Theory of Computation

**ASH2101008M**

NB: There are many mistakes. **Read at your own risk.**

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: deals with various definition and properties of mathematical model
* Computability theory: what can and cannot be computed
* Computational Complexity theory: it groups computable problem based on hardness

Model of Computation: mathematical abstraction of Computer

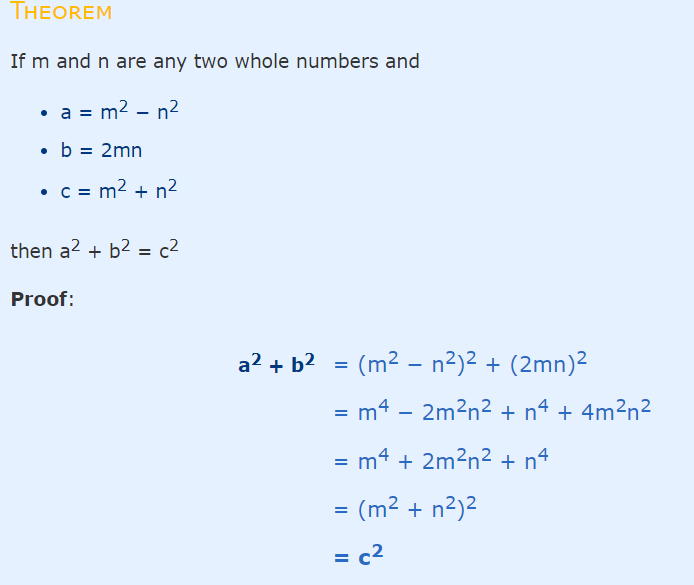
*Definition:* describe a object and notations.

*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.



*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

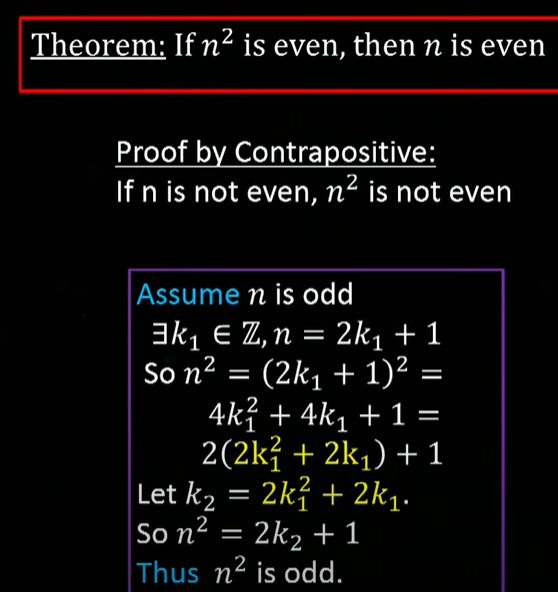
Example: sqrt (2) is irrational.

*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, n2 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/computing device
* mathematical model of a system that involves with input, output, state, transition etc.

Consists of

* States: Circle, description of the status of the system
* Transition: Arrow, input, one state to another

Basic Definition:

1. Symbols
   1. Symbols are indivisible objects or entity that cannot be defined.
2. Alphabets
   1. Finite set of symbols
   2. Σ
3. String
   1. Finite sequence of symbols
   2. Denoted by w,z,y,z
4. Empty String
   1. Denoted by ε
   2. The length of the empty string is 0
5. Length of a string
   1. Denoted by |W|
6. Power of Alphabets
   1. Set of string length k, Σk
      1. Σ0 = {ε}
   2. Set of all string including empty, Σ\* = Σ0 U Σ1 U ... Σn
   3. Set of all String w/o empty string, Σ+ = Σ1 U Σ2 .... U Σn
7. Concatenation of a string
   1. x = 01, y=10 concatenation of x and y, xy = 0110, yx = 1001
8. Language
   1. A language over an alphabet is a set of strings over that alphabet.
   2. Set of all Σ\*
   3. Empty language Φ

Membership Problem:

* Given a word and a Language, we want to check word belongs to the language or not, this is called membership problem.

3 requirements of automata:

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

Finite automata

* An abstract computing device
* 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 for one input | May have multiple next possible states for one input |
| Difficult to construct | Easier |
| Time less | Executing time more |
| All DFA = NFA | All NFA != DFA |
| δ: QxΣ -> Q | δ: QxΣ -> 2^Q |

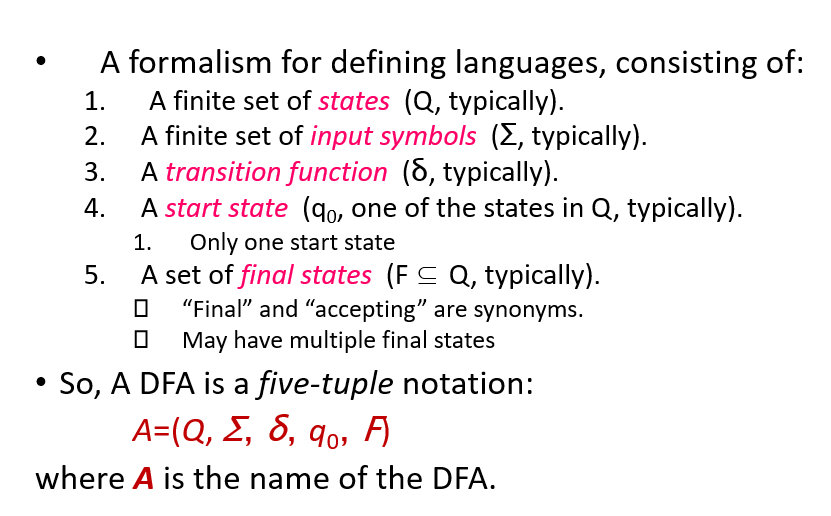
State:

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

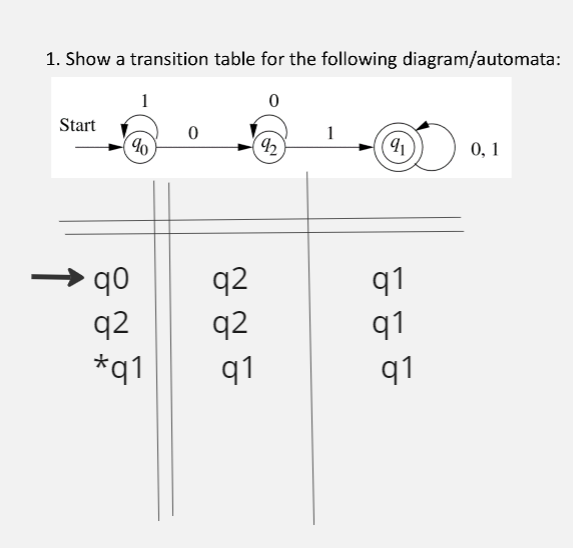
Transition:

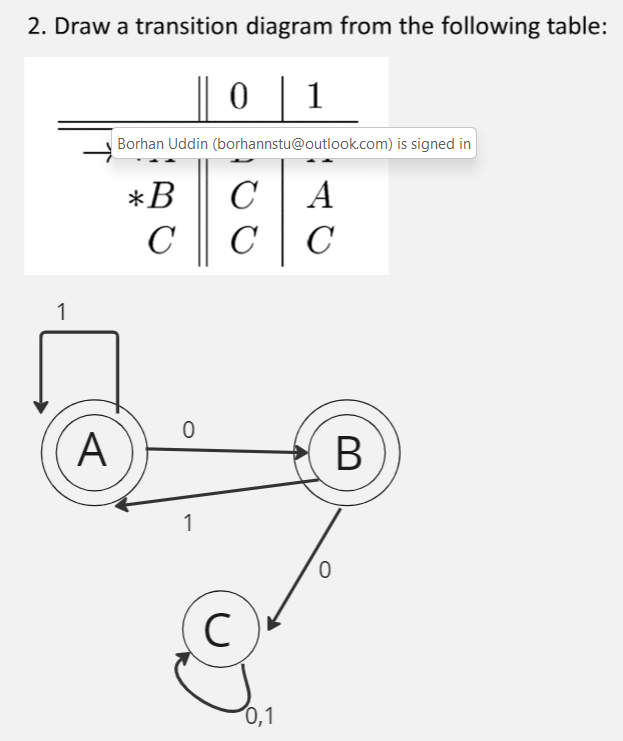
* set of actions to execute when a condition is fulfilled or an event received.
* Denoted by Arrow/Edge

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.**

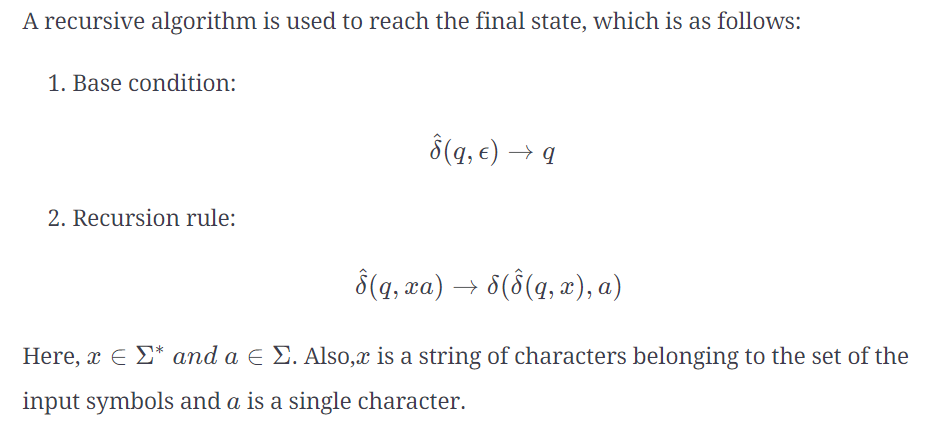
From Nazia ma’am’s Slide

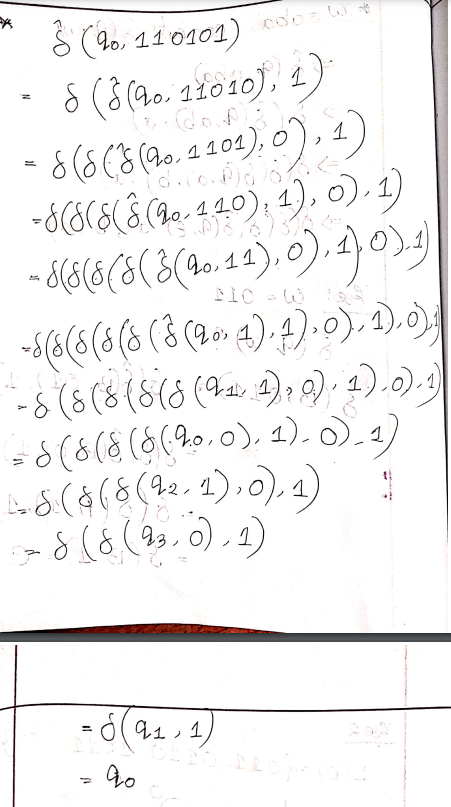
Lecture 3:



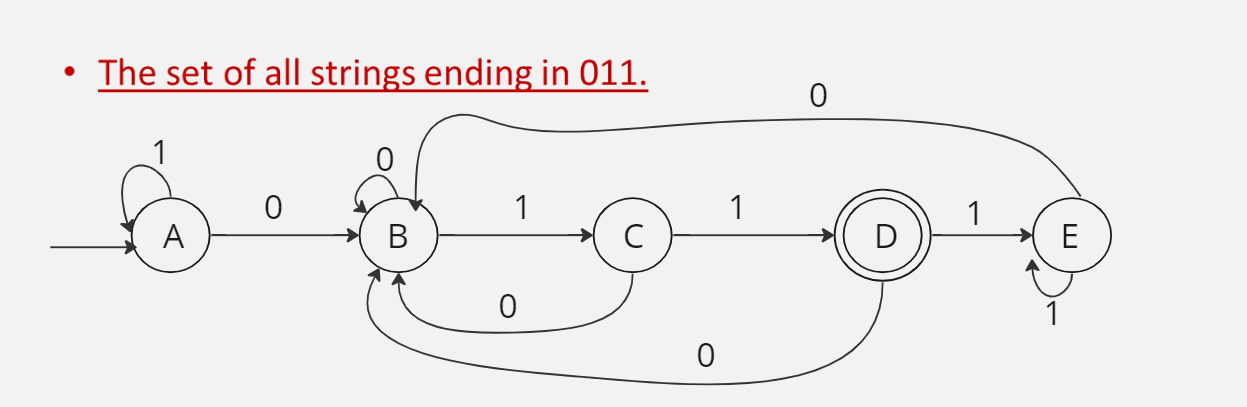
Extended Transition Functions

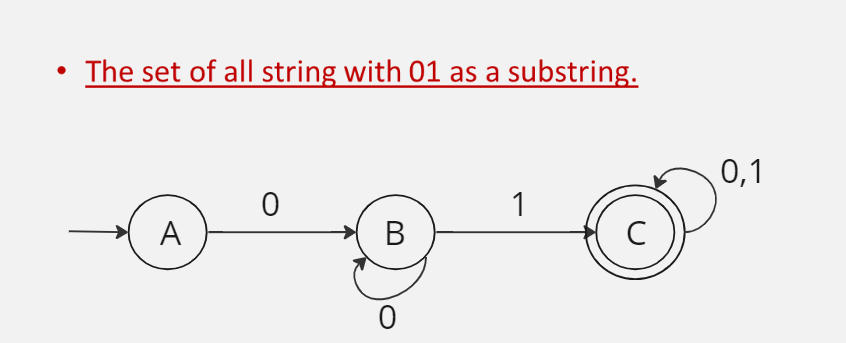
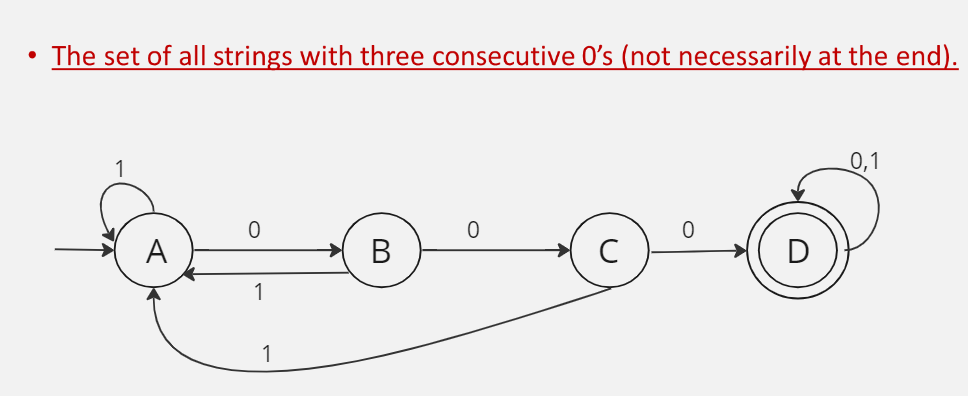
* Denoted by
* Takes state q and string w (where Transition function usually takes only alphabet)
* Induction steps

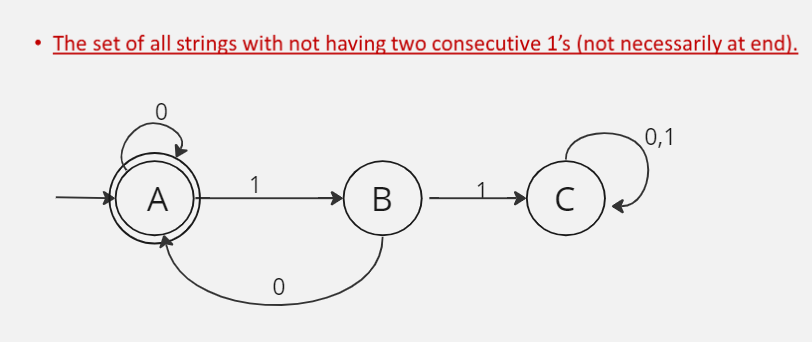


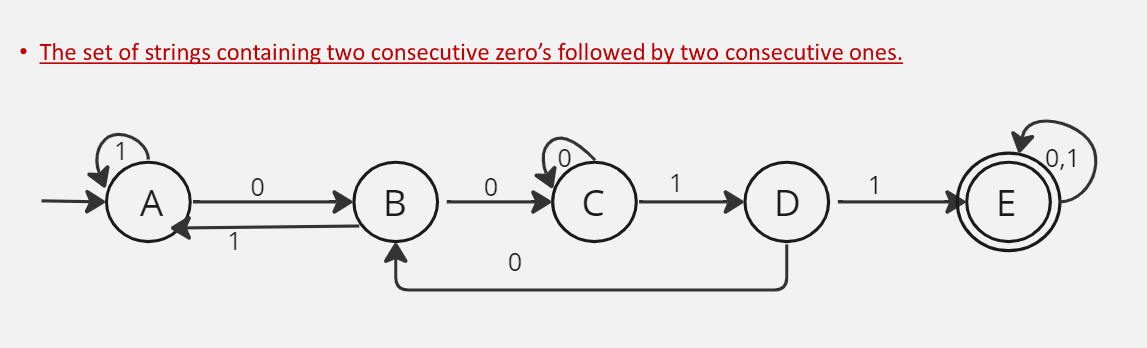


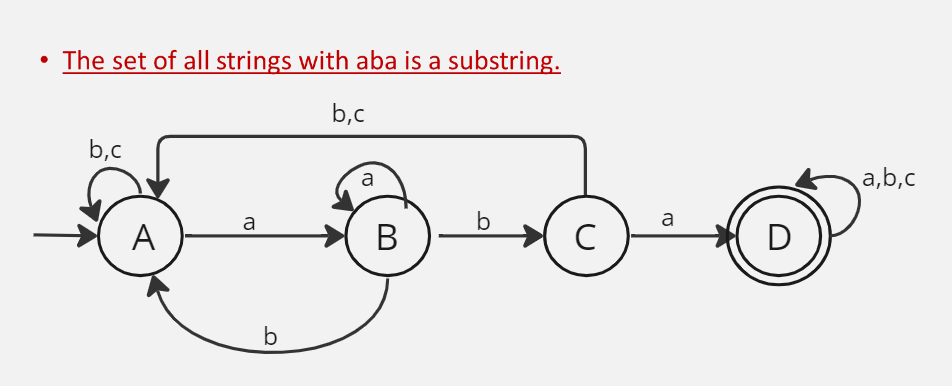
From : Pollock bhai

Lecture 4: 



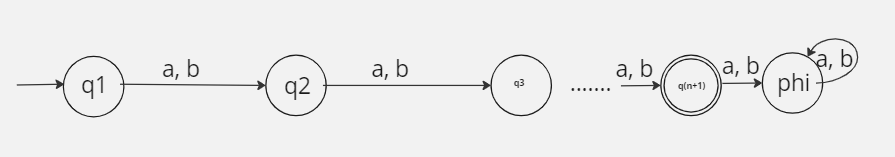




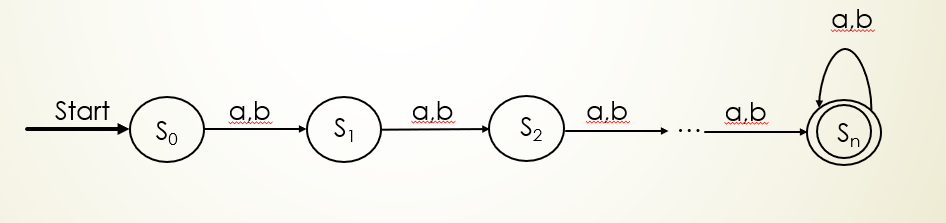


Lecture: 5

DFA with exactly alphabets

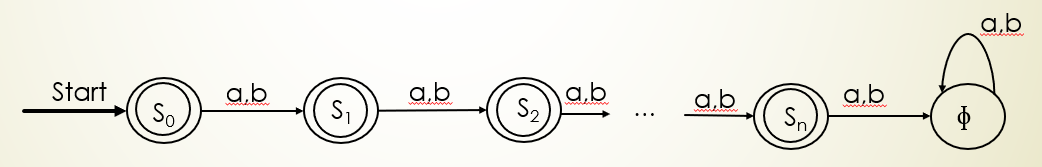
* + 1. Number of states: n+2

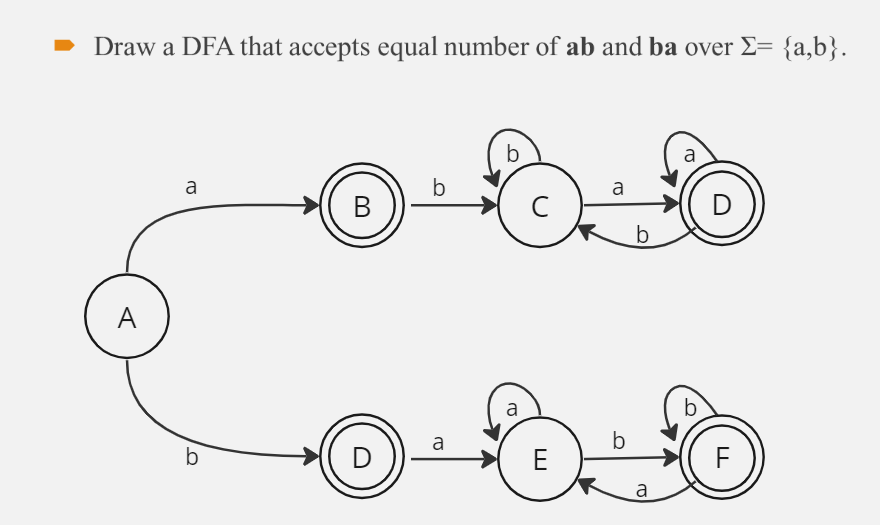
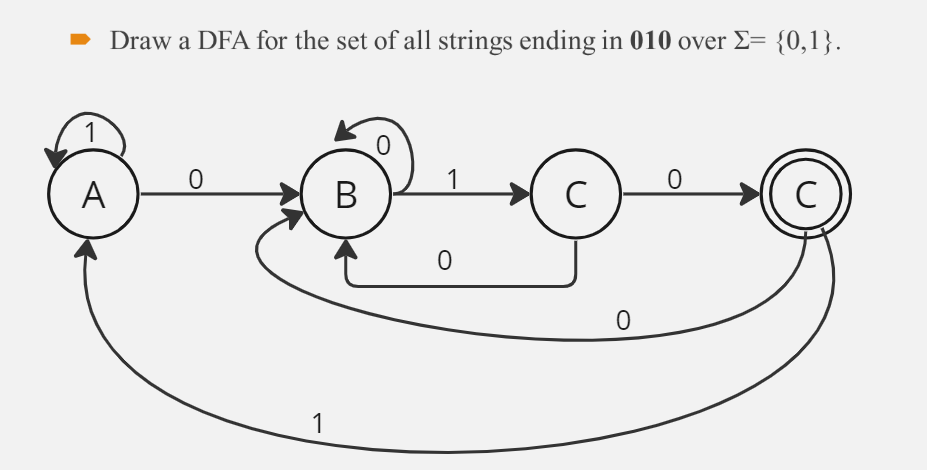
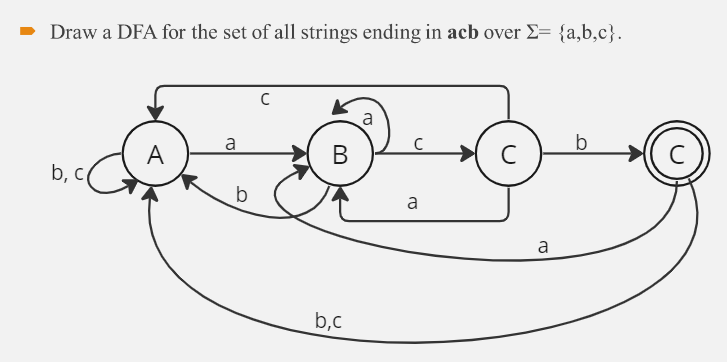
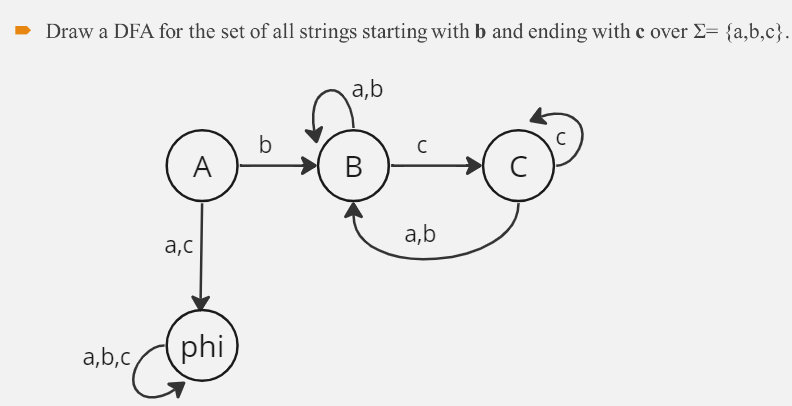
DFA with at least n alphabets

1. Number of states: n+1

DFA with at most n alphabets

1. Number of states: n+2





Minimizing DFA

* Equivalence Theorem

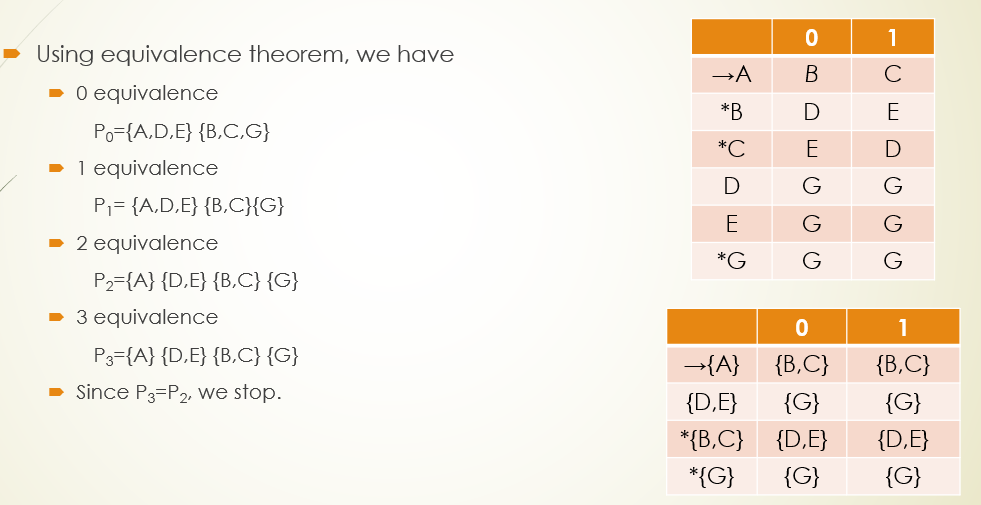
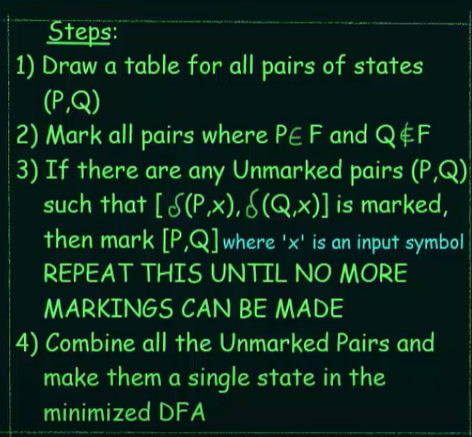


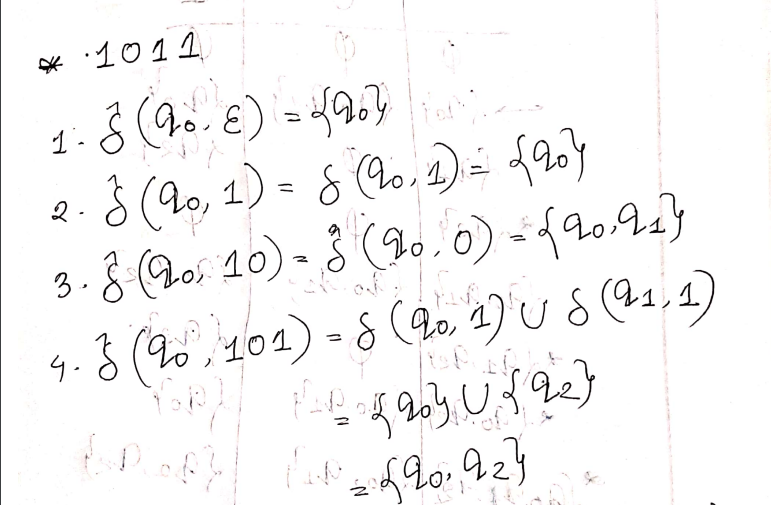
Table Filling Method / Myhill-Nerode Theorem



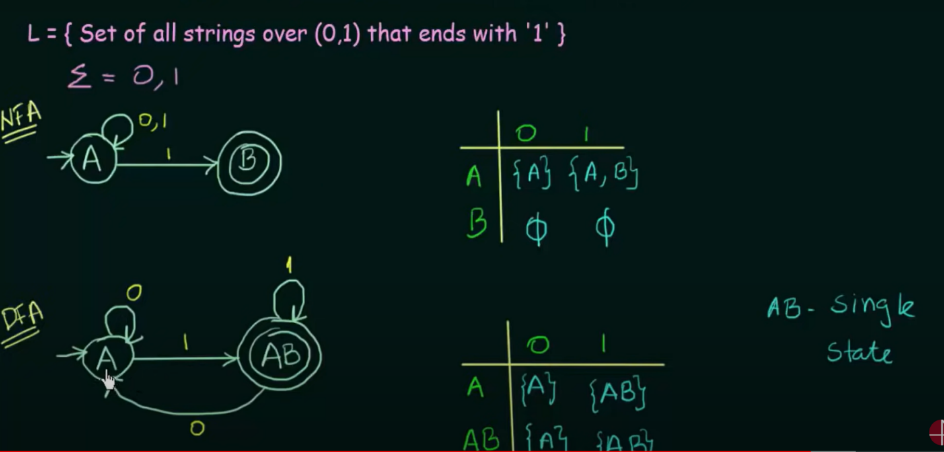
NFA

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

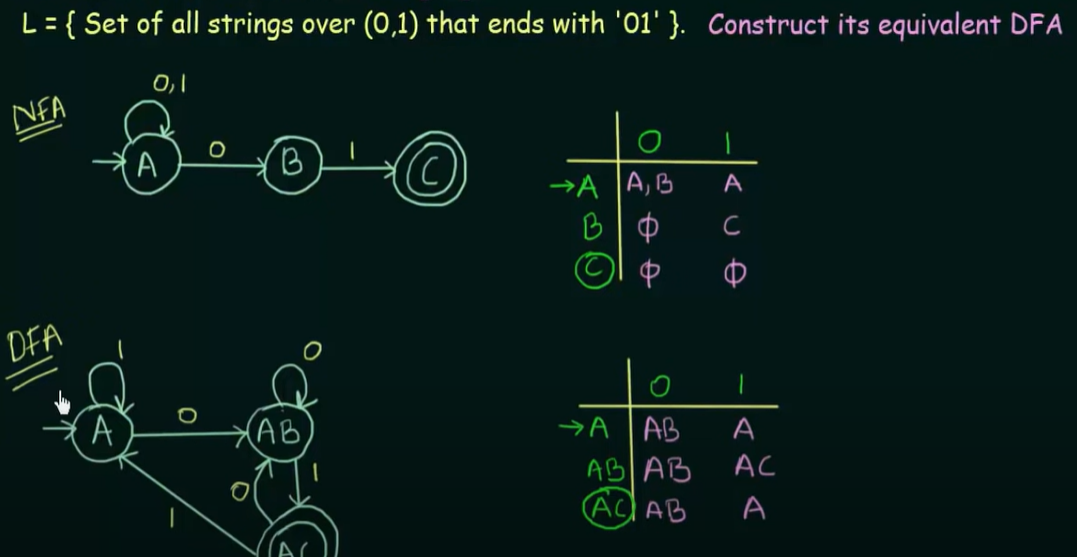
Extended Transition for NFA

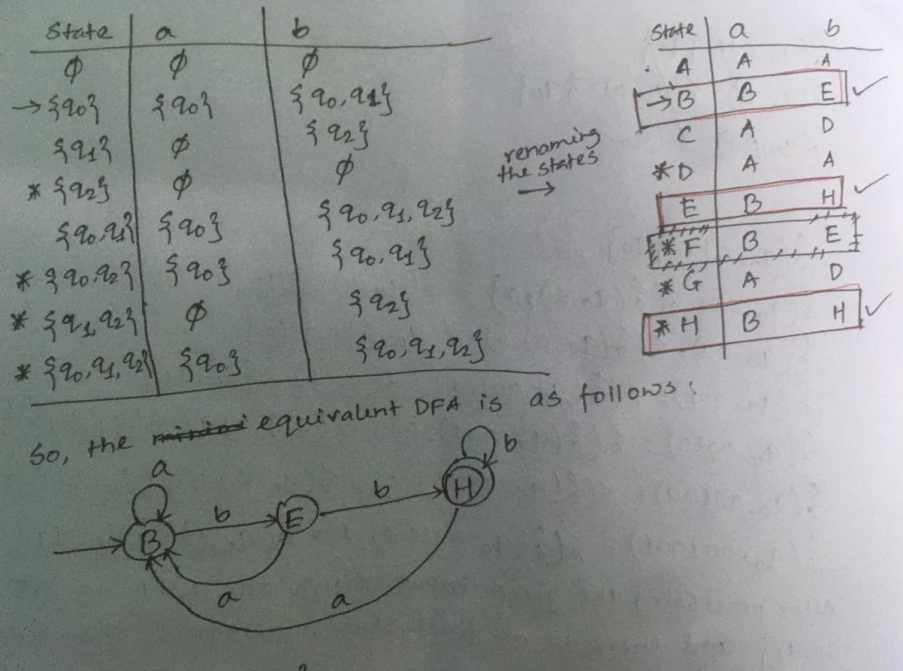


From: Pollock bhai

Conversion of NFA to DFA:

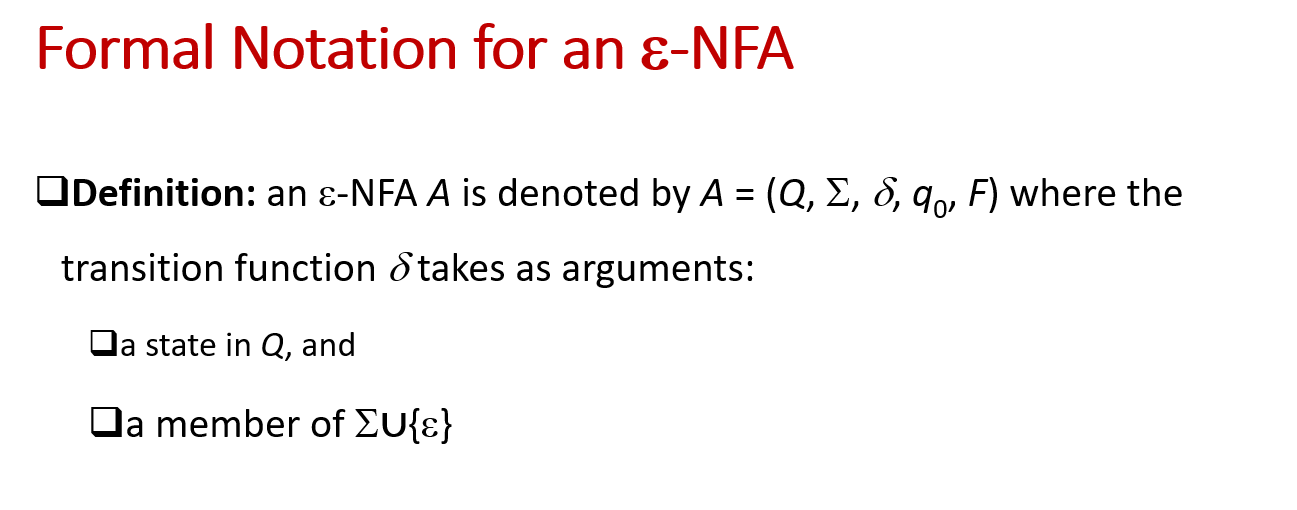
If there are any empty state/phi in NFA, a dead state should be added for that in NFA.



Set Construction Method:

**ε-transitions**

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

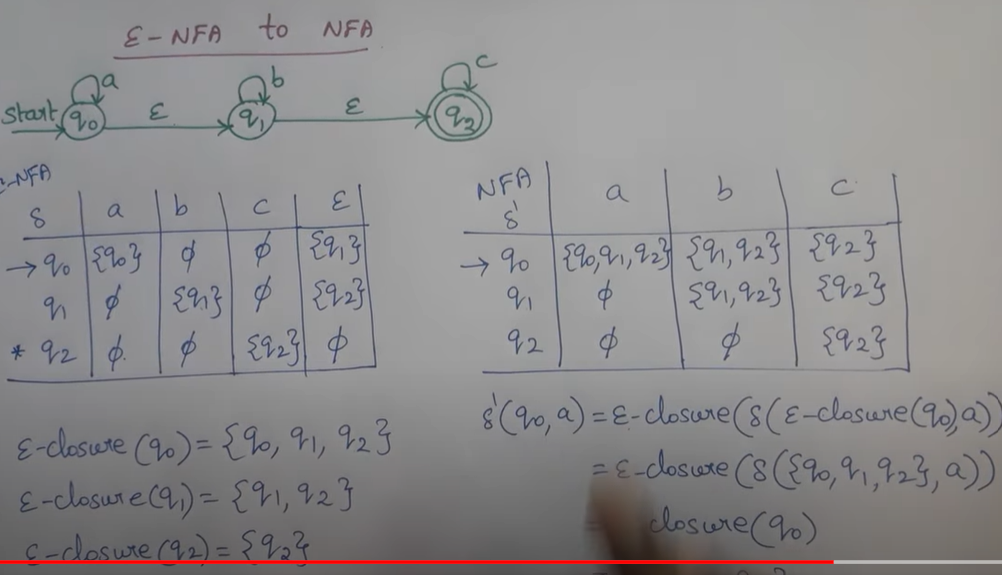


Epsilon-Closure

* 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
* Using epsilon-closure

Epsilon-NFA to NFA

Epsilon NFA to DFA

