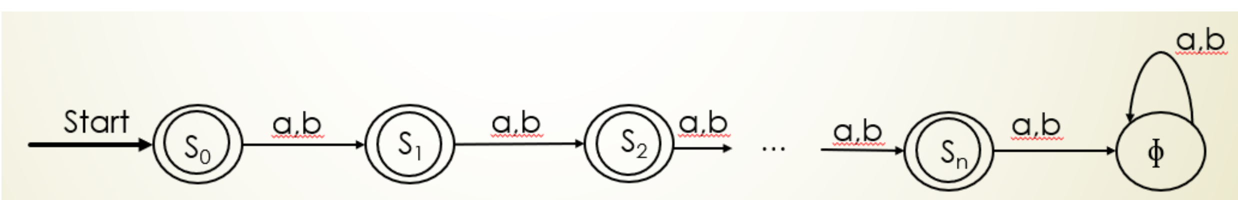
DFA with at least n alphabets

i. Number of states: n+1

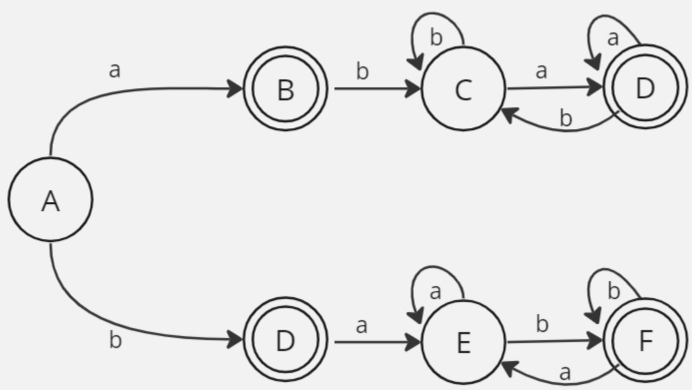


DFA with at most n alphabets

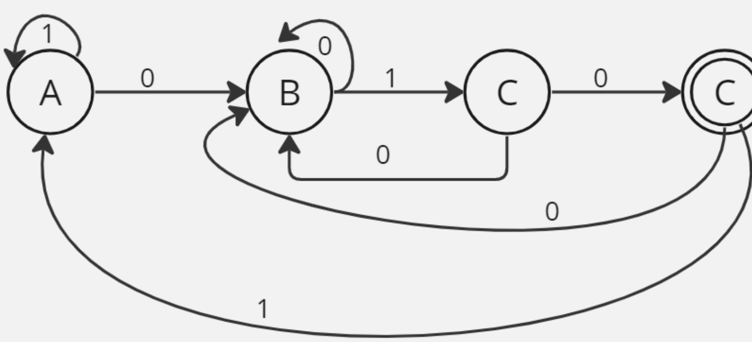
i. Number of states: n+2



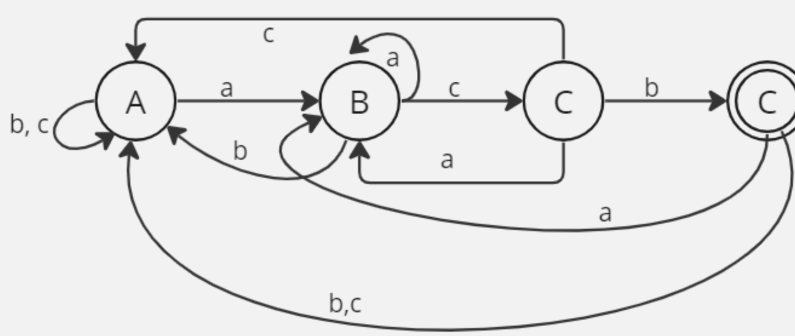
- Draw a DFA that accepts equal number of **ab** and **ba** over Σ= {a,b}.



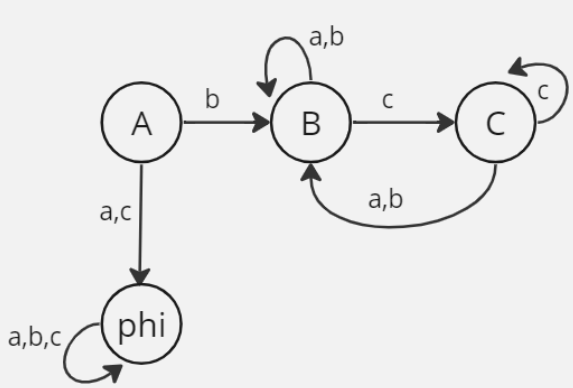
- Draw a DFA for the set of all strings ending in **010** over Σ= {0,1}.



- Draw a DFA for the set of all strings ending in **acb** over Σ= {a,b,c}.



- Draw a DFA for the set of all strings starting with **b** and ending with **c** over Σ= {a,b,c}.



Minimizing DFA

- Equivalence Theorem

- Using equivalence theorem, we have
  - 0 equivalence

    P_0={A,D,E} {B,C,G}
  - 1 equivalence

    P_1= {A,D,E} {B,C}{G}
  - 2 equivalence

    P_2={A} {D,E} {B,C} {G}
  - 3 equivalence

    P_3={A} {D,E} {B,C} {G}
  - Since P_3=P_2, we stop.

	0	1
→A	B	C
*B	D	E
*C	E	D
D	G	G
E	G	G
*G	G	G

	0	1
→{A}	{B,C}	{B,C}
{D,E}	{G}	{G}
*{B,C}	{D,E}	{D,E}
*{G}	{G}	{G}

Table Filling Method / Myhill-Nerode Theorem

Steps:

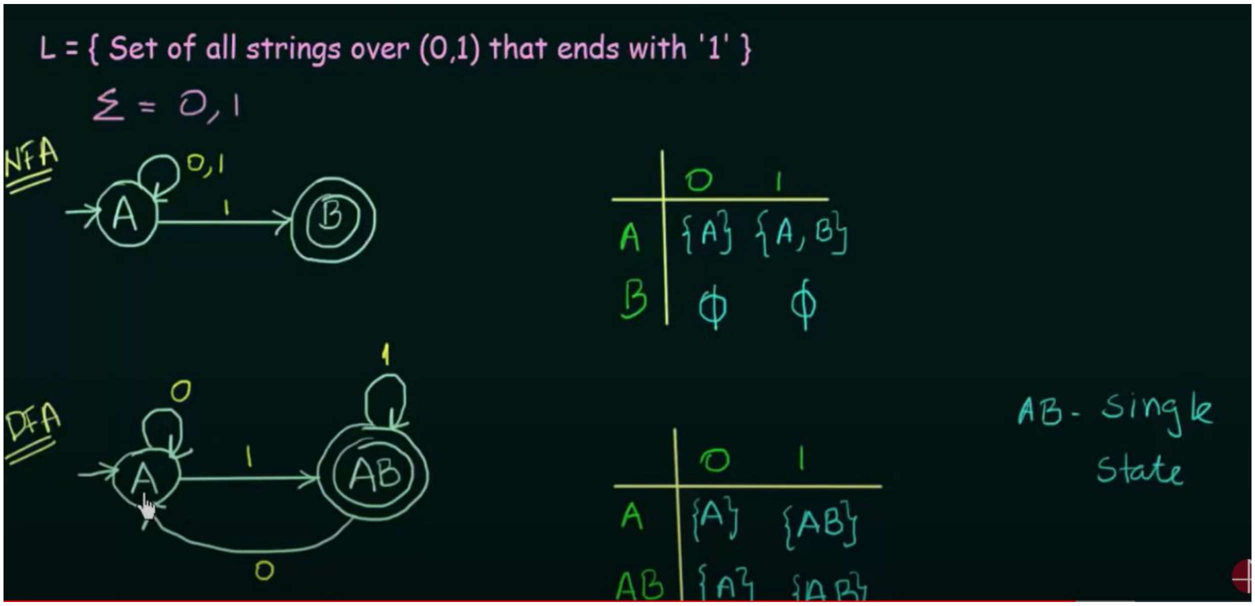
1) Draw a table for all pairs of states (P,Q)
2) Mark all pairs where P∈ F and Q∉F
3) If there are any Unmarked pairs (P,Q) such that [δ(P,x), δ(Q,x)] is marked, then mark [P,Q] where 'x' is an input symbol

   REPEAT THIS UNTIL NO MORE MARKINGS CAN BE MADE
4) Combine all the Unmarked Pairs and make them a single state in the minimized DFA

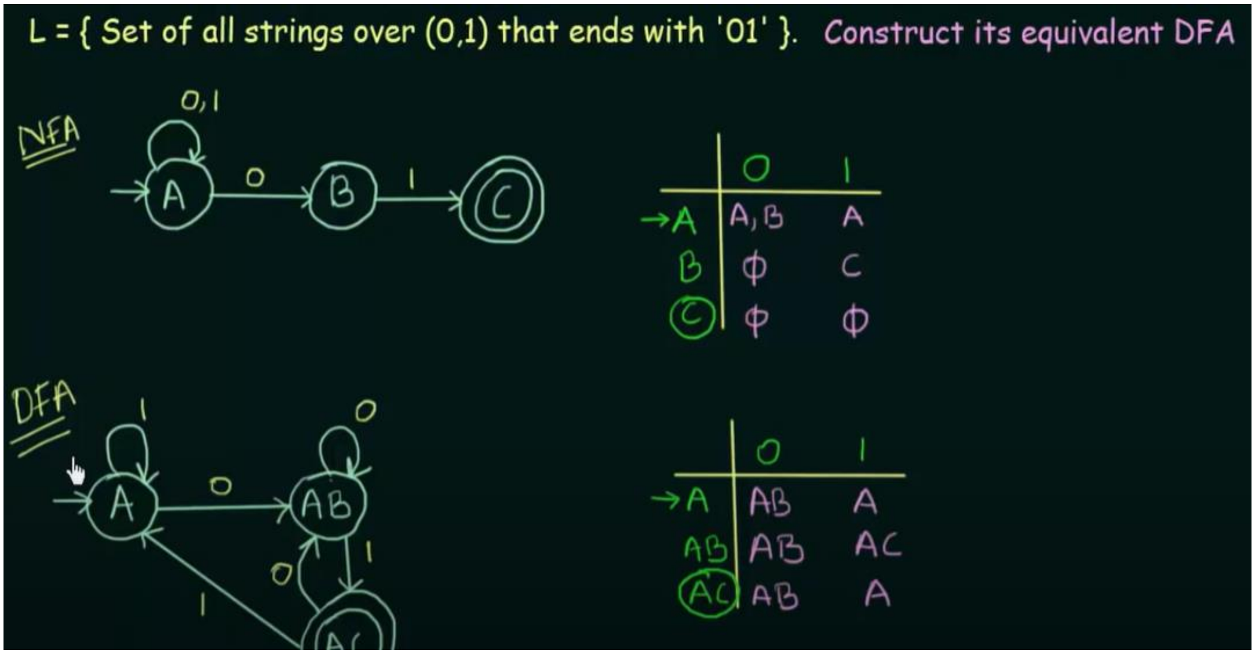
NFA

- Non-deterministic Finite Automata
- Transition from a state on input symbol can be to any set of states
- DFA: Q x Σ→ Q
- NFA: Q x Σ→ 2^Q
- No need to add dead state/trap state
- Input can be empty

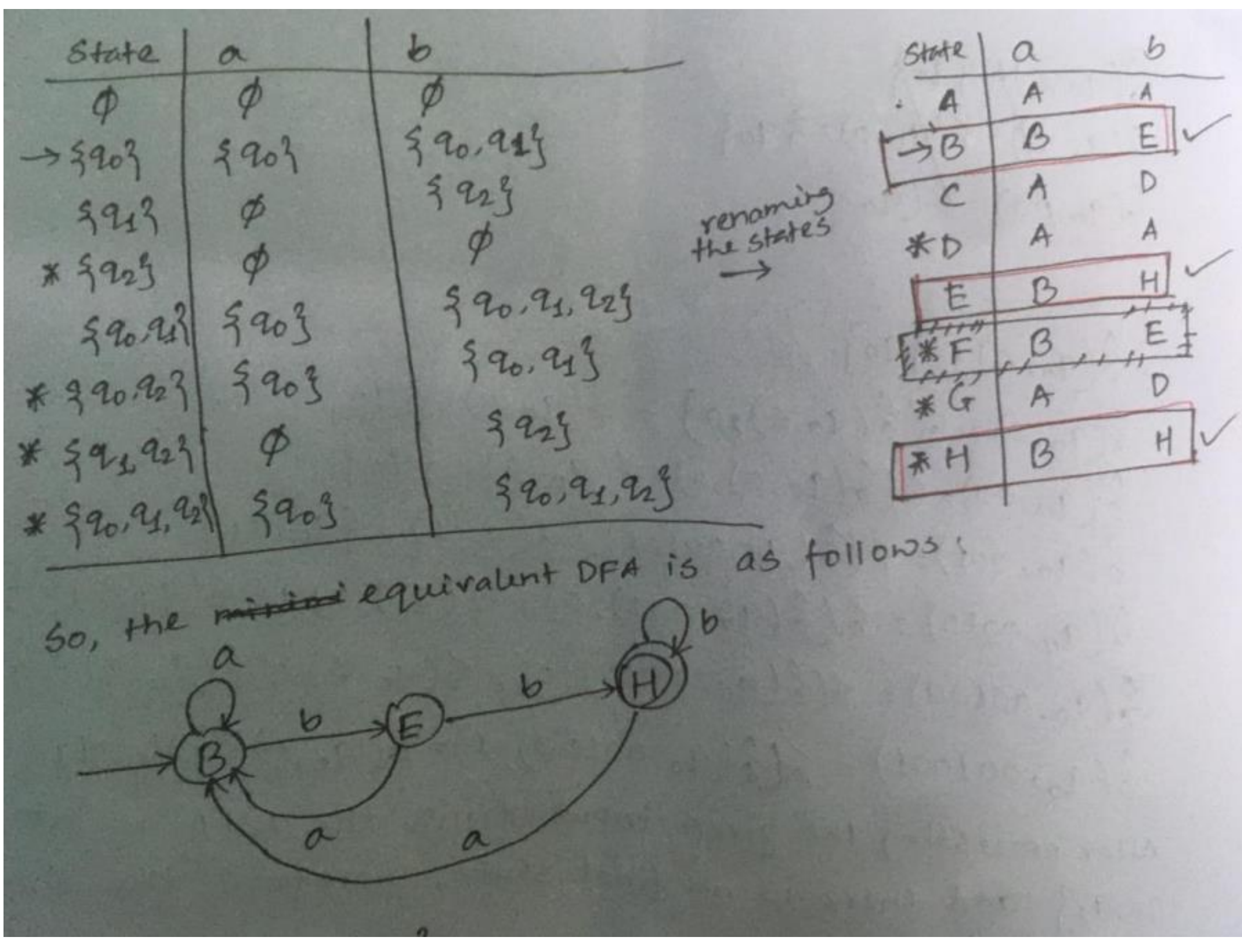
Conversion of NFA to DFA:



If there are any empty state/phi in NFA, a dead state should be added for that in NFA.



Set Construction Method:



ε-transitions

- NFA allows go to next state w/o any input
- ε means empty

Formal Notation for an ε-NFA

- **Definition:** an ε-NFA A is denoted by $A = (Q, \Sigma, \delta, q_0, F)$ where the transition function δ takes as arguments:
  - a state in Q, and
  - a member of $\Sigma \cup \{\varepsilon\}$

Epsilon-Closure

- Denoted by ε^*
- All the states than be reached from particular state only by seeing the ε symbol

Epsilon-NFA to NFA:

- If a state (let x) goes to final state only by seeing ε in E-NFA, then the state (x) will be a final state in the NFA