|| Jai Sri Gurudev||

Sri Adichunchanagiri Shikshana Trust®

SJB INSTITUTE OF TECHNOLOGY

No. 67, BGS Health & Education City, Dr. Vishnuvardhan Road, Kengeri, Bengaluru -560060

Accredited by NAAC, Accredited by NBA. Certified by ISO 9001-2015

Department of Computer Science & Engineering

Subject: Automata Theory & Computability

Subject Code: 18CS54

Sem: V

18CS54
Automata Theory & Computability

Module-1 Review of Mathematical Theory

Dr. Gopalakrishna M T,
Professor, CSE

gopalakrishnamt@sjbit.edu.in

ll Jai Sri Gurudev ll

S J B INSTITUTE OF TECHNOLOGY

Topics to be covered

- Introduction
- Mathematical Preliminaries & Terminology
- Languages
- Strings

Introduction

Computer Science stems from two starting points:

Mathematics: What can be computed?

And what **cannot be computed?**

Electrical Engineering: How can we build computers?

Not in this course.

Introduction

Computability Theory deals with the profound mathematical basis for Computer Science, yet it has some interesting practical ramifications that I will try to point out sometimes.

The question we will try to answer in this course is:

“What can be computed? What Cannot be computed and where is the line between the two?”

Computational Models

A **Computational Model** is a mathematical object (Defined on paper) that enables us to reason about computation and to study the properties and limitations of computing.

We will deal with Three principal computational models in increasing order of **Computational Power.**

Computational Models

We will deal with three principal models of computations:

1. Finite Automaton (in short FA).
   recognizes Regular Languages .
2. Stack Automaton.
   recognizes Context Free Languages .
3. Turing Machines (in short TM).
   recognizes Computable Languages .

Formal Language and Automata Theory

Formal Language and Automata Theory

What is Automata Theory ?

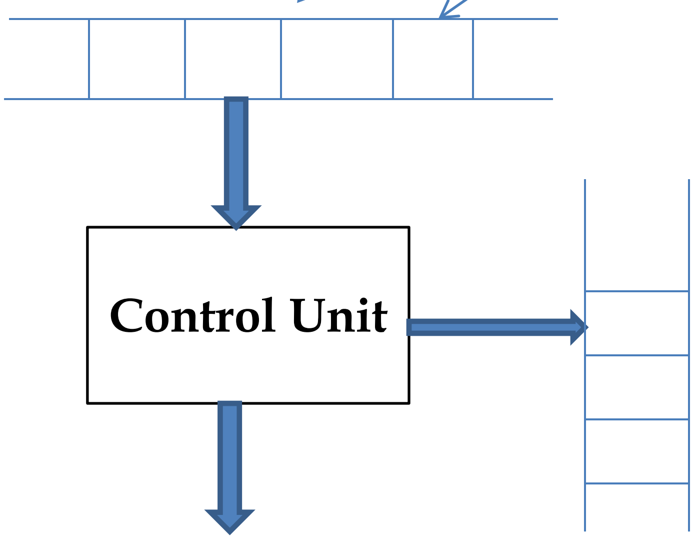
Study of Abstract Machines

Machine Which are not implemented but represented by Using some **Formal Notations**

It is also Basis for the theory of **Formal language.**

Invented by "**ALAN TURING**" in (**1912-1954**)

Input tape



Temporary Storage

Output

- An Automata is a Abstract Model of a Digital Computer, which operates in **discrete time frame.**
- The Automata reads the input, produce the output depending on the state it is in and can make decision in transferring the input into the output.

Different Kinds of Automata

Finite Automata (FA) < **Pushdown Automata(PDA)** < **Turing Machine(TM)**

Power of Automata's

What is a Finite Automata (FA)

Branch of Automata Theory

It is Mathematical Model of a Machine

Used to represent Behavioural Model of Machines

Input

String

Finite Automata

"Accept or Reject"

- Finite Automata consists of **finite set of states** and transitions from one state to another state, that occurs on input symbols chosen from an **input alphabets. OR**
- **Formal Definition FA**
  Defined by **5 Tuples** which is denoted by M.

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

where

Q is the finite set of states

Σ is the alphabet

$\delta : Q \times \Sigma \to Q$ is the transition function

$q_0 \in Q$ is the start state

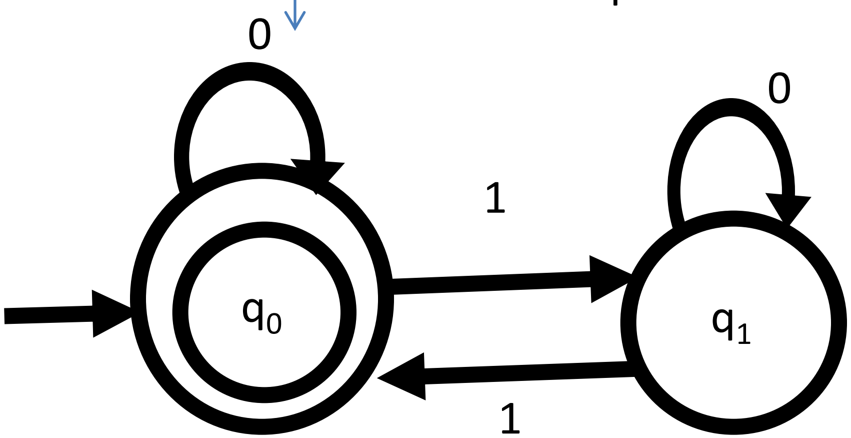
$F \subseteq Q$ is the set of accept states

How do you represent the FA?

Represented by Two Ways

Transition Diagram

Directed Graph

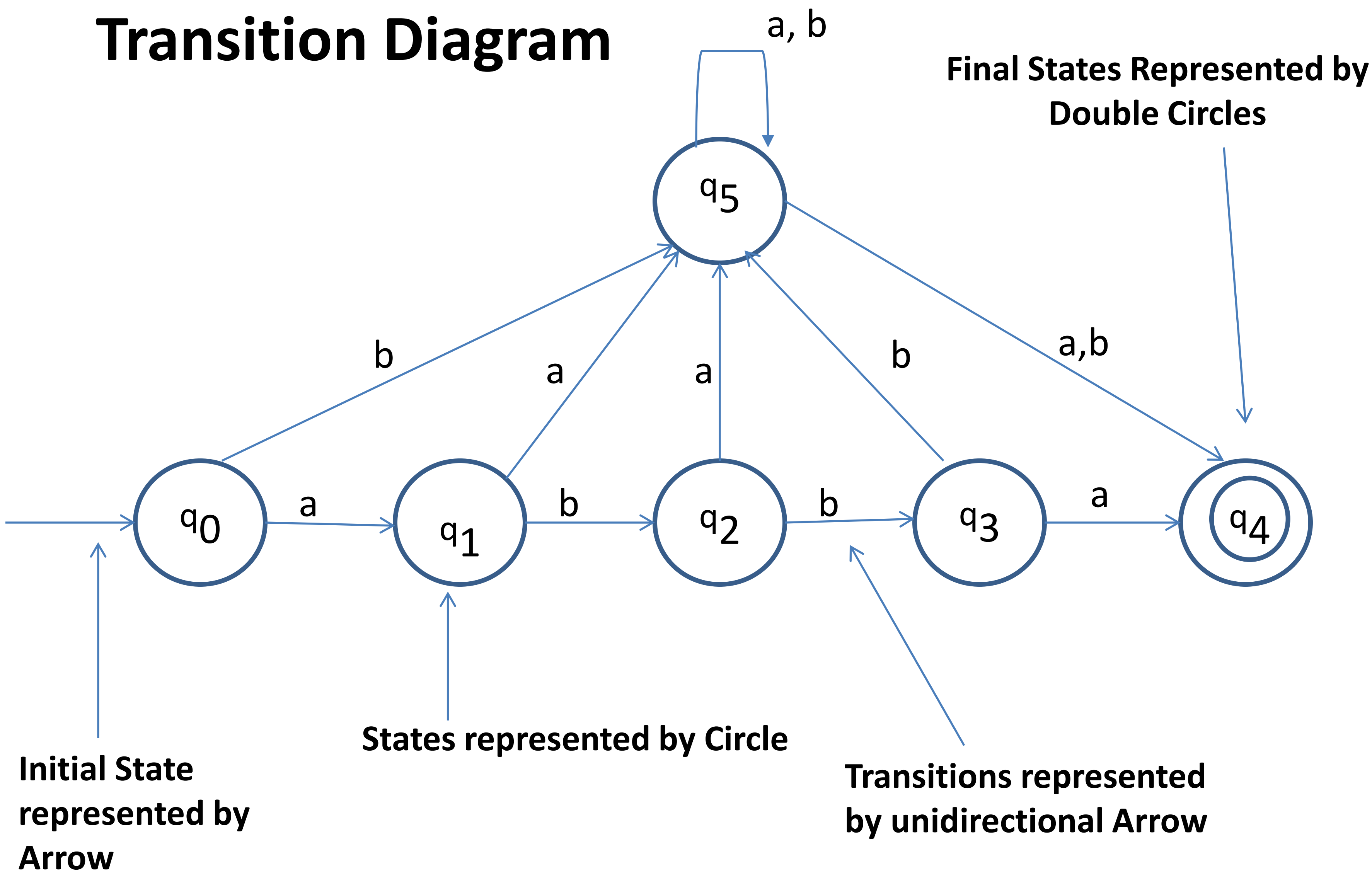


{ w | w has an even number of 1s}

Transitional Table

δ	**0**	**1**
$\mathbf{q_0}$	$\mathbf{q_0}$	$\mathbf{q_1}$
$\mathbf{q_1}$	$\mathbf{q_1}$	$\mathbf{q_0}$

Transition Diagram



Some Mathematical Preliminaries

Set

- A set is a collection of objects. 
- The objects in a set are called elements of the set.
- Examples:
  1. $A = \{11, 12, 21, 22\}$
  2. $B = \{11, 12, 21, 11, 12, 22)$

     Roster Notation

  3. $C = \{x \mid x \text{ is odd integer greater than } 1\}$
  4. $D = \{x \mid x \in B \text{ and } x \leq 11\}$

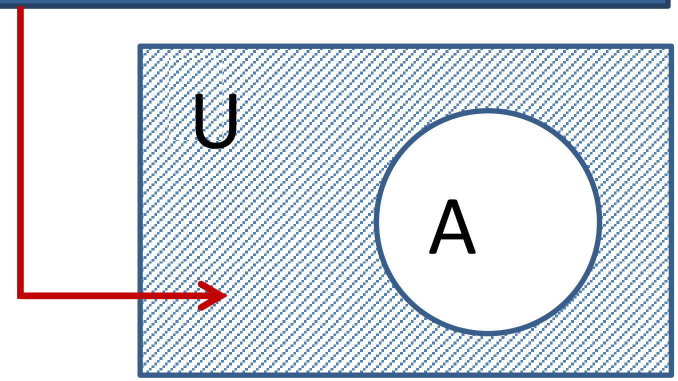
     Set-builder Notation

Operations on Sets

- Operations on the sets are:
  1. Complement
  2. Union
  3. Intersection
  4. Set Difference
  5. Symmetric Difference
  6. Cartesian product

- The complement of a set A is the set A' of everything that is not an element of A from Universal Set U.

$$A' = \{x \in U \mid x \notin A\}$$



- Example:

  U = {1,2,3,4,5}

  A = {1,2}

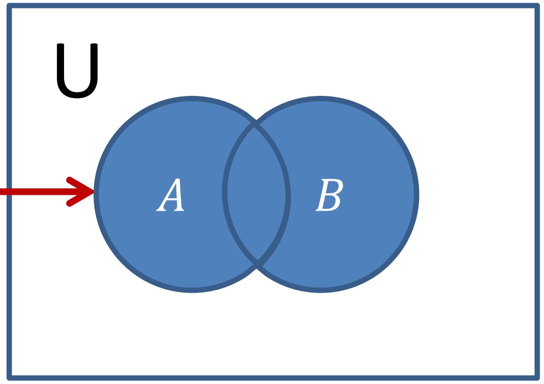
  A′ = {3,4,5}

Operations on Sets

- Operations on the sets are:
  1. Complement
  2. Union
  3. Intersection
  4. Set Difference
  5. Symmetric Difference
  6. Cartesian product

- The Union $(A \cup B)$ is a collection of all distinct elements from both the set A and B.

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$



- Example:

  A = {1, 3, 5, 7, 9}

  B = {1, 2, 3, 4, 5}

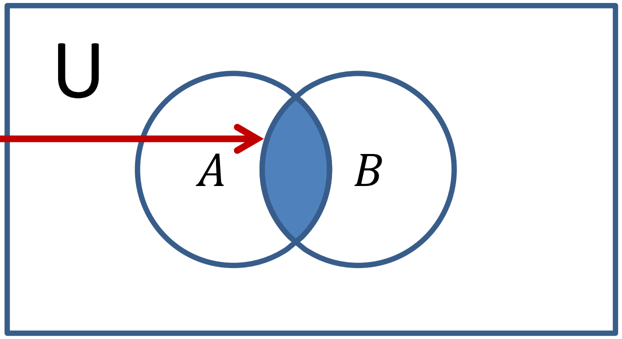
  A U B = {1, 2, 3, 4, 5, 7, 9}

Operations on Sets

- Operations on the sets are:
  1. Complement
  2. Union
  3. Intersection
  4. Set Difference
  5. Symmetric Difference
  6. Cartesian product

- The intersection $A \cap B$ of two sets A and B is the set that contains all elements of A that also belong to B, but no other elements.

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$



- Example:

  A = {1, 3, 5, 7, 9}

  B = {1, 2, 3, 4, 5}

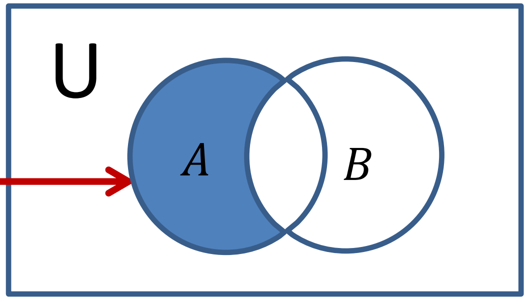
  $A \cap B = \{1, 3, 5\}$

Operations on Sets

- Operations on the sets are:
  1. Complement
  2. Union
  3. Intersection
  4. Set Difference
  5. Symmetric Difference
  6. Cartesian product

- The set difference *A* - *B* of two sets *A* and *B* is the set of everything in *A* but not in *B*.

$$A - B = \{x \mid x \in A \text{ and } x \notin B\}$$
$$= \{x \mid x \in A\} \cap \{x \mid x \notin B\}$$
$$= A \cap B'$$



- Example:

  A = {1, 3, 5, 7, 9}

  B = {1, 2, 3, 4, 5}

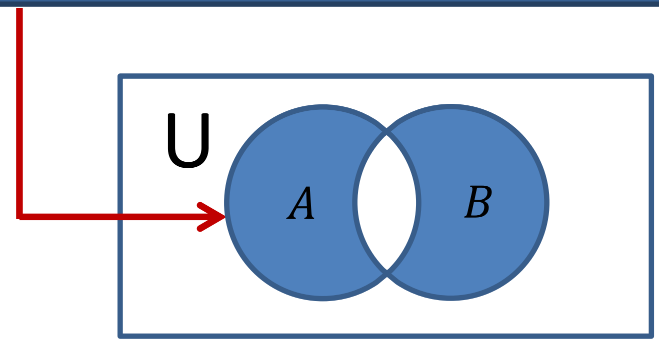
  A - B = {7, 9}

Operations on Sets

- Operations on the sets are:
  1. Complement
  2. Union
  3. Intersection
  4. Set Difference
  5. Symmetric Difference
  6. Cartesian product

- The symmetric difference $A \ominus B$ of two sets A and B is the set of everything in A but not in B or the set of everything in B but not in A.

$$A \ominus B = (A - B) \cup (B - A)$$



- Example:

  A = {1, 3, 5, 7, 9}

  B = {1, 2, 3, 4, 5}

  $A \ominus B$ = {7, 9, 2, 4}

Operations on Sets

- Operations on the sets are:
  1. Complement
  2. Union
  3. Intersection
  4. Set Difference
  5. Symmetric Difference
  6. Cartesian product

- The Cartesian product A x B of two sets A and B is the set of all ordered pairs (a, b) where a ∈ A and b ∈ B.

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$$

- Example:

  A = {1, 3, 5}

  B = {2, 4}

  A x B = {(1,2), (1,4), (3,2), (3,4), (5,2), (5,4)}

Set of identities

- Commutative laws

$$A \cap B = B \cap A$$
$$A\ U\ B = B\ U\ A$$

- Associative laws

$$A \cap (B \cap C) = (A \cap B) \cap C$$
$$A\ U\ (B\ U\ C) = (A\ U\ B)\ U\ C$$

- Distributive laws

$$A\ U\ (B \cap C) = (A\ U\ B) \cap (A\ U\ C)$$
$$A \cap (B\ U\ C) = (A \cap B)\ U\ (A \cap C)$$

Set of identities

- Idempotent laws

$$A \cup A = A$$
$$A \cap A = A$$

- Absorptive laws

$$A \cup (A \cap B) = A$$
$$A \cap (A \cup B) = A$$

- De Morgan laws

$$(A \cup B)' = A' \cap B'$$
$$(A \cap B)' = A' \cup B'$$

Set of identities

- Other complements laws

$$(A')' = A$$
$$A \cap A' = \Phi$$
$$A \cup A' = U$$

- Other empty set laws

$$A \cup \Phi = A$$
$$A \cap \Phi = \Phi$$

- Other universal set laws

$$A \cup U = U$$
$$A \cap U = A$$

Functions

- Domain: What can go into the function is called domain.
- Codomain: What may possibly come out from a function is codomain.
- Range: What actually come out from a function is range. The range of function is subset of codomain
- Example:

$$\boldsymbol{f}: \boldsymbol{N} \rightarrow \boldsymbol{N}, \boldsymbol{f}(\boldsymbol{x}) = \mathbf{2}\boldsymbol{x} + \mathbf{1}$$

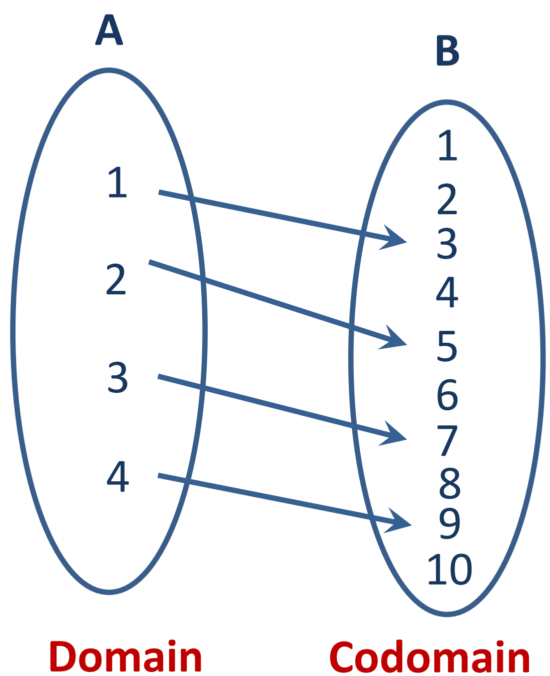
f(1)=2(1)+1= 3

f(2)=2(2)+1= 5

f(3)=2(3)+1= 7

f(4)=2(4)+1= 9

Range



- The range of function $\mathrm{f(x)} = \{3, 5, 7, 9\}$

Relations

Relations

- A relation on a set A is defined as subset of $A \times A$.
- The relation R is denoted as aRb where $a, b \epsilon A$ and pair $(a, b) \epsilon R$.
- Example:

$$N = \{1,2,3\}$$

$$N \times N = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}$$

- The **‘=‘** relation on $N \times N$ is :

  where

  1 = 1

  2 = 2

  3 = 3

Languages

Language

- A set of strings all of which are chosen from some Σ^*, where Σ is a particular alphabet, is called a language. If Σ is an alphabet, and $L \subseteq \Sigma^*$, then L is said to be language over alphabet Σ.
- Language comprises of:
  - Set of characters – Σ
  - Set of strings (words) defined from set of character - Σ^*
  - Language L is defined from Σ^*, and $L \subseteq \Sigma^*$ because Σ^* contains many string which may not satisfy the rules of language.
- Example:
  - Σ = {a, b}
  - Σ^* = {^, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, …}

Operations over Language

- Operations over the language are:
  1. Concatenation
  2. Union
  3. * (Kleene closure)
  4. +

If $L_1, L_2 \subseteq \Sigma^*$ then concatenation is defined as

$$L_1L_2 = \{xy \mid x \in L_1 \; and \; y \in L_2\}$$

Example:

L_1 = {hope, fear} and L_2 = {less, fully}

hopeless

L_1L_2 = {hopeless, hopefully, fearless, fearfully}

Operations over Language

- Operations over the language are:
  1. Concatenation
  2. Union
  3. * (Kleene closure)
  4. +

If $L_1, L_2 \subseteq \Sigma^*$ then union is defined as

$$L_1 \mid L_2 = \{x \mid x \in L_1 \ or \ x \in L_2\}$$

Example:

L_1 = {hope, fear} and L_2 = {less, fully}

$L_1 \mid L_2$ = {hope, fear, less, fully}

Operations over Language

- Operations over the language are:
  1. Concatenation
  2. Union
  3. * (Kleene closure)
  4. +

- If L is a set of words then by L^* we mean the set of all finite strings formed by concatenating words from S, where any word may be used as often we like, and where the null string is also included.

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Example: L = {ab}

$L*$ = {^, ab, abab, ababab, abababab, ….}

Operations over Language

- Operations over the language are:
    1. Concatenation
    2. Union
    3. * (Kleene closure)
    4. +

- If L is a set of words then by L^{+} we mean the set of all finite strings formed by concatenating words from L, where any word may be used as often we like, and where the null string is not included.

$$L^{+} = \bigcup_{i=1}^{\infty} L^{i}$$

Example: L = {ab}

L^{+} = {ab, abab, ababab, abababab, ...}

Power Set

Let A be the set, the set of all subset of set A is called power set of A and is denoted by $\mathbf{2^A}$

Example : A={1,2,3} $2^A = \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\},\{\emptyset\}\}$

Empty Set

Set Containing no Elements

Example: S={} or {Ø}

Finite and infinite set

If a set containing finite number of elements

Example : s ={1,2,3,4}, |s| =4

If a set containing an infinite number of elements

Example : Natural numbers

Grammars:

Set of Rules or protocols

Example A typical rule of English grammar is sentence can consists of a noun phrased followed by predicates

Ex. SIHI ate Slowly

Noun Verb Adverb

Rules

Sentence → Noun/Verb/Adverb

Noun → SIHI

Verb → ate

Adverb → Slowly

Language

Each Language consists of Alphabets from which the word, statements etc., can be derived

Alphabets An Alphabets is a finite and non empty set of symbols and is denoted by Σ

Symbols A Symbols is an abstract entity

Examples: Letter or Digits

Letter=A/B/C/D………………./Z/a/b/c/d/……………/z

Digits=0/1/2/3/…………/9

Strings The sequences of symbols from the alphabets(Σ)

Examples: Σ ={a,b,c}

Empty Strings Empty string is denoted by $\in$(epsilon) is consists of 0 symbols $| \in | = 0$

Alphabets and Strings :

We will use small alphabets $\Sigma = \{a, b\}$

Strings

a

ab

$abba$

$baba$

$aaabbbaabab$

$u = ab$

$v = bbbaaa$

$w = abba$

String Operations

$w = a_1 a_2 \cdots a_n$ $abba$

$v = b_1 b_2 \cdots b_m$ $bbbaaa$

Concatenation

$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$ $abbabbbaaa$

Reveres

$w = a_1 a_2 \cdots a_n$ $w^R = a_n \cdots a_2 a_1$

$ababaaabbb$ $bbbaaababa$

String Length

$$w = a_1 a_2 \cdots a_n$$

Length: $|w| = n$

Examples:

$$|abba| = 4$$

$$|aa| = 2$$

$$|a| = 1$$

Recursive Definition of Length

For any letter : $|a| = 1$

For any string : wa $\quad |wa| = |w| + 1$

Example :

$$
\begin{aligned}
|abba| &= |abb| + 1 \\
&= |ab| + 1 + 1 \\
&= |a| + 1 + 1 + 1 \\
&= 1 + 1 + 1 + 1 \\
&= 4
\end{aligned}
$$

Length of Concatenation

$$|uv| = |u| + |v|$$

Example:

$$u = aab, \quad |u| = 3$$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

Empty String

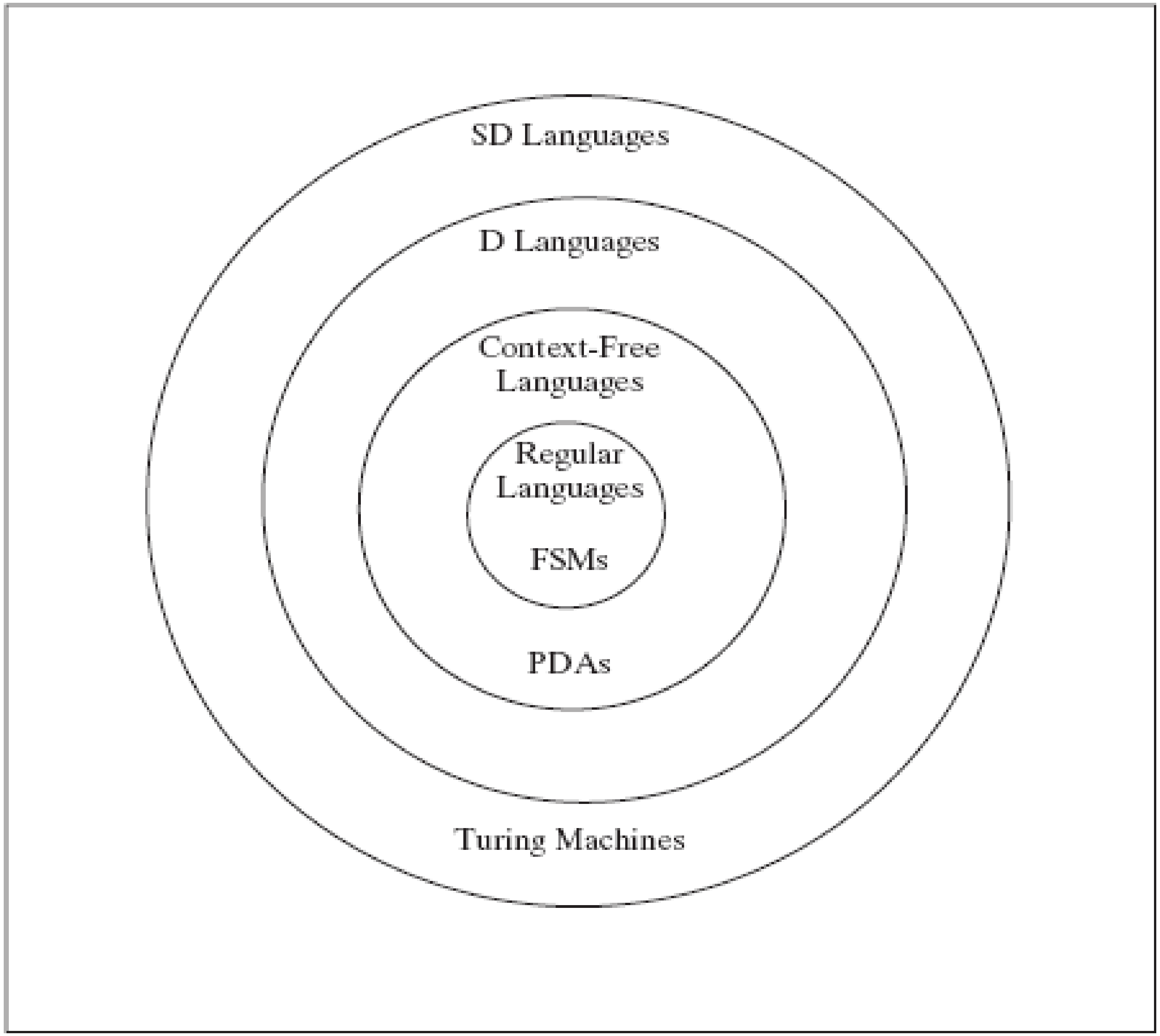
A string with no letters: $\in$

Observations: $|\in| = 0$

$$\in w = w \in = w$$

$$\in abba = abba \in = abba$$

Hierarchy of Languages



D=decidable
SD = Semidecidable

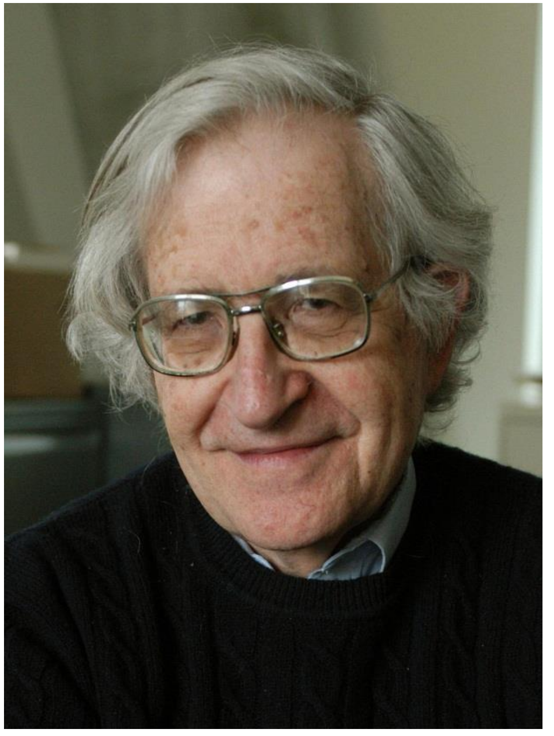
Chomsky Hierarchy of Languages

Languages from “simplest” to “complex”

Each is a subset of the ones below

- Regular
- Context Free
- Context Sensitive
- Recursively Enumerable

Can be defined by the type of Machine that will recognize it.

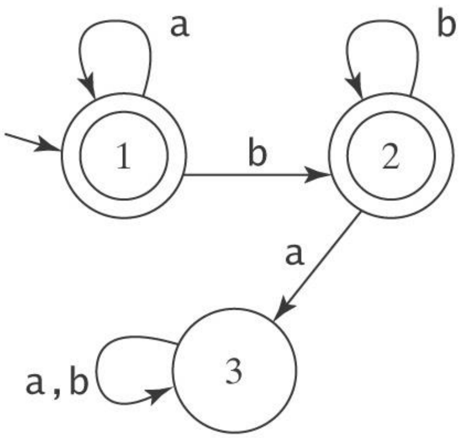


Noam Chomsky

Regular Languages

A Regular Language is one that can be recognized by a Finite State Machine.

An FSM to accept `a*b*`:



Finite Automata(FA)
or
Finite State Machine (FSM)

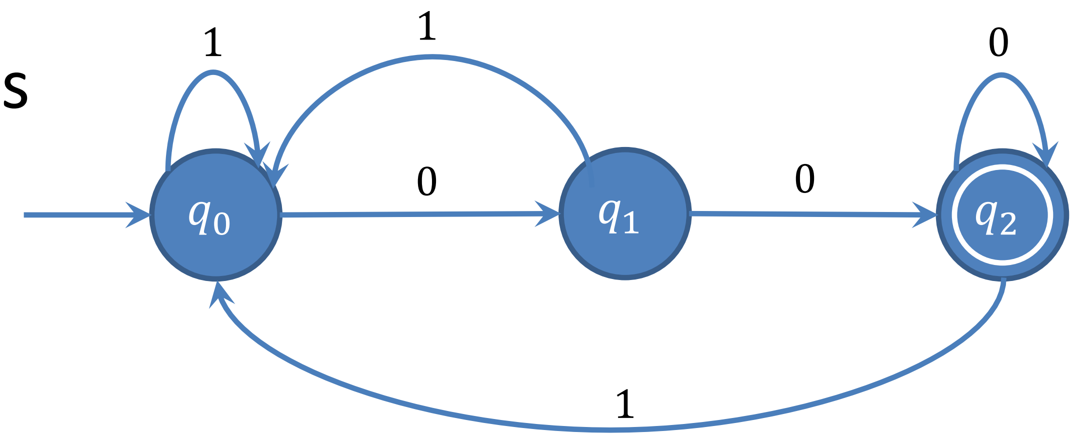
Finite Automata

- A finite automaton, or finite state machine is a 5-tuple $(Q, \Sigma, q_0, F, \delta)$ where
  - Q is finite set of states;
  - Σ is finite alphabet of *input symbols*;
  - $q_0 \in Q$ (*initial* state);
  - $F \subseteq Q$ (the set of *accepting* states);
  - δ is a function from $Q \times \Sigma \; to \; Q$ (the *transition* function).
- For any element q of Q and any symbol $a \in \Sigma$, we interpret $\delta(q, a)$ as the state to which the FA moves, if it is in state q and receives the input a.

Example: Finite Automata

- $M = (Q, \Sigma, q_0, F, \delta)$
  - $Q = \{q_0, q_1, q_2\}$
  - $\Sigma = \{0,1\}$
  - $q_0 = q_0$
  - $F = \{q_2\}$
  - δ is defined as

δ	Input	
State	**0**	**1**
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0



Applications of FA

- Lexical analysis phase of a compiler.
- Design of digital circuit.
- String matching.
- Communication Protocol for information exchange.

Automata Theory & Modern-day Applications

Algorithm
Design &
NP-Hardness

Compiler
Design &
Programming
Languages

Scientific
Computing
- biological systems
- speech recognition
- modeling

Automata
Theory &
Computability

Computer
Organization &
Architecture

Artificial Intelligence &
Information Theory

Computation models
serial vs. parallel
DNA computing, Quantum computing

Finite Automaton (FA or FSM)

- Informally, a state diagram that comprehensively captures all possible states and transitions that a machine can take while responding to a stream or sequence of input symbols
- Recognizer for "Regular Languages"

- Deterministic Finite Automata (DFA or DFSM)
  - The machine can exist in only one state at any given time
- Non-deterministic Finite Automata (NFA or NDFSM)
  - The machine can exist in multiple states at the same time

Deterministic Finite Automata - Definition

- A Deterministic Finite Automaton (DFA) consists of:
  - Q ==> a finite set of states
  - ∑ ==> a finite set of input symbols (alphabet)
  - q_0 ==> a start state
  - F ==> set of accepting states
  - δ ==> a transition function, which is a mapping between

    Q x ∑ ==> Q
- A DFA is defined by the 5-tuple:
  - {Q, ∑ , q_0,F, δ }

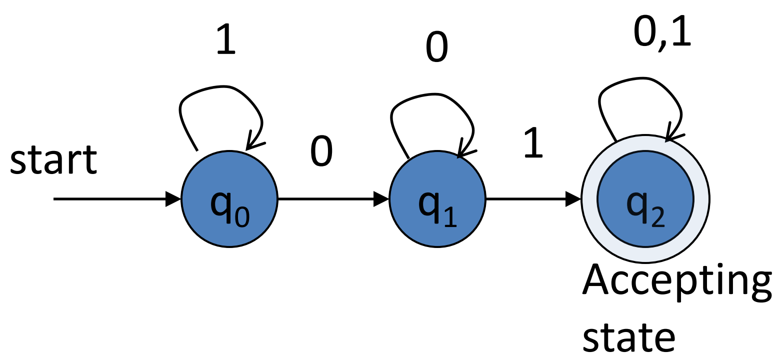
What does a DFA do on reading an input string?

- Input: a word w in Σ^*
- Question: Is w acceptable by the DFA?
- Steps:
  - Start at the "start state" q_0
  - For every input symbol in the sequence w do
    - Compute the next state from the current state, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed, the current state is one of the accepting states (F) then *accept w;*
  - Otherwise, *reject w.*

Regular expression: (0+1)*01(0+1)*

DFA for strings containing 01

- What makes this DFA deterministic?



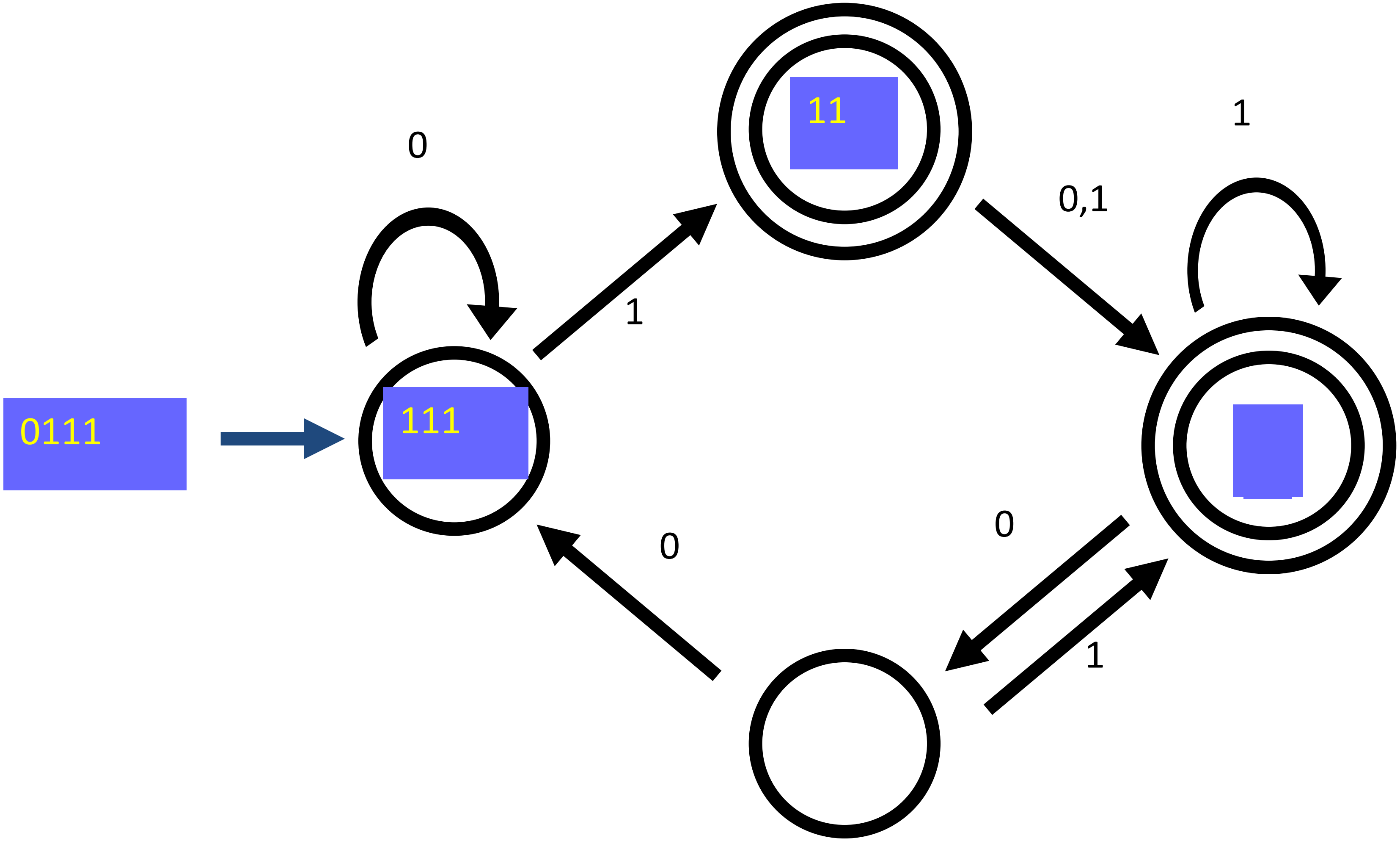
- What if the language allows empty strings?

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

symbols

δ	**0**	**1**
→ q_0	q_1	q_0
q_1	q_1	q_2
*q_2	q_2	q_2

states

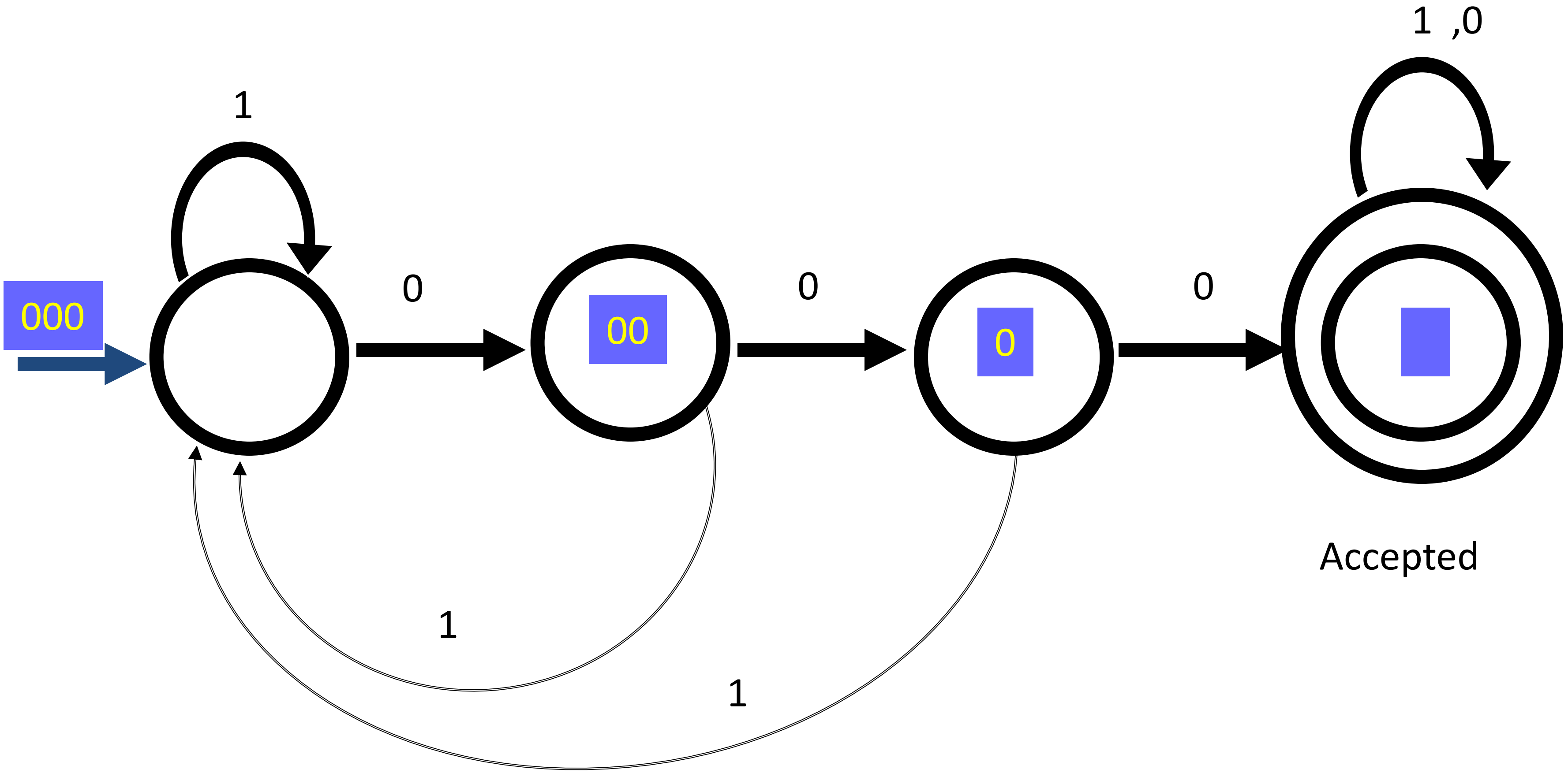


The machine accepts a string if the process ends in a double circle

1. Construct a DFA which accepts set of all strings of 0's and 1's having at least "3 consecutive zeros"

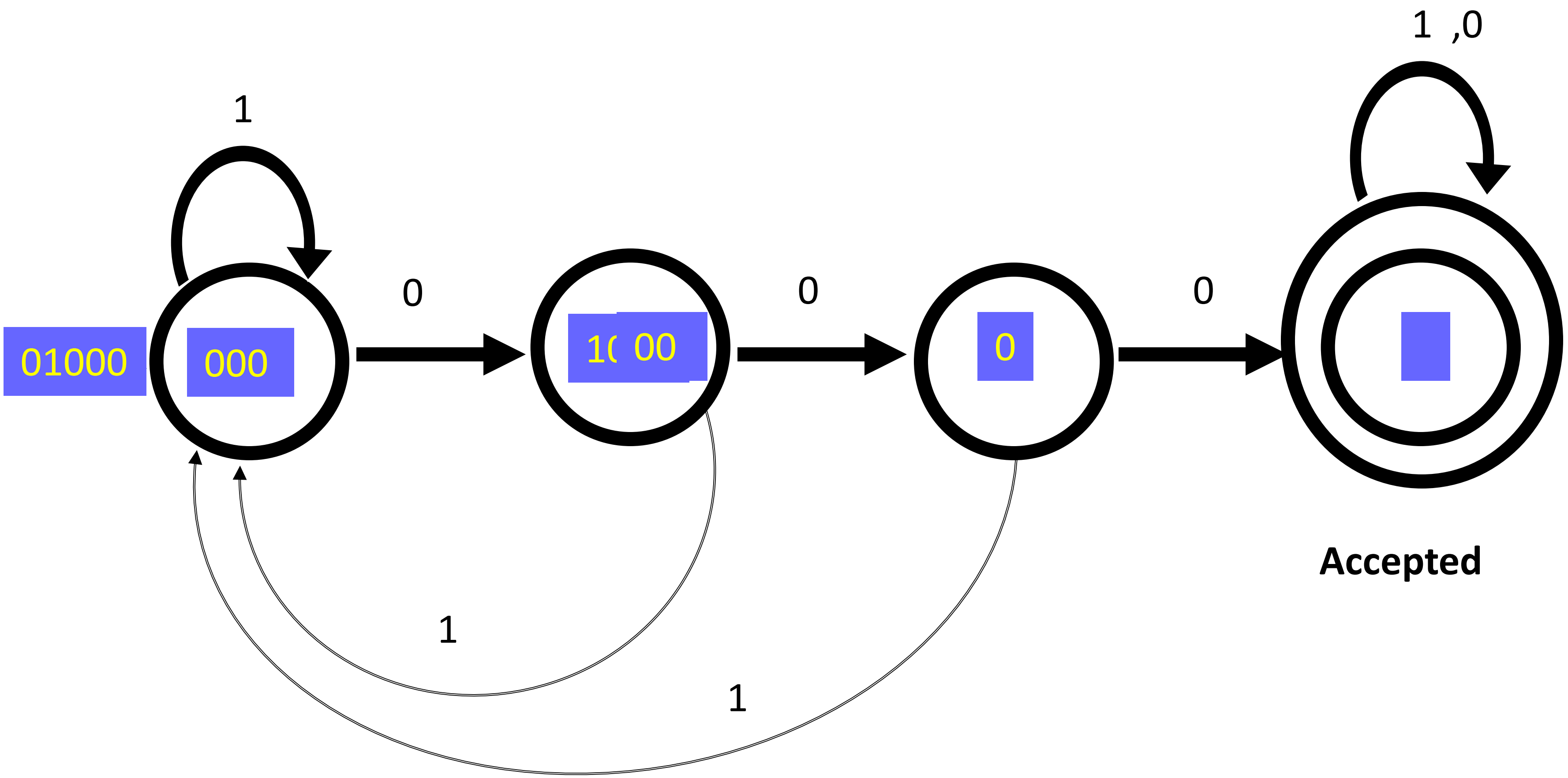
Minimum string

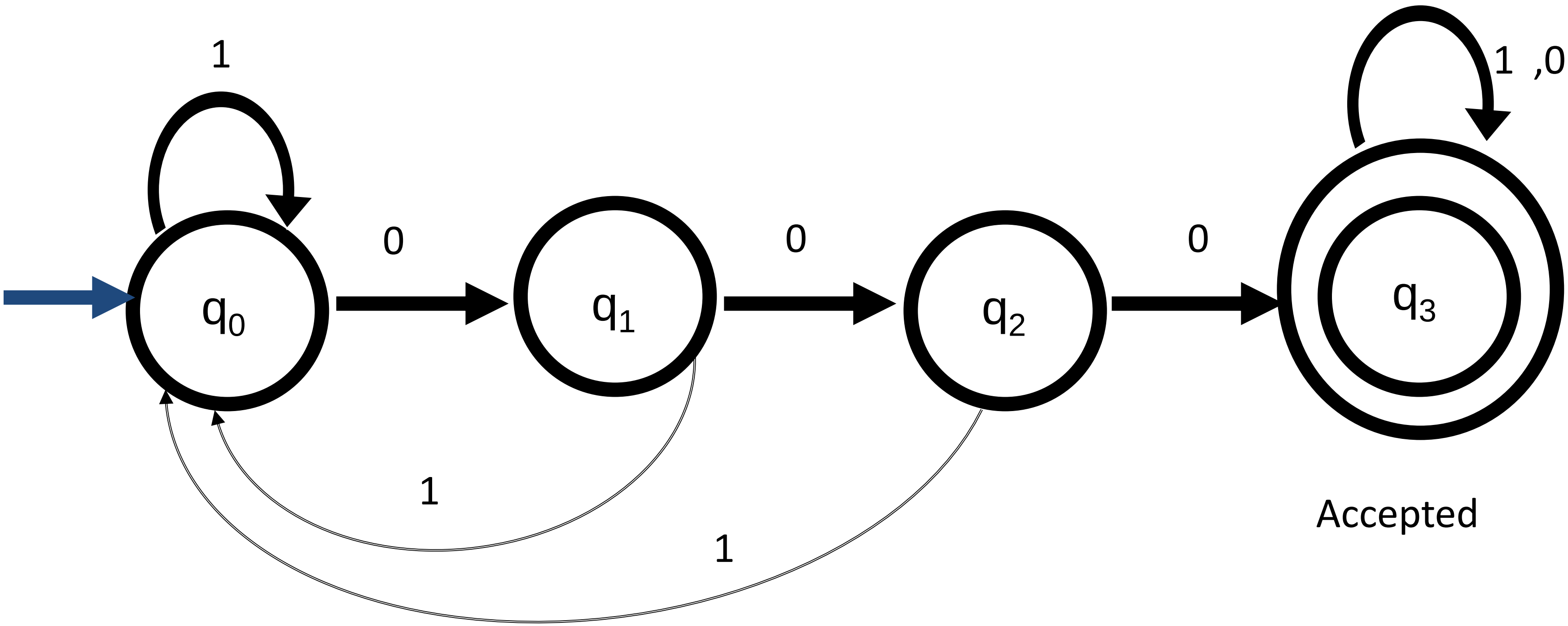
L= { 000, 1000, 0001, 0000, 01000, 001000, ..., }



Continued Example -1

L= { 000, 1000, 0001, 0000, 01000, 001000, ..., }





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

$Q = \{q_0, q_1, q_2, q_3\}$

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

$q_0 \in Q$ is start state

$F = \{q_3\} \subseteq Q$ accept states

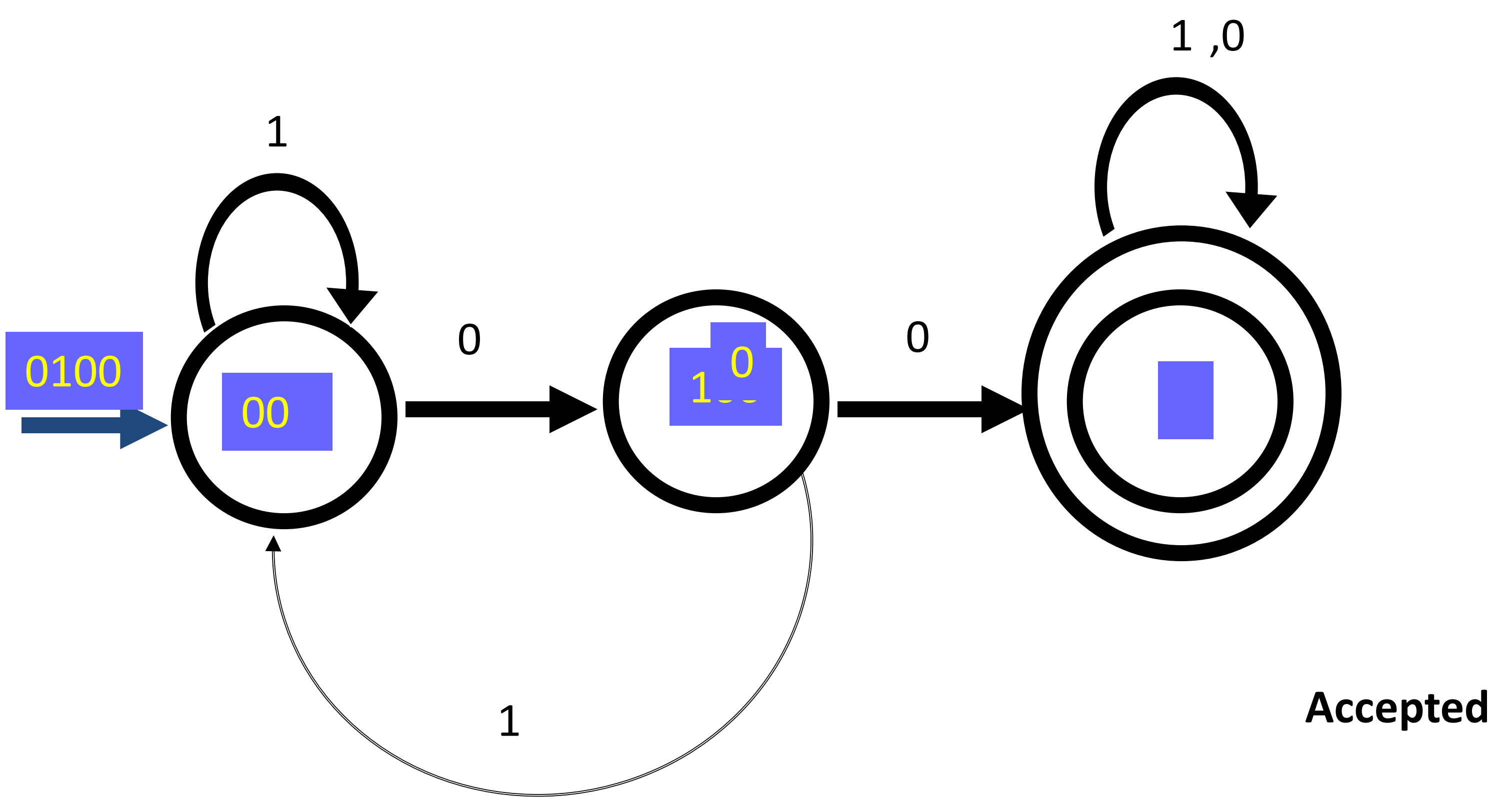
$\delta : Q \times \Sigma \rightarrow Q$ transition function

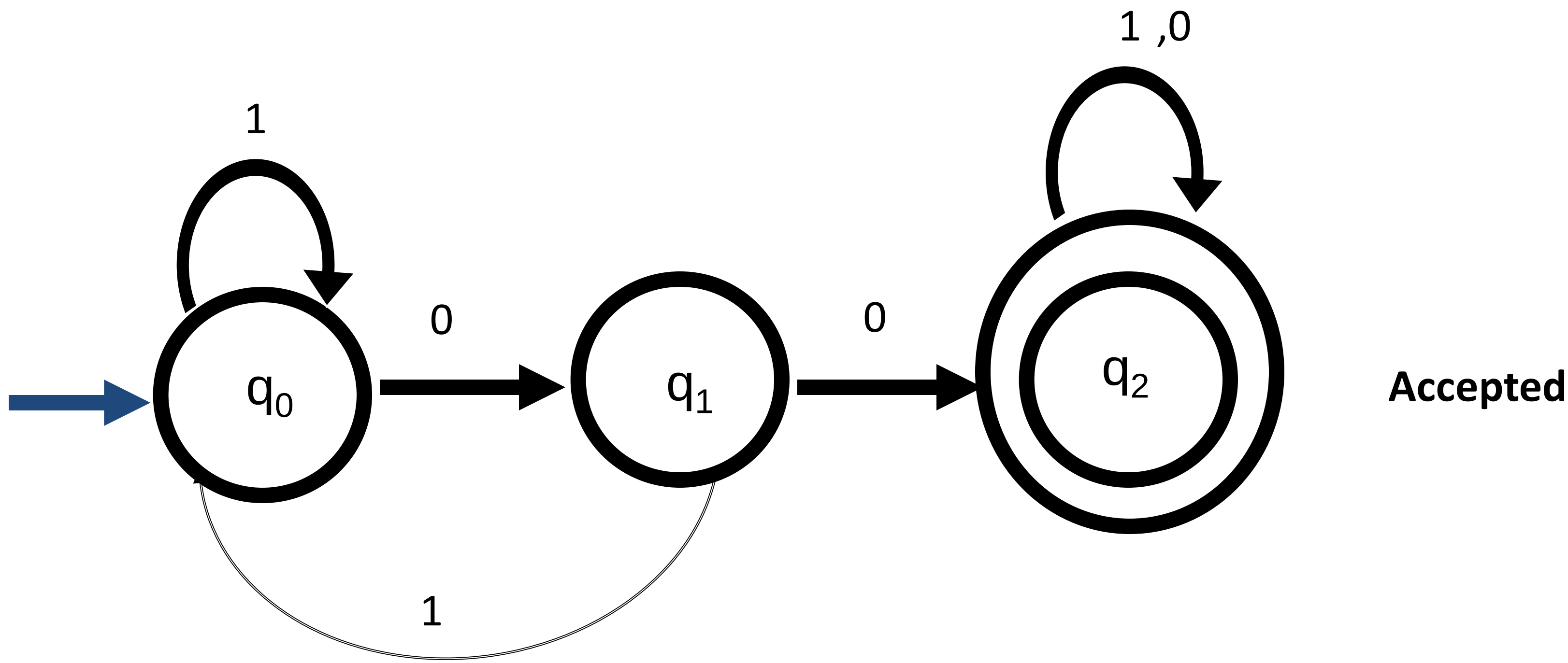
δ	**0**	**1**
-> $\mathbf{q_0}$	$\mathbf{q_1}$	$\mathbf{q_0}$
$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_0}$
$\mathbf{q_2}$	$\mathbf{q_3}$	$\mathbf{q_0}$
$\mathbf{q_3^*}$	$\mathbf{q_3}$	$\mathbf{q}$

2. Construct a DFA which accepts set of all strings of 0's and 1's having at least "2 consecutive zeros"

Minimum string

L= { 00, 100, 001, 000, 0100, ..., }





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

δ	**0**	**1**
-> q_0	q_1	q_0
q_1	q_2	q_0
q_2^*	q_2	q_2

$Q = \{q_0, q_1, q_2\}$

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

$q_0 \in Q$ is start state

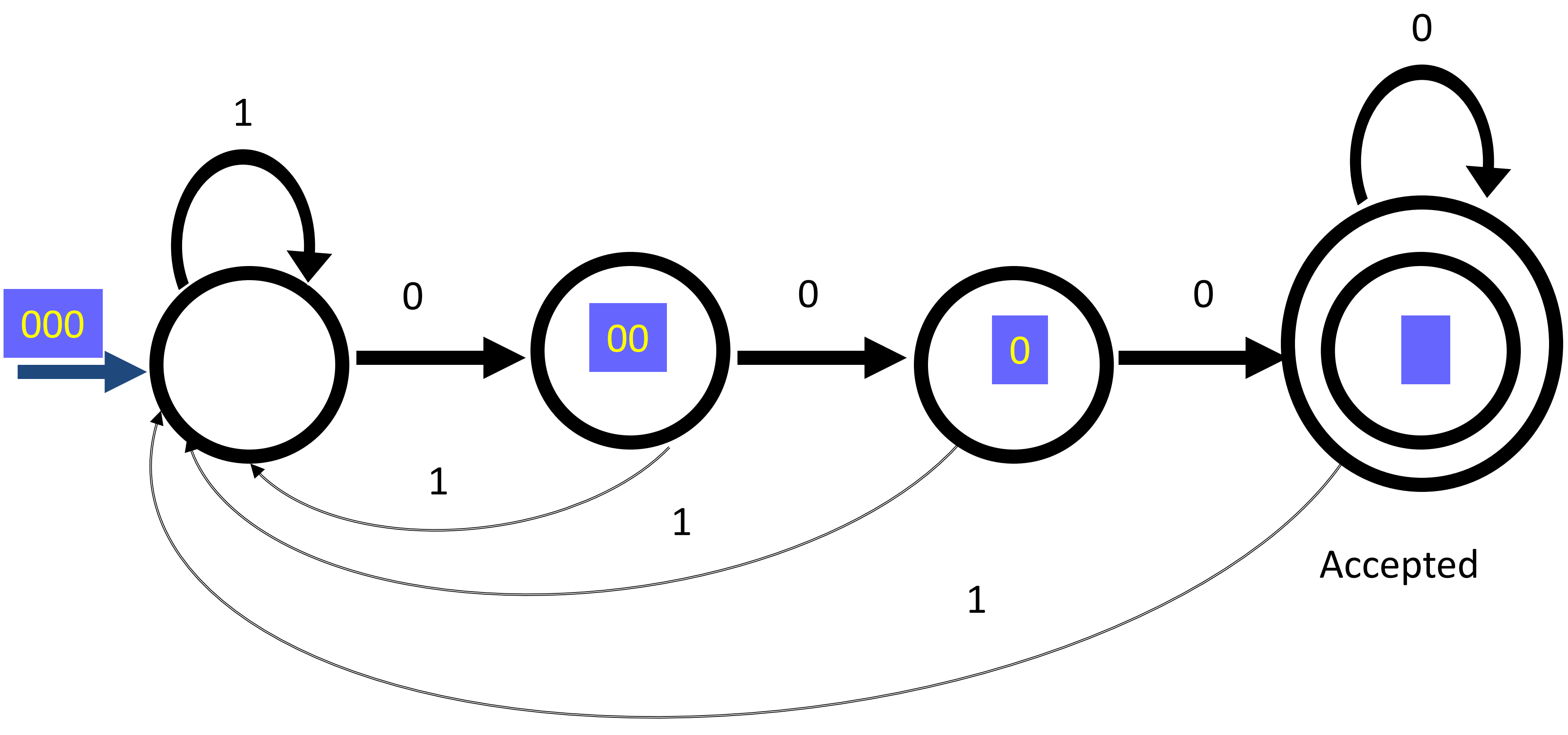
$F = \{q_2\} \subseteq Q$ accept states

$\delta : Q \times \Sigma \to Q$ transition function

3. Construct a DFA which accepts set of all strings of 0's and 1's ending with the '000'.

Minimum string

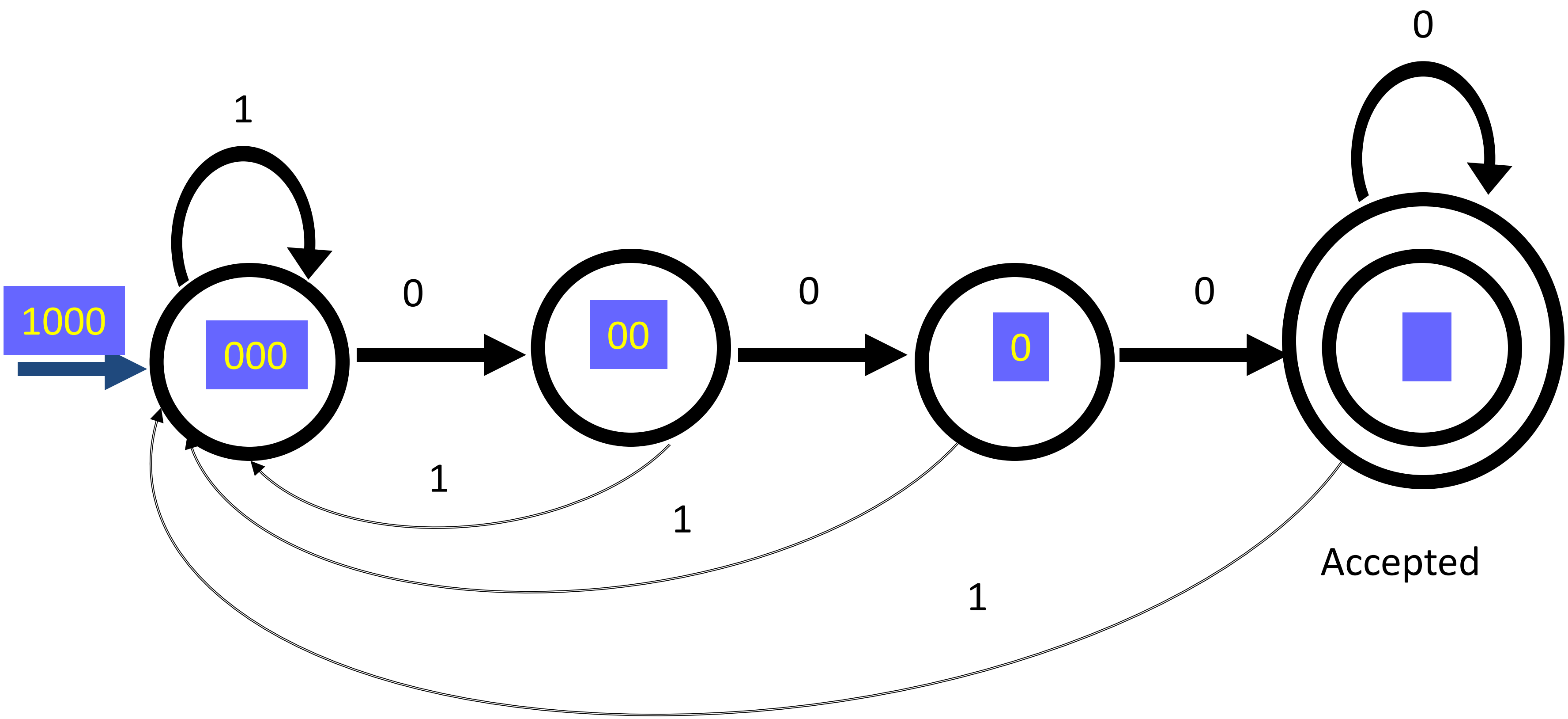
L= { 000, 1000, 0000, 01000, 001000, 0001000, ..., }



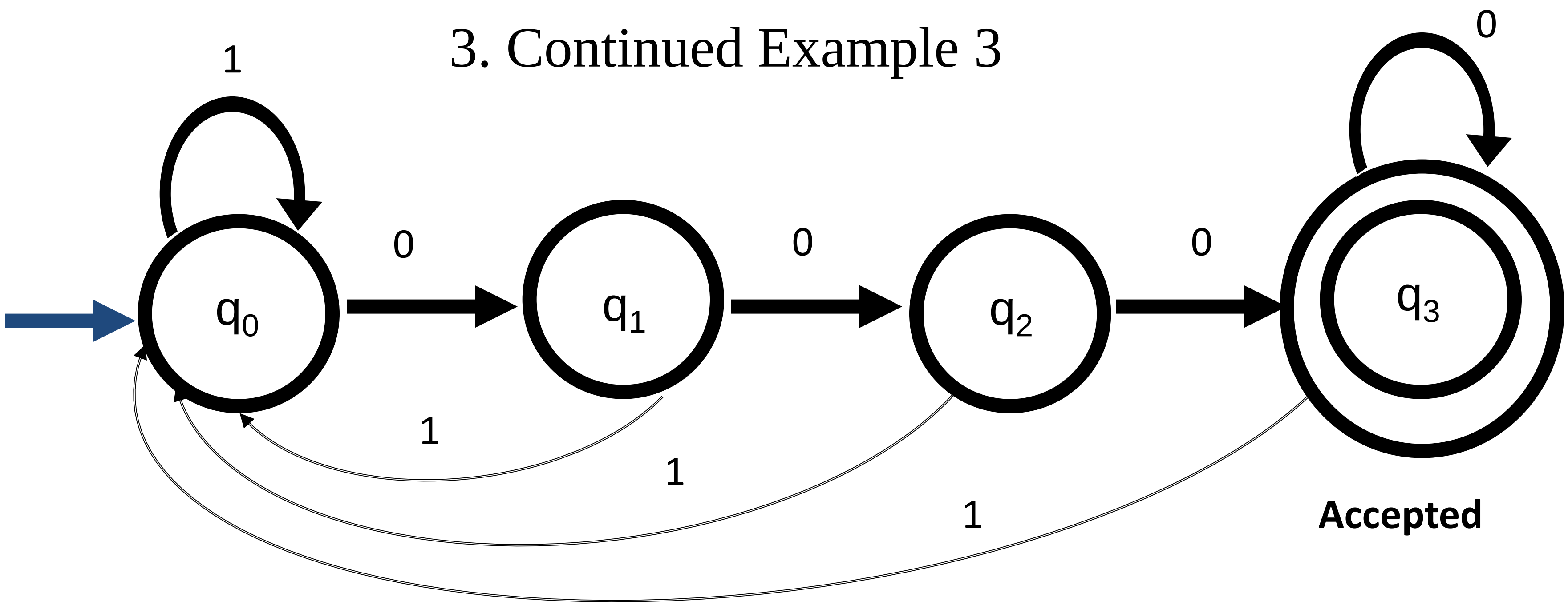
3. Construct a DFA which accepts set of all strings of 0's and 1's ending with the '000'.

Minimum string

L= { 000, 1000, 0000, 01000, 001000, 0001000, ..., }



3. Continued Example 3



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

δ	**0**	**1**
->q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q^*_3	q_3	q3

$Q = \{q_0, q_1, q_2, q_3\}$

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

$q_0 \in Q$ is start state

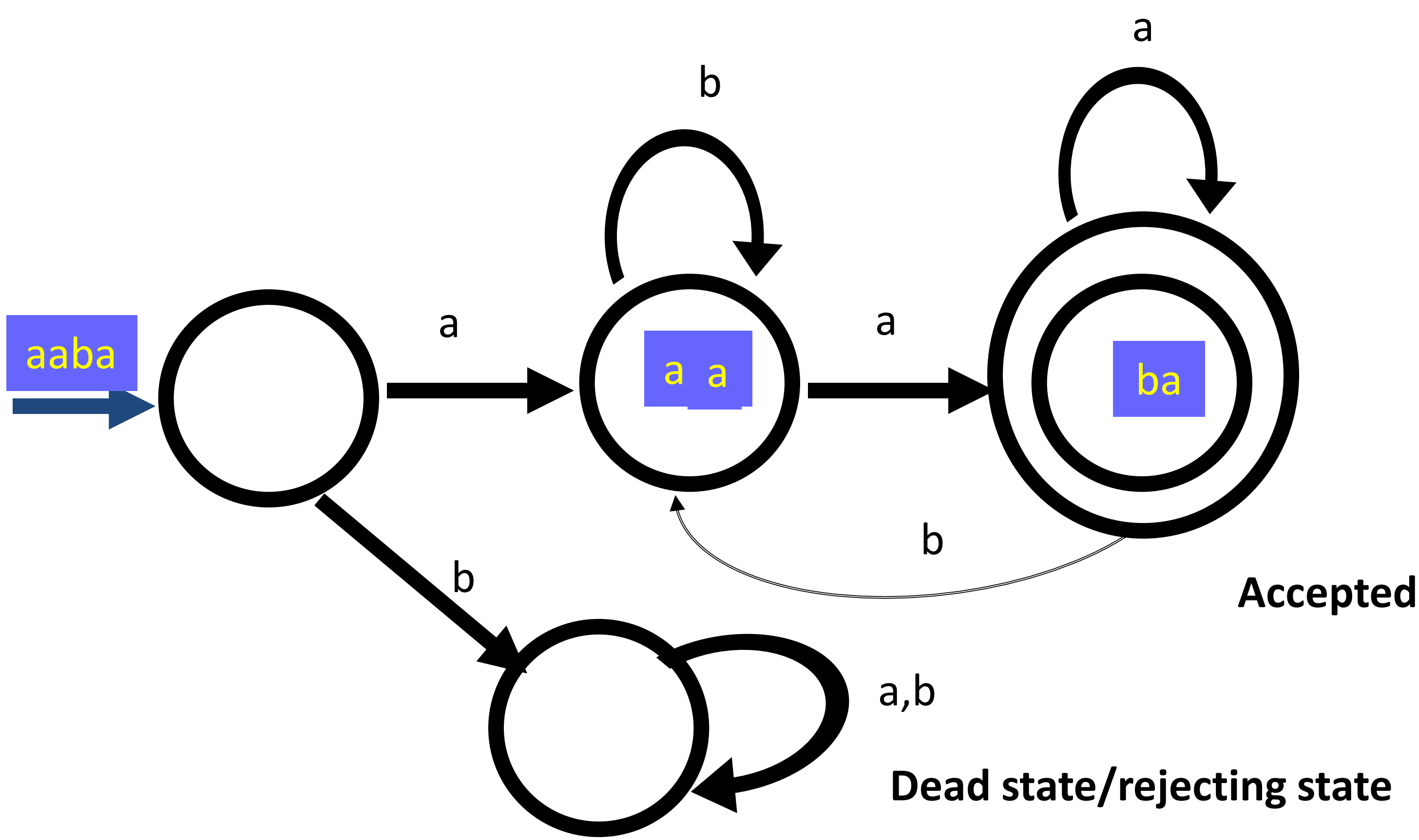
$F = \{q_3\} \subseteq Q$ accept states

$\delta : Q \times \Sigma \rightarrow Q$ transition function

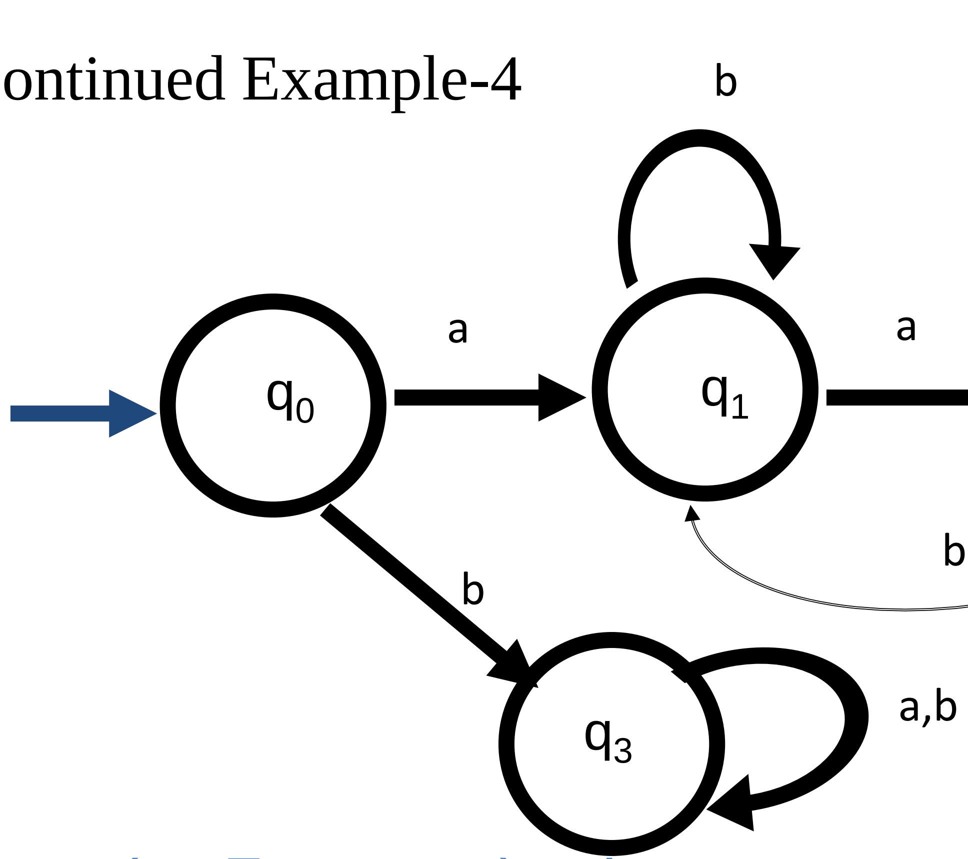
4. Construct a DFA which accepts set of all strings of a's and b's "starting with a ending with a"

Minimum string

L= { aa, aba, aaa, aaba, aaabbaa, ..., }



Continued Example-4



Accepted

Dead state/rejecting state

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

δ	a	b
->q_0	**q_1**	**q_3**
q_1	**q_2**	**q_1**
q^*_2	**q_2**	**q_1**
q_3	**q_3**	**q3**

$Q = \{q_0, q_1, q_2, q_3\}$

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

$q_0 \in Q$ is start state

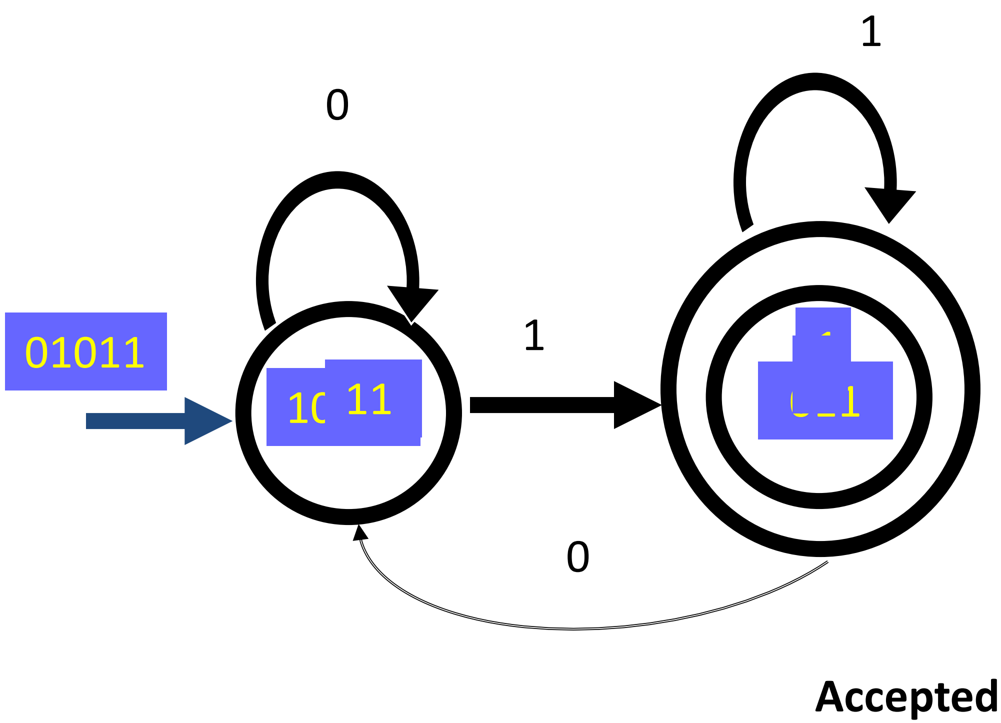
$F = \{q_2\} \subseteq Q$ accept states

$\delta : Q \times \Sigma \rightarrow Q$ transition function

5. Build a FSM with any string that ends in ‘1’ over the alphabet or (Binary ODD Number)

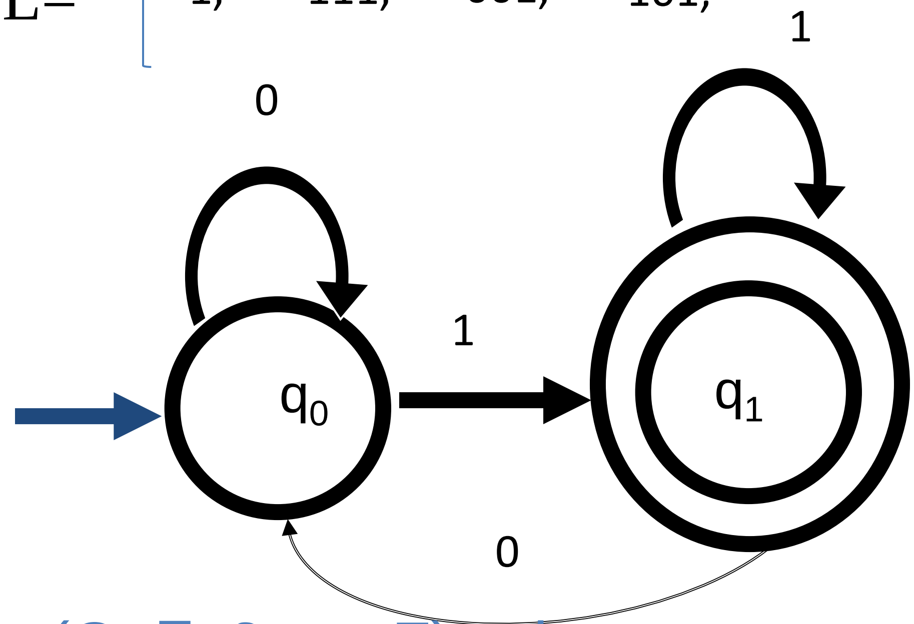
Minimum string

L= { 1, 111, 001, 101, ..., }



Continued Example 5

L= { 1, 111, 001, 101, ..., }



Accepted

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

$Q = \{q_0, q_1\}$

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

$q_0 \in Q$ is start state

$F = \{q_1\} \subseteq Q$ Accept states

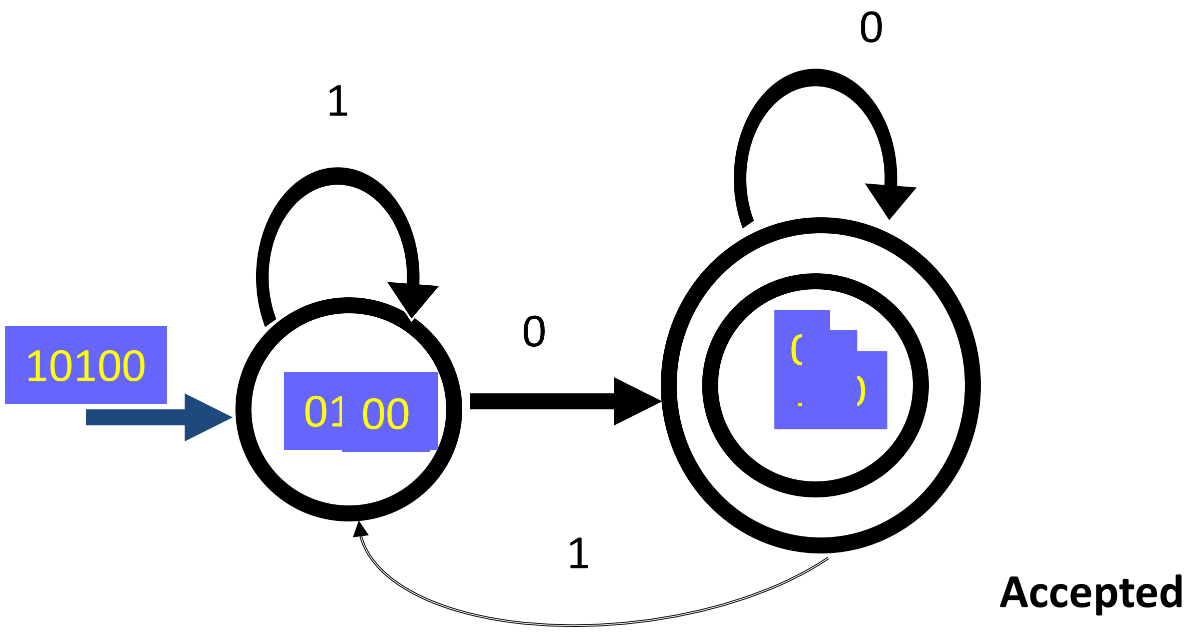
$\delta : Q \times \Sigma \rightarrow Q$ transition function

δ	**0**	**1**
->q_0	**q_0**	**q_1**
q^*_1	**q_0**	**q_1**

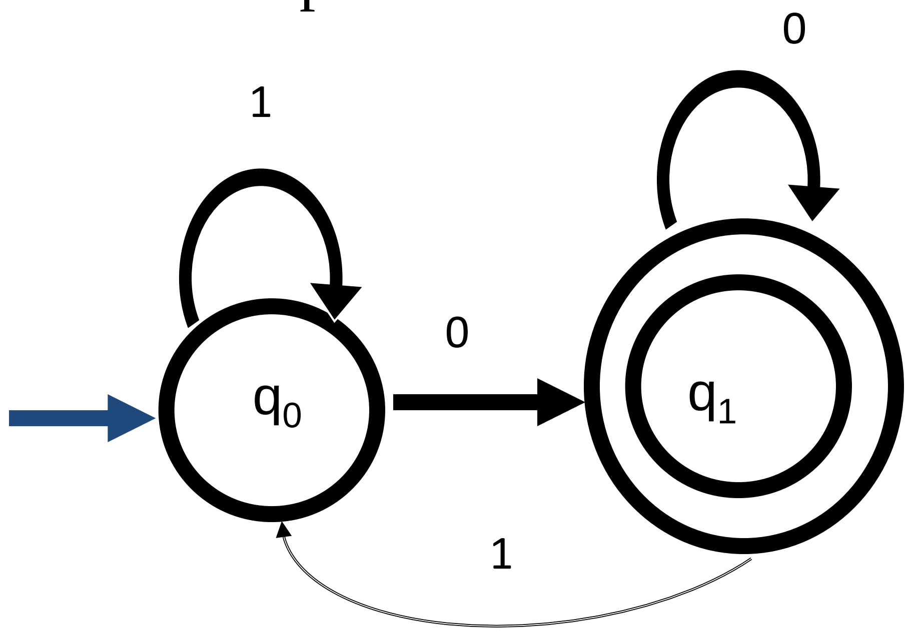
5. Build a FA with any string that ends in ‘0’ over the alphabet or (Binary EVEN Number)

Minimum string

L= { 00, 10, 010, ..., }



Continued Example



M = (Q, Σ, δ, q_0, F) where

$Q = \{q_0, q_1\}$

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

$q_0 \in Q$ is start state

$F = \{q_1\} \subseteq Q$ Accept states

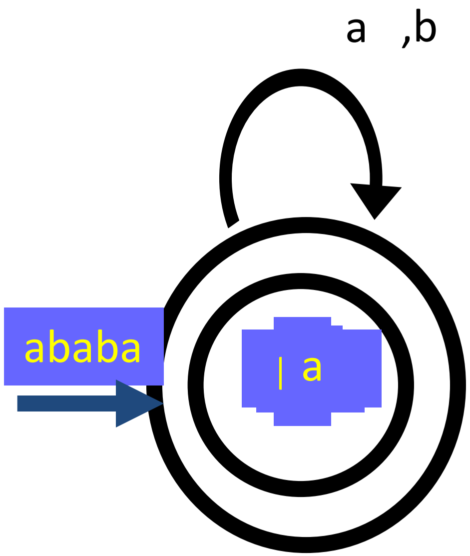
$\delta : Q \times \Sigma \rightarrow Q$ transition function

δ	**0**	**1**
->q_0	**q1**	**q0**
q^*_1	**q1**	**q_0**

5. Build a FA that accepts all strings over an alphabet $\Sigma = \{a,b\}$

Minimum string

L= { a, aaa, ab, bbbbb, ..., }

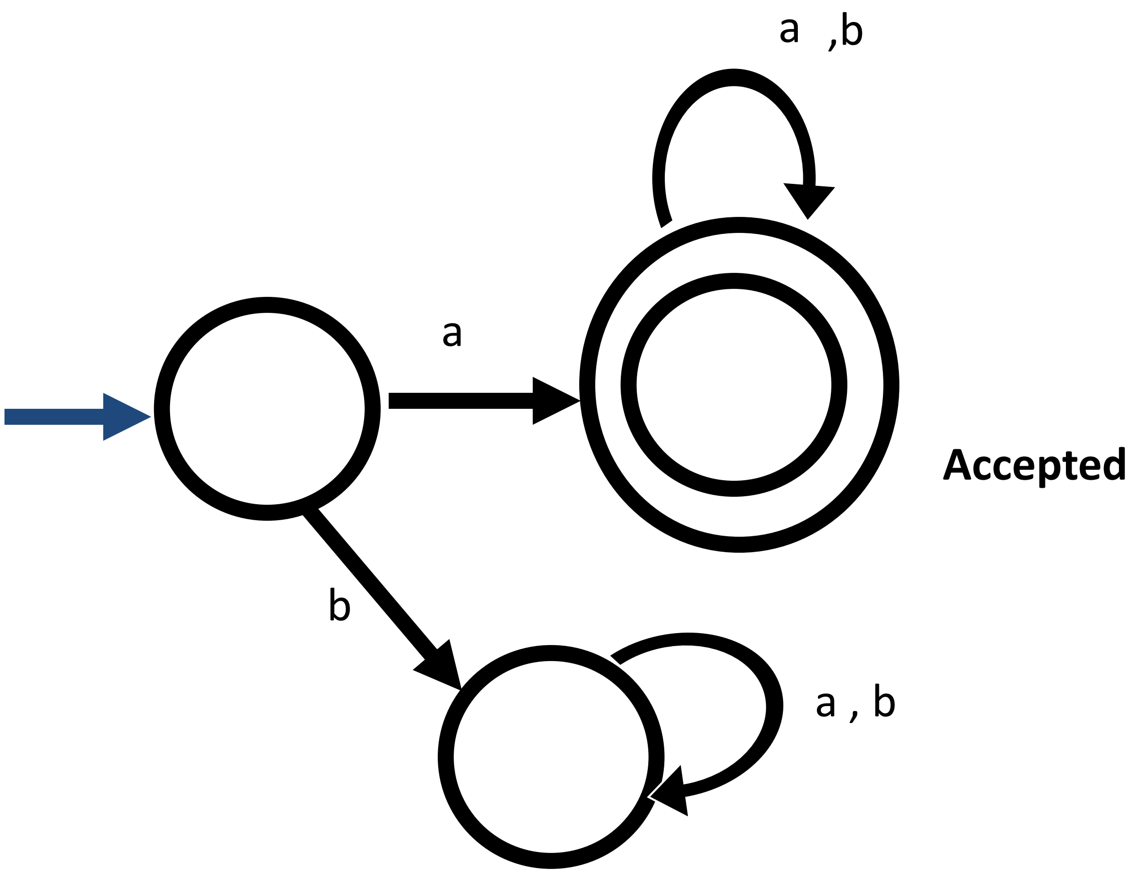


Accepted

5. Build a FA with any string that begins with ‘0’.

Minimum string

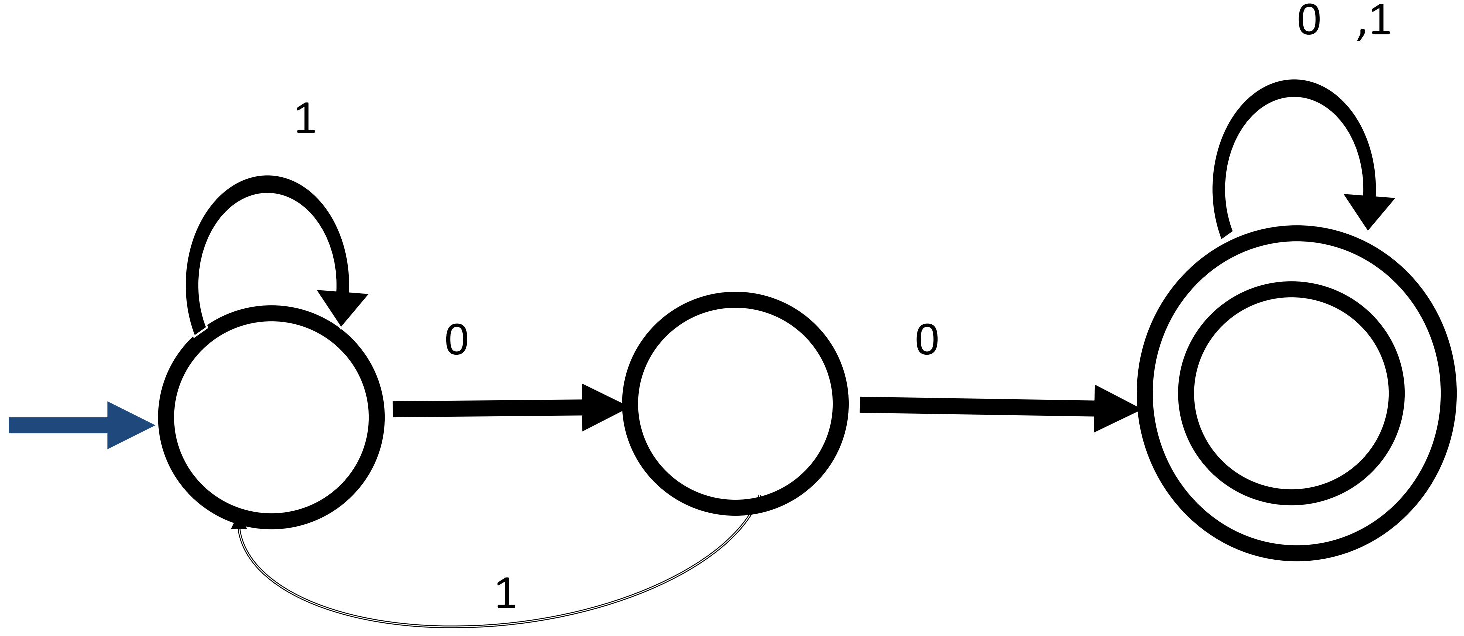
L= { a, aa, ab, ..., }



7.Obtain a DFA to accept string f 0’s and 1’s having a substring ‘00’

Minimum string

L= { 00, 000, 001, 100, 0100, ..., }

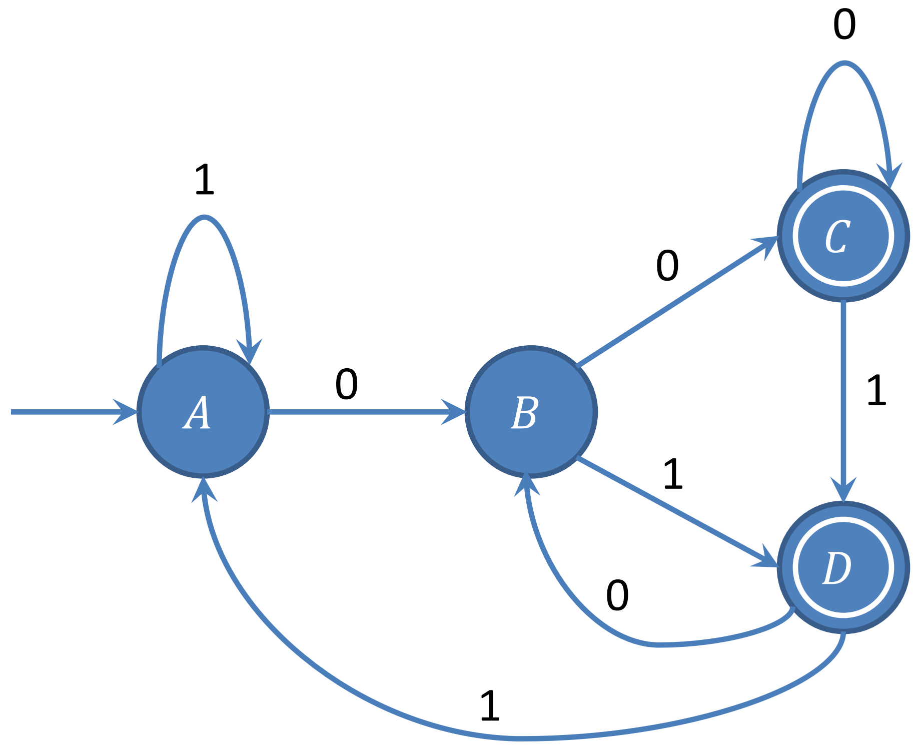


Accepted

Example: The string with next to last symbol as 0.

Minimum string

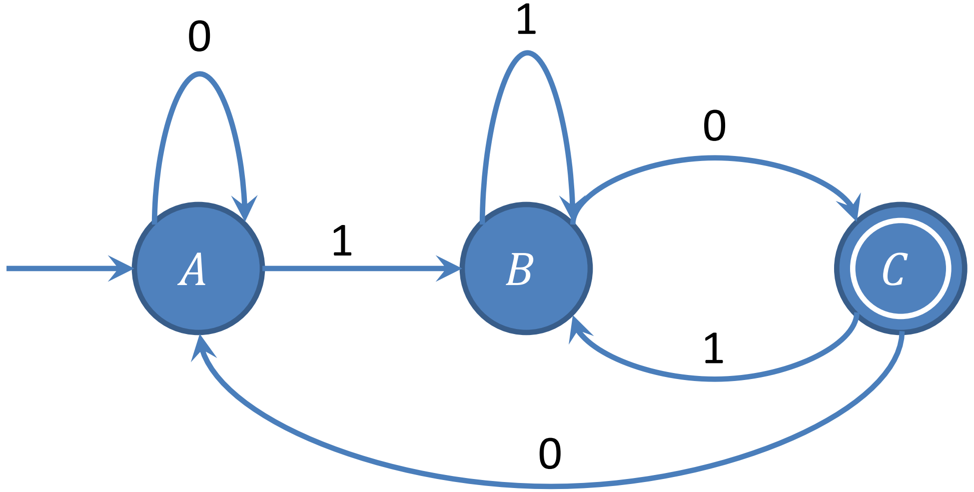
L= { 00, 01, 000, 1000, 0001, 00101 01101 ..., }



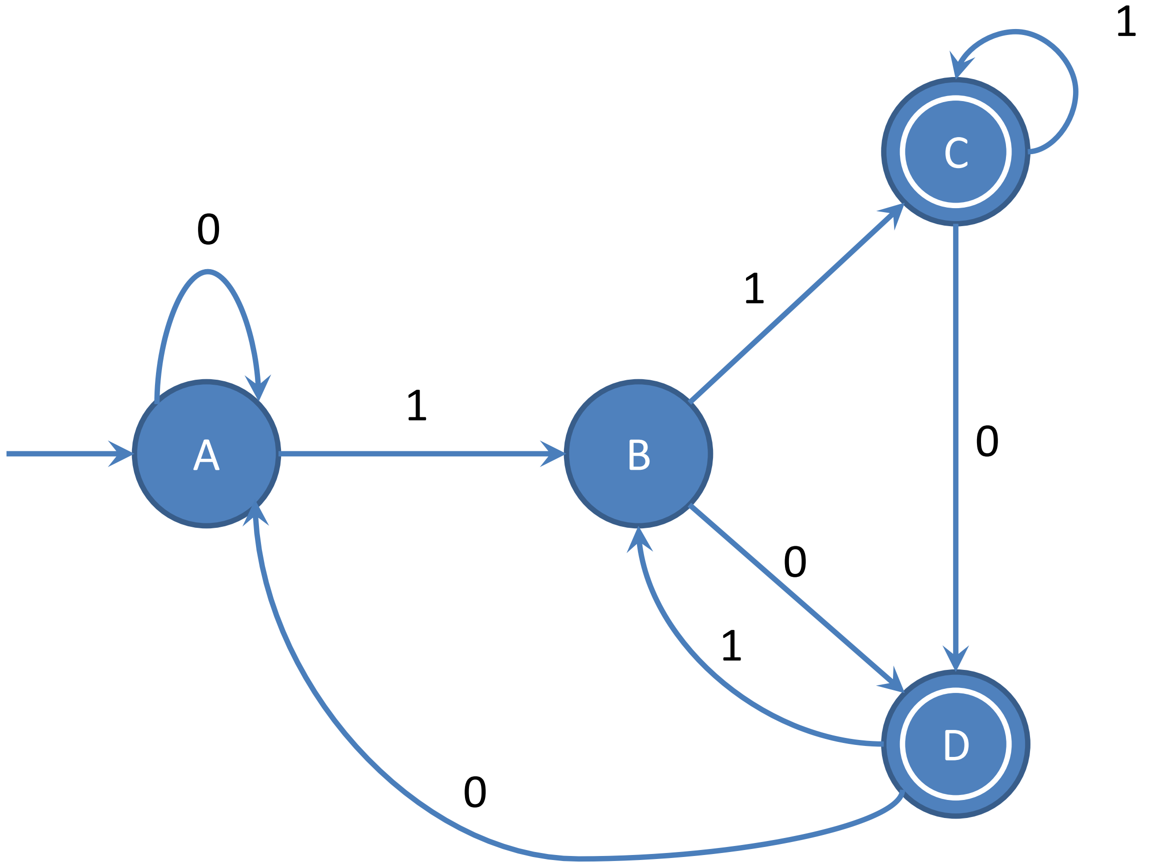
Example: The strings ending with 10.

Minimum string

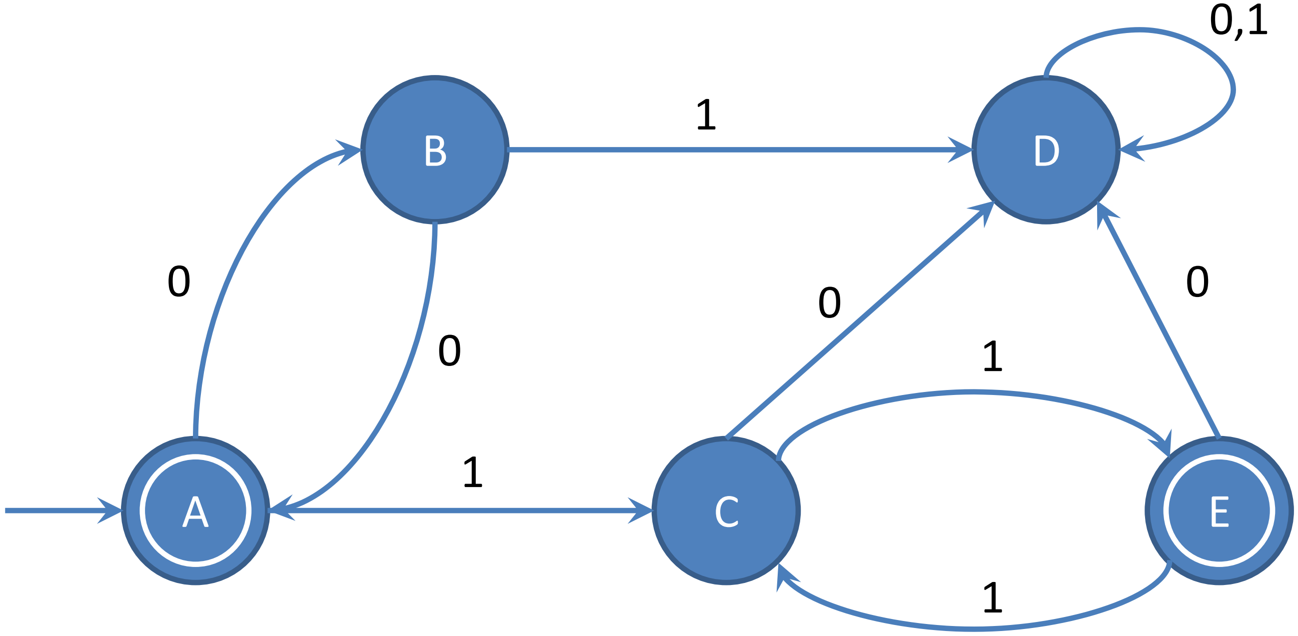
L= { 10, 010, 110, 1010, 10010, 101010 0001110 ..., }



The string ending in 10 or 11



The string corresponding to Regular expression {00}*{11}*



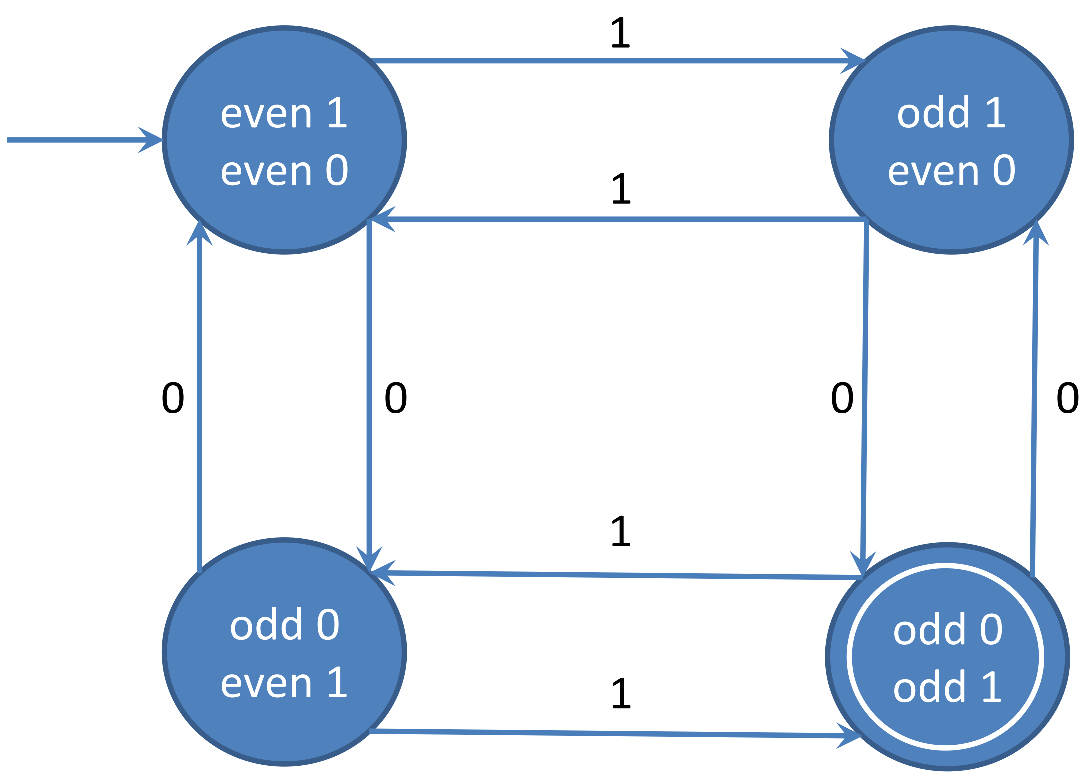
Construct a DFSM for accepting o’s and 1’s,

a. String consisting ODD Number of 0’s and 1’s.

b. String consisting EVEN Number of 0’s and 1’s.

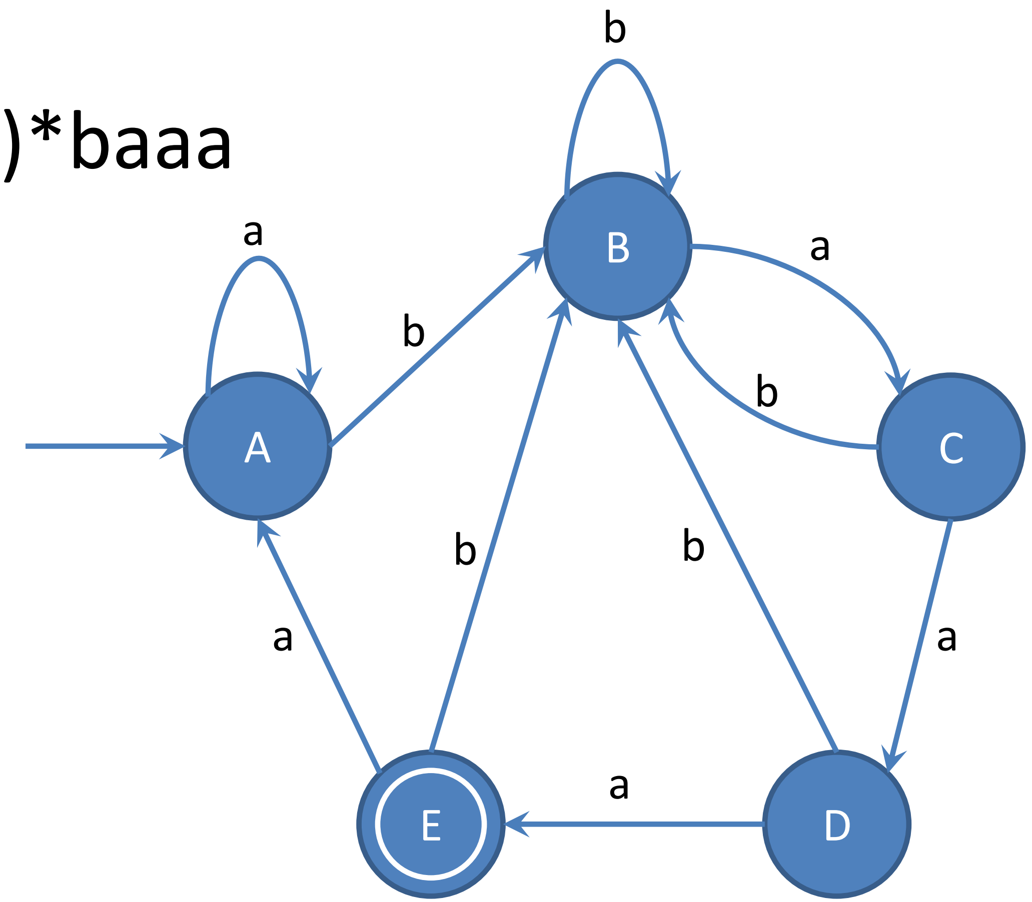
c. String consisting ODD Number of 0’s and Even Number of 1’s.

d. String consisting ODD Number of 1’s and Even number 1’s.



FA Examples

- (a+b)*baaa



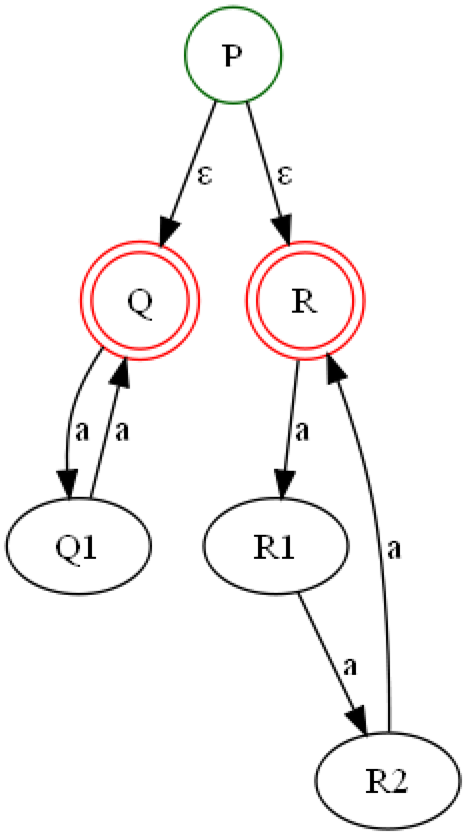
$\in$-Non-Deterministic Finite State Automata(Epsilon-NFA)

- We extend the class of NFAs by allowing instantaneous (ε) transitions:
    1. The automaton may be allowed to change its state without reading the input symbol.
    2. In diagrams, such transitions are depicted by labeling the appropriate arcs with ε.
    3. Note that this does not mean that ε has become an input symbol. On the contrary, we assume that *the symbol* ε *does not belong to any alphabet*.

NFA with ∊ move: If any FSM contains ε transaction or move, the finite automata is called NFA with ∊ move.

Example

- { a^n | n is even or divisible by 3 }



Formal Definition of $\in$-NDFSM

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

$Q:$ Set of states, i.e. $\{q_0, q_1, q_2\}$

$\Sigma:$ Input aplhabet, i.e. $\{a, b\}$

$\delta:$ Transition function

$$\delta : Qx\{\ \Sigma \cup \in\ \} \rightarrow 2^Q$$

$q_0:$ Initial state

$F:$ Accepting states

Note ε is never a member of Σ

ε-NDFSM

- ε -NFAs add a convenient feature but (in a sense) they bring us nothing new: they do not extend the class of languages that can be represented. Both NFAs and ε-NFAs recognize exactly the same languages.

ε-Closure

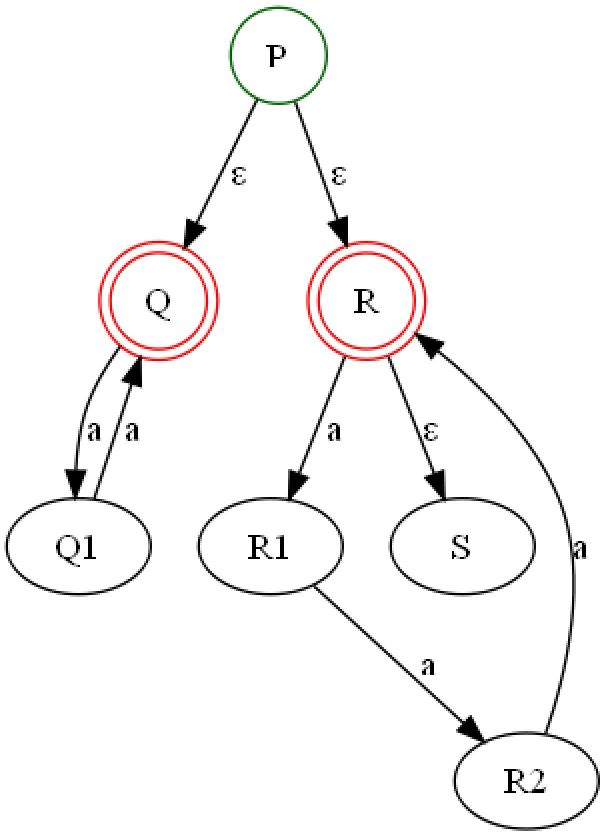
ε-Closure

- ε-closure of a state

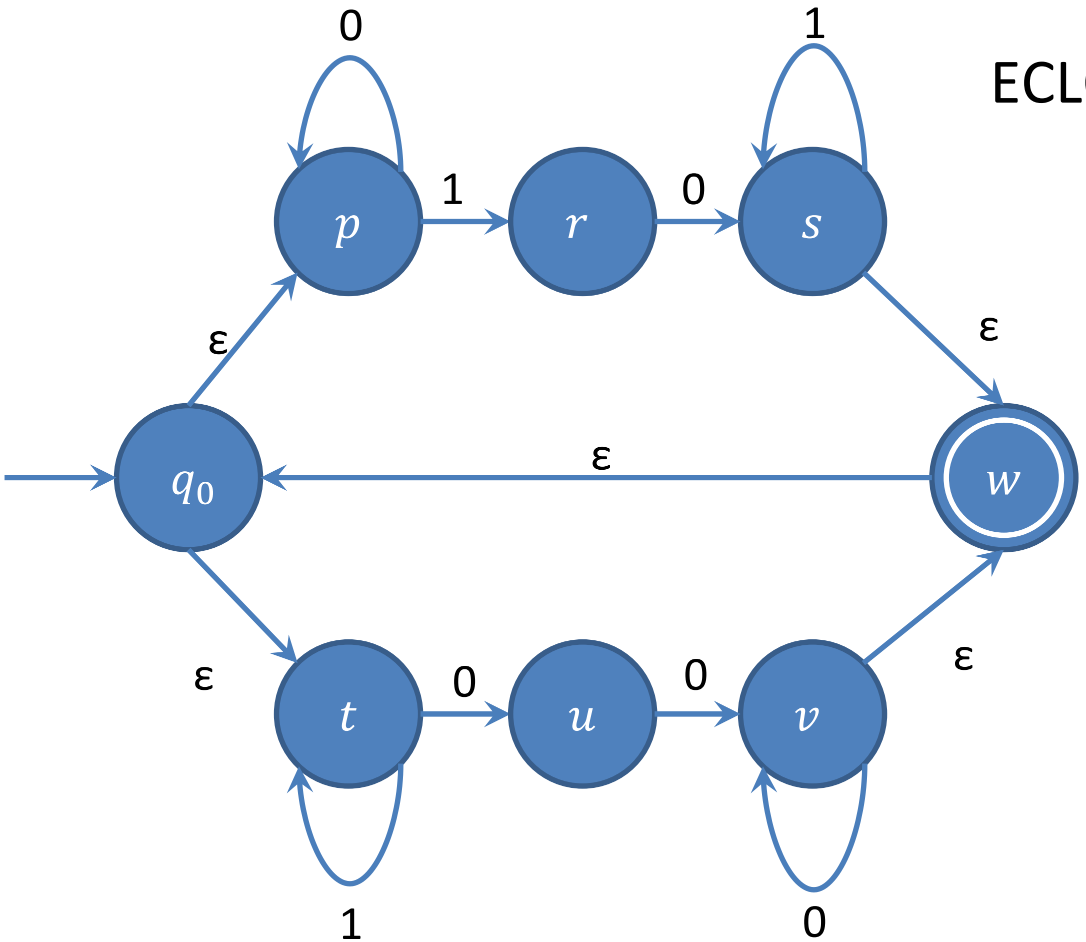
The ε-*closure* of the state q, denoted ECLOSE(q), is the set that contains q, together with all states that can be reached starting at q *by following only ε-transitions*.

In the above example:

ECLOSE(P) ={P,Q,R,S}
ECLOSE(R)={R,S}
ECLOSE(x)={x} for the remaining 5 states
{Q,Q1,R1,R2,R2}

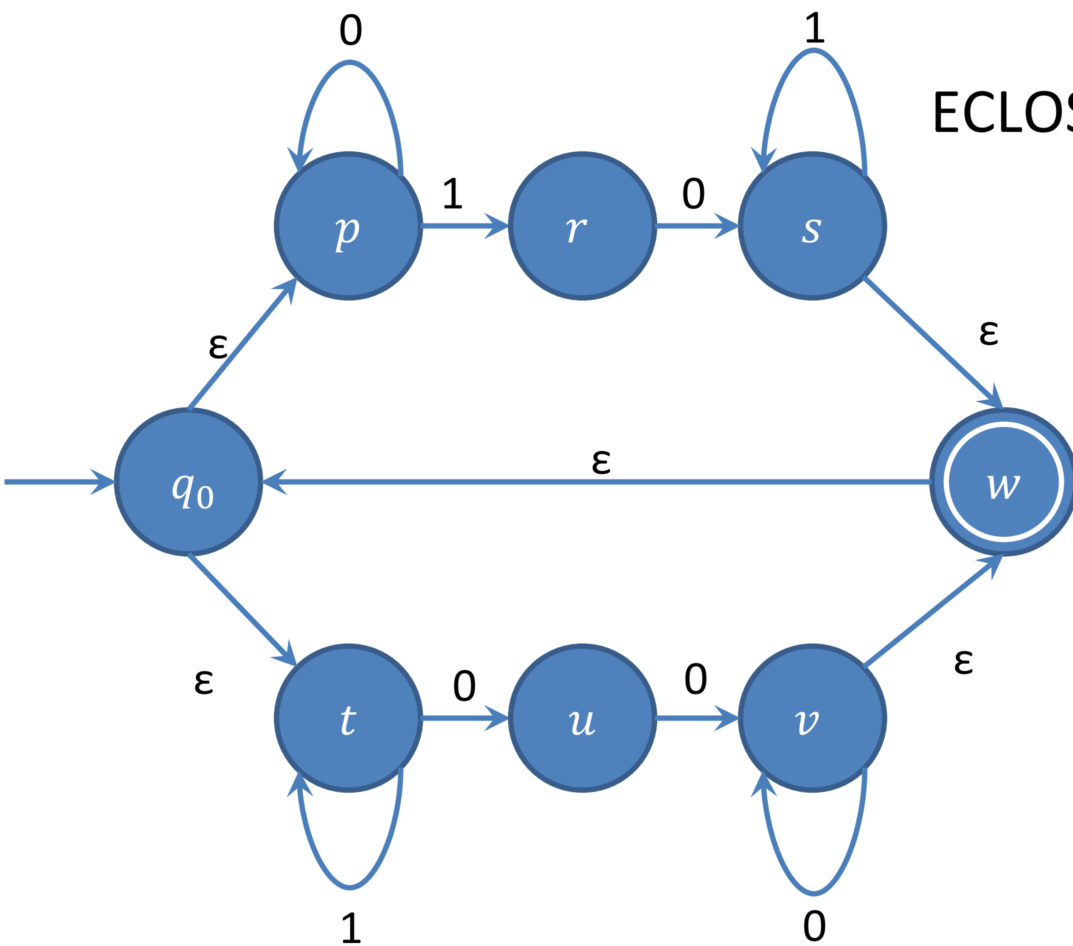


Applying Definitions of ECLOSE(S)



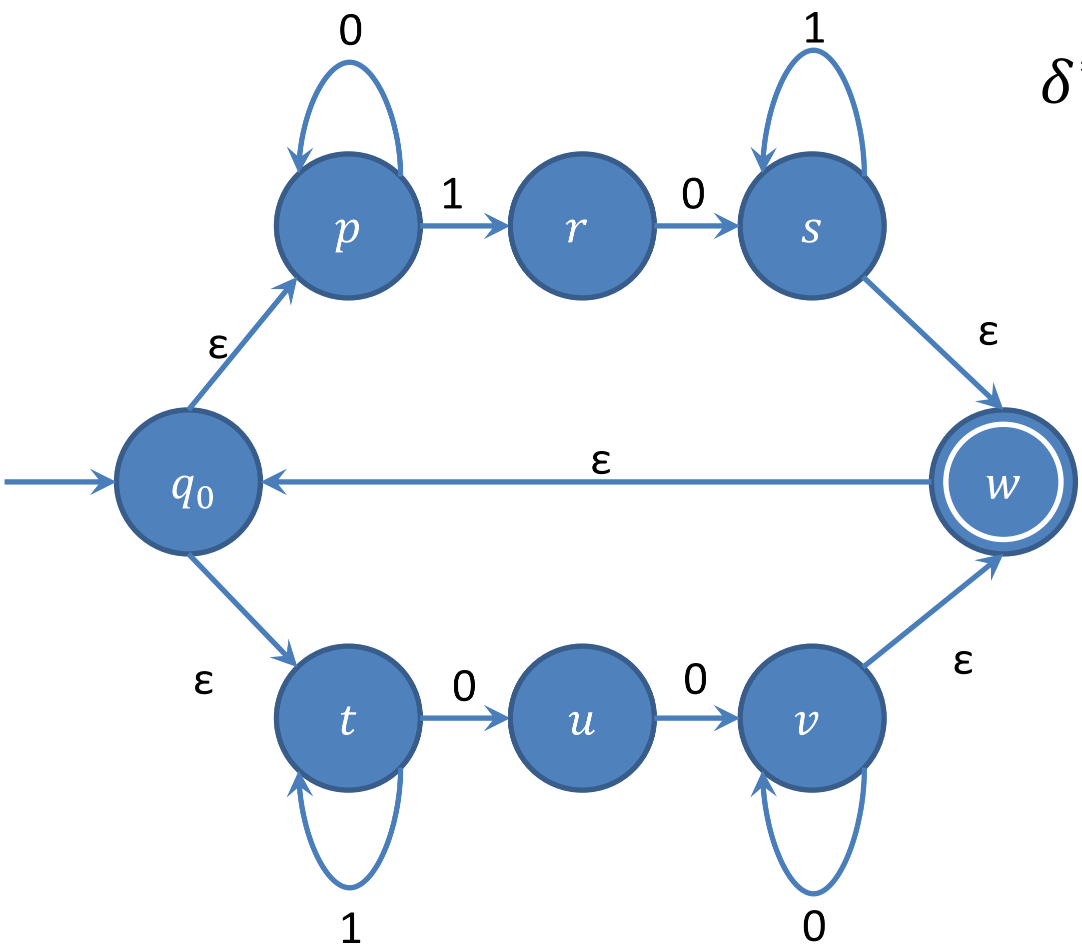
$\mathrm{ECLOSE}(\{q_0\}) = \{\, q_0, p, \; t \,\}$

Applying Definitions of ECLOSE(S)



$$\text{ECLOSE}(\{s\}) = \{\, s, w, q_0, p, t \,\}$$

Applying Definition of δ^*



$$\delta^*(q_0, \varepsilon) = \text{ECLOSE}(\{q_0\})$$
$$= \{q_0, p, t\}$$

NFA -ε to DFA

NFA - ε $\longrightarrow$ **NFA** $\longrightarrow$ **DFA**

Conversion from NFA with ε to DFA

Steps for converting NFA with ε to DFA:

Step 1: We will take the ε-closure for the starting state of NFA as a starting state of DFA.

Step 2: Find the states for each input symbol that can be traversed from the present. That means the union of transition value and their closures for each state of NFA present in the current state of DFA.

Step 3: If we found a new state, take it as current state and repeat step 2.

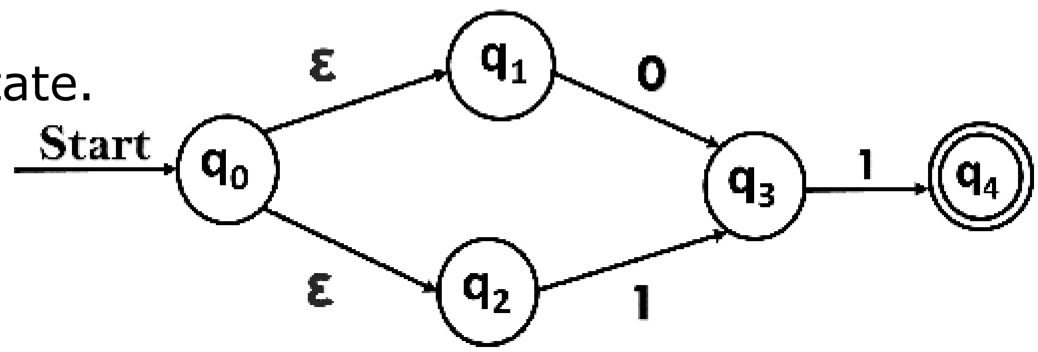
Step 4: Repeat Step 2 and Step 3 until there is no new state present in the transition table of DFA.

Step 5: Mark the states of DFA as a final state which contains the final state of NFA.

Example 1: Convert the NFA with ε into its equivalent DFA.

Solution:

Let us obtain ε-closure of each state.



ε-closure {q0} = {q0, q1, q2}
ε-closure {q1} = {q1}
ε-closure {q2} = {q2}
ε-closure {q3} = {q3}
ε-closure {q4} = {q4}

Now, let ε-closure {q0} = {q0, q1, q2} be state A. Hence

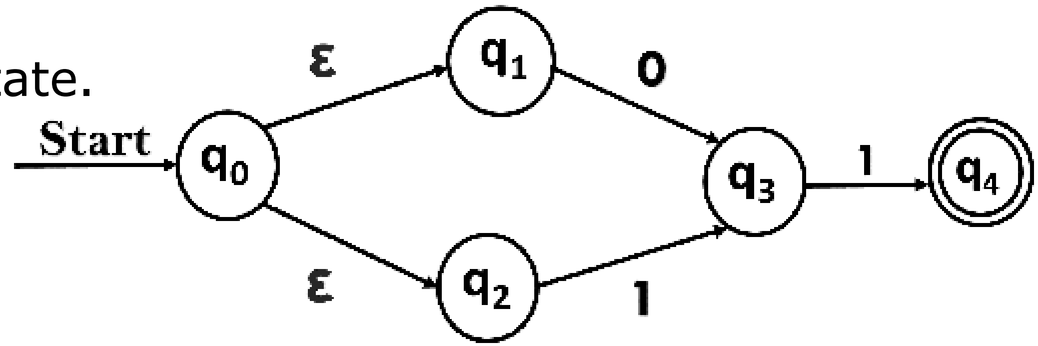
δ'(A, 0) = ε-closure {δ((q0, q1, q2), 0)}
= ε-closure {δ(q0, 0) ∪ δ(q1, 0) ∪ δ(q2, 0) }
= ε-closure {q3}
= {q3} call it as state B.

δ'(A, 1) = ε-closure {δ((q0, q1, q2), 1) }
= ε-closure {δ((q0, 1) ∪ δ(q1, 1) ∪ δ(q2, 1) }
= ε-closure {q3}
= {q3} = Same as B then Rewrite State B

Example 1: Convert the NFA with ε into its equivalent DFA.

Solution:

Let us obtain ε-closure of each state.

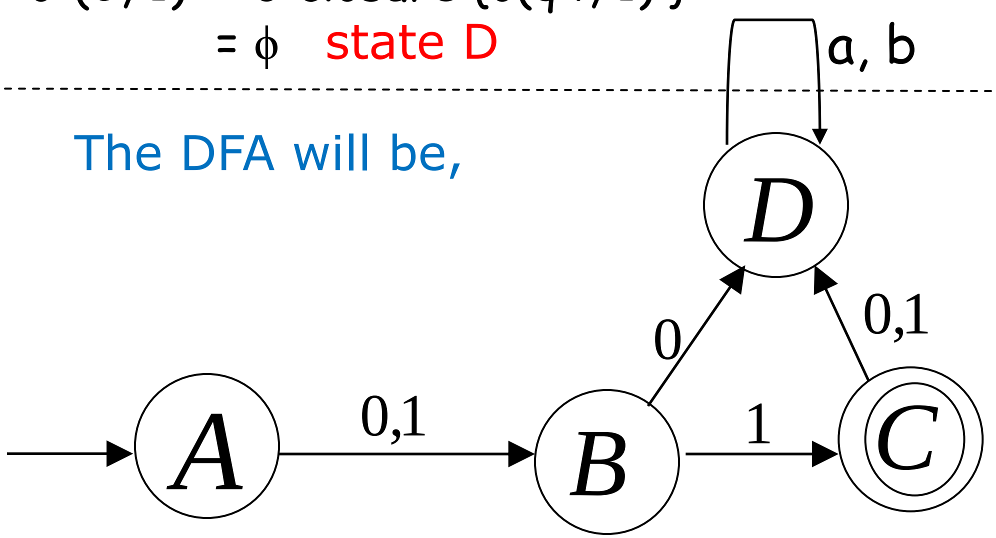


$$\delta'(B, 0) = \varepsilon\text{-closure } \{\delta(q3, 0)\}$$
$$= \phi$$
$$\delta'(B, 1) = \varepsilon\text{-closure } \{\delta(q3, 1)\}$$
$$= \varepsilon\text{-closure}\{q4\}$$
$$= \{q4\} \quad \text{New state C}$$

For state C:

$$\delta'(C, 0) = \varepsilon\text{-closure } \{\delta(q4, 0)\}$$
$$= \phi \quad \text{New state D}$$
$$\delta'(C, 1) = \varepsilon\text{-closure } \{\delta(q4, 1)\}$$
$$= \phi \quad \text{state D}$$

The DFA will be,



M = { Q, Σ, δ, q0, F}.

Q ={A,B, C, D }.

Σ = { 0,1}.

q0 = {q0}.

F = { C}.

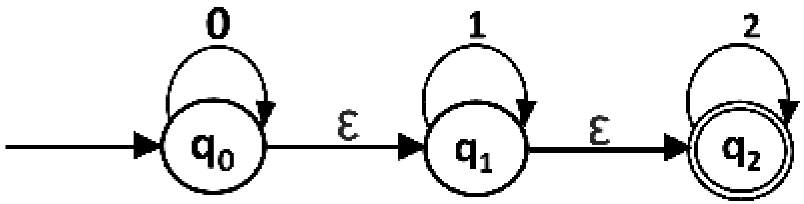
Transitional Table

δ	0	1
A	B	B
B	D	C
C*	D	D
D	D	D

Example 2: Convert the given NFA into its equivalent DFA.

Solution:

Let us obtain the ε-closure of each state.



ε-closure(q0) = {q0, q1, q2}
ε-closure(q1) = {q1, q2}
ε-closure(q2) = {q2}

Now we will obtain δ' transition.
Let ε-closure(q0) = {q0, q1, q2} call it as **state A**.

δ'(A, 0) = ε-closure{δ((q0, q1, q2), 0)}
= ε-closure{δ(q0, 0) ∪ δ(q1, 0) ∪ δ(q2,0)}
= ε-closure{q0}
= {q0, q1, q2} Same as state A

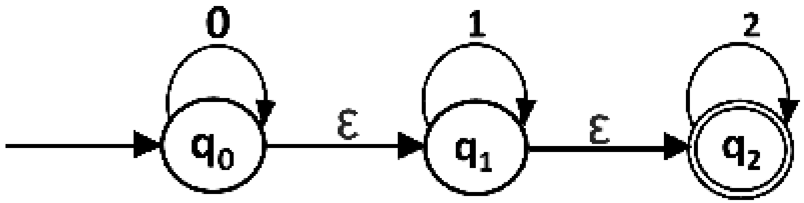
δ'(A, 1) = ε-closure{δ((q0, q1, q2), 1)}
= ε-closure{δ(q0, 1) ∪ δ(q1, 1) ∪ δ(q2,1)}
= ε-closure{q1}
= {q1, q2} call it as state B

δ'(A, 2) = ε-closure{δ((q0, q1, q2), 2)}
= ε-closure{δ(q0, 2) ∪ δ(q1, 2) ∪ δ(q2,2)}
= ε-closure{q2}
= {q2} call it state C

Example 2: Convert the given NFA into its equivalent DFA.

Solution: Continue....

Now we will find the transitions on states B and C for each input. Hence



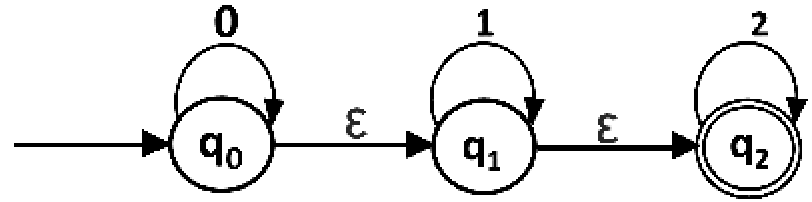
$$
\begin{aligned}
\delta'(B, 0) &= \varepsilon\text{-closure}\{\delta((q1, q2), 0)\} \\
&= \varepsilon\text{-closure}\{\delta(q1, 0) \cup \delta(q2,0)\} \\
&= \varepsilon\text{-closure}\{\phi\} \\
&= \phi \text{ ->State D}
\end{aligned}
$$

$$
\begin{aligned}
\delta'(B, 1) &= \varepsilon\text{-closure}\{\delta((q1, q2), 1)\} \\
&= \varepsilon\text{-closure}\{\delta(q1, 1) \cup \delta(q2,1)\} \\
&= \varepsilon\text{-closure}\{q1\} \\
&= \{q1, q2\} \qquad \text{i.e. state B itself}
\end{aligned}
$$

$$
\begin{aligned}
\delta'(B, 2) &= \varepsilon\text{-closure}\{\delta((q1, q2), 2)\} \\
&= \varepsilon\text{-closure}\{\delta(q1, 2) \cup \delta(q2, 2)\} \\
&= \varepsilon\text{-closure}\{q2\} \\
&= \{q2\} \qquad \text{i.e. state C itself}
\end{aligned}
$$

Example 2: Convert the given NFA into its equivalent DFA.

Solution: Continue….



Now we will obtain transitions for C:

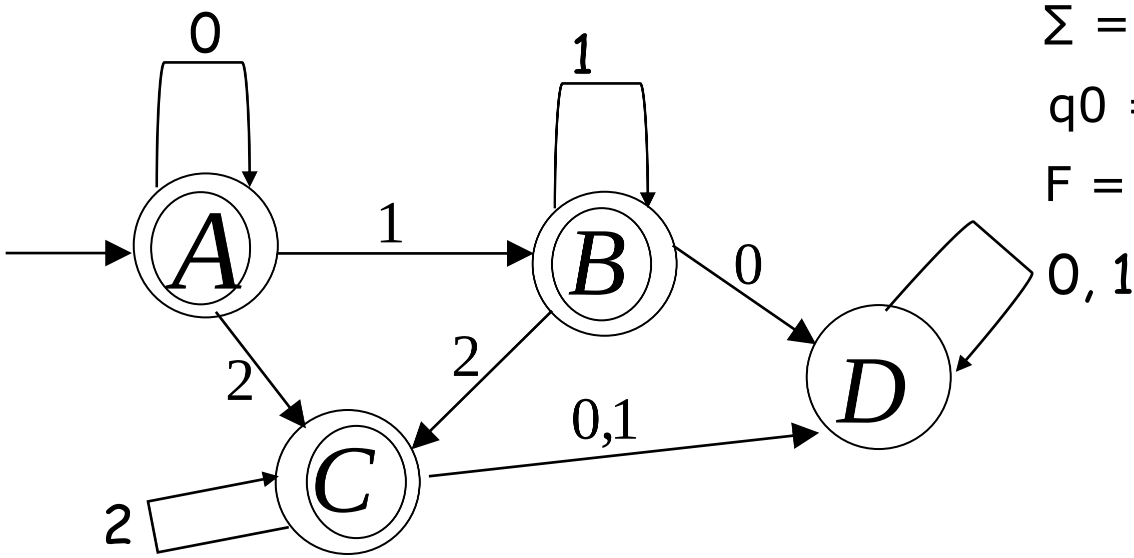
$\delta'(C, 0) = \varepsilon\text{-closure}\{\delta(q2, 0)\}$
$= \varepsilon\text{-closure}\{\phi\}$
$= \phi \rightarrow D$

$\delta'(C, 1) = \varepsilon\text{-closure}\{\delta(q2, 1)\}$
$= \varepsilon\text{-closure}\{\phi\}$
$= \phi \rightarrow D$

$\delta'(C, 2) = \varepsilon\text{-closure}\{\delta(q2, 2)\}$
$= \{q2\}$

As A = {q0, q1, q2} in which final state q2 lies hence A is final state.
B = {q1, q2} in which the state q2 lies hence B is also final state.
C = {q2}, the state q2 lies hence C is also a final state..

Hence the DFA is



$M = \{ Q, \Sigma, \delta, q0, F\}$.

$Q = \{A, B, C, D\}$.

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

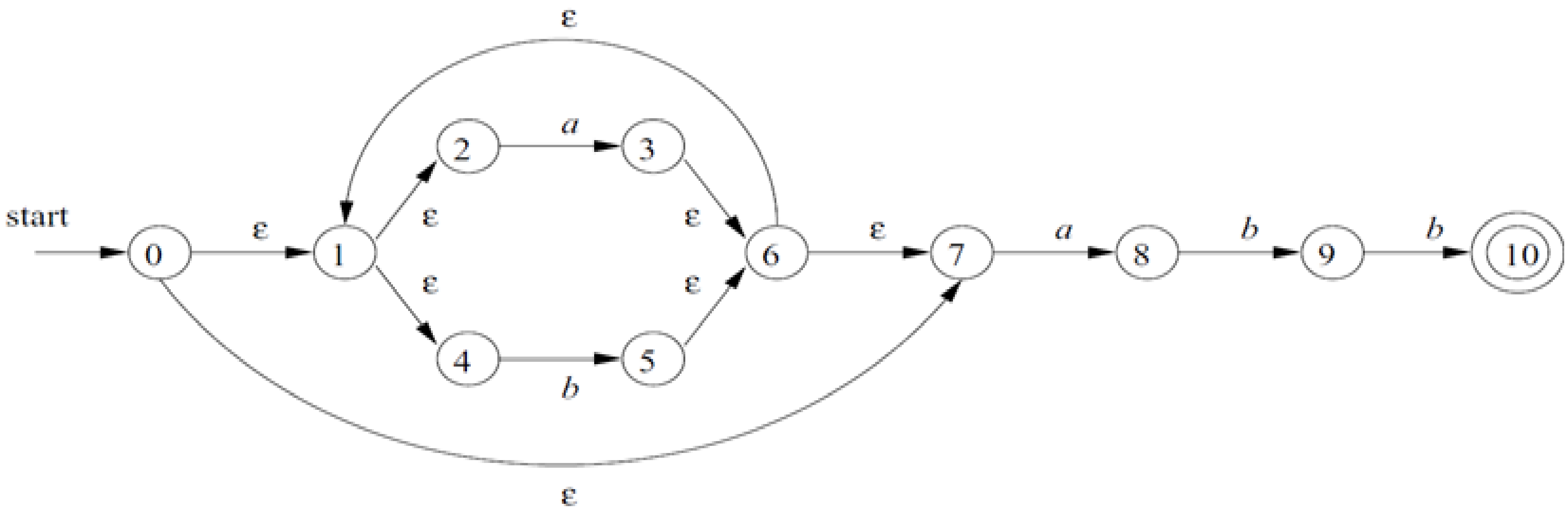
$q0 = \{q0\}$.

$F = \{C\}$.

Transitional Table

δ	0	1
A*	B	B
B*	D	C
C*	D	D
D	D	D

3: Convert the given NFA into its equivalent DFA



ε-closure{0} ={0,1,2,4,7} ->A
δ(A, a)= {3,8}=ε-closure{3,8}={3,6,7,1,2,4,8}=>{1,2,3,4,6,7,8}->B
δ(A, b)= {5}=ε-closure{5}={5,6,7,1,2,4}=>{1,2,4,5,6,7}->C

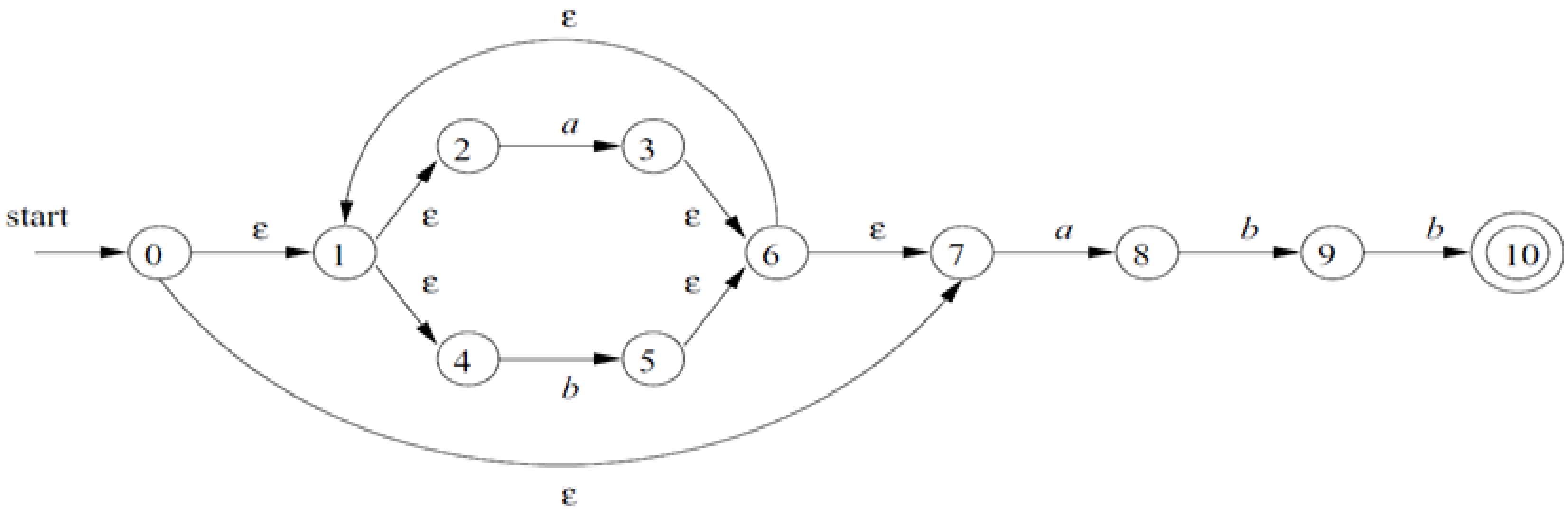
δ(B, a)= {3,8}=ε-closure{3,8}={3,6,7,1,2,4,8}=>{1,2,3,4,6,7,8}->same as B
δ(B, b)= {5,9}=ε-closure{5,9}={1,2,4,5,6,7,9}->D

δ(C, a)= {3,8}=ε-closure{3,8}={3,6,7,1,2,4,8}=>{1,2,3,4,6,7,8}->same as B
δ(C, b)= {5}=ε-closure{5}={5,6,7,1,2,4}=>{1,2,4,5,6,7}-> same as C

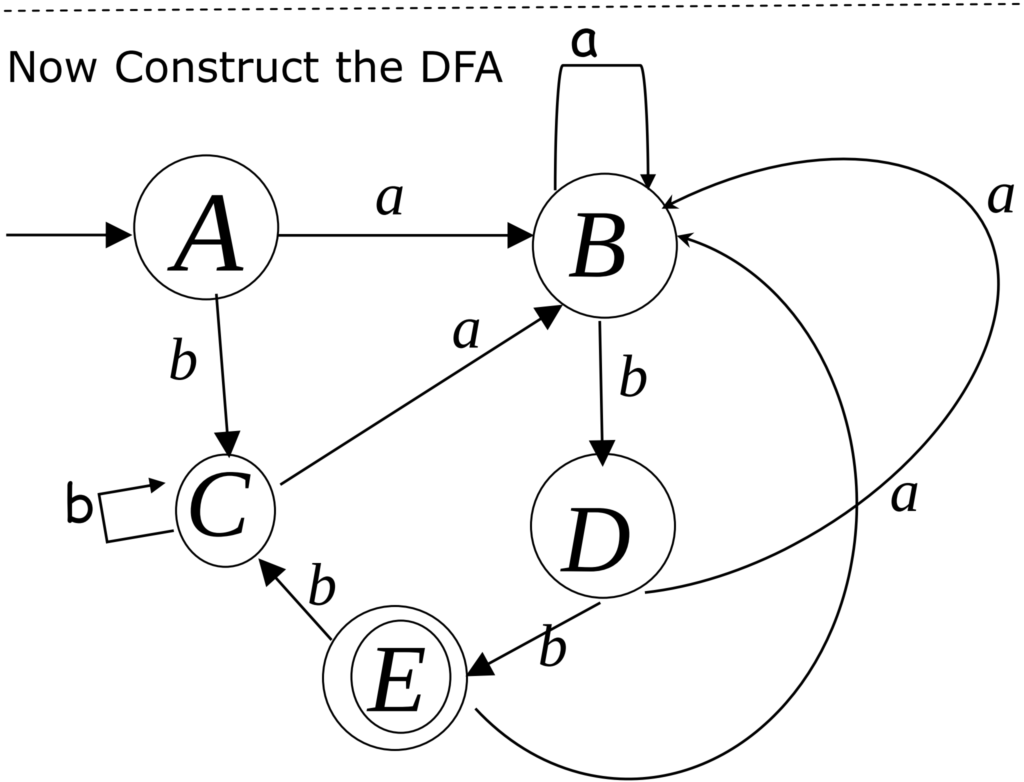
δ(D, a)= {3,8}=ε-closure{3,8}={3,6,7,1,2,4,8}=>{1,2,3,4,6,7,8}->same as B
δ(D, b)= {5}=ε-closure{5,10}={5,6,7,1,2,4,10}=>{1,2,4,5,6,7,10}-> E

δ(E, a)= {3,8}=ε-closure{3,8}={3,6,7,1,2,4,8}=>{1,2,3,4,6,7,8}->same as B
δ(E, b)= {5}=ε-closure{5}={5,6,7,1,2,4}=>{1,2,4,5,6,7}-> same as C

3: Convert the given NFA into its equivalent DFA



Now Construct the DFA

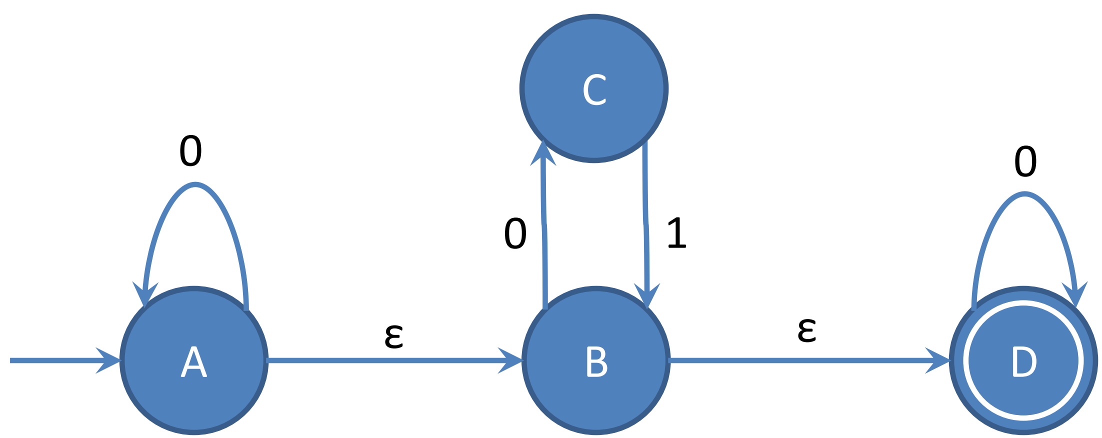


Transitional Table

δ	a	b
->A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C

Conversion from NFA- ε to DFA

q	$\delta(q, \boldsymbol{\varepsilon})$	$\delta(q, 0)$	$\delta(q, 1)$
A	{B}	{A}	ϕ
B	{D}	{C}	ϕ
C	ϕ	ϕ	{B}
D	ϕ	{D}	ϕ

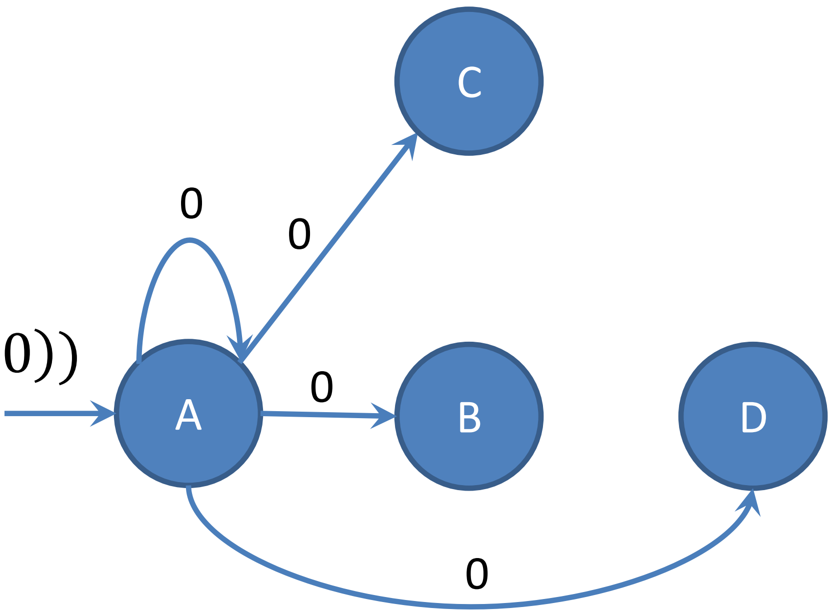


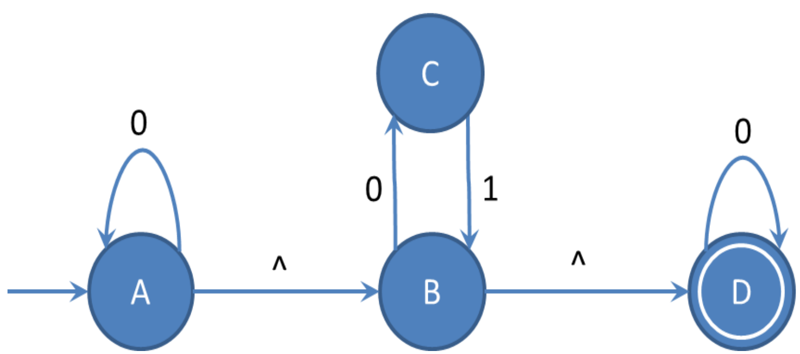
Conversion from NFA- ε to DFA

Step 1: To convert NFA - ε to NFA

$$\delta^*(A, \varepsilon) = ECLOSE\{A\} = \{A, B, D\}$$

$$\begin{aligned}\delta^*(A, 0) &= \varepsilon cls(\delta(A, 0) \cup \delta(B, 0) \cup \delta(D, 0)) \\ &= \varepsilon cls(\{A, C, D\}) \\ &= \{A, B, C, D\}\end{aligned}$$

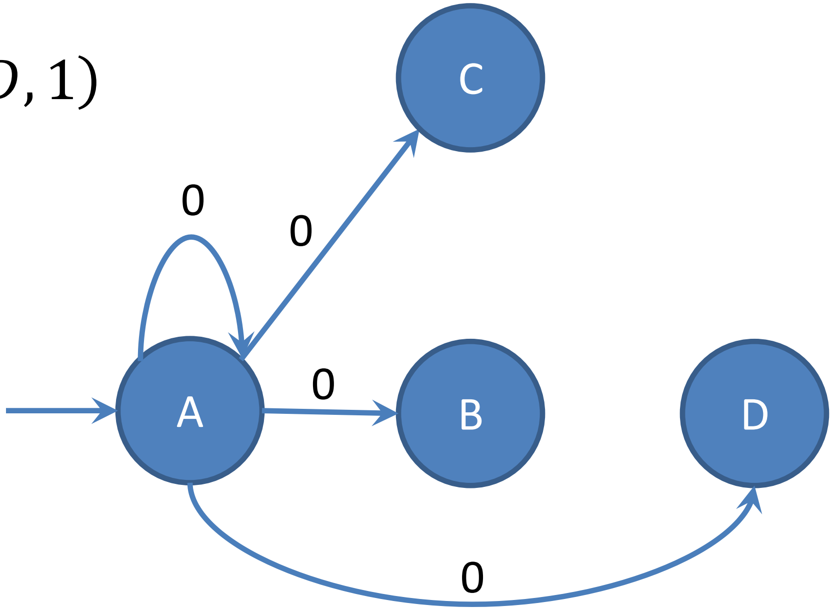




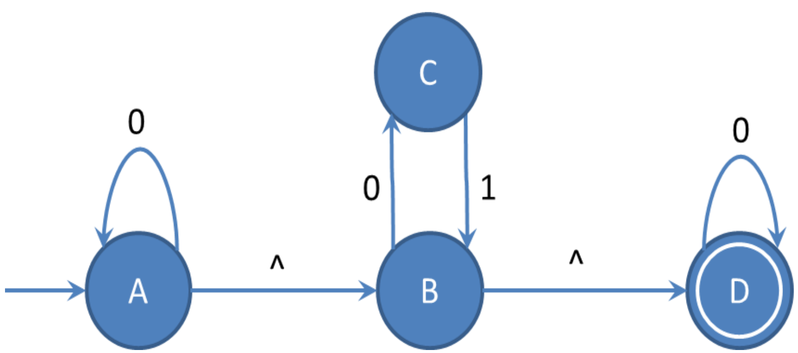
q	$\delta^*(q, 0)$	$\delta^*(q, 1)$
A	$\{A, B, C, D\}$	

Conversion from NFA- ε to DFA

$$\delta^*(A,1) = \varepsilon cls\big(\delta(A,1)\big) \cup \delta(B,1) \cup \delta(D,1)$$
$$= \varepsilon cls(\phi)$$
$$= \phi$$



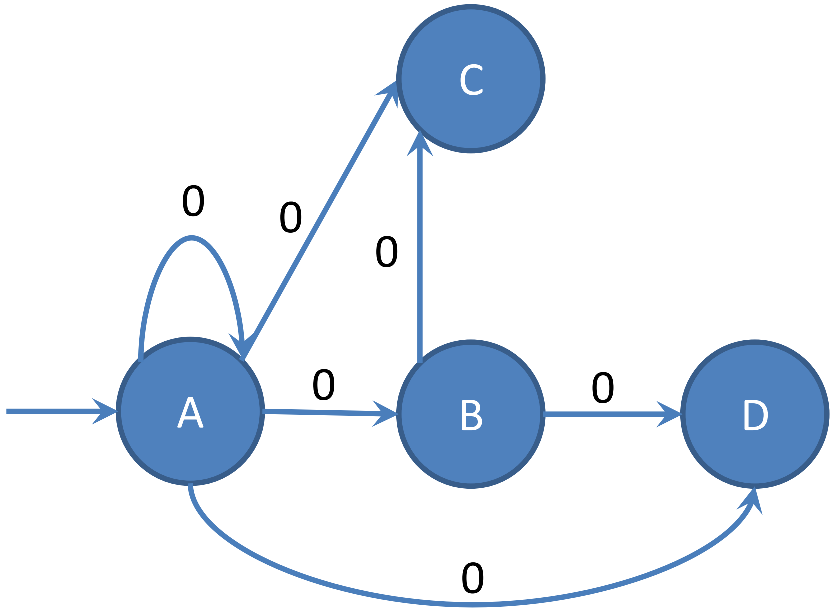
q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	



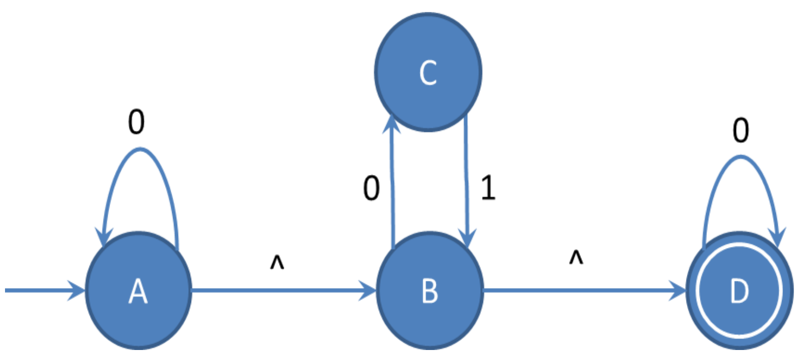
Conversion from NFA- ε to DFA

$\color{red}{\delta^*(B,\varepsilon)} = \varepsilon cls\{B\} = \{B, D\}$

$$\delta^*(B,0) = \Lambda(\delta(B,0) \cup \delta(D,0))$$
$$= \Lambda(\{C, D\})$$
$$= \{C, D\}$$

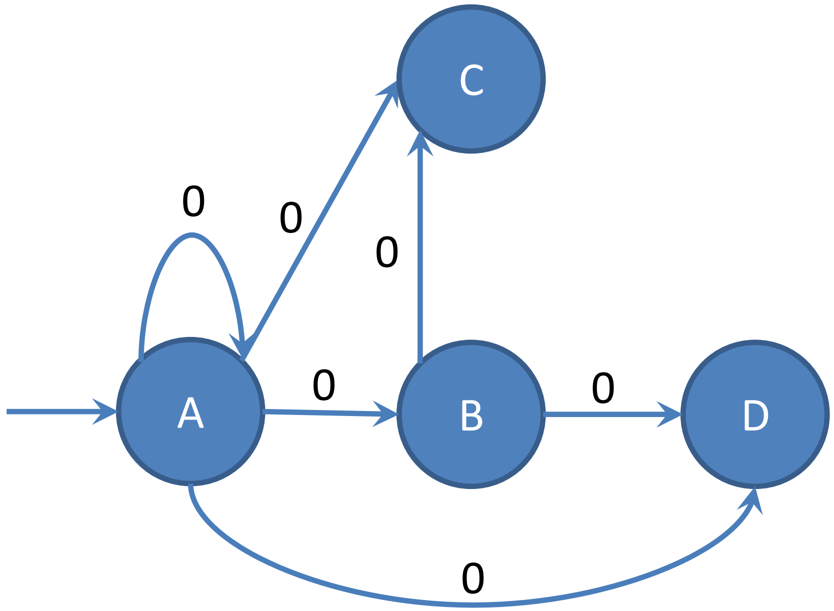


q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A, B, C, D\}$	ϕ
B		

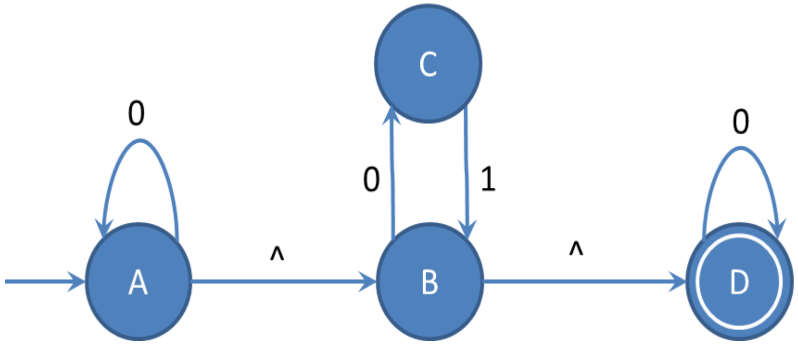


Conversion from NFA- ε to DFA

$$\delta^*(B,1) = \varepsilon cls(\delta(B,1) \cup \delta(D,1))$$
$$= \varepsilon cls(\phi)$$
$$= \phi$$



q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	ϕ
B	$\{C,D\}$	



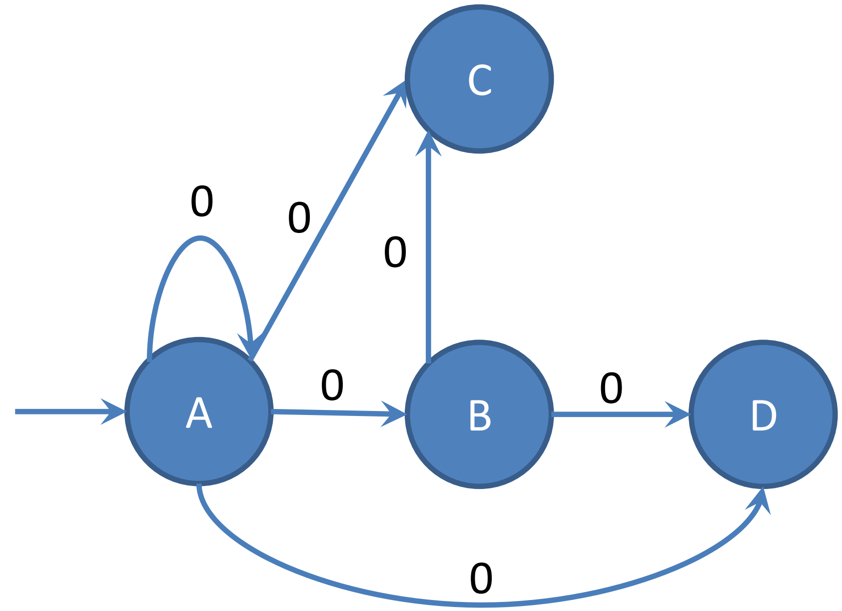
Conversion from NFA- ε to DFA

$$\delta^*(C, \varepsilon) = \varepsilon cls\{C\} = \{C\}$$

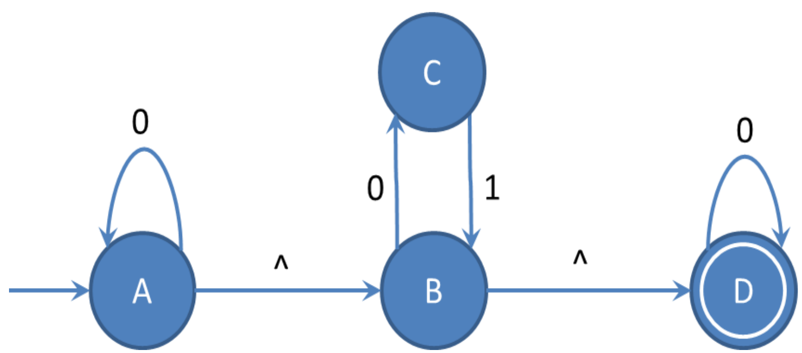
$$\delta^*(C, 0) = \varepsilon cls(\delta(C, 0))$$

$$= \varepsilon cls(\phi)$$

$$= \phi$$

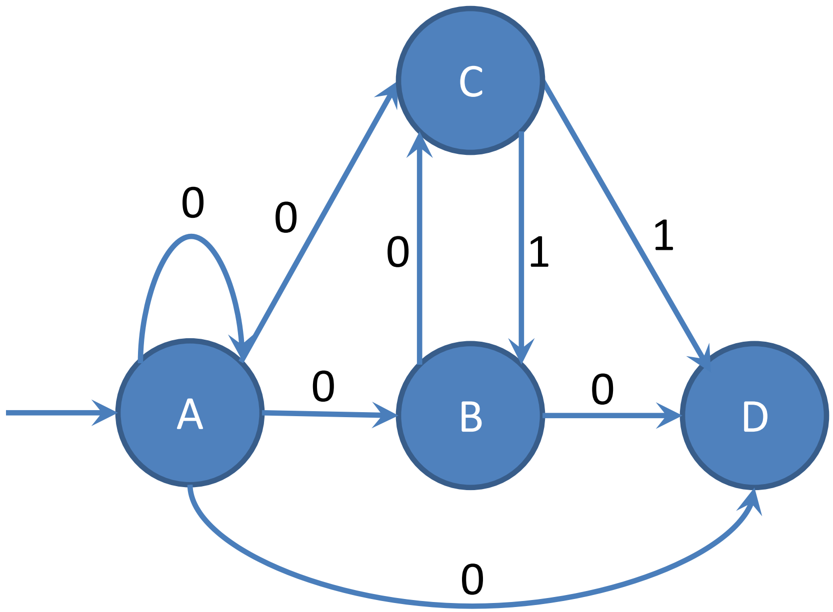


q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	ϕ
B	$\{C,D\}$	ϕ
C		

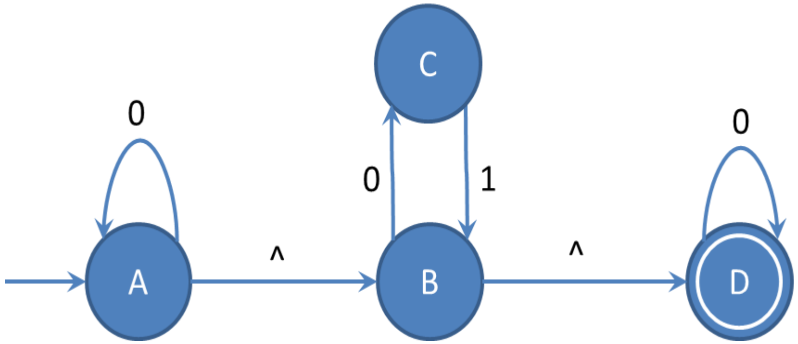


Conversion from NFA- ε to DFA

$$\delta^*(C,1) = \varepsilon cls(\delta(C,1))$$
$$= \varepsilon cls(\{B\})$$
$$= \{B,D\}$$



q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	ϕ
B	$\{C,D\}$	ϕ
C	ϕ	



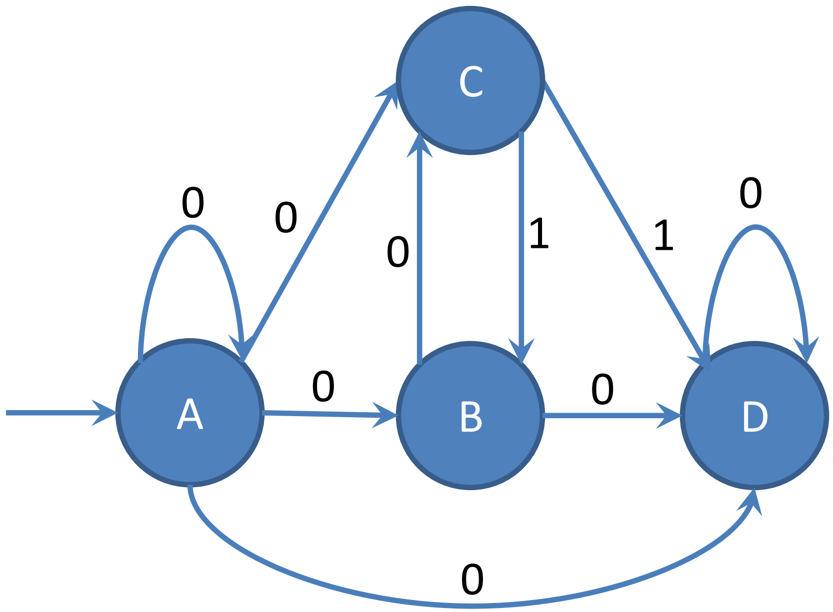
Conversion from NFA- ε to DFA

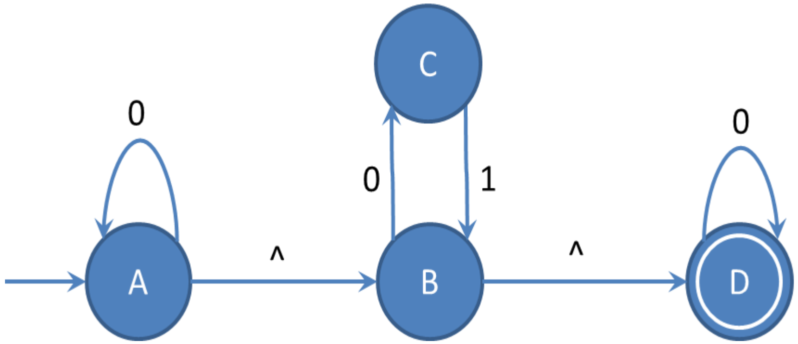
$\delta^*(D, \varepsilon) \quad = \Lambda\{D\} = \{D\}$

$\delta^*(D, 0) \quad = \varepsilon cls(\delta(D, 0))$

$= \varepsilon cls(\{D\})$

$= \{D\}$

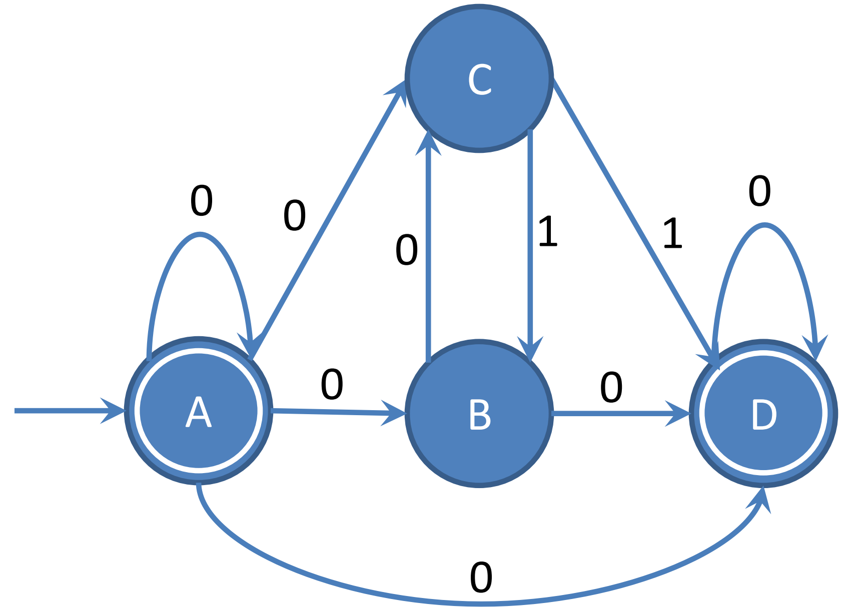




q	$\delta^*(q, 0)$	$\delta^*(q, 1)$
A	$\{A, B, C, D\}$	ϕ
B	$\{C, D\}$	ϕ
C	ϕ	$\{B, D\}$
D		

Conversion from NFA- ε to DFA

$$\delta^*(D,1) = \varepsilon(\delta(D,1))$$
$$= \varepsilon(\phi)$$
$$= \phi$$



q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	ϕ
B	$\{C,D\}$	ϕ
C	ϕ	$\{B,D\}$
D	$\{D\}$	

Accepting State $= \begin{cases} A \cup \{q_0\} & \text{if } \varepsilon cls(\{q_0\}) \cap A \neq \phi \text{ in } M \\ A & \text{otherwise} \end{cases}$

Resulting NFA

Conversion from NFA- ε to DFA

q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	ϕ
B	$\{C,D\}$	ϕ
C	ϕ	$\{B,D\}$
D	$\{D\}$	ϕ

Step 2: To convert NFA to FA

$\delta(\{A\},0) = \{A,B,C,D\}$

$\delta(\{A\},1) = \phi$

$\delta(\{A,B,C,D\},0) = (\delta(A,0) \cup \delta(B,0) \cup \delta(C,0) \cup \delta(D,0))$

$= \{A,B,C,D\} \cup \{C,D\} \cup \{\phi\} \cup \{D\} = \{A,B,C,D\}$

$\delta(\{A,B,C,D\},1) = (\delta(A,1) \cup \delta(B,1) \cup \delta(C,1) \cup \delta(D,1))$

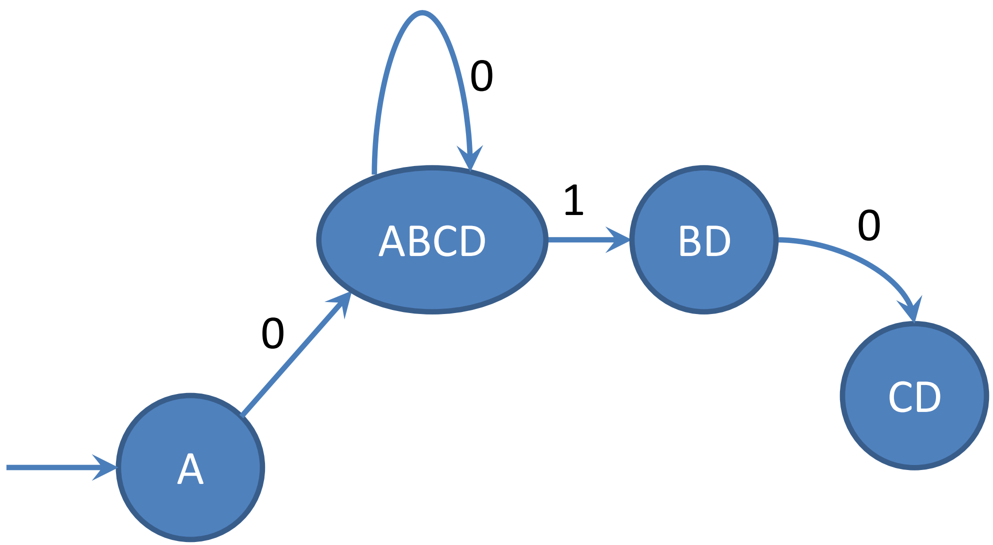
$= \{B,D\}$

$\delta(\{B,D\},0) = (\delta(B,0) \cup \delta(D,0))$

$= \{C,D\}$

$\delta(\{B,D\},1) = (\delta(B,1) \cup \delta(D,1))$

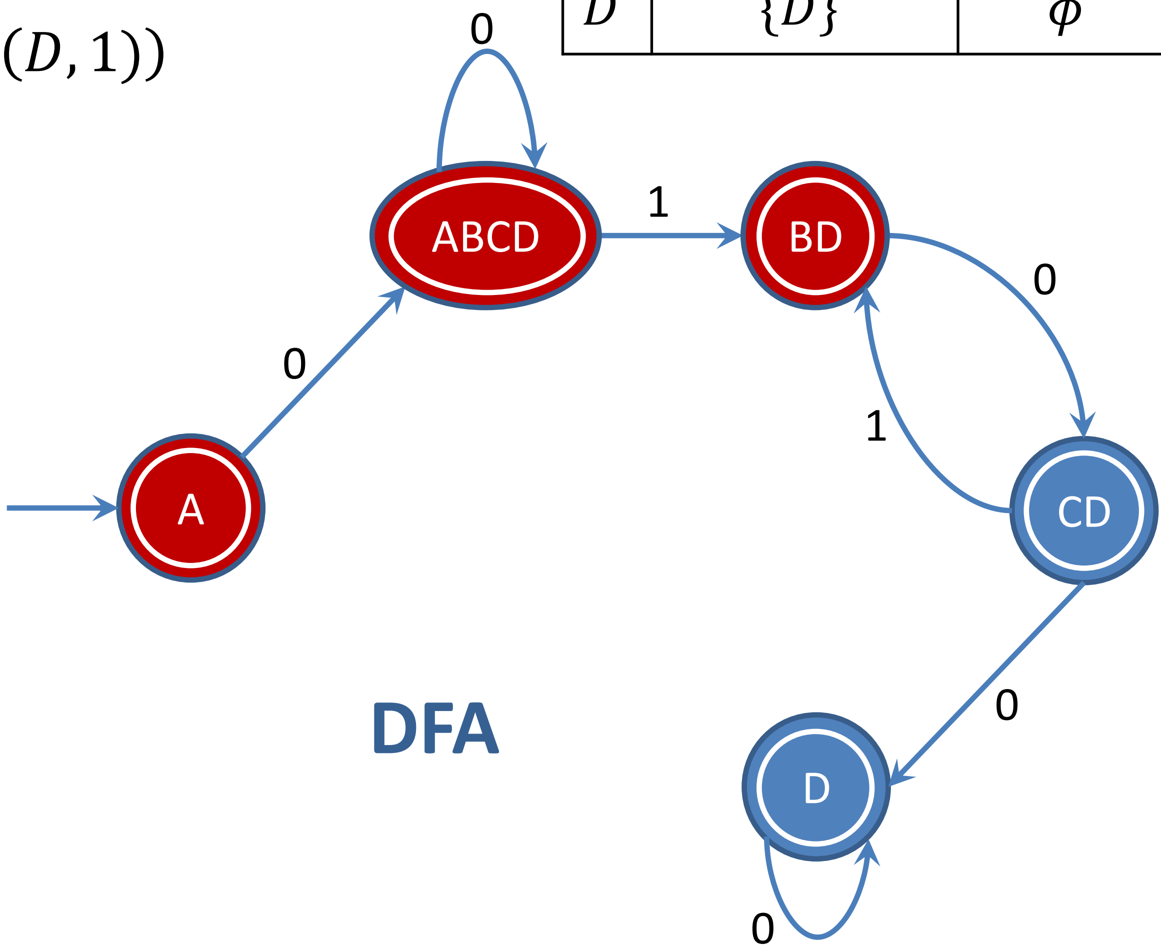
$= \phi$



Conversion from NFA- ε to DFA

q	$\delta^*(q,0)$	$\delta^*(q,1)$
A	$\{A,B,C,D\}$	ϕ
B	$\{C,D\}$	ϕ
C	ϕ	$\{B,D\}$
D	$\{D\}$	ϕ

$$\delta(\{C,D\},0) = (\delta(C,0) \cup \delta(D,0))$$
$$= \{D\}$$
$$\delta(\{C,D\},1) = (\delta(C,1) \cup \delta(D,1))$$
$$= \{B,D\}$$
$$\delta(\{D\},0) = \{D\}$$
$$\delta(\{D\},1) = \phi$$



DFA

Difference between DFA and NFA

SR.NO.	DFA	NFA
1	DFA stands for Deterministic Finite Automata.	NFA stands for Nondeterministic Finite Automata.
2	For each symbolic representation of the alphabet, there is only one state transition in DFA.	No need to specify how does the NFA react according to some symbol.
3	DFA cannot use Empty String transition.	NFA can use Empty String transition.
4	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
5	In DFA, the next possible state is distinctly set.	In NFA, each pair of state and input symbol can have many possible next states.
6	DFA is more difficult to construct.	NFA is easier to construct.
7	DFA rejects the string in case it terminates in a state that is different from the accepting state.	NFA rejects the string in the event of all branches dying or refusing the string.
8	Time needed for executing an input string is less.	Time needed for executing an input string is more.
9	All DFA are NFA.	Not all NFA are DFA.
10	DFA requires more space.	NFA requires less space then DFA.

Converting NDFSM to DFSM

Steps for converting NDFSM to DFSM

- **Step 1**: Initially Q' = ϕ

- **Step 2**: Add q_0 of NFA to Q'. Then find the transitions from this start state.

- **Step 3**: In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.

- **Step 4**: In DFA, the final state will be all the states which contain F(final states of NFA)

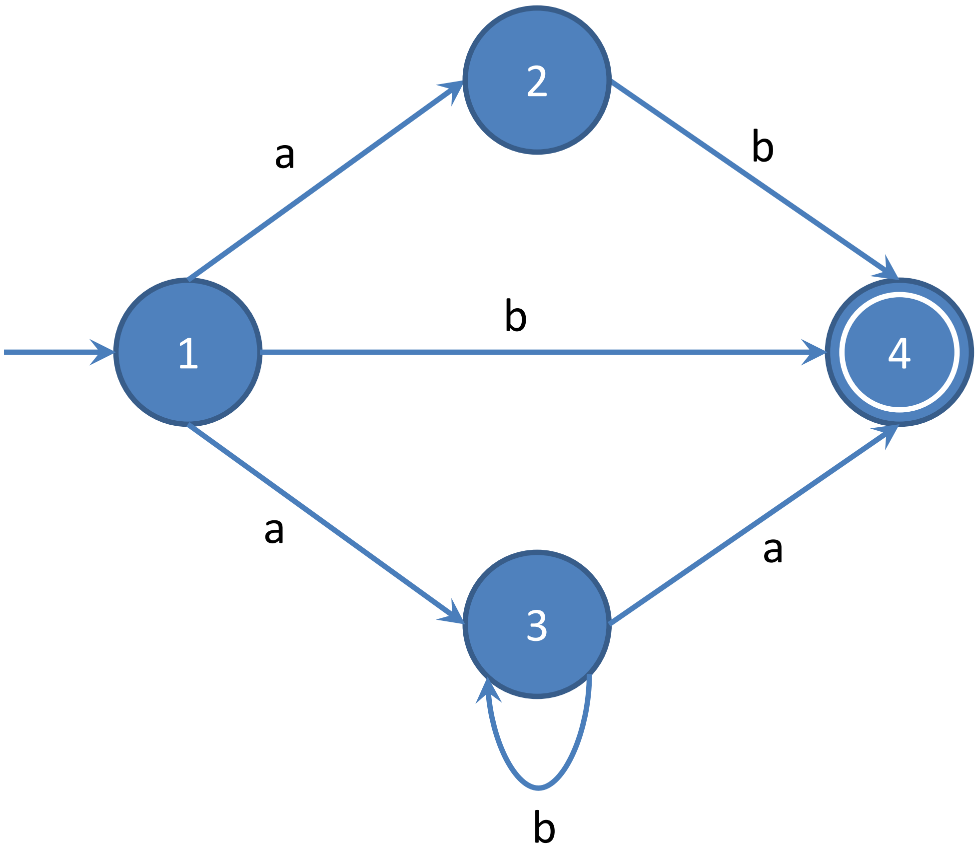
Convert the given NDFSM to DFSM

Step1: The start state of NDFSM is the start state of DFSM.

Step2: if $\delta(q, a)$ $=\{q_1,q_2,q_3,\ldots\ldots\ldots\ldots\ldots\ldots q_n\}$ is the transitions defined for NDFSM, then $[q_1,q_2,q_3,\ldots\ldots\ldots\ldots q_n]$is a single state of in DFSM from q on input symbol a.

Step3: $[q_1,q_2,q_3,\ldots\ldots\ldots\ldots\ldots\ldots q_n]$ is the final state of DFSM if $[q_1,q_2,q_3,\ldots\ldots\ldots\ldots q_n]$ contains a final state of NDFSM

Example 1: Conversion from NFA to FA



NDFSM

q	$\delta(q,a)$	$\delta(q,b)$
1	{2,3}	{4}
2	$\{\phi\}$	{4}
3	{4}	{3}
4	$\{\phi\}$	$\{\phi\}$

Transition Table

Example 1: Conversion from NFA to FA

$\delta_1(1, a) = \{2,3\}$

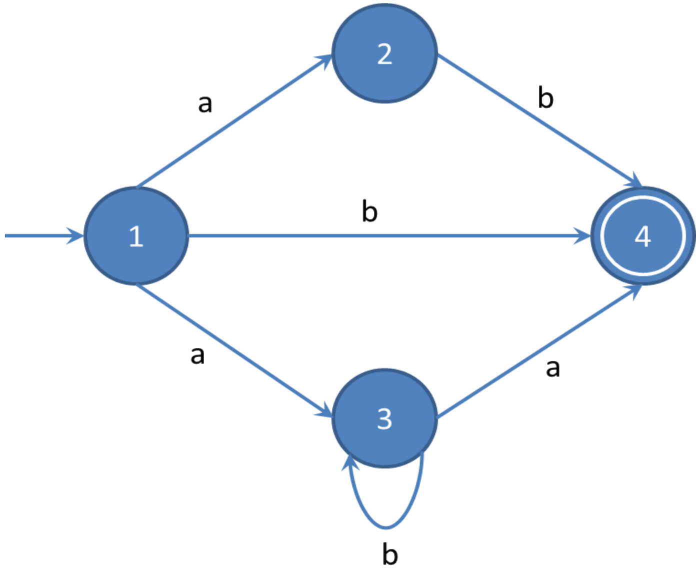
$\delta_1(1, b) = \{4\}$

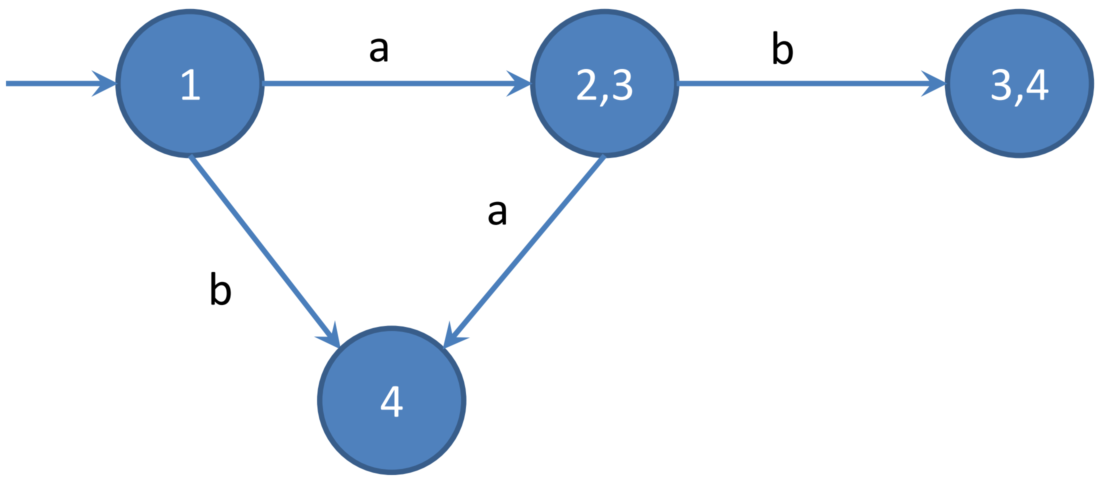
$$\delta_1(\{2,3\}, a) = \delta(2, a) \cup \delta(3, a) = \{4\}$$

$$\delta_1(\{2,3\}, b) = \delta(2, b) \cup \delta(3, b) = \{3,4\}$$

$\delta_1(4, a) = \{\emptyset\}$

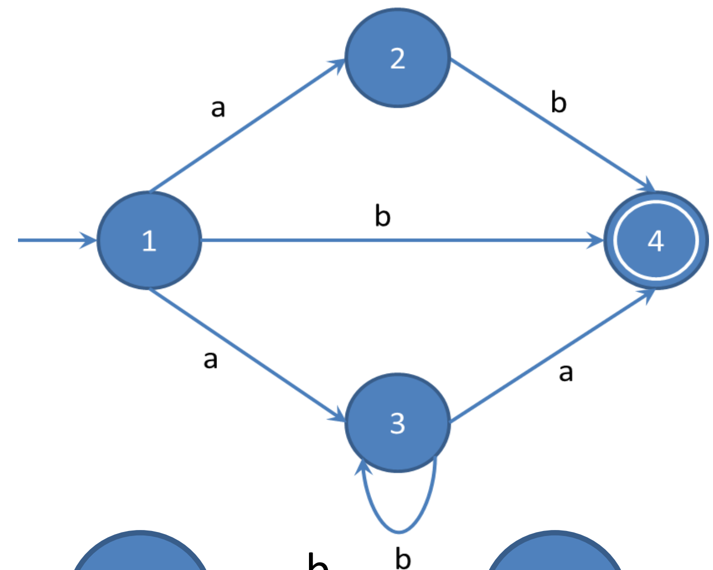
$\delta_1(4, b) = \{\emptyset\}$

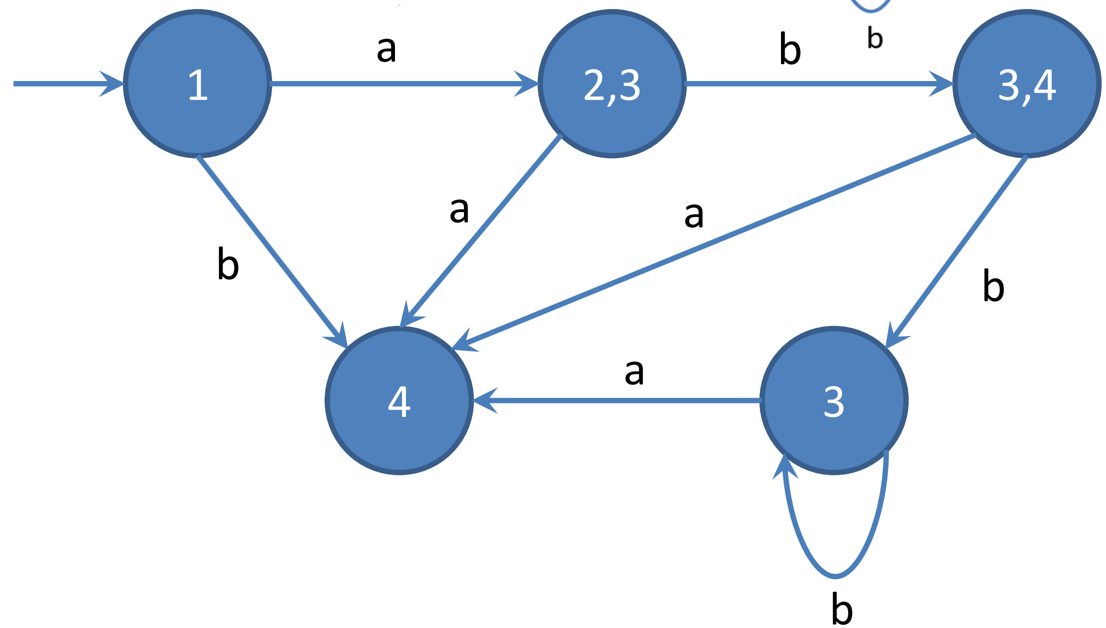




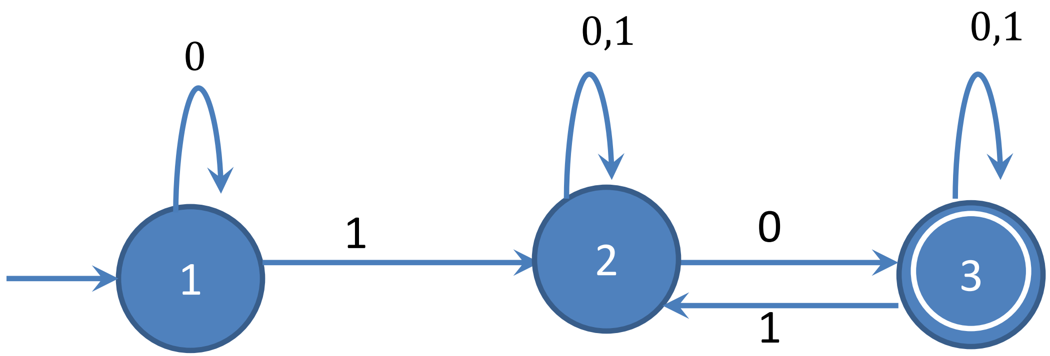
Example 1: Conversion from NFA to FA

$$\delta_1(\{3,4\}, a) = \delta(3, a) \cup \delta(4, a)$$
$$= \{4\}$$
$$\delta_1(\{3,4\}, b) = \delta(3, b) \cup \delta(4, b)$$
$$= \{3\}$$
$$\delta_1(3, a) = \{4\}$$
$$\delta_1(3, b) = \{3\}$$





Ex-2:Convert the given NDFSM to DFSM



NDFSM

q	$\delta(q,0)$	$\delta(q,1)$
→1	{1}	{2}
2	{2,3}	{2}
3*	{3}	{2,3}

Transition Table

Ex-2: Conversion from NFA to FA

$$\delta_1(1,0) = \{1\}$$

$$\delta_1(1,1) = \{2\}$$

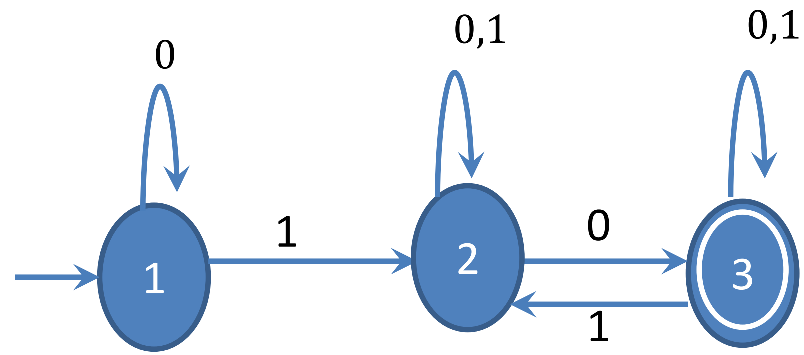
$$\delta_1(2,0) = \{2,3\}$$

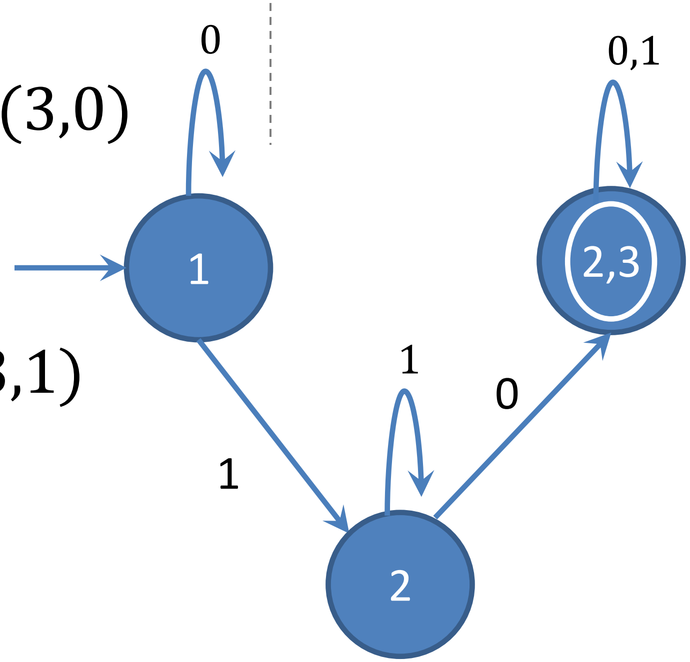
$$\delta_1(2,1) = \{2\}$$

$$\delta_1(\{2,3\},0) = \delta(2,0) \cup \delta(3,0) = \{2,3\}$$

$$\delta_1(\{2,3\},1) = \delta(2,1) \cup \delta(3,1) = \{2,3\}$$

NDFSM

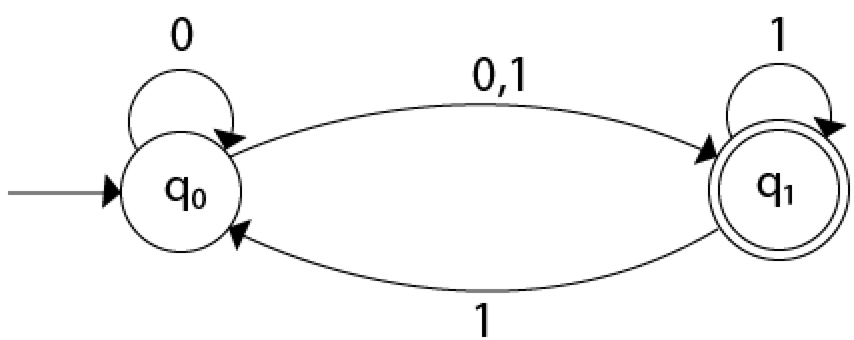




DFSM

Ex- 3: Conversion from NFA to DFA

δ	Input	
State	0	1
->q_0	$\{q_0, q_1\}$	$\{q_1\}$
$* \; q_1$	$\{\phi\}$	$\{q_0, q_1\}$

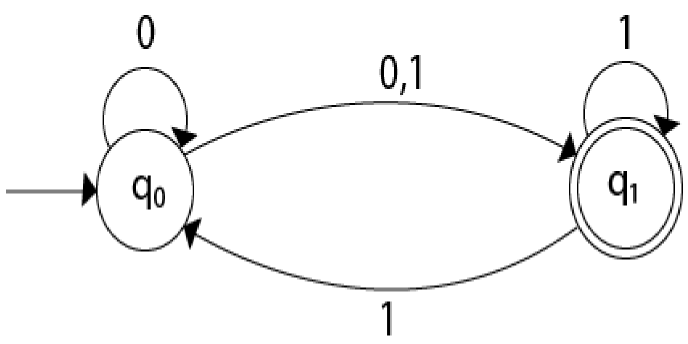


Ex- 3: Conversion from NFA to DFA

Now we will obtain δ' transition for state q0.

1.δ'([q0], 0) = {q0, q1}
= [q0, q1] (**new** state generated)

δ'([q0], 1) = {q1} = [q1]



The δ' transition for state q1 is obtained as:

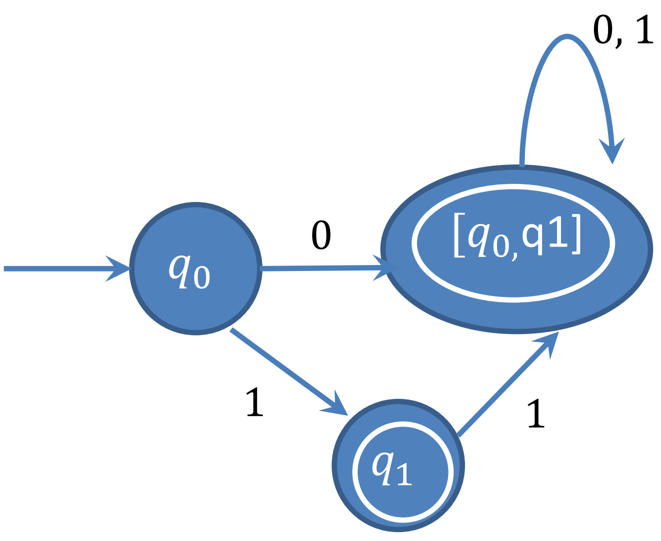
2. δ'([q1], 0) = ϕ
δ'([q1], 1) = [q0, q1]

Now we will obtain δ' transition on [q0, q1].

3.δ'([q0, q1], 0) = δ(q0, 0) ∪ δ(q1, 0)
= {q0, q1} ∪ ϕ
= {q0, q1}
= [q0, q1]

Similarly,

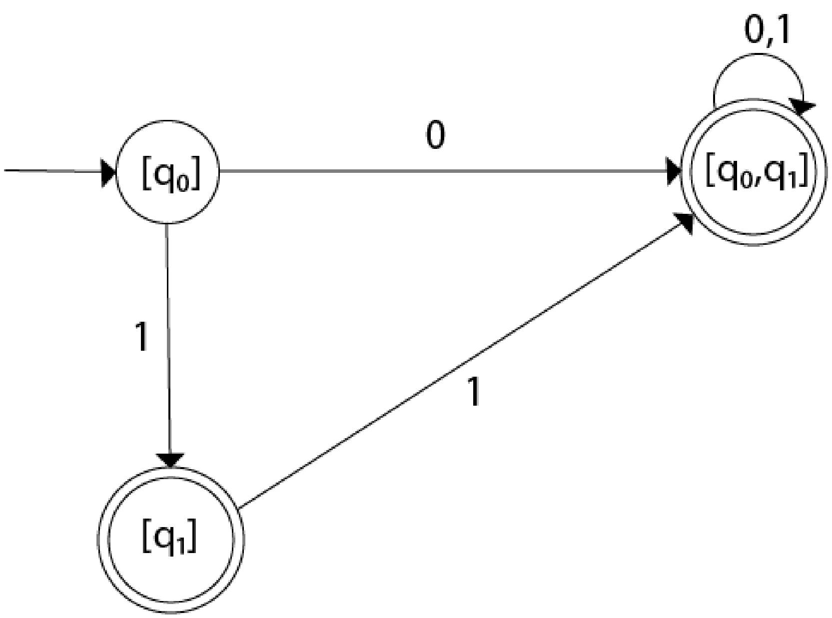
4.δ'([q0, q1], 1) = δ(q0, 1) ∪ δ(q1, 1)
= {q1} ∪ {q0, q1}
= {q0, q1}
= [q0, q1]



Ex- 3: Conversion from NFA to DFA

As in the given NFA, q1 is a final state, then in DFA wherever, q1 exists that state becomes a final state. Hence in the DFA, final states are [q1] and [q0, q1]. Therefore set of final states F = {[q1], [q0, q1]}.

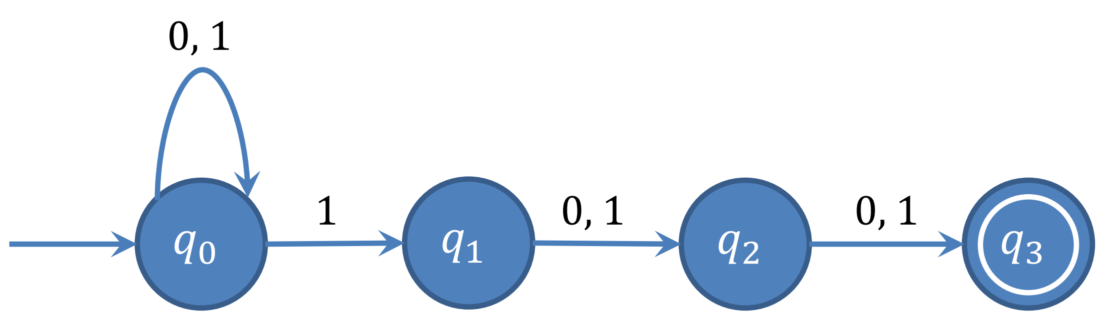
The transition table for the constructed DFA will be:



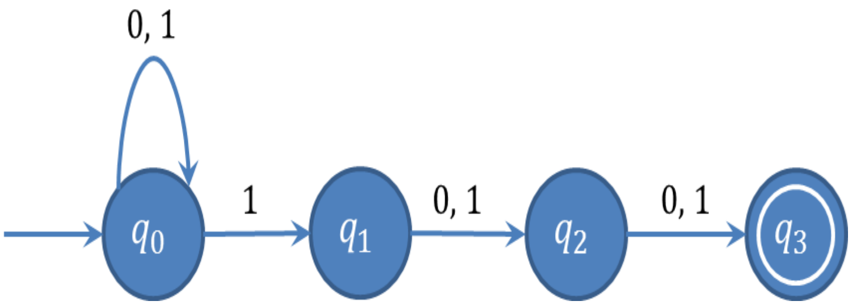
State	0	1
→[q0]	[q0, q1]	[q1]
*[q1]	ϕ	[q0, q1]
*[q0, q1]	[q0, q1]	[q0, q1]

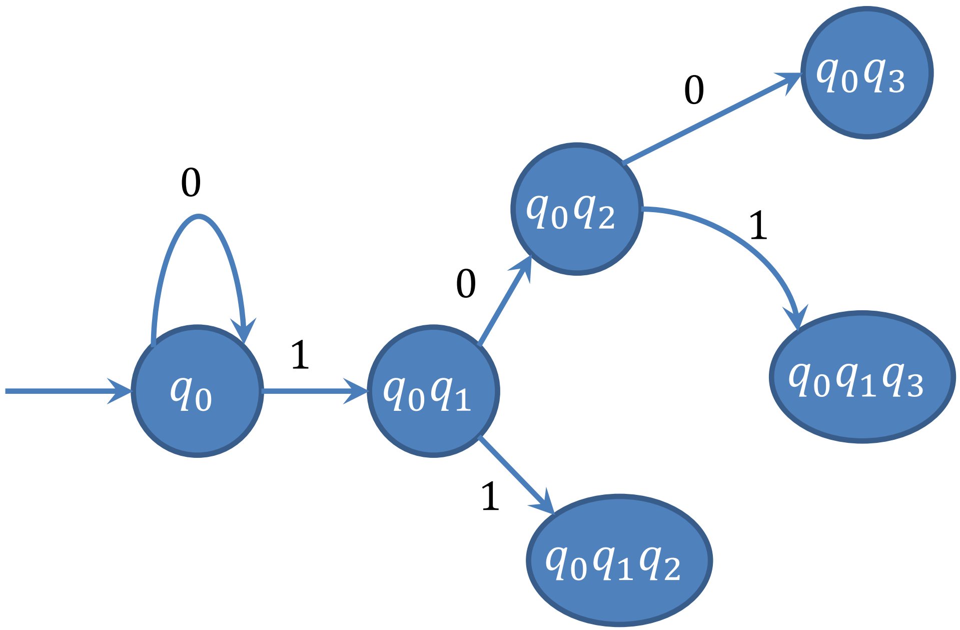
Ex- 4: Conversion from NFA to DFA

δ	Input	
State	0	1
q_0	$\{q_o\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
q_3	$\emptyset$	$\emptyset$



Ex- 4: Conversion from NFA to DFA





$\delta_1(\{q_0\},0) = \{q_0\}$

$\delta_1(\{q_0\},1) = \{q_0,q_1\}$

$\delta_1(\{q_0,q_1\},0) = \delta(\{q_0\},0) \cup \delta(\{q_1\},0) = \{q_0\} \cup \{q_2\} = \{q_0,q_2\}$

$\delta_1(\{q_0,q_1\},1) = \delta(\{q_0\},1) \cup \delta(\{q_1\},1) = \{q_0,q_1\} \cup \{q_2\} = \{q_0,q_1,q_2\}$

$\delta_1(\{q_0,q_2\},0) = \delta(\{q_0\},0) \cup \delta(\{q_2\},0) = \{q_0\} \cup \{q_3\} = \{q_0,q_3\}$

$\delta_1(\{q_0,q_2\},1) = \delta(\{q_0\},1) \cup \delta(\{q_2\},1) = \{q_0,q_1\} \cup \{q_3\} = \{q_0,q_1,q_3\}$

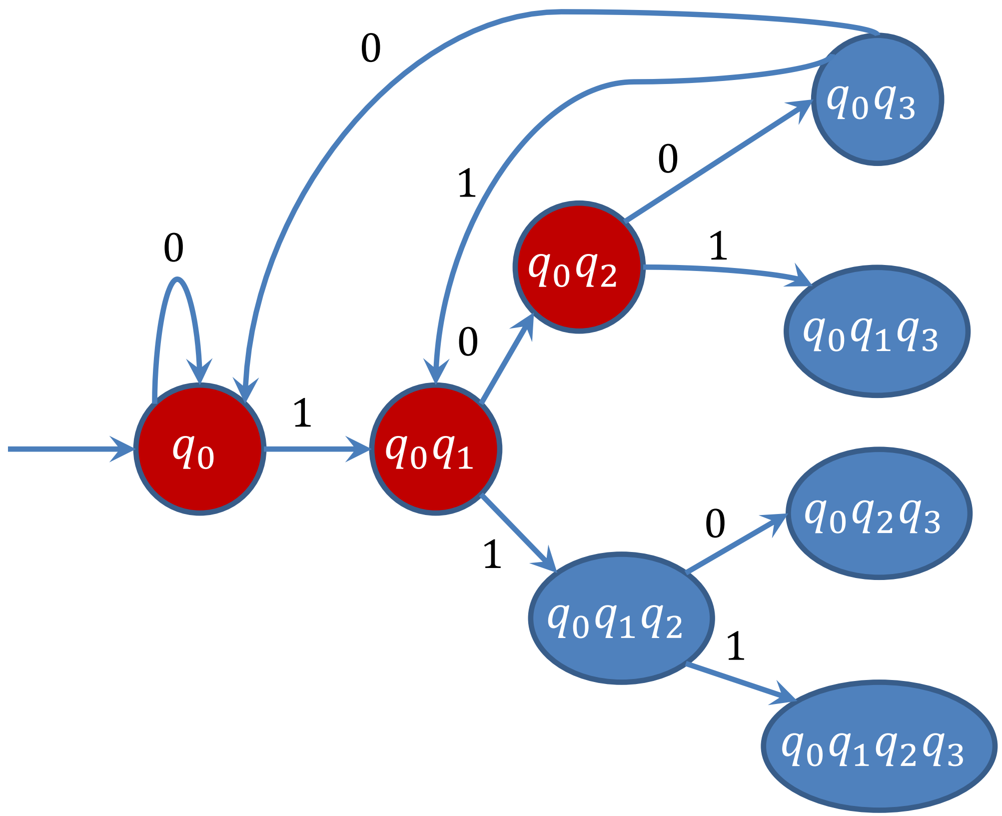
Ex- 4: Conversion from NFA to DFA

$$\delta_1(\{q_0, q_1, q_2\}, 0) = \delta(\{q_0\}, 0) \cup \delta(\{q_1\}, 0) \cup \delta(\{q_2\}, 0) = \{q_0, q_2, q_3\}$$

$$\delta_1(\{q_0, q_1, q_2\}, 1) = \delta(\{q_0\}, 1) \cup \delta(\{q_1\}, 1) \cup \delta(\{q_2\}, 1) = \{q_0, q_1, q_2, q_3\}$$

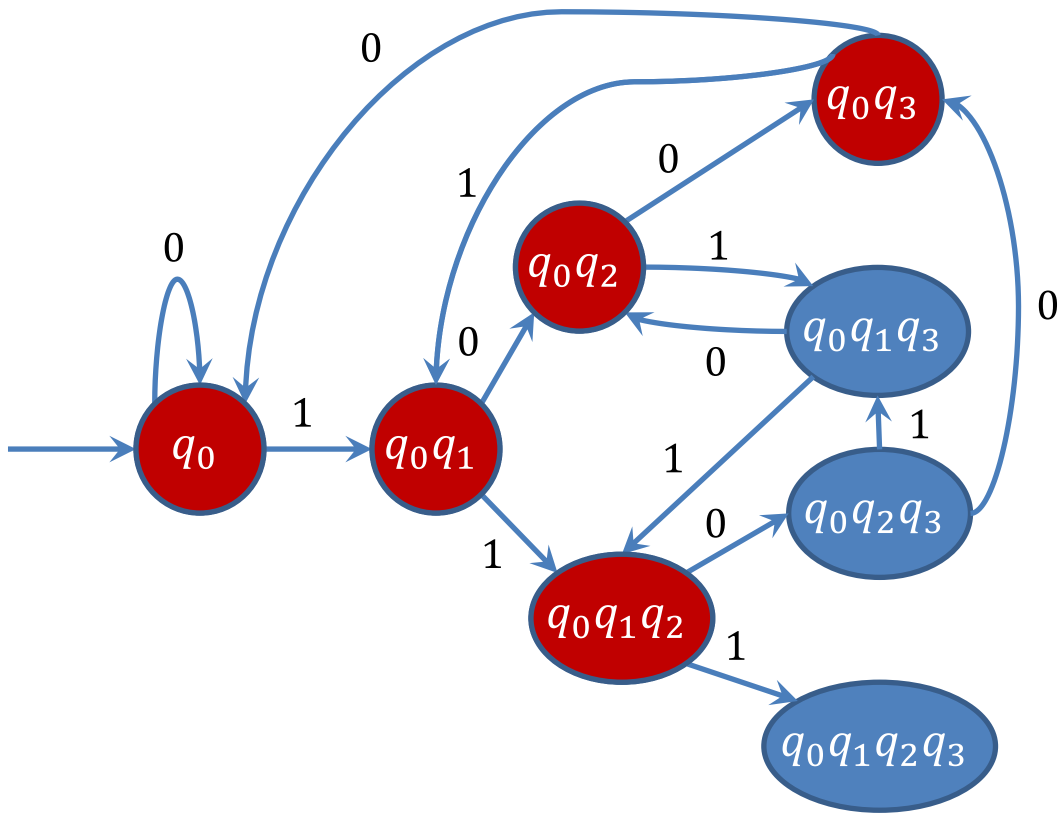
$$\delta_1(\{q_0, q_3\}, 0) = \delta(\{q_0\}, 0) \cup \delta(\{q_3\}, 0) = \{q_0\}$$

$$\delta_1(\{q_0, q_3\}, 1) = \delta(\{q_0\}, 1) \cup \delta(\{q_3\}, 1) = \{q_0, q_1\}$$



Ex 4: Conversion from NFA to DFA

$$\delta_1(\{q_0,q_1,q_3\},0) = \delta(\{q_0\},0) \cup \delta(\{q_1\},0) \cup \delta(\{q_3\},0) = \{q_0,q_2\}$$
$$\delta_1(\{q_0,q_1,q_3\},1) = \delta(\{q_0\},1) \cup \delta(\{q_1\},1) \cup \delta(\{q_3\},1) = \{q_0,q_1,q_2\}$$
$$\delta_1(\{q_0,q_2,q_3\},0) = \delta(\{q_0\},0) \cup \delta(\{q_2\},0) \cup \delta(\{q_3\},0) = \{q_0,q_3\}$$
$$\delta_1(\{q_0,q_2,q_3\},1) = \delta(\{q_0\},1) \cup \delta(\{q_2\},1) \cup \delta(\{q_3\},1) = \{q_0,q_1,q_3\}$$

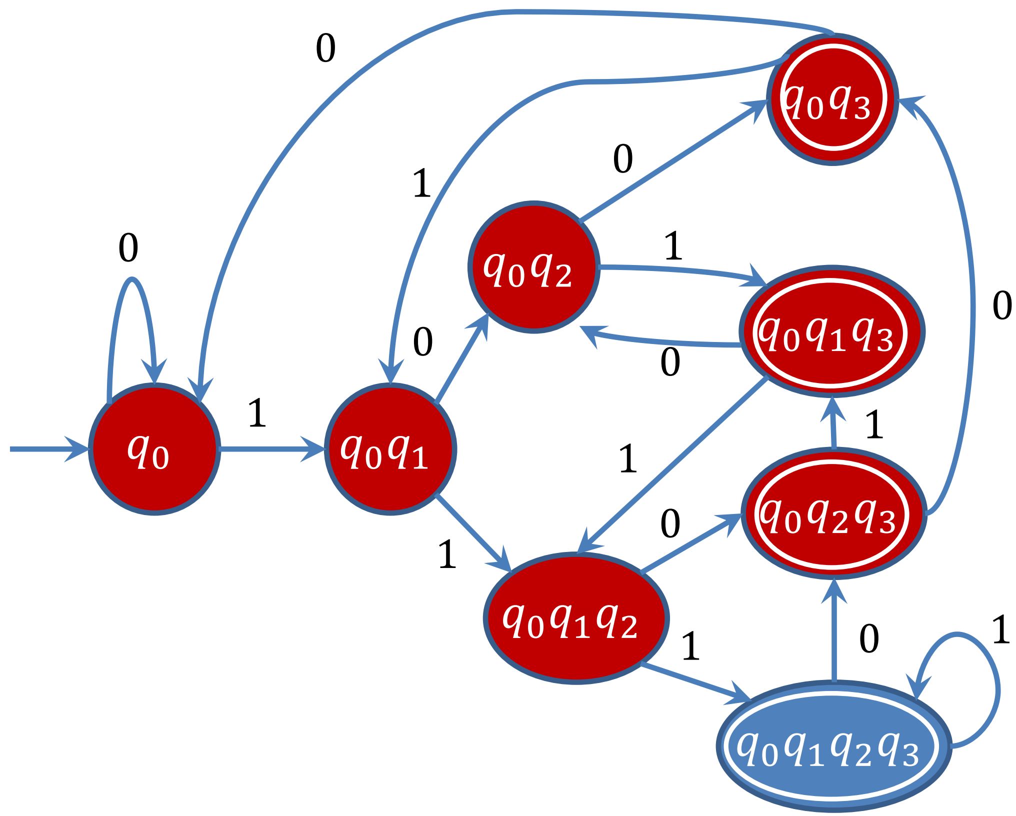


Ex 4: Conversion from NFA to DFA

$$\delta_1(\{q_0,q_1,q_2,q_3\},0) = \delta(\{q_0\},0) \cup \delta(\{q_1\},0) \cup \delta(\{q_2\},0) \cup \delta(\{q_3\},0)$$
$$= \{q_0,q_2,q_3\}$$

$$\delta_1(\{q_0,q_1,q_2,q_3\},1) = \delta(\{q_0\},1) \cup \delta(\{q_1\},1) \cup \delta(\{q_2\},1) \cup \delta(\{q_3\},1)$$
$$= \{q_0,q_1,q_2,q_3\}$$

- As now no new states are obtained, the process stops and we need to define the accepting states.
- To define accepting states, the states which contain the accepting states of NFA will be accepting states of final FA.



State Minimization

Minimizing a DFA

- To *minimize* a DFA is to find an equivalent DFA that has the least possible number of states.
- If the DFA is going to be used to write a program (e.g., a compiler) or to design hardware, then there may be a significant benefit in minimizing it before implementing it.

Minimizing a DFA

- The problem is to determine which states are distinguishable and which are indistinguishable.
- To minimize a DFA, we want to identify its equivalence classes of indistinguishable states and replace them with single states.

Minimizing a DFA by Table filling Method

- A pair of states (p , q) is said to be *distinguishes*, *if there is a string w such that, either*

$$\delta(p, w) \in F \;\&\; \delta(q, w) \notin F$$

OR

$\delta(p, w) \notin F$ & $\delta(q, w) \in F$, *then (p,q) are distinguishes states*

Minimizing a DFA by Table filling Method

- A pair of states (p , q) is said to be Indistinguishable, *if there is a string w such that, either*

$$\delta(p, w) \in F \ \& \ \delta(q, w) \in F$$

OR

$\delta(p, w) \in Q\text{-}F \ \& \ \delta(q, w) \in Q\text{-}F$, *then (p,q) are* Indistinguishable *states*

Rules

1. Basis: for each p in Q-F & q in F mark (p,q)
2. Induction steps: for any pair (p,q), if there is some input 'a' such that *$\delta(p, a)$, $\delta(q, a)$ is mark, then mark (p,q)* .
3. *Repeat step2 until no more pair can be marked.*

1. Minimize the following DFA

Transitional Table

δ	**a**	**b**
->A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C

Solutions

δ	a	b
->A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C

Pairs of states

(A,B) (A,C) (A,D) (A,E)

(B,C) (B,D) (B,E)

(C,D) (C,E)

(D,E)

Table

B				
C				
D	x_1	x_1	x_1	
E	x_0	x_0	x_0	x_0
	A	**B**	**C**	**D**

Basis Rule

$(B,B) \xleftarrow{a} (A,B) \xrightarrow{b} (C,D)$

$(B,B) \xleftarrow{a} (A,C) \xrightarrow{b} (C,C)$

$\cancel{(B,B) \xleftarrow{a} (A,D) \xrightarrow{b} (C,E)}$

$(B,B) \xleftarrow{a} (B,C) \xrightarrow{b} (D,C)$

$\cancel{(B,B) \xleftarrow{a} (B,D) \xrightarrow{b} (D,E)}$

$\cancel{(B,B) \xleftarrow{a} (C,D) \xrightarrow{b} (C,E)}$

Repeat step 2 for remaining pair of state

Solutions

Table

B	x_2			
C		x_2		
D	x_1	x_1	x_1	
E	x_0	x_0	x_0	x_0
	A	**B**	**C**	**D**

Pairs of states

(A,B) (A,C)

(B,C)

Therefore, states of the reduced DFA is {(Ac),B,D,E}

~~$(B,B) \xleftarrow{a} (A,B) \xrightarrow{b} (C,D)$~~

$(B,B) \xleftarrow{a} (A,C) \xrightarrow{b} (C,C)$

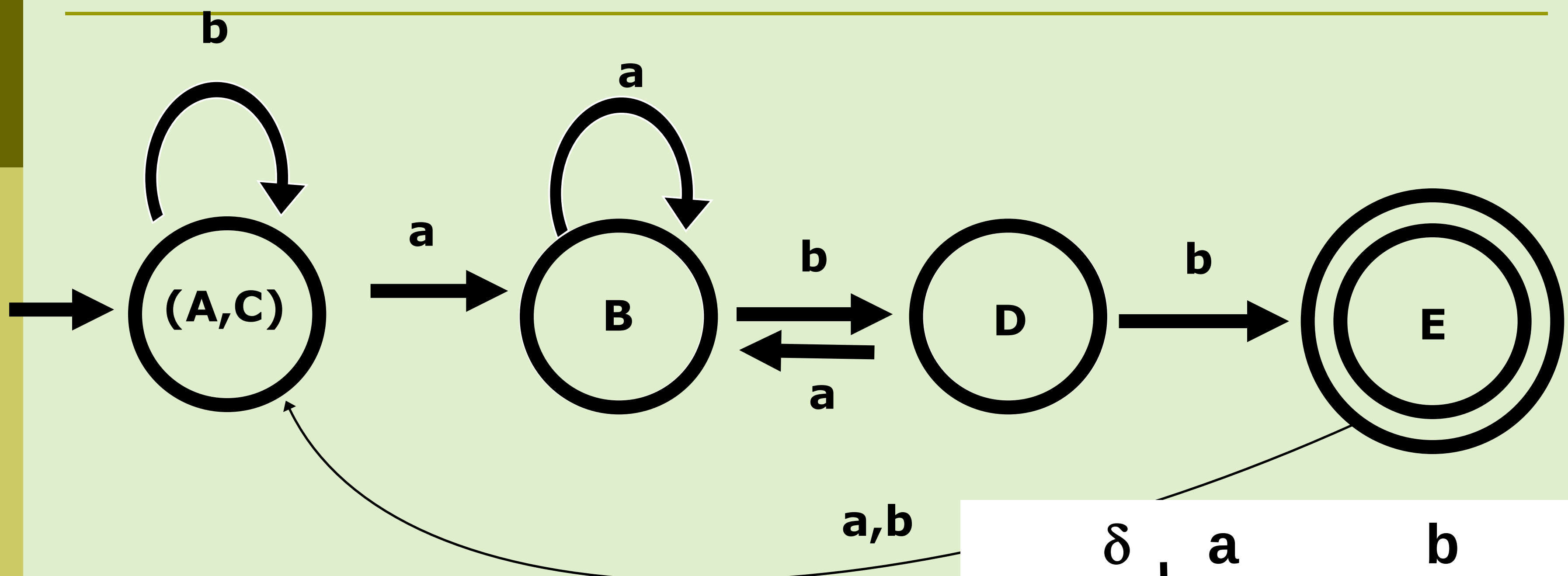
~~$(B,B) \xleftarrow{a} (B,C) \xrightarrow{b} (D,C)$~~

Repeat step 2 for remaining pair of state

$(B,B) \xleftarrow{a} (A,C) \xrightarrow{b} (C,C)$

No more pair of states can be marked, then stop

Therefore, states of the reduced DFA is {(Ac),B,D,E}



δ	a	b
(A,C)	B	(A,C)
B	B	D
D	B	E
E*	(A,C)	(A,C)

M = { Q, Σ, δ, q0, F}.

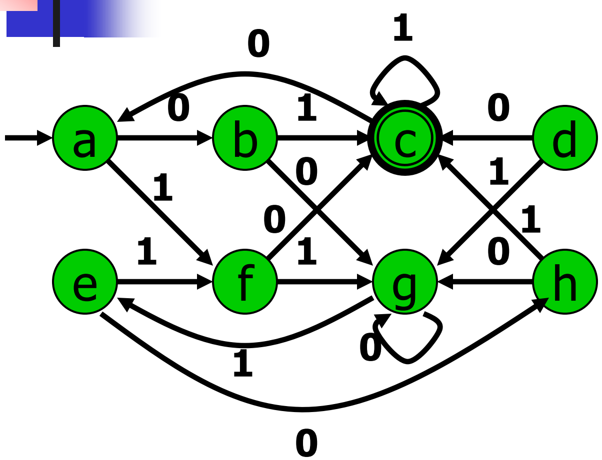
Q ={(A,C),B, D, E }.

Σ = { a,b}.

q0 = {q0}.

F = { E}.

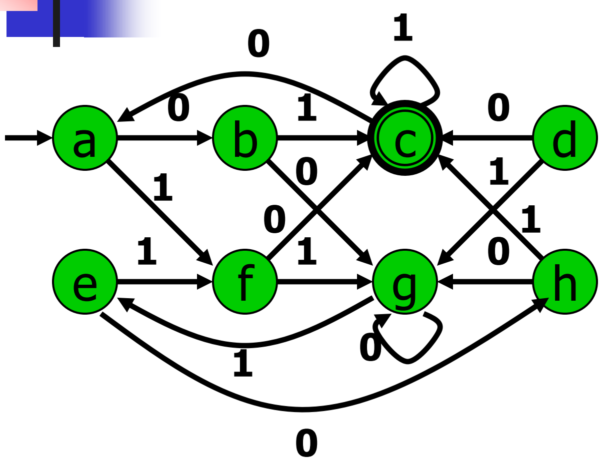
DFA Minimization: Example



1. Initialize table entries:
   Unmarked, empty list

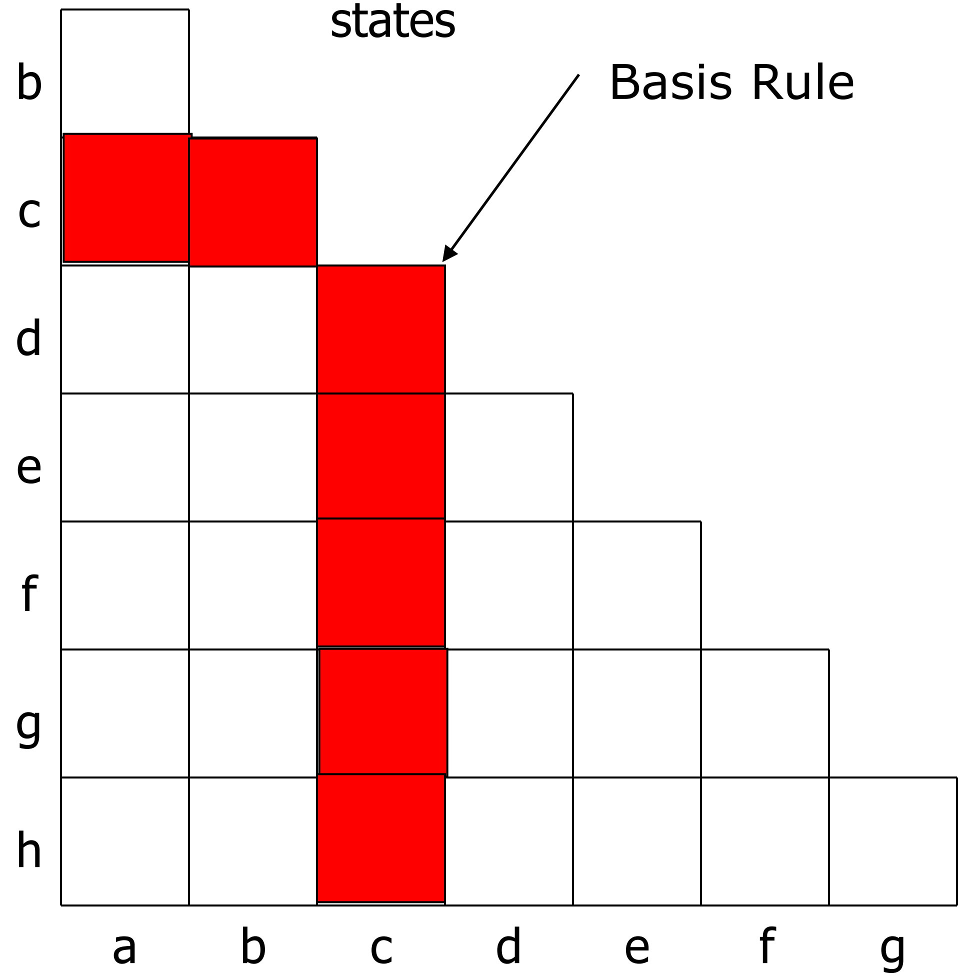
b							
c							
d							
e							
f							
g							
h							
	a	b	c	d	e	f	g

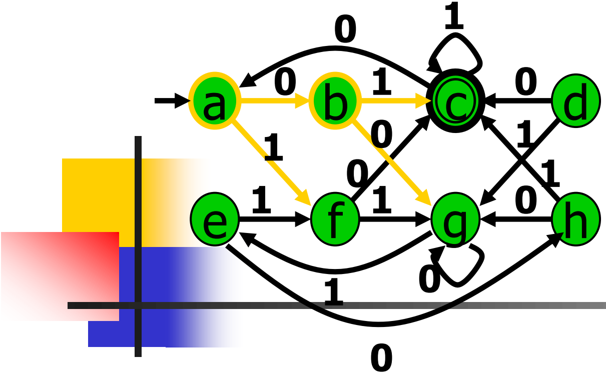
DFA Minimization: Example



(a,b) (a,c) (a,d) (a,e) (a,f) (a,g) (a,h)
(b,c) (b,d) (b,e) (b,f) (b,g) (b,h)
(c,d) (c,e) (c,f) (c,g) (c,h)
(d,e) (d,f) (d,g) (d,h)
(e,f) (e,g) (e,h)
(f,g) (f,h)
(g,h)

2. Mark pairs of final & non final states





Repeat step 2 for remaining pair of state

(a,b) (a,d) (a,e) (a,f) (a,g) (a,h)
(b,d) (b,e) (b,f) (b,g) (b,h)
(d,e) (d,f) (d,g) (d,h)
(e,f) (e,g) (e,h)
(f,g) (f,h)
(g,h)

~~$(b,g) \xleftarrow{0} \delta(a,b) \xrightarrow{1} (f,c)$~~

~~$(b,c) \xleftarrow{0} \delta(a,d) \xrightarrow{1} (f,g)$~~

$(b,h) \xleftarrow{0} \delta(a,e) \xrightarrow{1} (f,f)$

~~$(b,c) \xleftarrow{0} \delta(a,f) \xrightarrow{1} (f,g)$~~

$(b,g) \xleftarrow{0} \delta(a,g) \xrightarrow{1} (f,e)$

~~$(b,g) \xleftarrow{0} \delta(a,h) \xrightarrow{1} (f,c)$~~

~~$(g,c) \xleftarrow{0} \delta(b,d) \xrightarrow{1} (c,g)$~~

~~$(g,h) \xleftarrow{0} \delta(b,e) \xrightarrow{1} (c,f)$~~

~~$(g,c) \xleftarrow{0} \delta(b,f) \xrightarrow{1} (c,g)$~~

~~$(g,g) \xleftarrow{0} \delta(b,g) \xrightarrow{1} (c,e)$~~

$(g,g) \xleftarrow{0} \delta(b,h) \xrightarrow{1} (c,c)$

~~$(c,h) \xleftarrow{0} \delta(d,e) \xrightarrow{1} (g,f)$~~

$(c,c) \xleftarrow{0} \delta(d,f) \xrightarrow{1} (g,g)$

~~$(c,g) \xleftarrow{0} \delta(d,g) \xrightarrow{1} (g,e)$~~

~~$(c,g) \xleftarrow{0} \delta(d,h) \xrightarrow{1} (g,c)$~~

~~$(h,c) \xleftarrow{0} \delta(e,f) \xrightarrow{1} (f,g)$~~

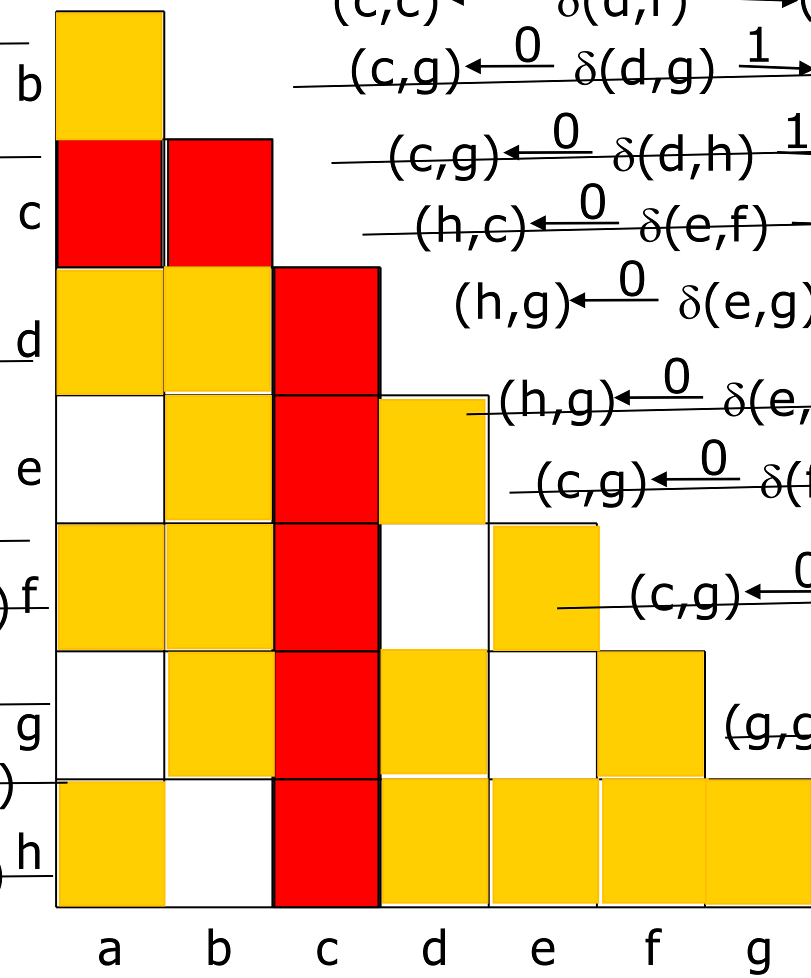
$(h,g) \xleftarrow{0} \delta(e,g) \xrightarrow{1} (f,e)$

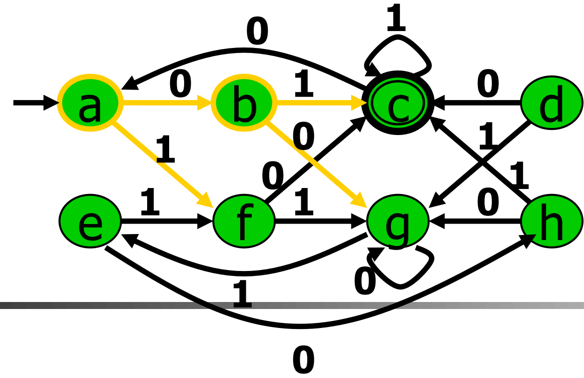
~~$(h,g) \xleftarrow{0} \delta(e,h) \xrightarrow{1} (f,c)$~~

~~$(c,g) \xleftarrow{0} \delta(f,g) \xrightarrow{1} (g,e)$~~

~~$(c,g) \xleftarrow{0} \delta(f,h) \xrightarrow{1} (g,c)$~~

~~$(g,g) \xleftarrow{0} \delta(g,h) \xrightarrow{1} (e,c)$~~





(a,e) (a,g)

(b,h)

(d,f)

(e,g)

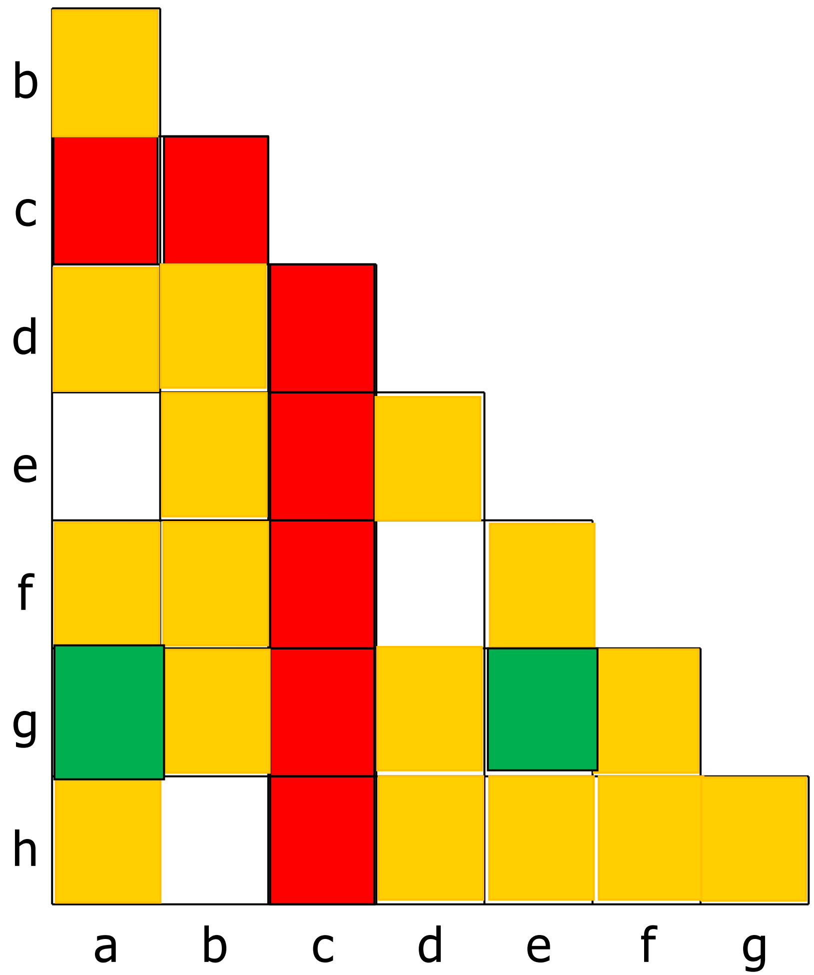
$(b,h) \xleftarrow{0} \delta(a,e) \xrightarrow{1} (f,f)$

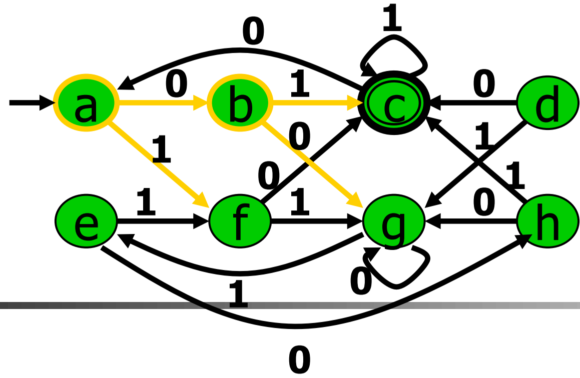
$(c,c) \xleftarrow{0} \delta(d,f) \xrightarrow{1} (g,g)$

~~$(b,g) \xleftarrow{0} \delta(a,g) \xrightarrow{1} (f,e)$~~

$(g,g) \xleftarrow{0} \delta(b,h) \xrightarrow{1} (c,c)$

~~$(h,g) \xleftarrow{0} \delta(e,g) \xrightarrow{1} (f,e)$~~



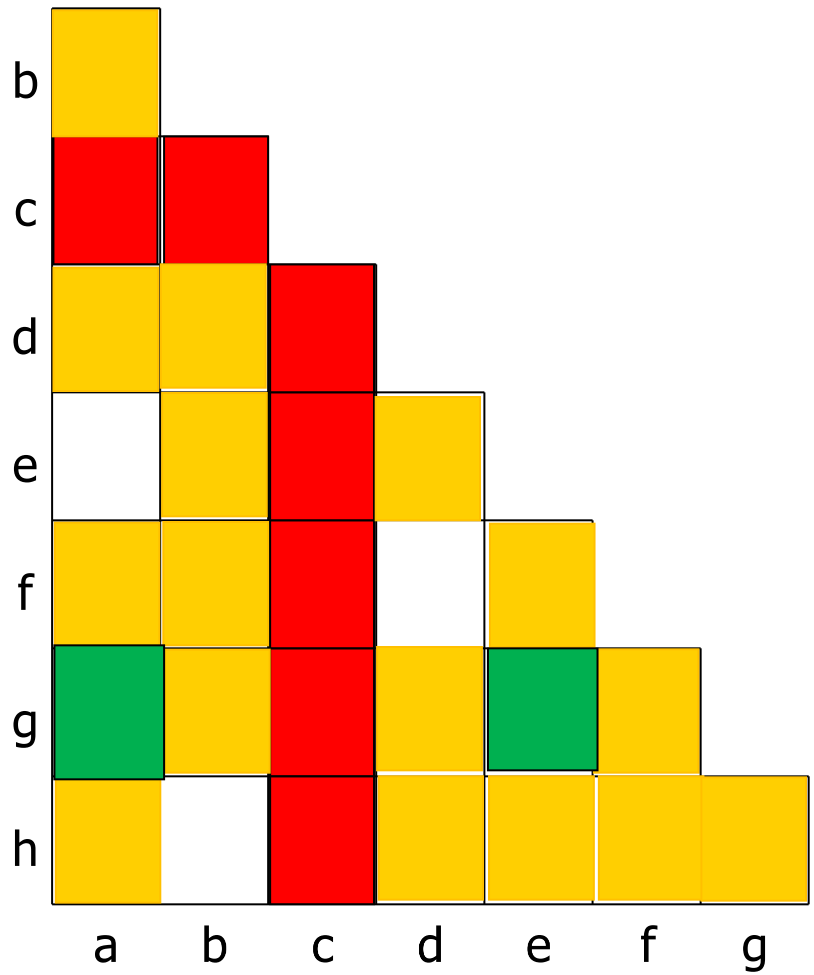


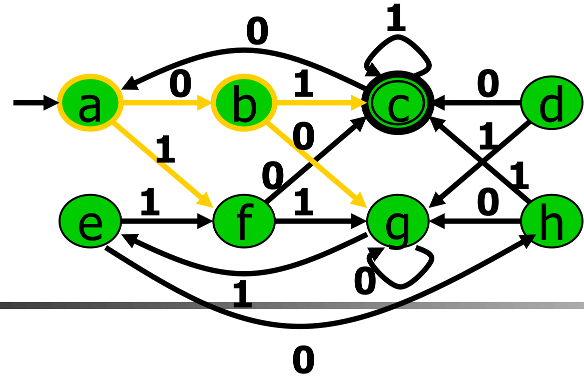
(a,e) (b,h) (d,f)

$(b,h) \xleftarrow{0} \delta(a,e) \xrightarrow{1} (f,f)$

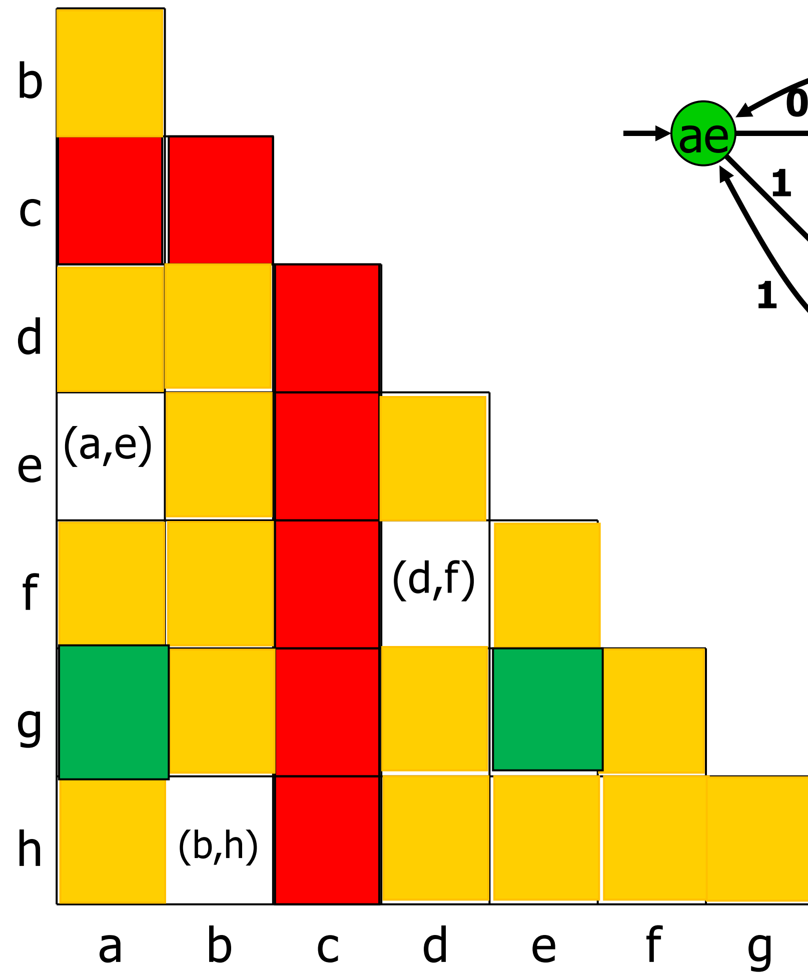
$(c,c) \xleftarrow{0} \delta(d,f) \xrightarrow{1} (g,g)$

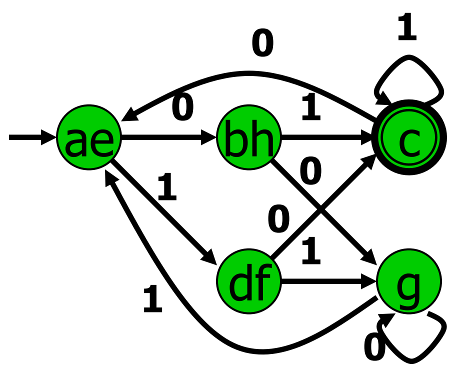
$(g,g) \xleftarrow{0} \delta(b,h) \xrightarrow{1} (c,c)$





$a \equiv e$
$b \equiv h$
$d \equiv f$





Minimal DFA

Example: Minimize FA

Final state is {6}

And, Non-Final state is {1,2,3,4,5,7}

(6, 1), (6,2), (6, 3), (6, 4), (6, 5), (6, 7) are distinguish pairs.

Consider pair **(1,2)**

$\delta(1,a)=2$ $\delta(1,b)=3$

$\delta(2,a)=4$ $\delta(2,b)=5$

Consider pair **(1,3)**

$\delta(1,a)=2$ $\delta(1,b)=3$

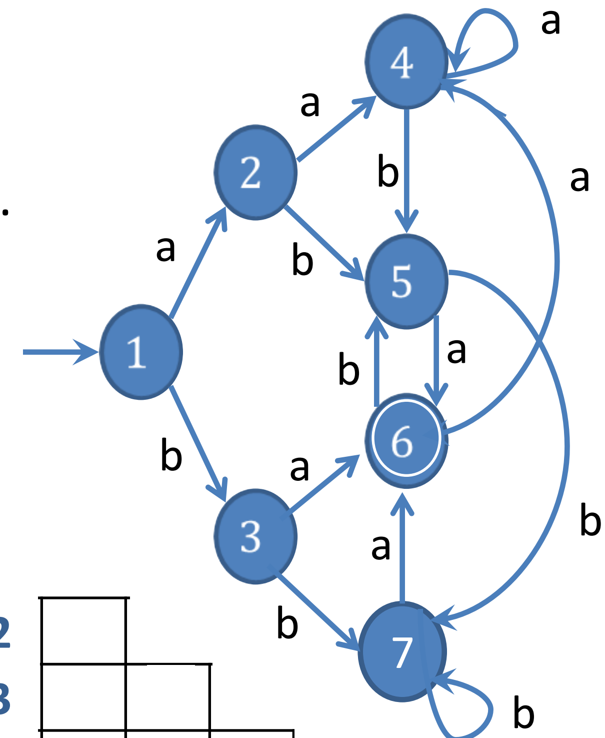
$\delta(3,a)=6$ $\delta(3,b)=7$

pair (2,6) is distinguish, so (1,3) is distinguished pair.

Consider pair **(1,4)**

$\delta(1,a)=2$ $\delta(1,b)=3$

$\delta(4,a)=4$ $\delta(4,b)=5$



	1	2	3	4	5	6
2						
3						
4						
5						
6						
7						

Example: Minimize FA

Consider pair **(1,5)**

δ(1,a)=2 δ(1,b)=3

δ(5,a)=6 δ(5,b)=7

pair (2,6) is distinguish, so (1,5) is distinguished pair.

Consider pair **(1,7)**

δ(1,a)=2 δ(1,b)=3

δ(7,a)=6 δ(7,b)=7

pair (2,6) is distinguish, so (1, 7) is distinguished pair.

Consider pair **(2,3)**

δ(2,a)=4 δ(2,b)=5

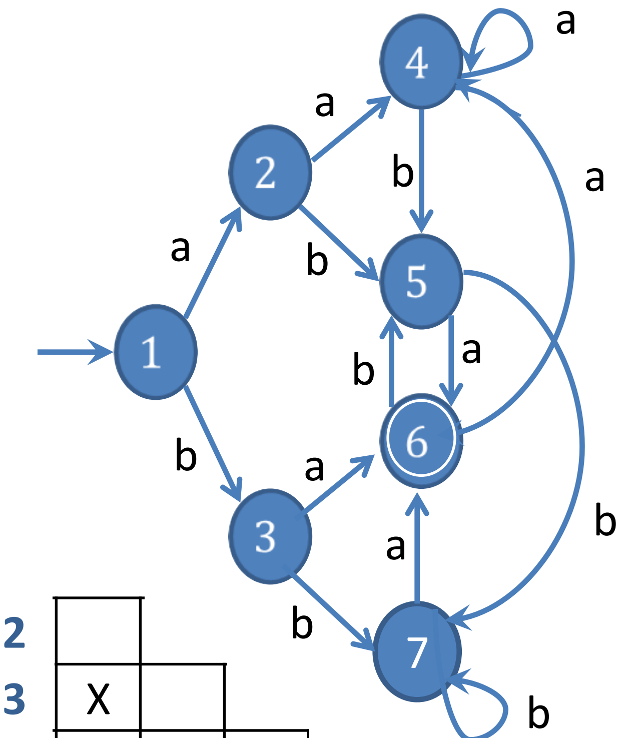
δ(3,a)=6 δ(3,b)=7

pair (4,6) is distinguish, so (2, 3) distinguished pair.

Consider pair **(2,4)**

δ(2,a)=4 δ(2,b)=5

δ(4,a)=4 δ(4,b)=5



	1	2	3	4	5	6
2						
3	X					
4						
5						
6	X	X	X	X	X	
7						X

Example: Minimize FA

Consider pair **(2,5)**

$\delta(2,a)=4$ $\delta(2,b)=5$

$\delta(5,a)=6$ $\delta(5,b)=7$

pair (4,6) is distinguish, so (2,5) is distinguished pair.

Consider pair **(2,7)**

$\delta(2,a)=4$ $\delta(2,b)=5$

$\delta(7,a)=6$ $\delta(7,b)=7$

pair (4,6) is distinguish, so (2,7) is distinguished pair.

Consider pair **(3,4)**

$\delta(3,a)=6$ $\delta(3,b)=7$

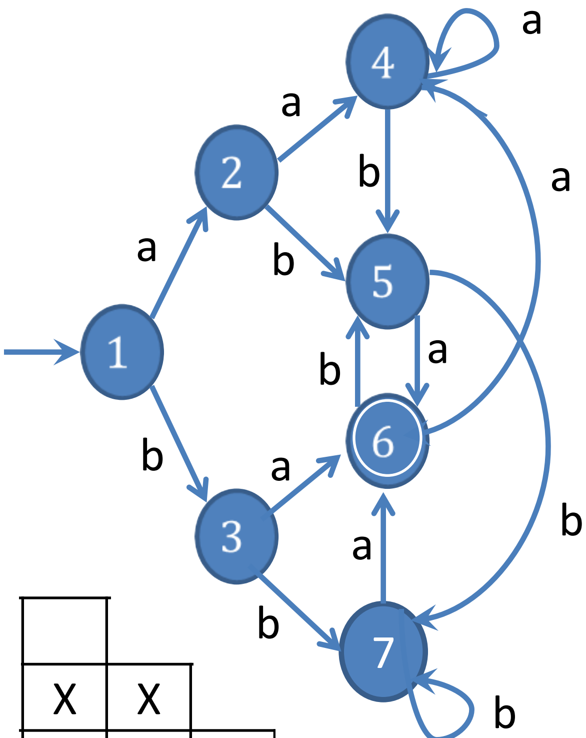
$\delta(4,a)=4$ $\delta(4,b)=5$

pair (6,4) is distinguish, so (3,4) is distinguish pair.

Consider pair **(3,5)**

$\delta(3,a)=6$ $\delta(3,b)=7$

$\delta(5,a)=6$ $\delta(5,b)=7$



2						
3	X	X				
4						
5	X					
6	X	X	X	X	X	
7	X					X
	1	2	3	4	5	6

Example: Minimize FA

Consider pair **(3,7)**

$\delta(3,a)=6$ $\delta(3,b)=7$

$\delta(7,a)=6$ $\delta(7,b)=7$

Consider pair **(4,5)**

$\delta(4,a)=4$ $\delta(4,b)=5$

$\delta(5,a)=6$ $\delta(5,b)=7$

pair (6,4) is distinguish, so (4,5) is distinguish.

Consider pair **(4,7)**

$\delta(4,a)=4$ $\delta(4,b)=5$

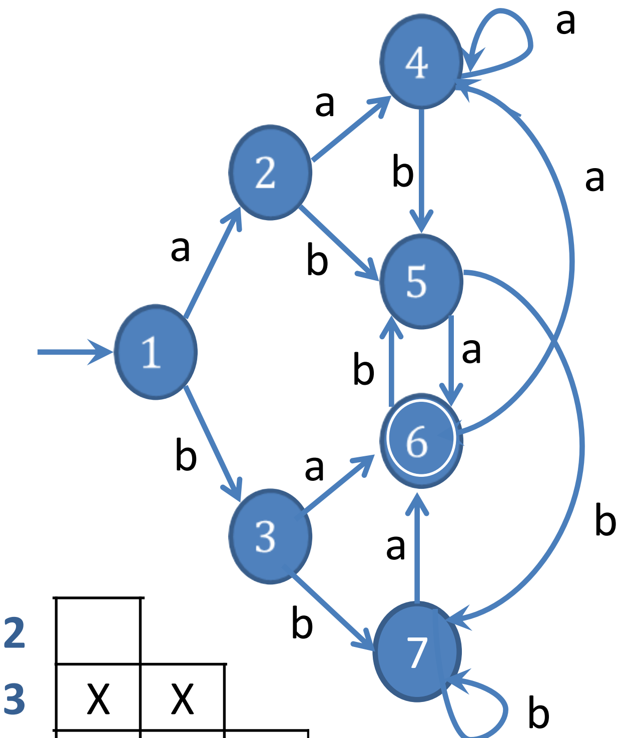
$\delta(7,a)=6$ $\delta(7,b)=7$

pair (4,6) is distinguish, so (4,7) is distinguish.

Consider pair **(5,7)**

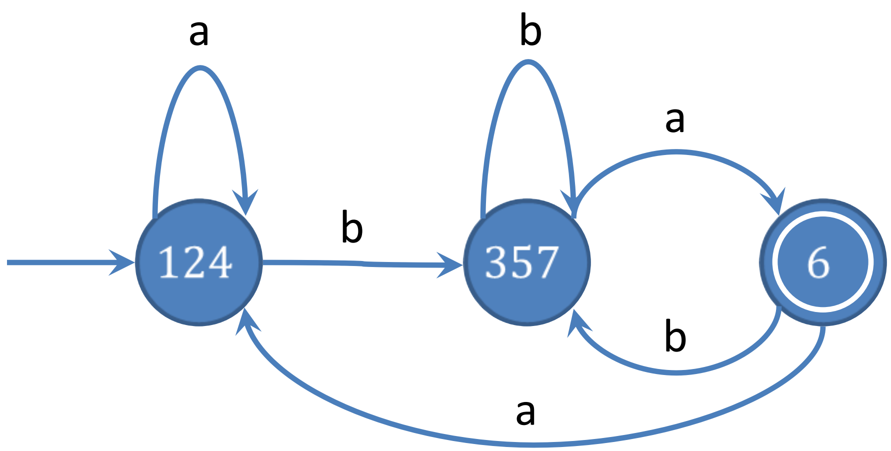
$\delta(5,a)=6$ $\delta(5,b)=7$

$\delta(7,a)=6$ $\delta(7,b)=7$

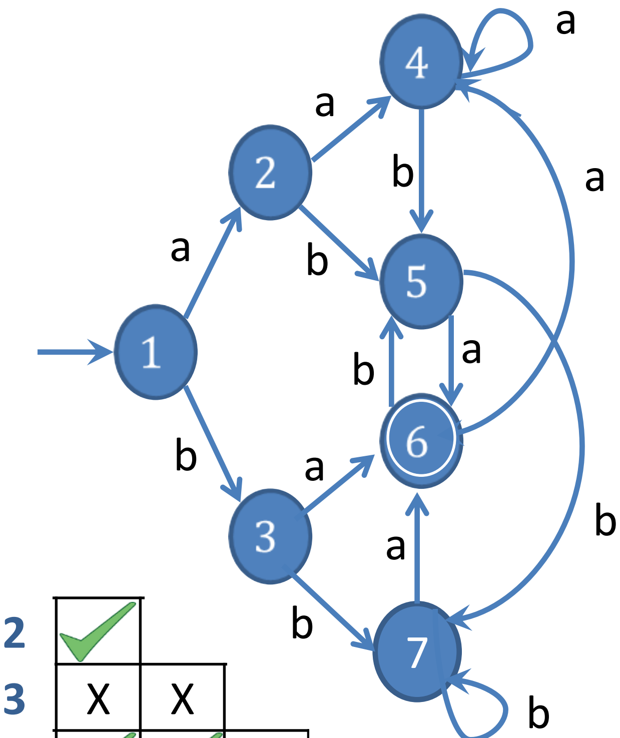


	1	2	3	4	5	6
2						
3	X	X				
4			X			
5	X	X				
6	X	X	X	X	X	
7	X	X				X

Example: Minimize FA



Minimized FA



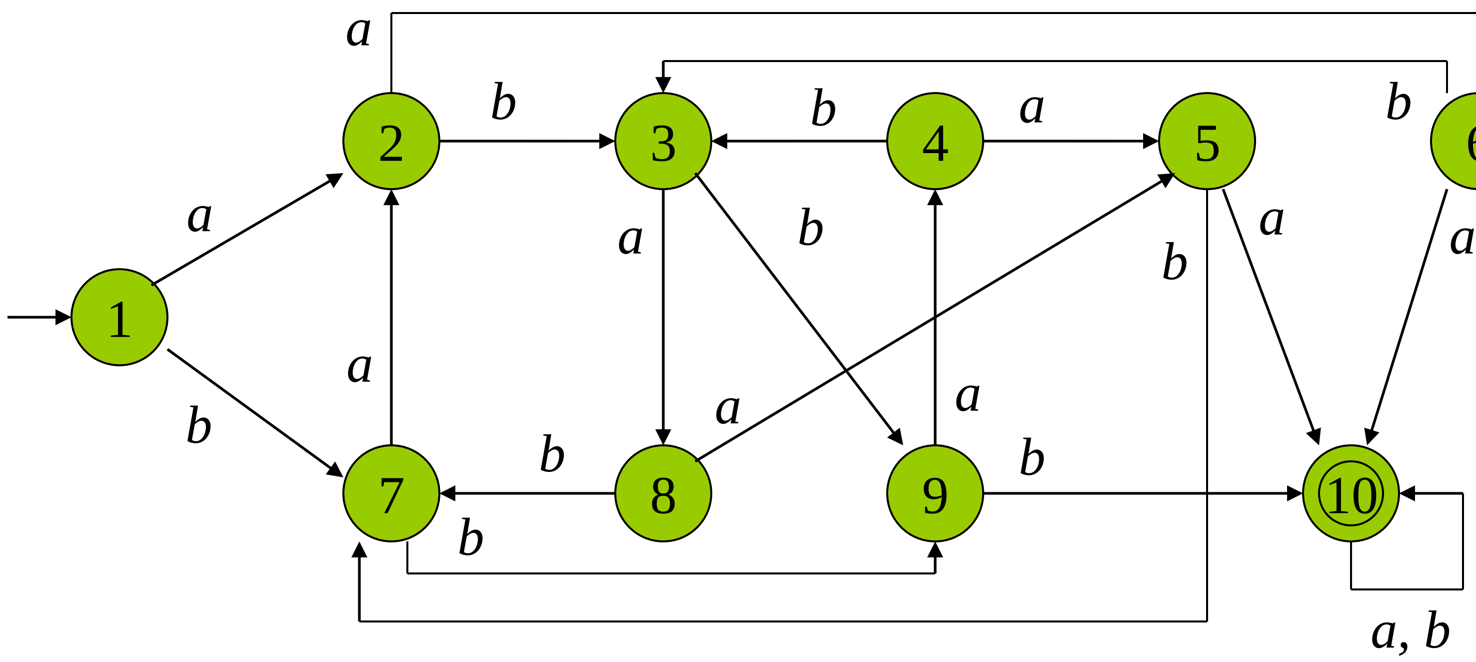
2	✓					
3	X	X				
4	✓	✓	X			
5	X	X	✓	X		
6	X	X	X	X	X	
7	X	X	✓	X	✓	X
	1	2	3	4	5	6

1=2 1=4 2=4 → 1=2=4

3=5 3=7 5=7 → 3=5=7

Example of Minimization

- Minimize the following DFA.



Example of Minimization

- Now we can see that the language of this DFA is

$$\{w \in \Sigma^* \mid w \text{ contains } aaa \text{ or } bbb\}.$$

Example

- Find a minimal DFA that accepts the language

$$\{w \in \Sigma^* \mid w \text{ contains 010 and 101}\}.$$

DFA Minimization: Correctness

Why is new DFA no larger than old DFA?

Only removes states, never introduces new states.

Obvious.

Why is new DFA equivalent to old DFA?

Only identify states that provably have same behavior.

Could prove $x \in L(M) \leftrightarrow x \in L(M')$ by inductions on derivations.

What About NFA Minimization?

This algorithm doesn’t find a unique minimal NFA.

?

Is there a (not necessarily unique) minimal NFA?

?

Of course.

NFA Minimization

In general, minimal NFA not unique!

Example NFAs for $\mathbf{0}^+$:

Both minimal, but not isomorphic.