قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 1

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

First Lecture

* **Set**:

A set is a collection of objects without repetition. Each object in a set is called an element of the set for example D denotes the set of days

D= {Sun., Mon., Tue., Wed., thru., Fri., Sat.}

D= {X | X is a day of a week}

T= {0,1,2,3,4,5,6,7,8,9}

- If an element X is an element of a set A Then we write X ∈ A and if X is not an element of A we write X ∉ A thus
  Monday ∈ D
  May ∉ D
- We say a set A is a subset of set B written A C B
  {1, 2, 4} C {1, 2, 4, 5, 3}
  {1, 2, 6} ₵ {1, 2, 3 ,4 ,5 }
- Two sets A and B are sided equal written A = B iff A and B contain the same elements.

❖ **The Basic operation on sets are**

- Unary operation ex. Complement.
- Binary operation ex. Union ( ), intersection ( ) and difference.

$\bar{A}$ ={X | X ∉ A} consist of all elements in the universe are not in A.

A∪B ={X | X ∈ A or X ∈ B}

A∩B ={X | X ∈ A and X ∈ B}

A\B ={X |X ∈ A and X ∉ B}

2^A the power set of A, is the set of all subset of A.

Example:

If $U=\{0,1,2,3,4,5,6,7,8,9\}$

$A= \{0,1,3,5\}$

$B= \{2,3,5\}$ then find

$\bar{A}= \{2,4,6,7,8,9\}$

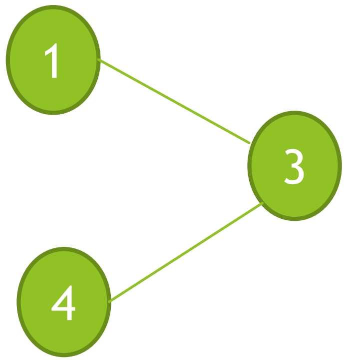
$A\cup B= \{0,1,3,5,2\}$

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

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

$\mathbf{2^A} =\{\emptyset,\{0\},\{1\},\{3\},\{5\},\{0,1\},\{0,3\},\{0,5\},\{\{1,3\},\{1,5\},\{3,5\},$
$\{0,1,3\},\{0,1,5\},\{1,3,5\},\{0,3,5\},\{0,1,3,5\}\}$

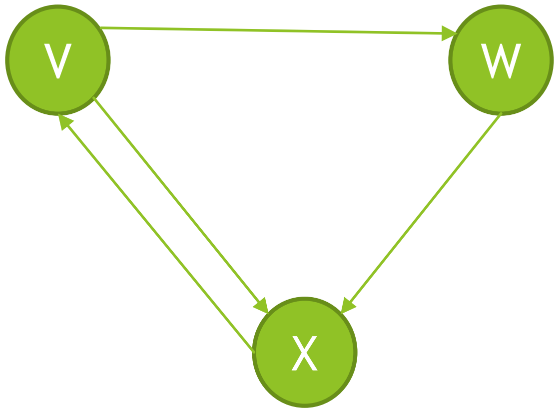
- **A graph** denoted G= (V, E), consist of a finite set of vertices (or node) **V** and a set of pairs of vertices **E** called edges.
- <u>**Example:**</u>



V= {1, 3, 4}
E= {(1, 3), (3, 4)}

- **A directed graph** (dgraph), also denoted G= (V, E) consist of a finite set of vertices V and a set of ordered pairs of vertices E called arcs.

Example: The digraph G=(V,E) where V={V,W,X} and E={(V,W),(V,X),(X,V), (W,X)}



❖ Products of Sets

Let A1, A2 be two sets then the product of A1 and A2 consist of all the pair (a1, a2) where a1 is in A1 and a2 is in A2

A1*A2={(a1,a2)| a1 ∈ A1 and a2 ∈ A2 }

Example:

If A1 ={0, 1} and A2={x, y, z}

Then

A1*A2 ={(0,x),(0,y),(0,z),(1,x),(1,y),(1,z)}

❖ **concatenation of string:**

The concatenation of two strings is the string formed by writing the first followed by the second with no intervening space. The concatenation of x,y over denoted by xy.

abcb is string .

If X=a1 a2 a3 a4….an and Y=b1 b2 b3 b4 ….bn

XY=a1 a2 a3 a4….an b1b2b3b4….bn

YX=b1 b2 b3 b4….bn a1 a2 a3 a4 ….an

- **Symbol**: indivisible item(letter or digit)
- **Alphabet** is a finite set of element which is called symbols.
- **A string (or word)** is a finite Sequence of symbols juxtaposed (repeating are allowed).

Example:

a ,b ,c are symbols and

abcb is string .

-The empty word denoted by $\boldsymbol{\epsilon}$ or $\boldsymbol{\lambda}$ is the string contains no symbols.

Language: A formal language is a set of string of symbols from some an alphabet.
Closure: concatenation of Σ with itself for all length of string.

Example

$\Sigma = \{a, b, c\}$

$\Sigma^0 = \{\epsilon\}$

$\Sigma^1 = \{a, b, c\}$

$\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$

$\Sigma^3 = \{abc, aaa, aba, aab, baa, bba\ldots\ldots\}$

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \ldots\ldots$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \ldots\ldots$

$\Sigma^{*} = \Sigma^{+} \cup \{\epsilon\}$

$\Sigma^{+} = \Sigma^{*} - \{\epsilon\}$

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory

2nd Class/1st Sem

Lecture 2

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Finite State System

Finite State Systems:

The finite automaton is a mathematical model of system, with discrete inputs and outputs. The system can be in any one of a finite number of configuration or states. The state of the system summarizes the information inputs that are needed to determine the behavior of the system on subsequent inputs.

In computer science we find many examples of finite state systems:-

1- Switching circuit, such as the control unit of a computer.

2- The design of several common types of computer algorithms and programs. For example the lexical analysis and text editors.

Basic definitions:

A finite automaton (FA) consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet Σ.

for each input symbol there is exactly one transition out of each state(possibly back to the state itself).

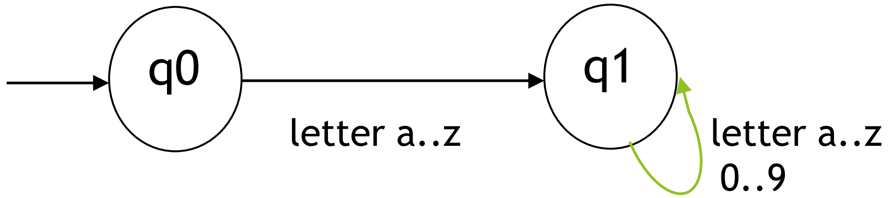
One state, usually denoted q_0, is the initial state in which the automaton starts. Some states are design as final or accepting states.

A directed graph, called a transition diagram is associated with a FA as follows. The vertices of the graph correspond to the states of the FA.

<u>Basic definitions:</u>

If there is a transition from state q to state p on input a, then there is an arc labeled a from state q to state p in the transition diagram. The FA accept a string x if the sequence of transitions correspond to the symbols of x leads from the start state to an accepting state.

Example:



transition diagram of identifier

Deterministic Finite Automaton (DFA):

Correspond It is an acceptor for any state and input character has at most one transition state that the acceptor change to. If no transition state is specified the input string is rejected.

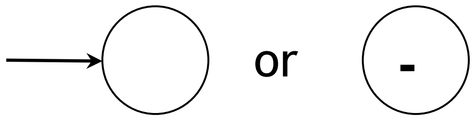
A DFA is a 5-tuple M= $(Q, \Sigma, \delta, q_0, F)$ Where

Q is a set of state.

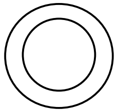
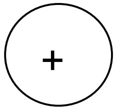
Σ is an input alphabet.

δ is a transition function $\delta = Q * \Sigma = Q$

q_0 $q_0 \in Q$ is the initial state (the initial state is marked with an incoming arrow



F is a set of final states F C Q
(The final states are depicted using double circles

 or 

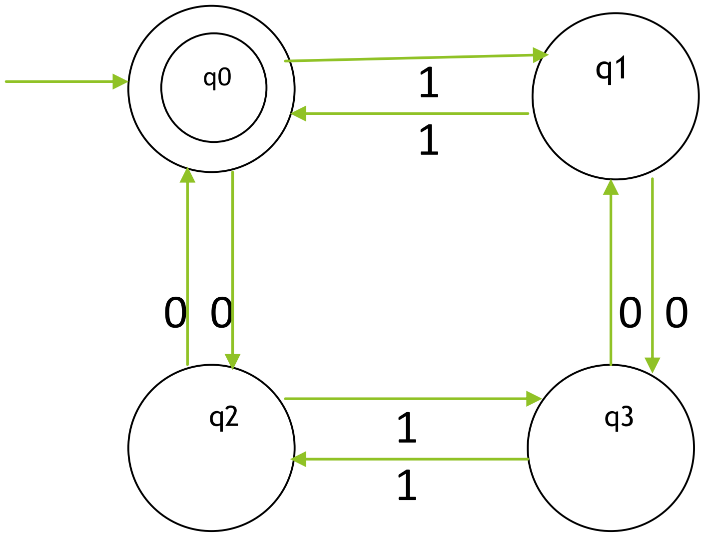
Accepted string → δ (q0, w) =p Є F → w accepted
Else w reject

The language accepted by M, designated L (M),
Language of Automata: **L (M)** :{ w: δ (q0, w) =p, p Є F}

<u>Example1</u>: Design a DFA that accepted the set of all strings with an even number of 0's and 1's

L={11,1111,111111,11111111,….
00,0000,000000,00000000,….
0011,1100,001111,111100,110000,000011,….
0101,1010,010111,…. }

Solution: 1-transition Diagram



Q= {q0, q1, q2, q3}
Σ = {0, 1}
F= {q0}
2-transition function
Suppose the string (110101) is input to M
δ(q0,110101)=δ(δ(qo,1),10101)=δ(δ(q1,1),0101)=δ(δ(q0,0),101)=
δ(δ(q2,1)01)=(δ(δ(q3,0),1)= δ (δ(q1,1)=q0 ЄF The string is accepted.
- The path o this string

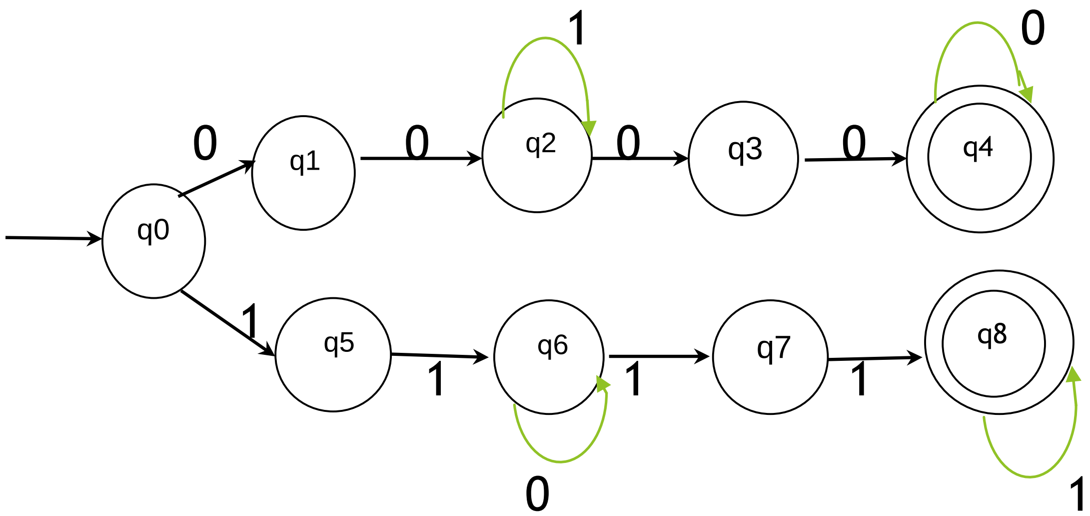
$\longrightarrow q0 \xrightarrow{1} q1 \xrightarrow{1} q0 \xrightarrow{0} q2 \xrightarrow{1} q3 \xrightarrow{0} q1 \xrightarrow{1} q0$

3-transition table

Input / State	0	1
-+q0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

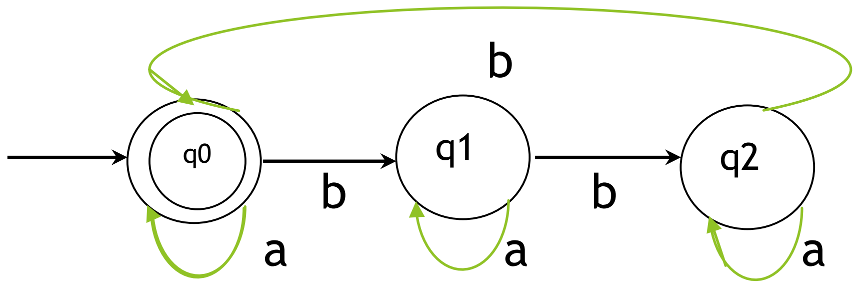
<u>Example2</u>: Design a DFA that accepted the set of all strings that begin and end with the same double letter, either of the form 00...00, 11...11, Σ= {0, 1}.

L= {0000,00000,000000,0000000, …
00100,001100,0011111100,001111000, …
1111,11111,111111,11111111, …
11011,110011,1100000011, 110000111…}



<u>Example3</u>: Design a DFA that accepted the set of all strings that have number of b's divisible by 3, Ɛ= {a, b}.

L= {bbb,bbbbbb,bbbbbbbbb,abbb,abbba,aabbb,aabbbbbba,…}



Example4: Design a DFA that accepted the set of all strings that have total number of 0's divisible by 3, Ɛ= {0, 1}

Example5: Design a DFA that accepted the set of all strings that ending with double letter Ɛ= {a, b}

Example6: Design a DFA that accepted the set of all strings that ending in 00.

Example7: Design a DFA that accepted the set of all strings that all zero must be 3 consecutive 0’s

Example8: Design a DFA that accepted the set of all strings that does not contain 3 consecutive 1's.

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 3

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Finite State System

Non Deterministic Finite Automata

(NFA)

Non Deterministic Finite Automaton (NFA):

It is the FA that allows one or more transition from a state on the same input symbol.

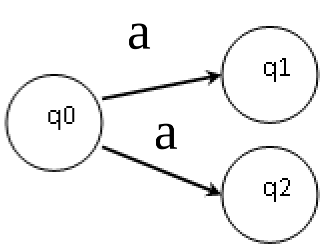
NFA is a 5-tuple M= (Q, Σ, δ, q_0, F) Where

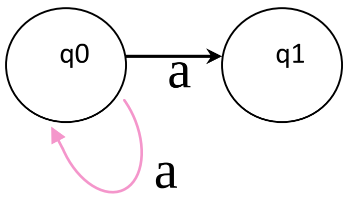
Where Q, Σ and F have the same meaning as for a DFA, but δ is a map from

$Q * \Sigma$ to 2^Q.

(2^Q is the power set of Q, the set of all subset of Q)

-Transition diagram





Non Deterministic Finite Automaton (NFA):

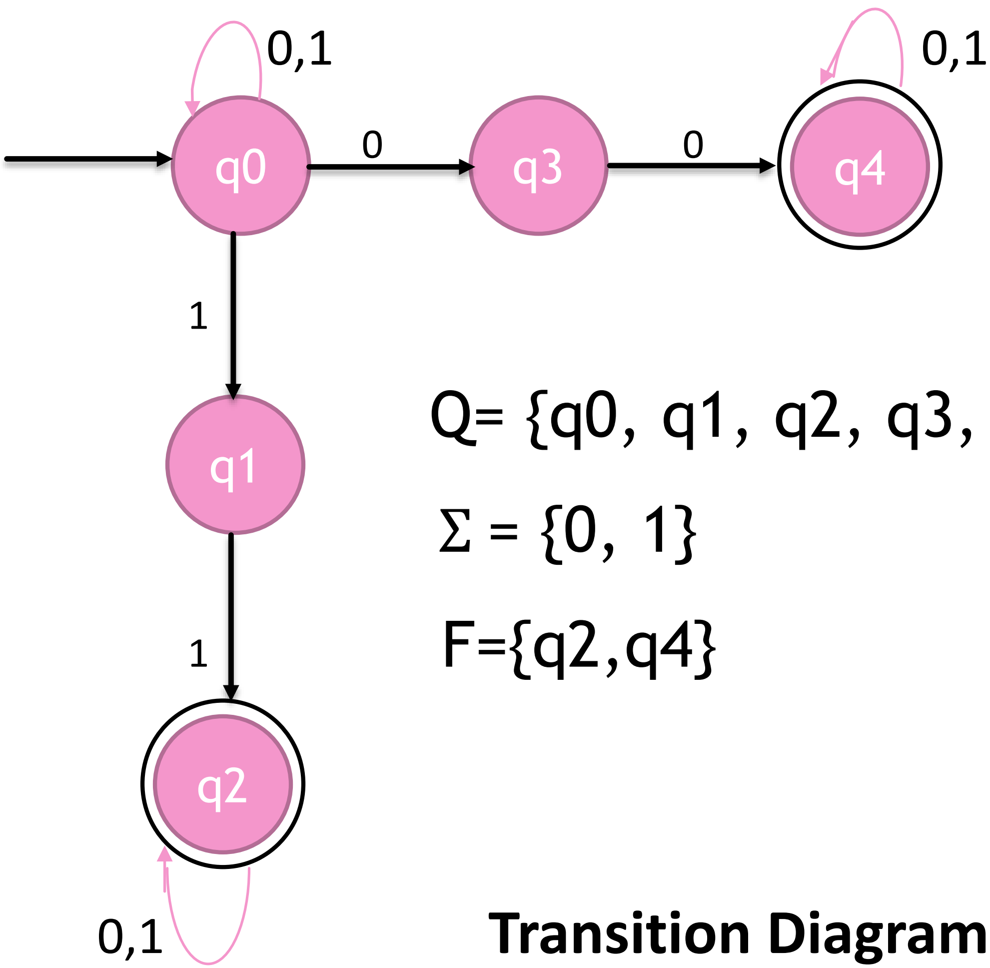
-Transition Table

Σ / Q	

-Transition Function

$$\delta(\{q0, q1\}, a) = \delta(q0, a) \cup \delta(q1, a)$$

Example1: NFA accept all string with either two consecutive 0's or two consecutive 1's?



$Q = \{q0, q1, q2, q3, q4\}$

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

$F = \{q2, q4\}$

Transition Diagram

Σ / Q	0	1
-q0	{q0,q3]	{q0,q1}
q1	∅	{q2}
+q2	{q2}	{q2}
q3	{q4}	∅
+q4	{q4}	{q4}

Transition Table

-Transition Function

Suppose (01001) input to machine

δ (q0,01001)= $\delta(\delta$(q0,0),1001)= δ({q0,q3},1001)= $\delta(\delta$(q0,1) ∪ δ(q3,1),001) = δ({q0,q1},001)= $\delta(\delta$(q0,0) ∪ δ(q1,0),01)= δ({q0,q3},01)= $\delta(\delta$(q0,0) ∪ δ(q3,0),1)= δ({q0,q3,q4},1)= δ(q0,1) ∪ δ(q3,1) ∪ δ(q4,1)={q0,q1,q4}

-An input string is accepted by NFA if there exists a sequence of transition for the given string that leads from the initial state to some final state

$\{W \in \Sigma^* \mid \delta(q0, w) \cap F \neq \emptyset\}$

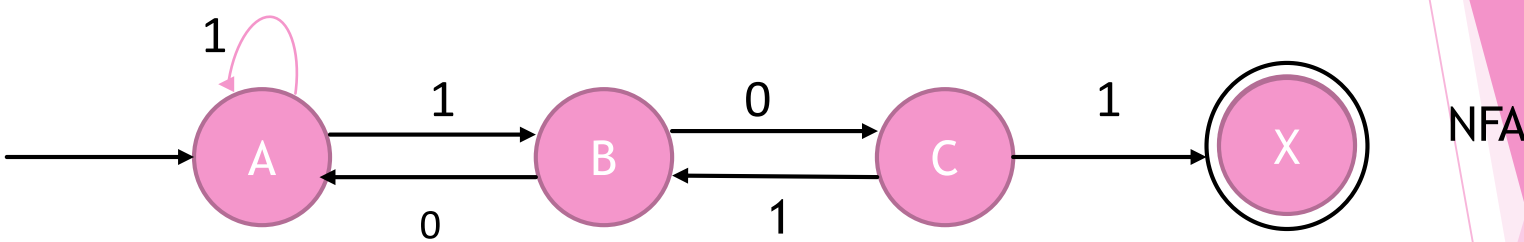
$\{q0, q1, q4\} \cap \{q2,q4\} \neq \emptyset$

The equivalence of DFA's and NFA's

For every NFA can construct an equivalent DFA (one which accepts the same language).

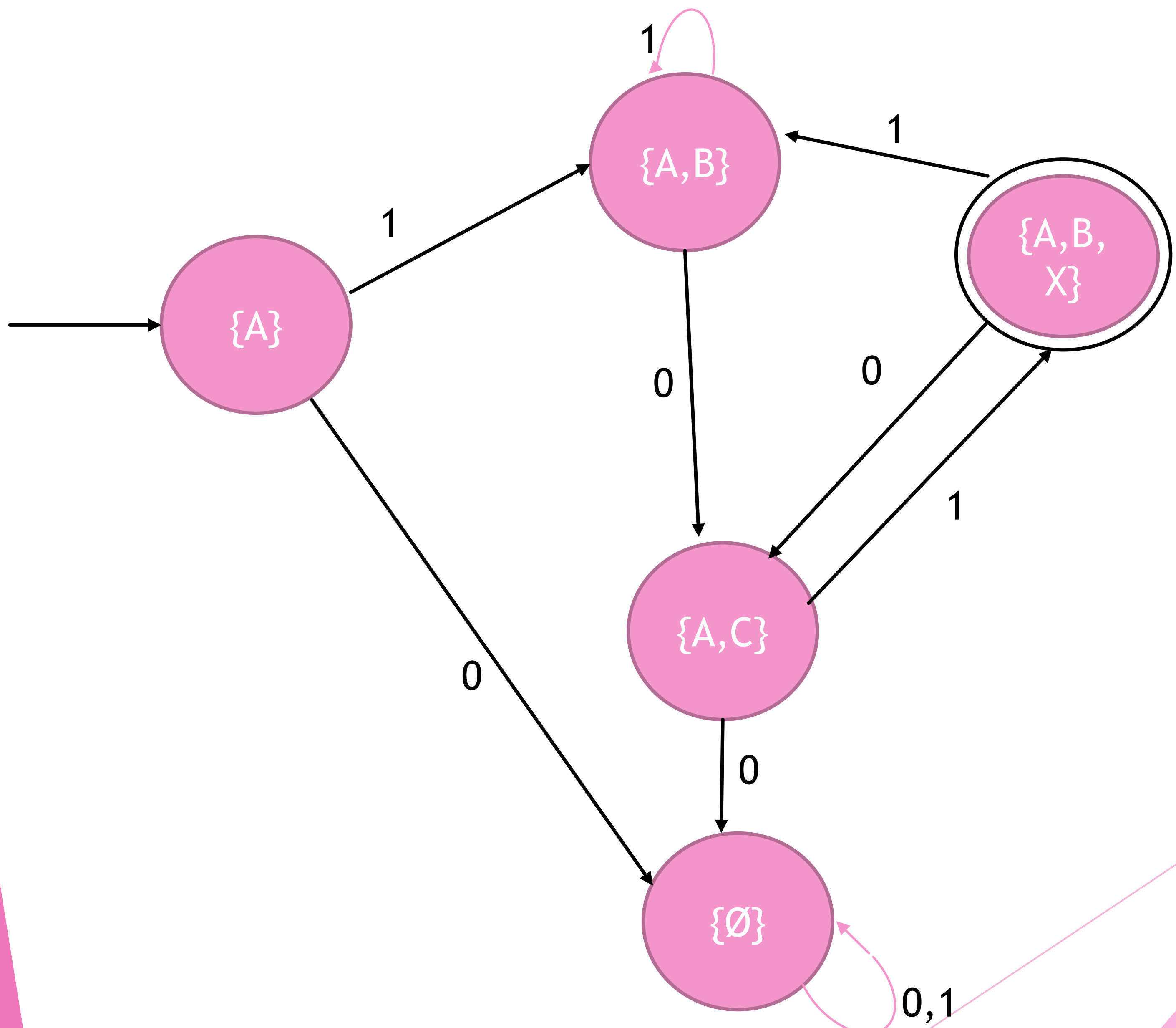
Definition

Let L be a set accepted by a non deterministic finite automata. Then there exists deterministic finite automata that accept L.

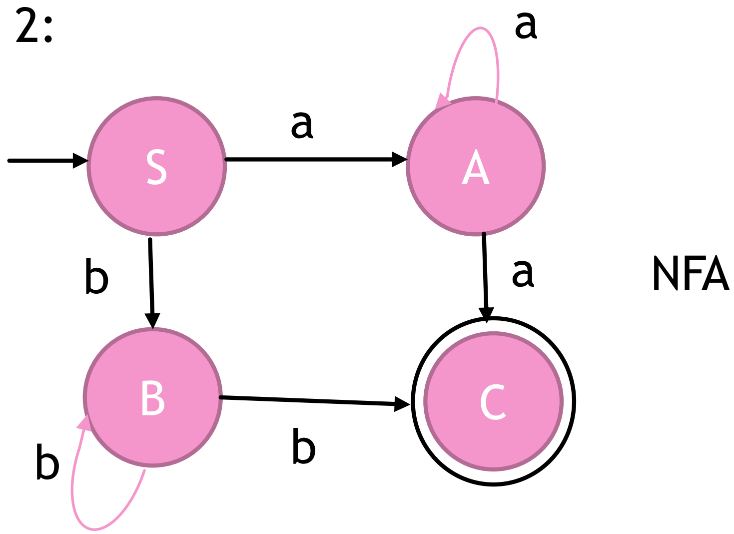


2^A={Ø,{A},{B},{C},{X},{A,B},{A,C},{A,X},{B,C},{B,X},{C,X},{A,B,C},{A,B,x},{B,C, x},{A, C, X}, {A, B, C, X}}

δ ({A}, 0) = Ø
δ ({A}, 1) ={A,B}
δ ({A,B}, 0) ={A,C}
δ ({A,B}, 1) ={A,B}
δ ({A,C}, 0) = Ø
δ ({A, C}, 1) = {A, B, X}
δ ({A, B, X}, 0) = {A, C}
δ ({A, B, X}, 1) = {A, B}


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

Example 2:



sol :

δ ({S}, a) = {A}

δ ({S}, b) = {B}

δ ({A}, a) = {A,C}

δ({A}, b) = Ø

δ ({B}, a) = Ø

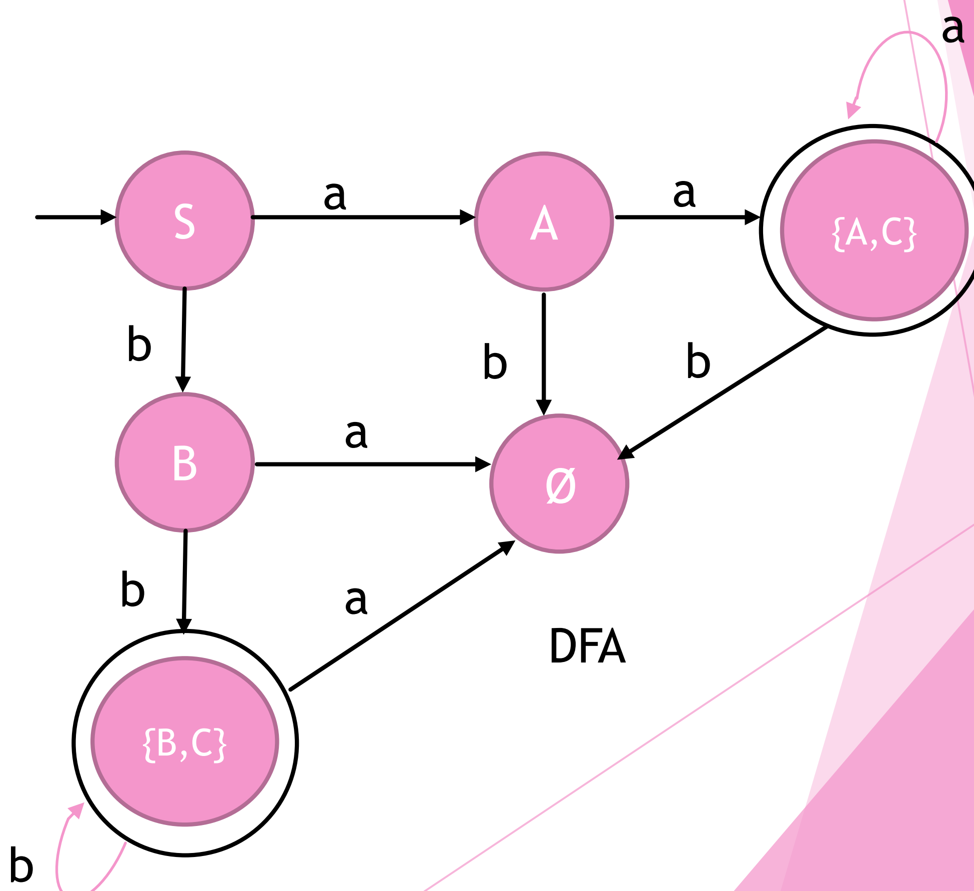
δ ({B}, b) = {B,C}

δ ({A,C}, a) = {A,C}

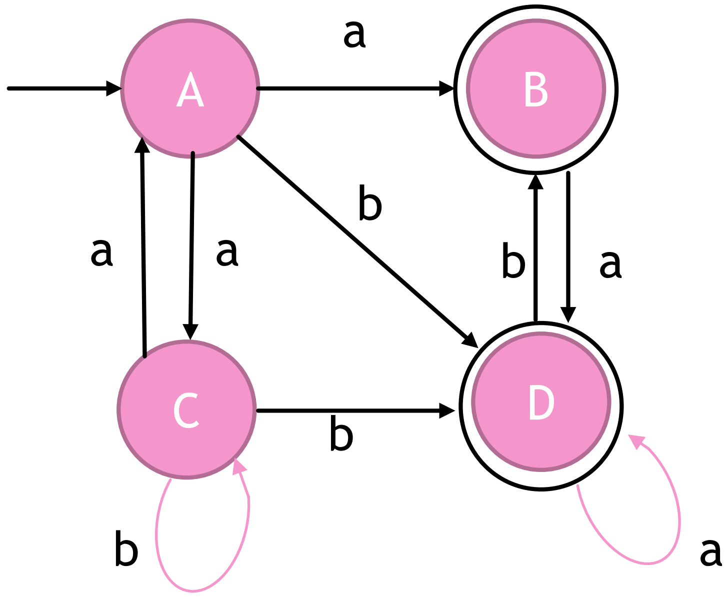
δ({A,C}, b) = Ø

δ ({B,C}, a) = Ø

δ ({B,C}, b) = {B,C}



HW:- Convert the following NFA to the DFA



Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 4

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Finite State System

Non Deterministic Finite Automata with

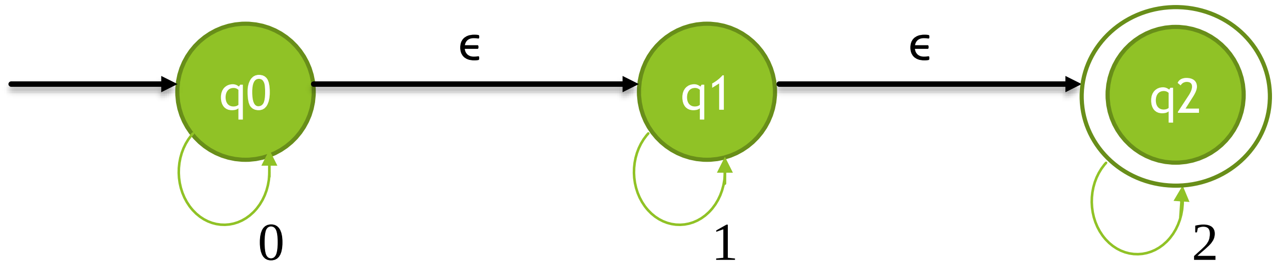
ϵ - moves

Finite Automata with ϵ - moves

Transition of NFA may be extended to include empty input ϵ.
NFA accepts a siring w if there is some path labeled w from the initial state to a final state.
Of course, edges labeled ϵ may be included in the path, although the ϵ's do not appear explicitly in w.

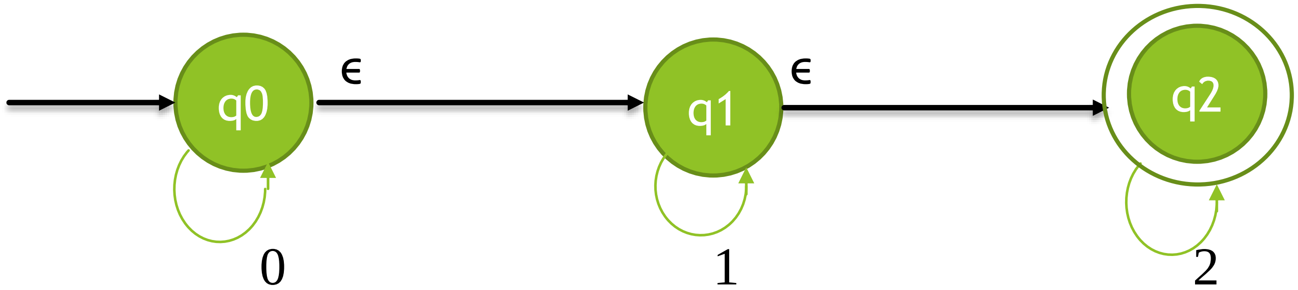
Example:-

Consider an NFA with ϵ-moves which accepts the language consisting of any number (including zero) of 0's followed by any number of 1's followed by any number of 2's.



Definition

Nondeterministic finite automata with ϵ -moves to be 5-tuple (Q, Ɛ, δ, q_0, F) the transition function map Q*(Ɛ ᴜ {ϵ}) to $_2{}^Q$



Transition diagram

	0	1	2	ϵ
-q0	{q0}	Ø	Ø	{q1}
q1	Ø	{q1}	Ø	{q2}
+q2	Ø	Ø	{q2}	Ø

Transition table

The transition function δ can be extended to a function δ^ that map Q * Ɛ * to $_2{}^Q$.

- We use **ϵ-closure (q)** to denote the set of all P such that there is a path from q to P labeled ϵ.

ϵ-closure (q0) = {q0, q1, q2}

Equivalence of NFA's with and without ϵ -moves

Theorem: If L is accepted by NFA with ϵ-transition, then L is accepted by an NFA without ϵ-transitions.

M= (Q, Ɛ, δ, q_0, F) be an NFA with ϵ-transition

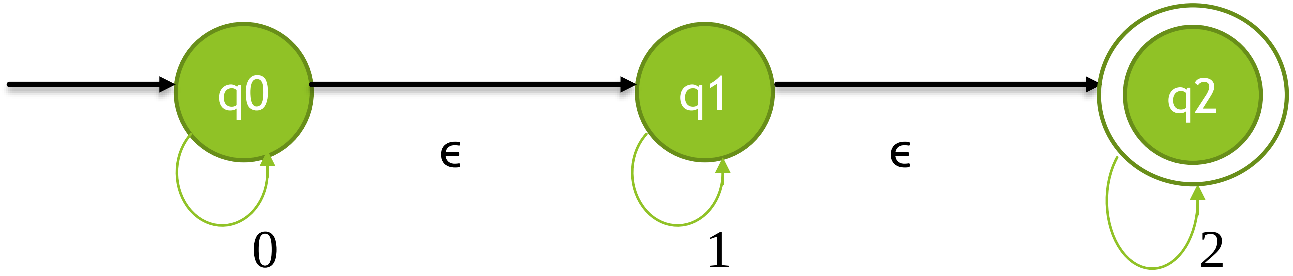
M^{-}= (Q, Ɛ, δ^{-}, q_0, F^{-}) where

$$F^{-} = \begin{cases} F \cup \{q0\} & \text{if } \epsilon\text{-closure } (q0) \text{ contains a state of } F \\ F & \text{otherwise} \end{cases}$$

- $\hat{\delta}$ (q0, ϵ) = **ϵ-closure** (q0)

- $\hat{\delta}$ (q0, 0) = **ϵ-closure** (δ ($\hat{\delta}$ (q0, ϵ), 0))
  = ϵ-closure (δ ({q0, q1, q2}, 0))
  = ϵ-closure (δ (q0, 0) ∪ δ (q1, 0) ∪ δ (q2, 0))
  = ϵ-closure ({q0} ∪ Ø ∪ Ø)
  = ϵ-closure (q0) = **{q0, q1, q2}**

Example1 :- Construct DFA equivalent the following NFA with ϵ



$\delta\hat{}\ (q0, \epsilon) =$ **ϵ-closure** $(q0) = \{q0, q1, \mathbf{q2}\}$

$\delta\hat{}\ (q1, \epsilon) =$ **ϵ-closure** $(q1) = \{q1, \mathbf{q2}\}$

$\delta\hat{}\ (q2, \epsilon) =$ **ϵ-closure** $(q2) = \{\mathbf{q2}\}$

$\delta\hat{}\ (q0, 0) =$ **ϵ-closure** $(\delta\ (\delta\hat{}\ (q0, \epsilon), 0))$

$= \epsilon\text{-closure}\ (\delta\ (\{q0, q1, q2\}, 0))$

$= \epsilon\text{-closure}\ (\delta\ (q0, 0) \cup \delta\ (q1, 0) \cup \delta\ (q2, 0))$

$= \epsilon\text{-closure}\ (\{q0\} \cup \emptyset \cup \emptyset)$

$= \epsilon\text{-closure}\ (q0) = \mathbf{\{q0, q1, q2\}}$

$\delta\hat{}\ (q0, 1) =$ **ϵ-closure** $(\delta\ (\delta\hat{}\ (q0, \epsilon), 1))$

$= \epsilon\text{-closure}\ (\delta\ (\{q0, q1, q2\}, 1))$

$= \epsilon\text{-closure}\ (\delta\ (q0, 1) \cup \delta\ (q1, 1) \cup \delta\ (q2, 1))$

$= \epsilon\text{-closure}\ (\emptyset \cup \{q1\} \cup \emptyset)$

$= \epsilon\text{-closure}\ (q1) = \mathbf{\{q1, q2\}}$

$\hat{\delta}$ (q0, 2) = ϵ-closure (δ ($\hat{\delta}$ (q0, ϵ), 2))

= ϵ-closure (δ ({q0, q1, q2}, 2))

= ϵ-closure (δ (q0, 2) ∪ δ (q1, 2) ∪ δ (q2, 2))

= ϵ-closure (Ø ∪ Ø ∪ {q2})

= ϵ-closure (q2) = **{q2}**

$\hat{\delta}$ (q1, 0) = ϵ-closure (δ ($\hat{\delta}$ (q1, ϵ), 0))

= ϵ-closure (δ ({q1, q2}, 0))

= ϵ-closure (δ (q1, 0) ∪ δ (q2, 0))

= ϵ-closure (Ø ∪ Ø)

= ϵ-closure (Ø) = **{Ø}**

$\hat{\delta}$ (q1, 1) =

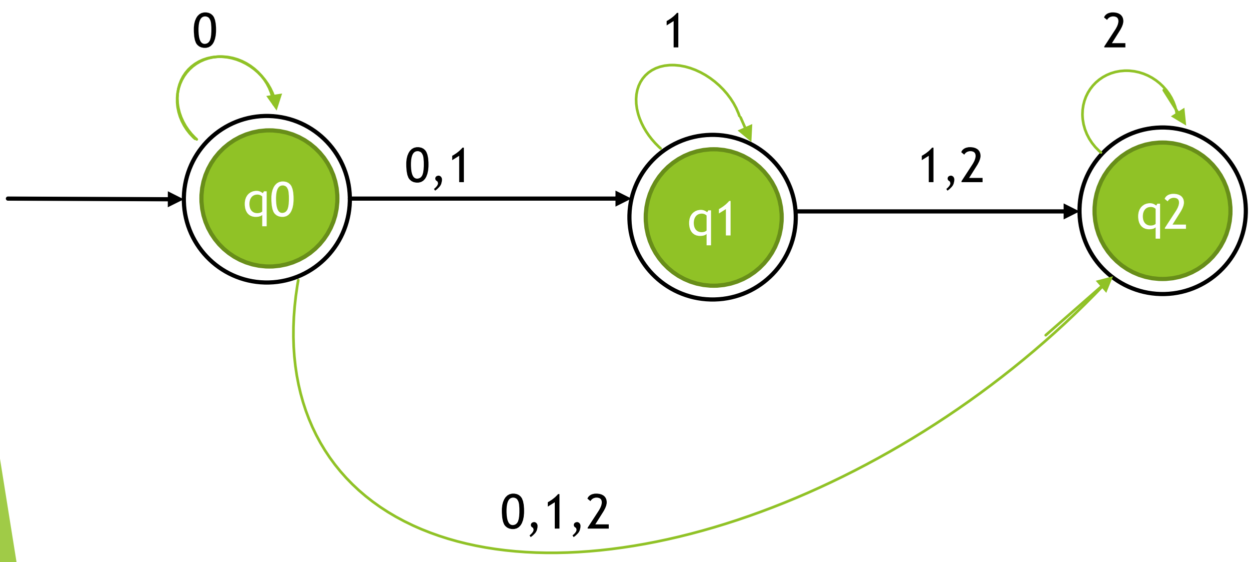
$\hat{\delta}$ (q1, 2) =

$\hat{\delta}$ (q2, 0) =

$\hat{\delta}$ (q2, 1) =

$\hat{\delta}$ (q2, 2) =

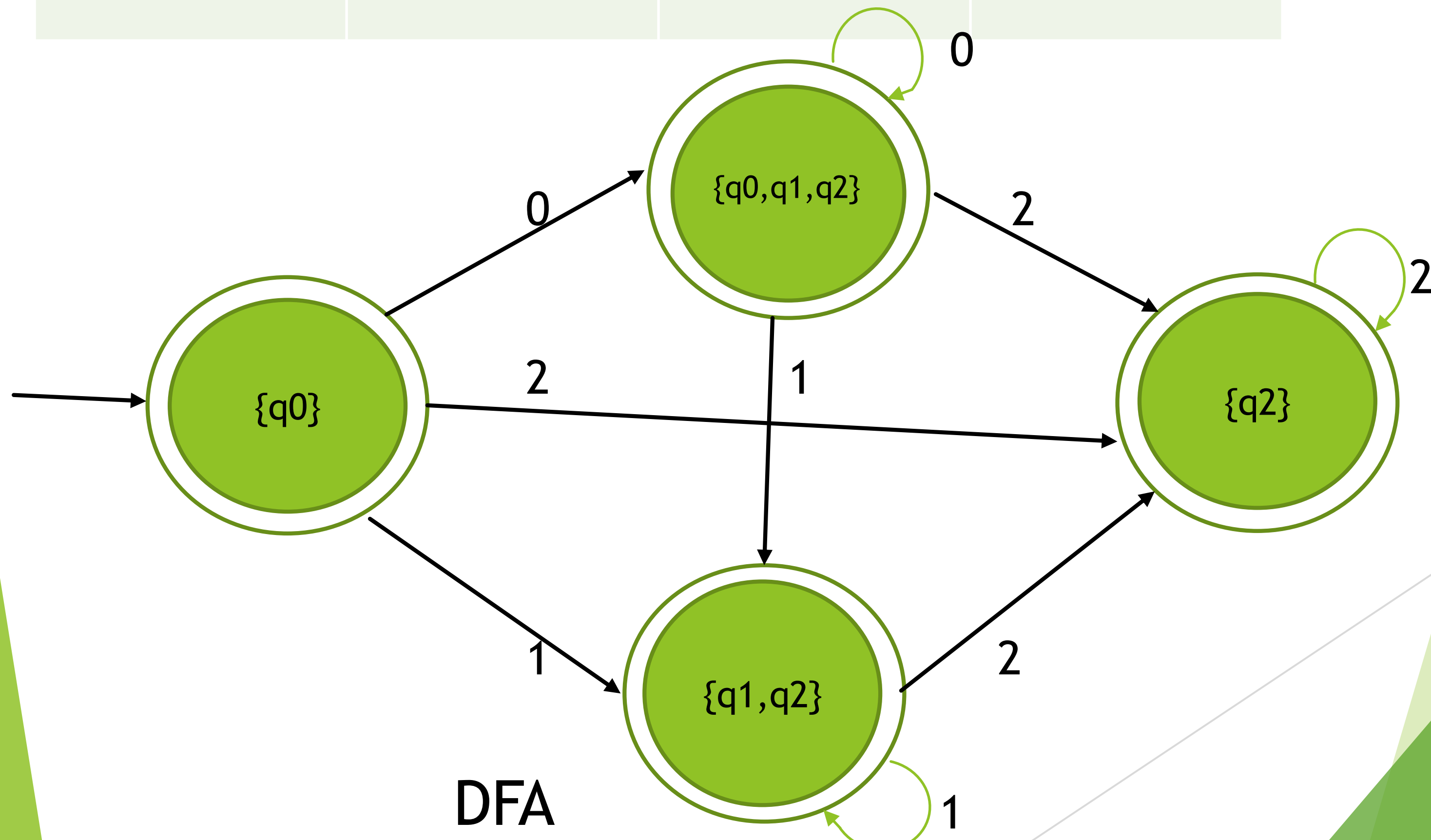
	0	1	2
-+q0	{q0,q1,q2}	{q1,q2}	{q2}
+q1	Ø	{q1,q2}	{q2}
+q2	Ø	Ø	{q2}



NFA with out ϵ

NFA without ϵ to DFA

	0	1	2
+-q0	{q0,q1,q2}	{q1,q2}	{q2}
+{q0,q1,q2}	{q0,q1,q2}	{q1,q2}	{q2}
+{q1,q2}	Ø	{q1,q2}	{q2}
+{q2}	Ø	Ø	{q2}



DFA

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory

2nd Class/1st Sem

Lecture 5

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Regular Expression

Regular Expression:

- The languages accepted by finite automata are easily described by simple expressions called Regular Expressions.
- Let Σ be an alphabet. The regular expressions over Σ and the sets that they denote are defined recursively as follows

  1- Ø is a regular expression and denotes the empty set {}.

  2- ϵ is a regular expression and denotes the set { ϵ }

  3- For each a in Σ, a is regular expression and denotes the set {a}.

  4- If r and s are regular expressions denoting the language R and S, respectively then (r+s), (rs), and (r^*) are regular expressions denote the sets RuS, RS and R^* respectively.

Regular Expression:

- In writing regular expressions we can omit many parentheses if we assume that * has higher precedence than concatenation or +, and that concatenation has higher precedence than +, for example ((0(1*)) +0) may be written 01*+ 0.

We may abbreviate the expression rr* by r^+ .

$r^* =\{\epsilon, r, rr, rrr, rrrr, \ldots\}$

$r^+ =\{r, rr, rrr, rrrr, \ldots\}$

$rr^* = r(\epsilon, r, rr, rrr, rrrr, \ldots) = r, rr, rrr, rrrr, \ldots = r^+$

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

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

$L^0 = \epsilon$

Language Example

example1:

Let L1= {10, 1} and L2= {011, 11}

L1L2 = {10011, 1011, 111}

L1+L2= {10, 1, 011, 11}

$L1^*$= {ϵ, 10, 1, 1010, 11, …}

$L1^+$= {10, 1, 1010, 11,…}

example2:

L1= {01, 0}, L2= {ϵ, 0, 10}

L1L2= {01, 010, 0110, 0, 00}

L2L1= {01, 0, 001, 00, 1001, 100}

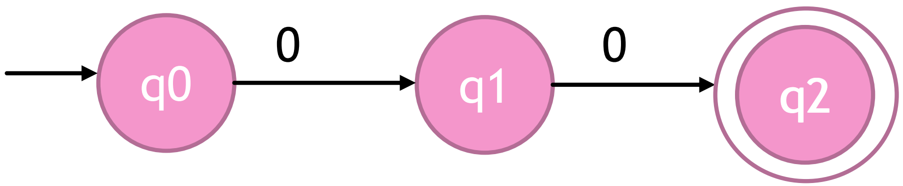
L1ϵ=L1= {01, 0}

$L1^*$={ϵ, 01,0,0101,00,010,001,…}

Examples of Regular Expression

- **00** is a regular expression representing {00}.

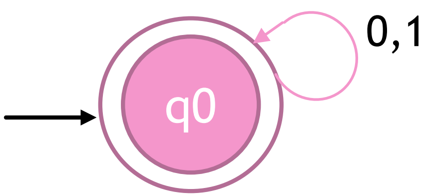
  L={00}



FA

- (0+1) * is **a regular expression denotes all strings** of 0's and 1's.

  L= {ϵ, 0, 1, 00, 11, 01, 10, …}



- **0^*+1^*** is a regular expression denotes all strings of 0's and 1's

  $0^* = \{\epsilon, 0, 00, 000, \ldots\}$
  $1^* = \{\epsilon, 1, 11, 111, \ldots\}$
  $L = 0^*+1^* = \{\epsilon, 0, 1, 00, 11, 000, 111, \ldots\}$

- **$0^*1^*2^*$** is a regular expression denotes the language of any number of 0's followed by any number of 1's followed by any number of 2's.
- **$(0+1)^*011$** is a regular expression denotes the language of mixed group of 0's and 1's ended by 011.

- **$((0+1)^*\ 00\ (0+1)^*$** is a regular expression denotes the language of **all string of 0's and 1's with at least two consecutive 0's.**

- Finite Language L that contains all the strings of a's of b's of length exactly three L= {aaa, aab, aba, abb, bab, bba, bbb,baa}. The first letter of each word in L is either a or b, the second letter of each word in L is either is either a or b, the third letter of each word in L is either a or b so we may write **L=((a+b)(a+b)(a+b)).**
- **$a(a+b)^*b$** is a regular expression denotes the language of all words that begin with a and end with b .
- **$(a+b)^*aa(a+b)^*$** is a regular expression denotes the language all words over the alphabet Σ= {a,b} with at least two consecutive a's
- **(a+b)*abb** is a regular expression denotes the language of all string of a's and b's ending in abb.

- The language defined by the regular expression **a^*b^*** is the set of all the string of a's and b's L={ ϵ, a, b, aa, bb, ab, aaa, aab, abb, bbb, aaaa,…}.
- Language of expressions **a*b* ≠(ab)***

Since the language defined by the expression on the right contains the word abab, which the language defined by the expression on the left does not.

- Consider the language T defined over the alphabet Σ= {a, b, c} ,

T= {a, c, ab, cb, abb, cbb, abbb, cbbb, abbbb, cbbbb, abbbbb, cbbbbb,…}.
All the words in T begin with an a or c and then are followed by some number of b's.

Symbolical we may write this as **T= ((a+c) b*).**

- $(b+ba)^*$ is a regular expression denotes the language of all string of a's and b's beginning with b and not having two consecutive a's.
- $\mathbf{(a+b)^*(aa+bb)(a+b)^*}$ is a regular expression denotes the language

  of all words over the alphabet $\Sigma = \{a,b\}$ with at least two.

  consecutive a's or two consecutive b's
- The language defined by the regular expression **ab*a** is the set of all string of a's and b's that have at least two letters, that begin and end with a's

  $b^* = \{\epsilon, b, bb, bbb, bbbb, \dots\}$

  $L = \{aa, aba, abba, abbba, abbbba, \dots\}$

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 6

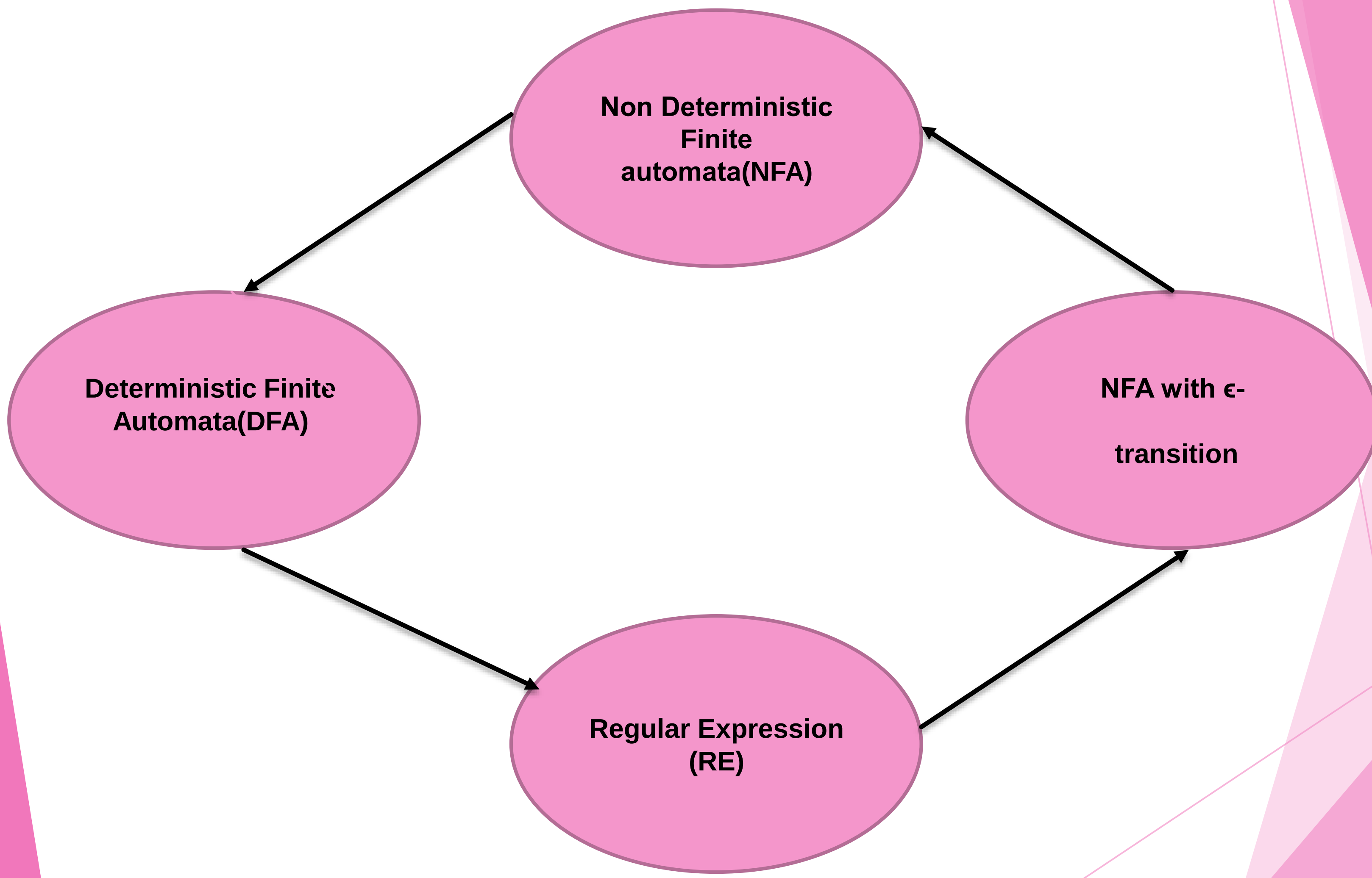
ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Equivalence of finite automata and regular expression

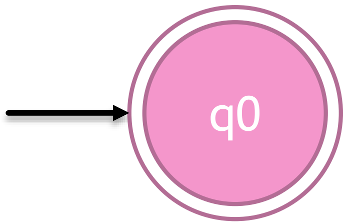
Equivalence of finite automata and regular expression:



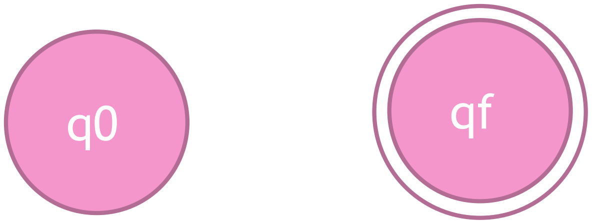
Theorem

Let r be a regular expression. Then there exists an NFA with ϵ-transitions that accepts L(r).

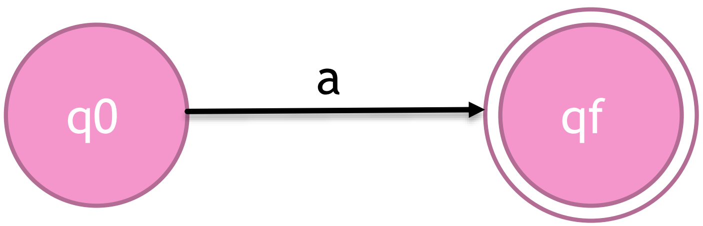
Case 0 (zero operation)



(a) r= ϵ



(b) r= Ø

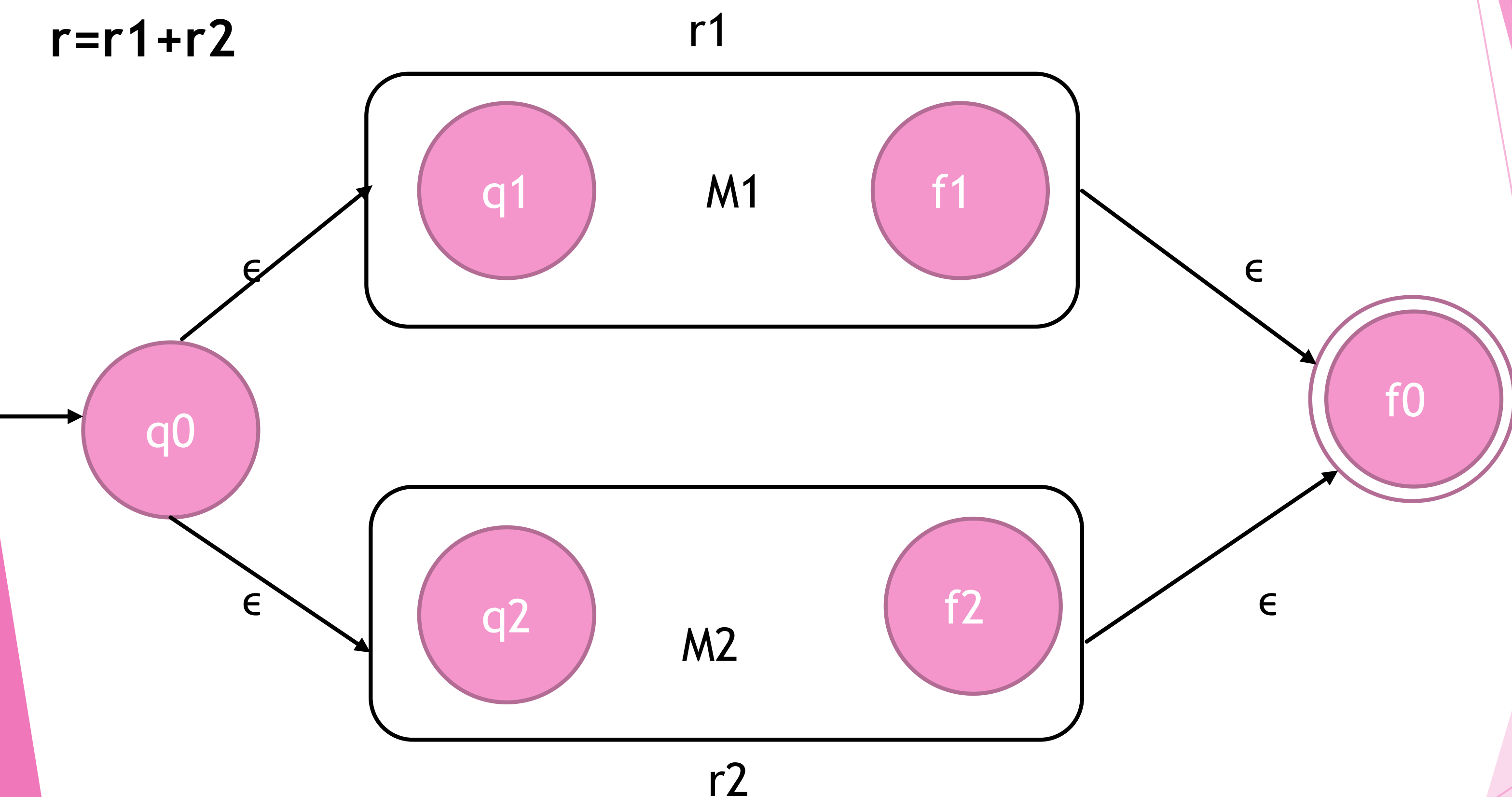


(C) r=a

Regular Expression:

Case 1 (one or more operators)

r=r1+r2



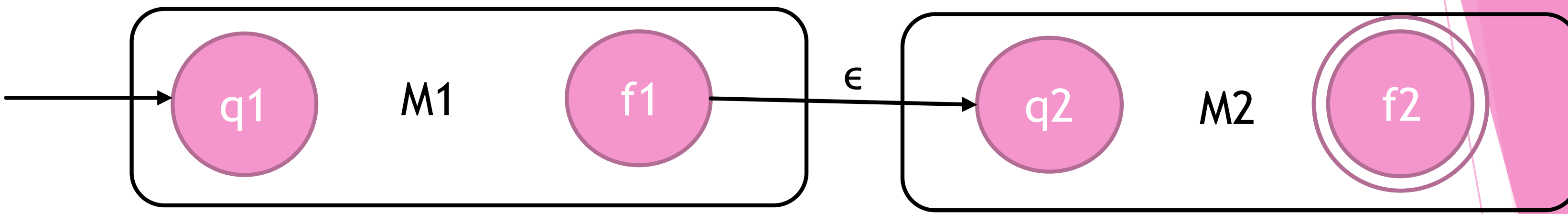
M1=(Q1, Σ1,δ1,q1,f1)

M1=(Q2, Σ2,δ2,q2,f2) let q0 be a new initial state and f0 anew final state

M=(Q1∪ Q2 ∪{q0,f0}, Σ1∪ Σ2, δ,q0,f0}

Case 2

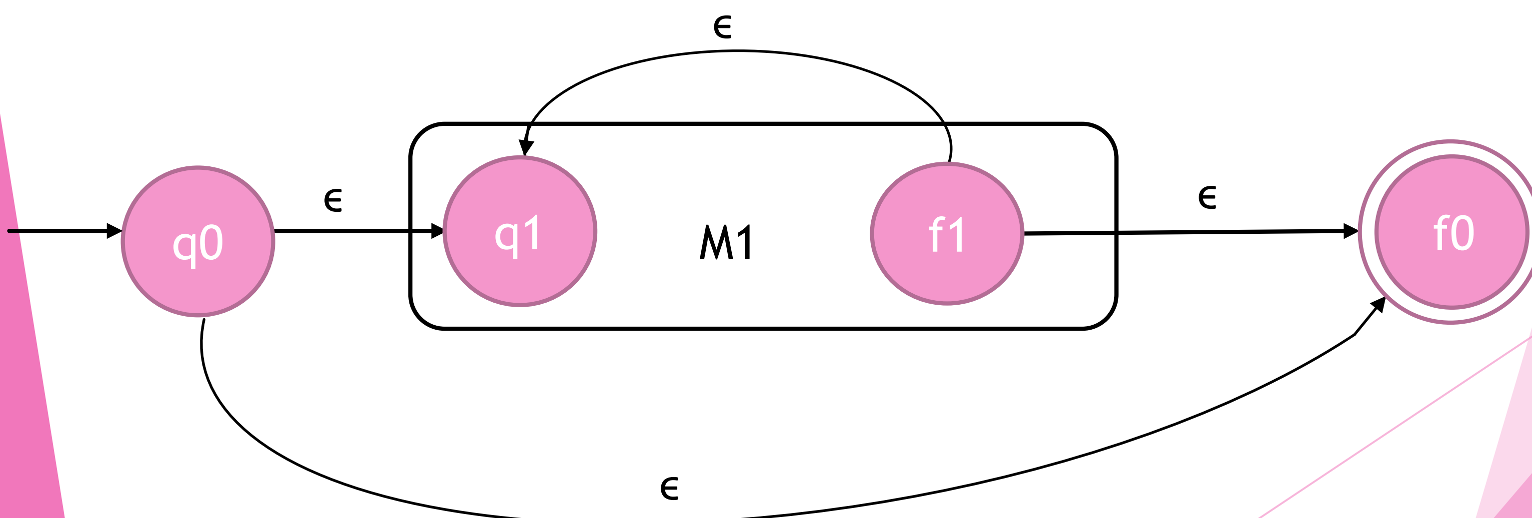
r=r1r2



$M=(Q1\cup Q2,\ \Sigma1\cup \Sigma2,\ \delta, q1, f2\}$

Case 3

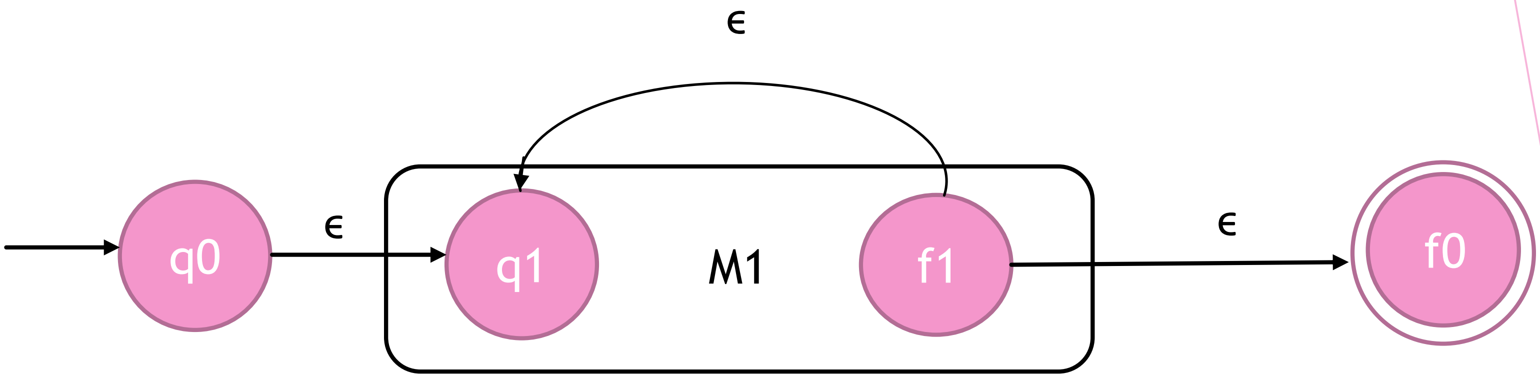
r=r1*



$M=(Q1\cup \{q0, fo\},\ \Sigma 1, \delta, q0, f0\}$

Case 3

$r = r1^{+}$



$$M = (Q1 \cup \{q0, fo\},\ \Sigma 1, \delta, q0, f0\}$$

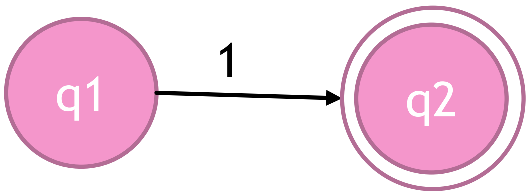
Example1

Construct an NFA with ϵ for the following regular expression

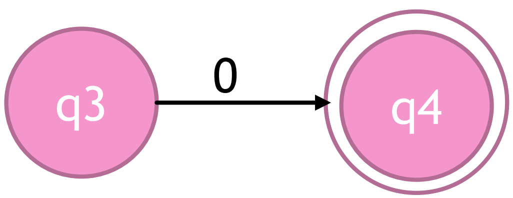
RE=01*+1

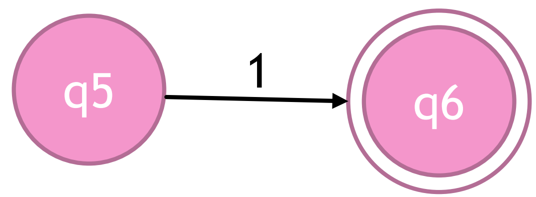
r= r1 +r2

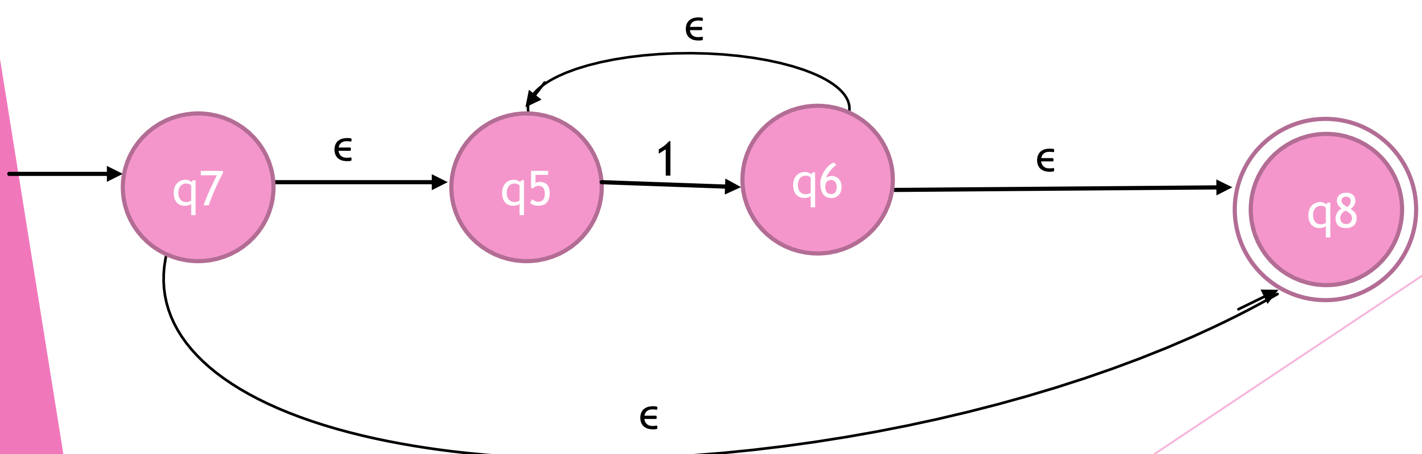
r2=1



r1= 01*



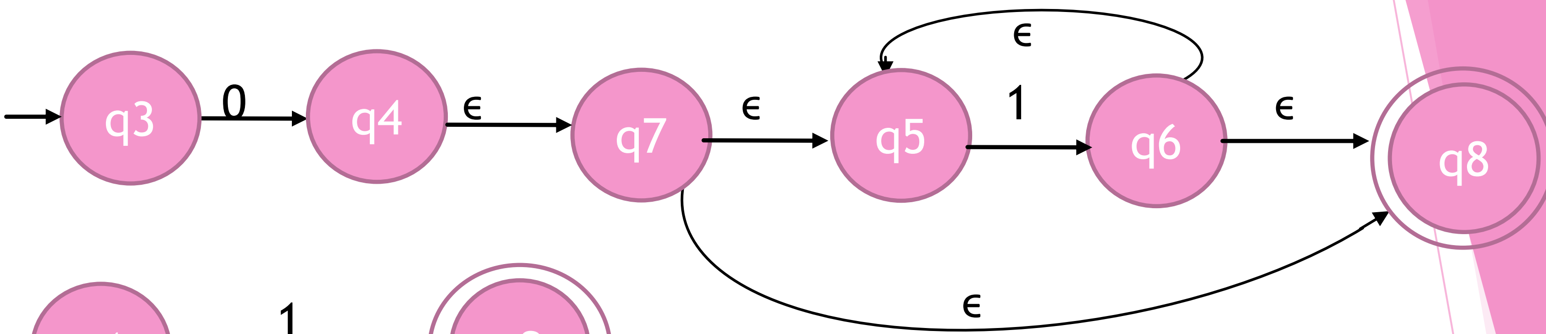




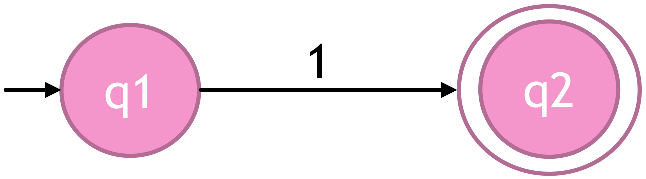
1*

Example1:

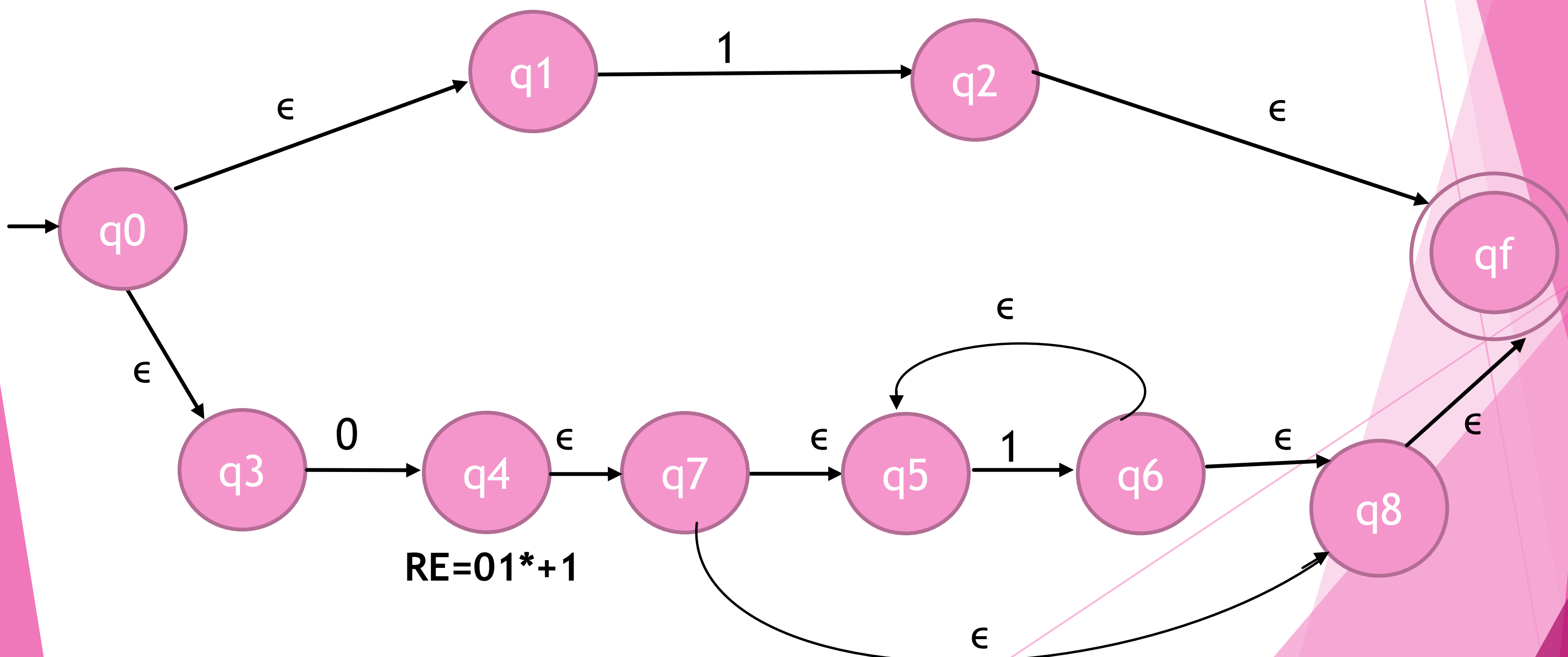
Construct an NFA with ϵ for the following regular expression **RE=01*+1**



r1= 01*



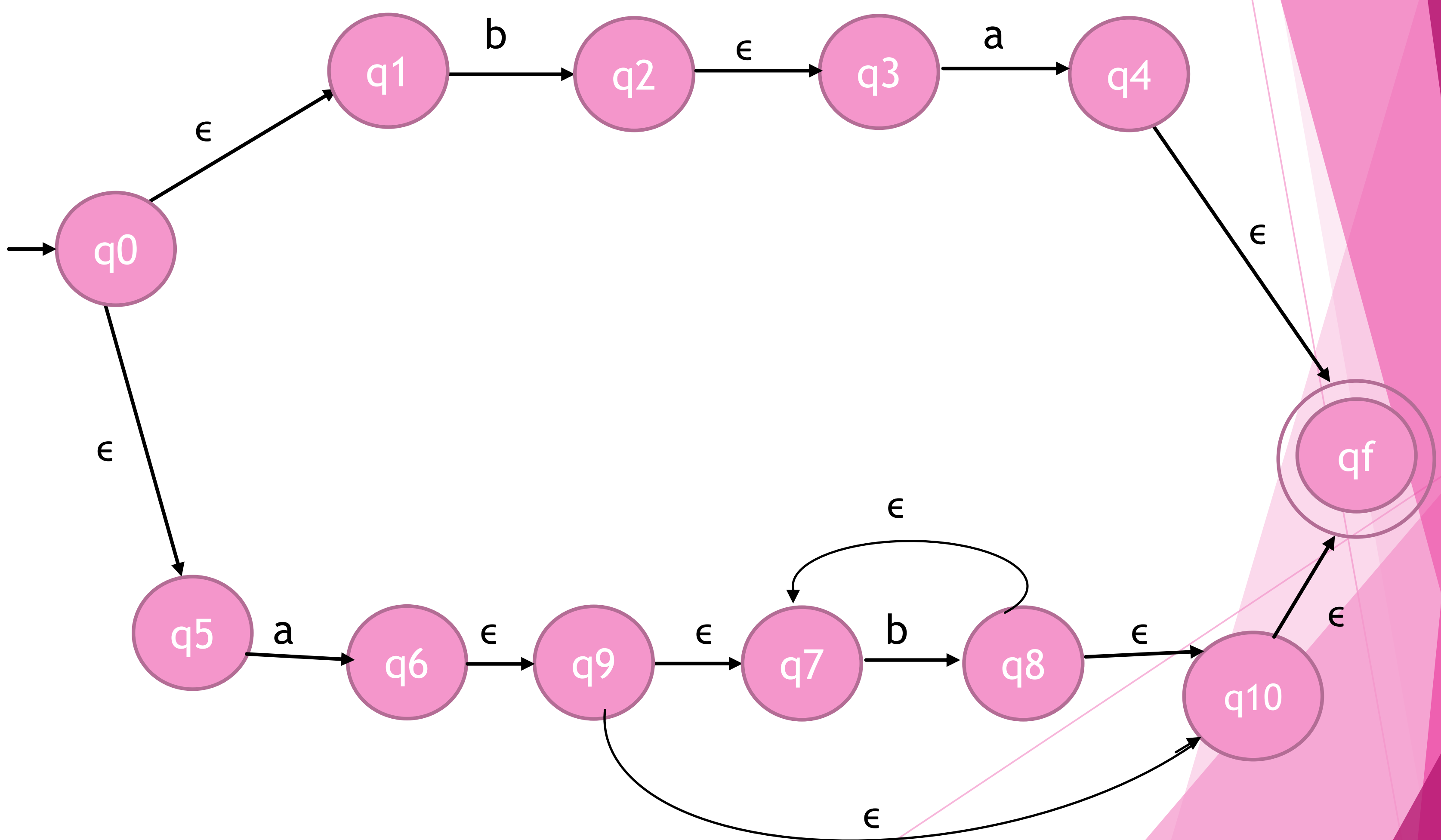
r2= 1



Example2

Construct an NFA with ϵ for the following regular expression

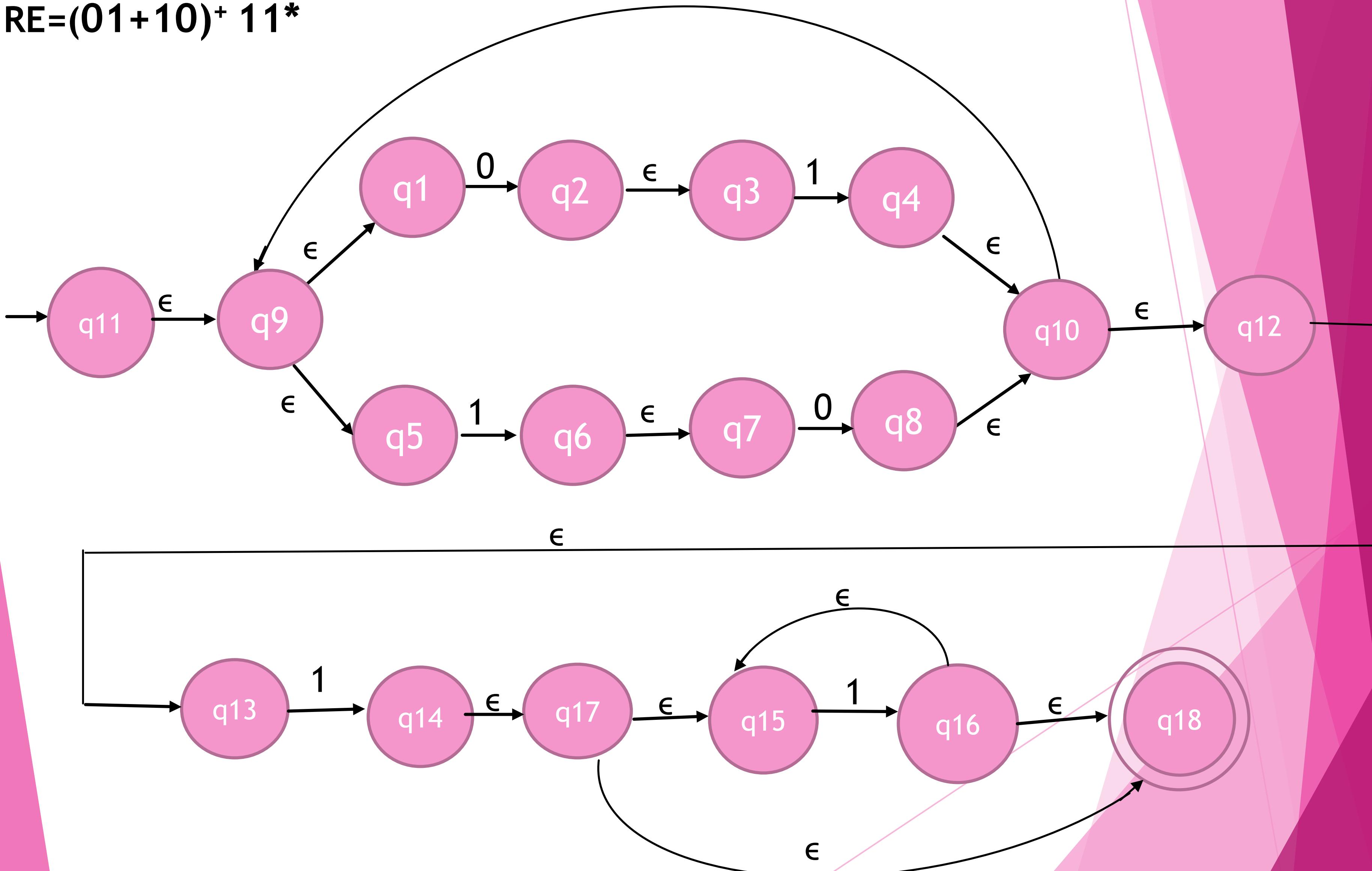
RE=(ba+ab*)



Example3

Construct an NFA with ϵ for the following regular expression

RE=(01+10)$^{+}$ 11*



Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 7

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Transition Graph (TG)

Transition Graph:

A transition graph abbreviated TG is a collection of $(\mathrm{Q}, \Sigma, \delta, q0, F)$

Q is a set of states

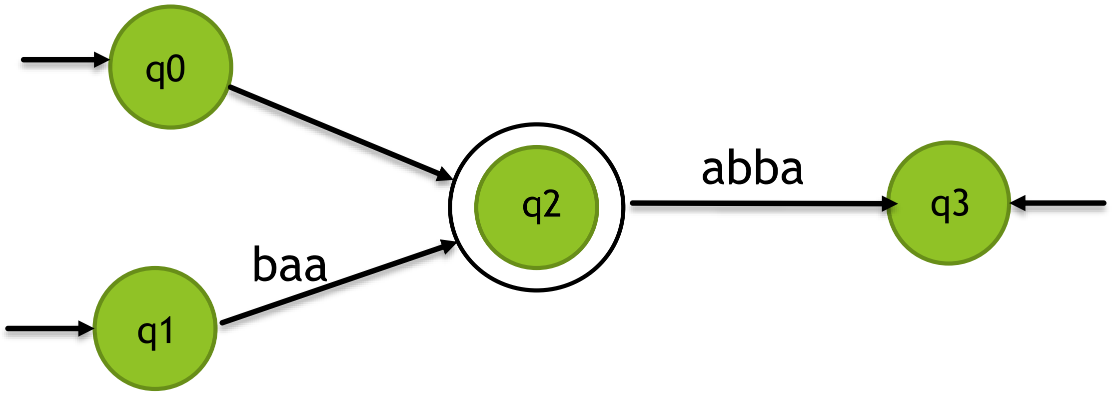
Σ is an input alphabet

δ A finite set of transition that show how to go from one state to another based on reading specified substring of input letters(possibly even the null string Є).

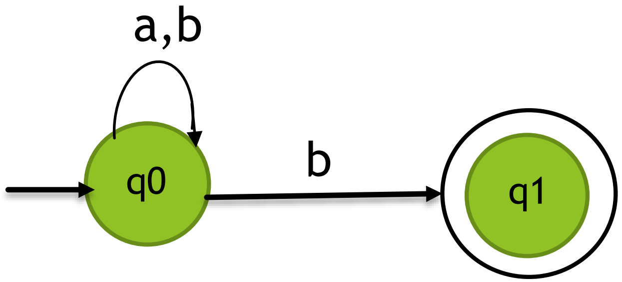
q0 $q0 \in Q$ is the initial state.

F is a set of final states F$\subseteq Q$(may be none)

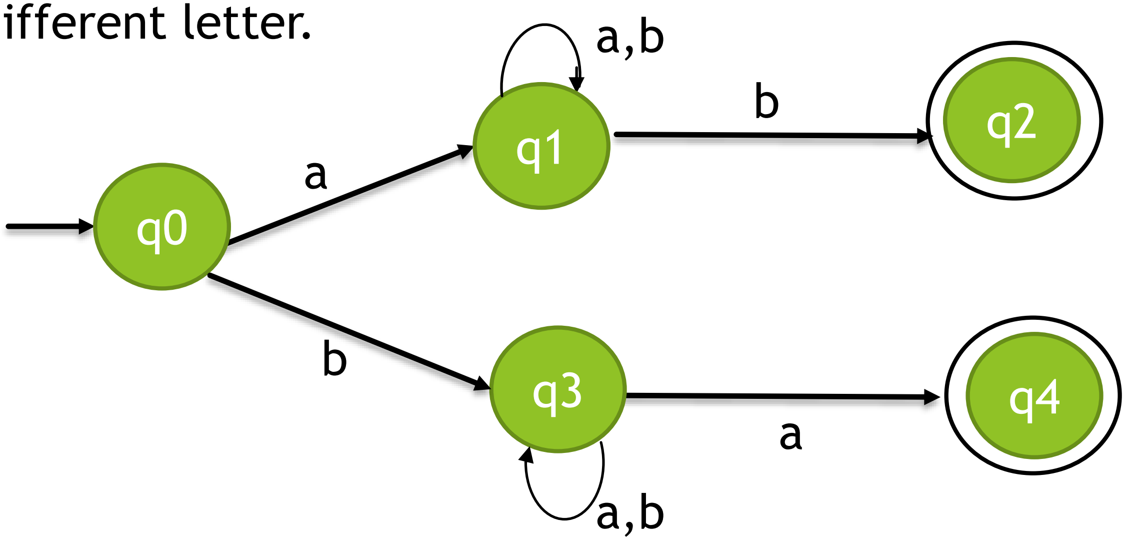
Example1:



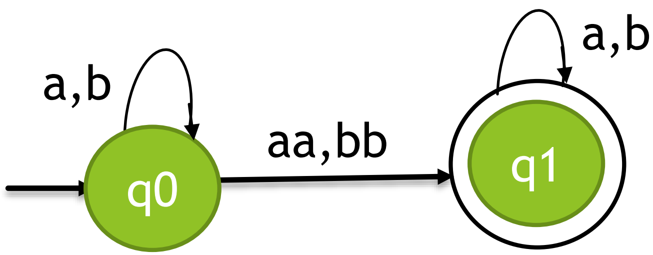
Example2:



Example3: the following TG accept a language of all words that begin and end with different letter.

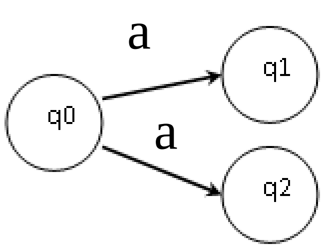


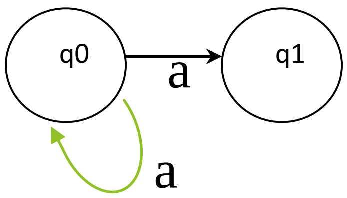
Example 4: Design a TG for that recognize all words that contain double letter $\Sigma=\{a,b\}$



Transition Graph (TG)
It is the FA that allows one or more transition from a state on the same input symbol.
NFA is a 5-tuple M= (Q, Σ, δ, q_0, F) Where
Where Q, Σ and F have the same meaning as for a DFA, but δ is a map from
Q * Σ to 2^Q.
(2^Q is the power set of Q, the set of all subset of Q)

-Transition diagram





Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

VIEDEO LECTURES

Computation Theory

2nd Class/1st Sem

Lecture 8

ا.م وجدان عبد الامير حسن

Computation Theory

Kleene's Theorem

Kleene's theorem

Any language that can be defined by

1-Regular Expression

2-Finite Automata

3-Transition Graph

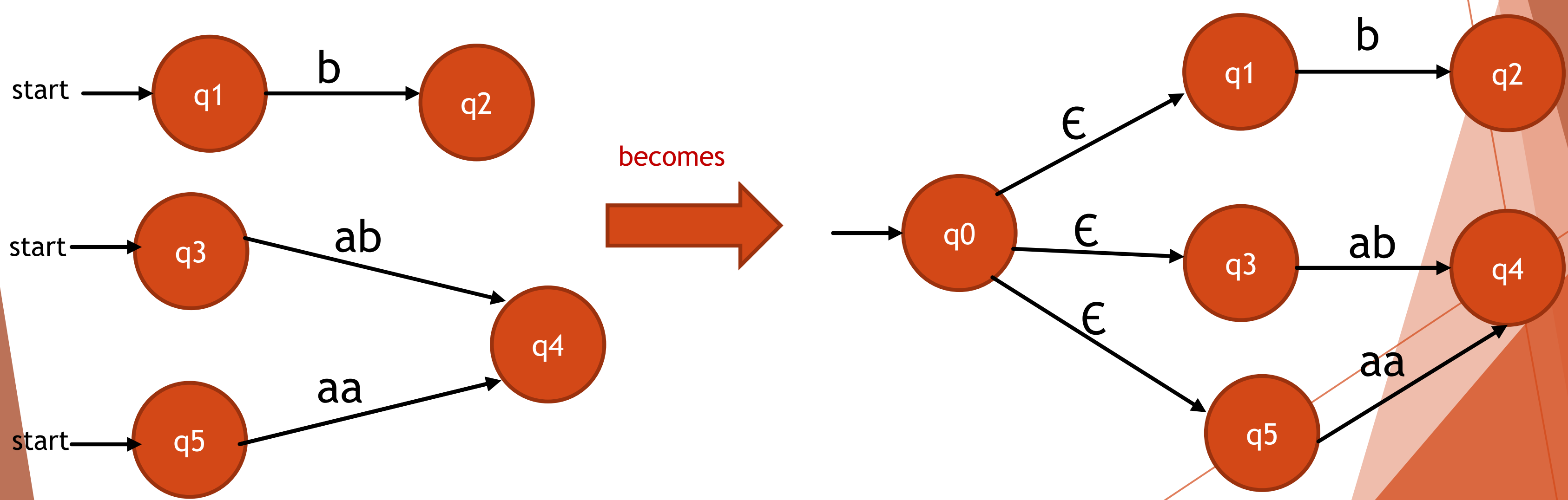
Part1:-Every language that can be defined by a finite automata can also be defined by a transition graph.

Every finite automata is itself a transition graph. therefore, any language that has been defined by a finite automata has already been defined by a transition graph.

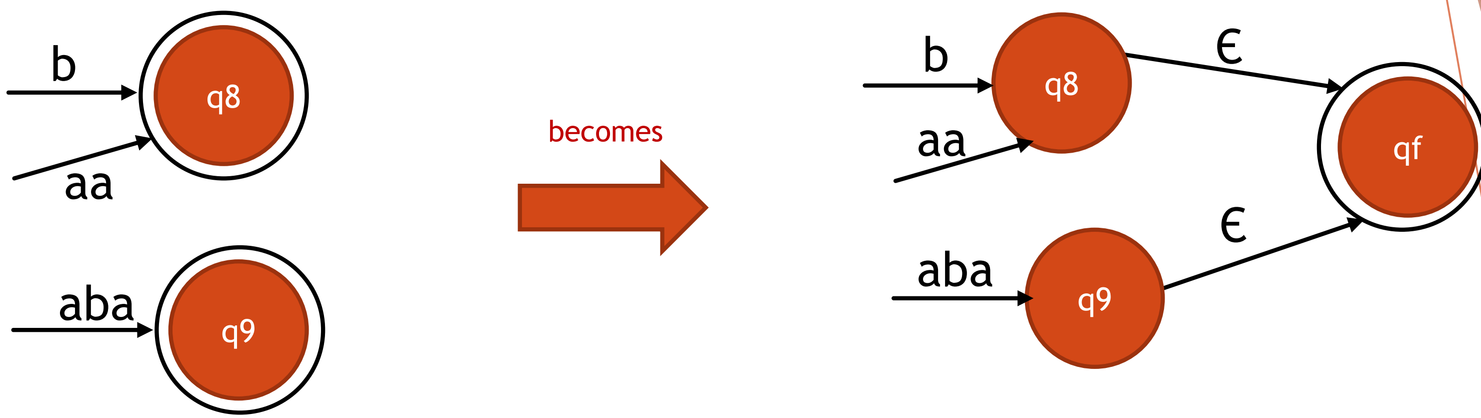
<u>Part2:-</u>Every language that can be defined by a transition graph can also be defined by a regular expression.

This means that we present a procedure that stars out with a transition graph and ends with a regular expression that defined the same language.

First want to simply T so that it has only one start state

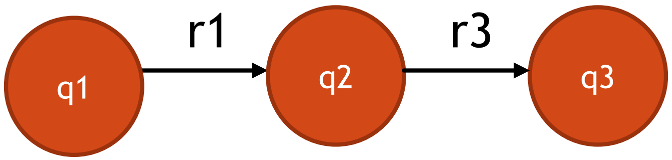


Another simplification we can make in TG is that it can be modified to have a unique final state without changing the language it accepts.

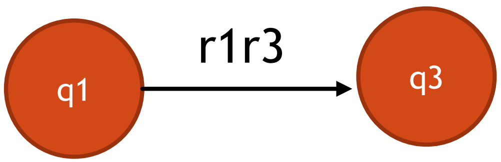


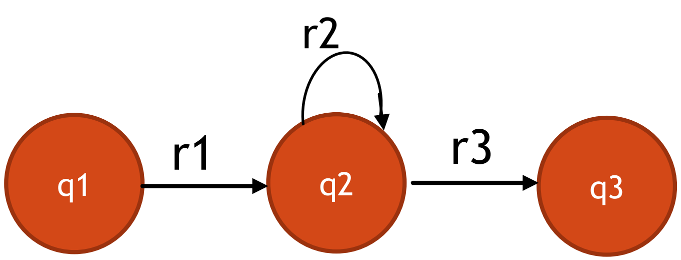
It should be clear that the addition of these two new states does not affect the language that TG(transition graph) accepts. Any word accepted by old TG is also accepted by the new TG, and any word rejected by the old TG is also rejected by the new TG.

Pass operation:- If three states in arrow connected by edges labeled with regular expression, we can eliminate the middleman and go directly from one outer state to the other by a new edge labeled with a regular expression that is the concatenation of the two previous labels.

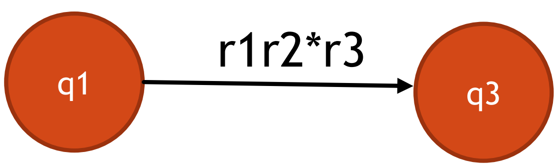


We can replace this with

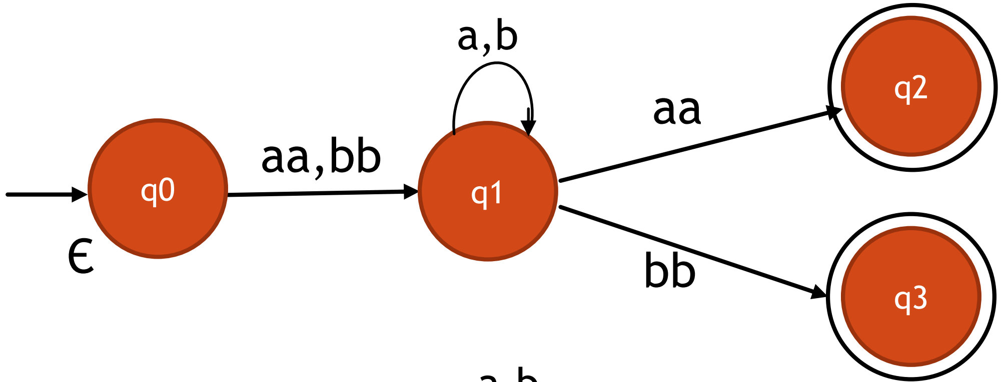


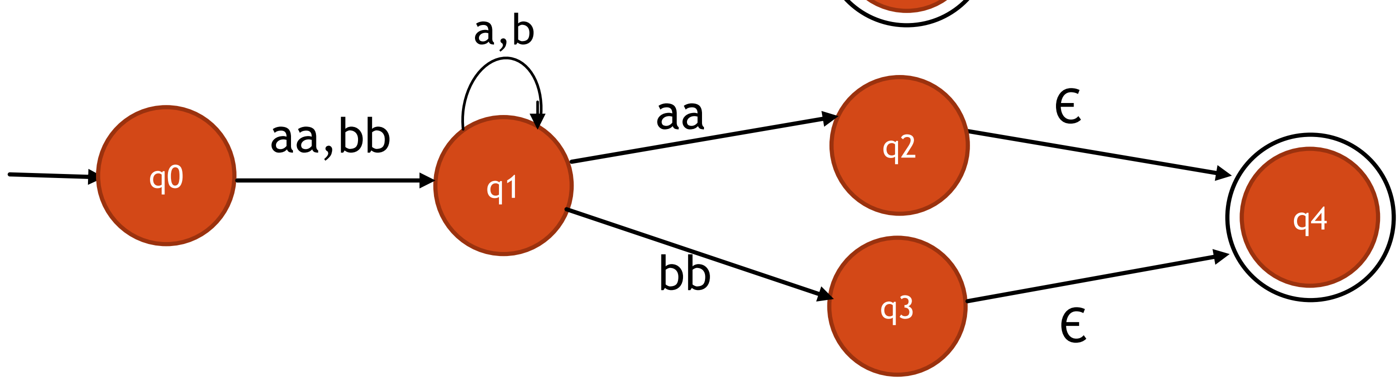


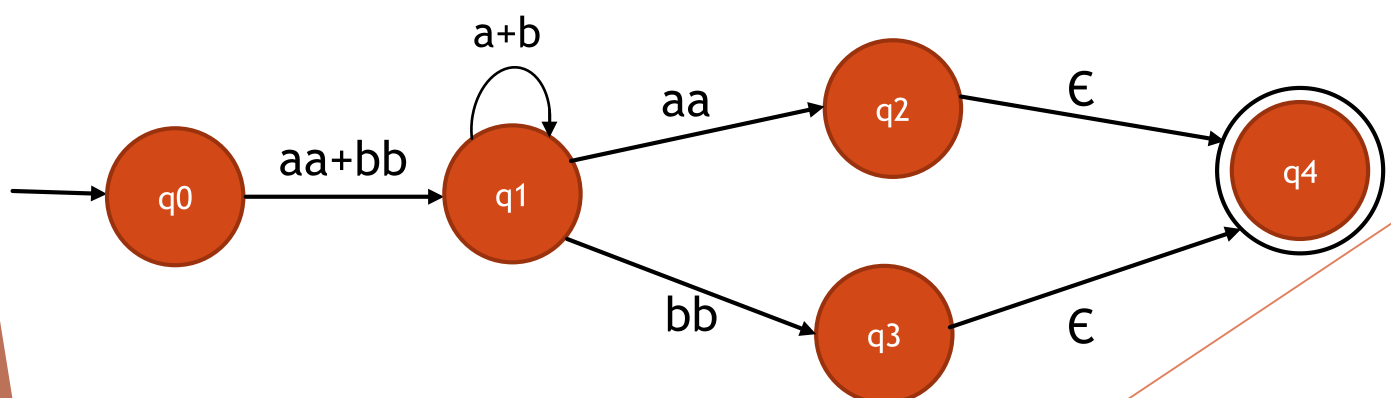
becomes



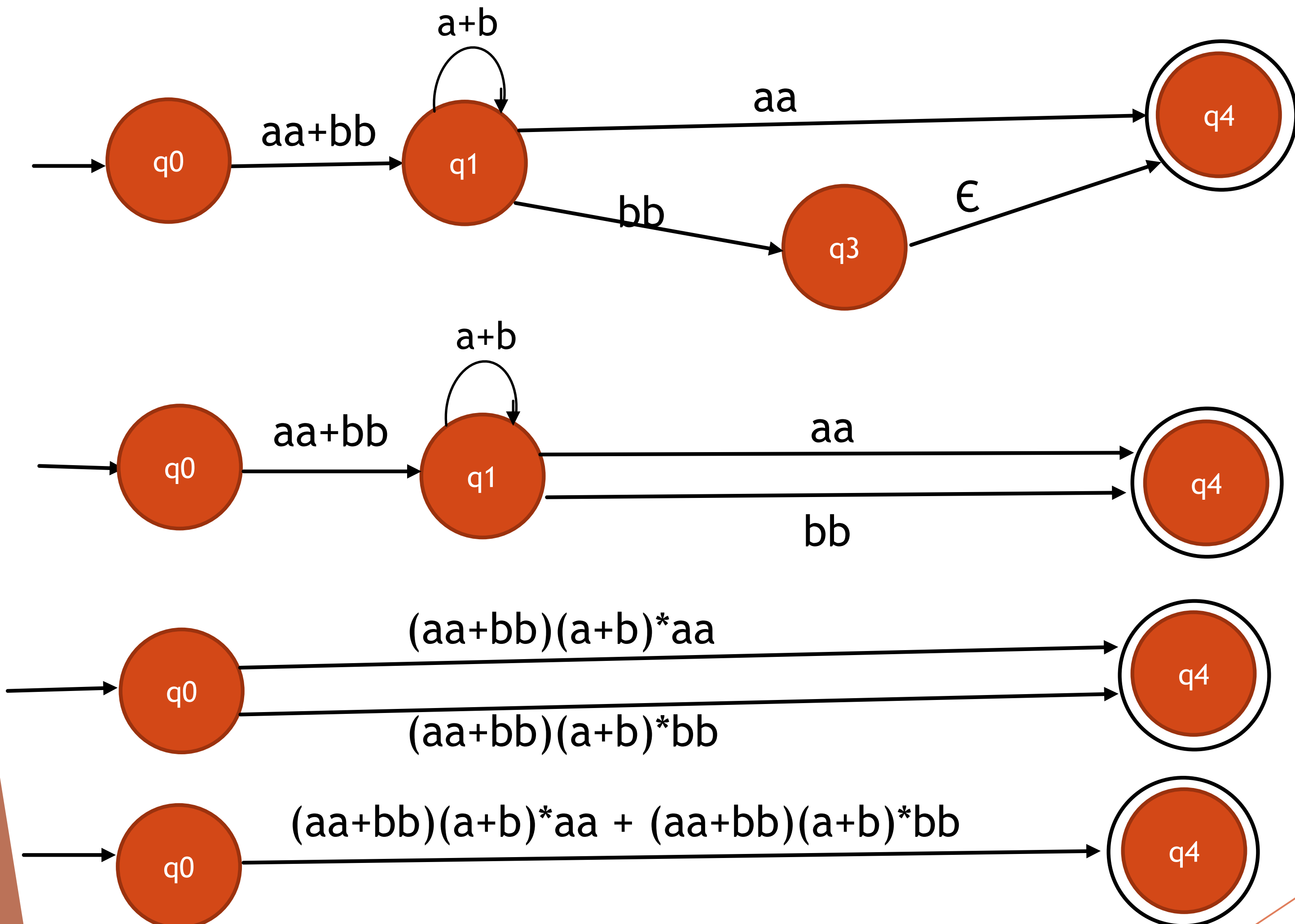
Example:-TG which accepts all words that begin and end with double letters





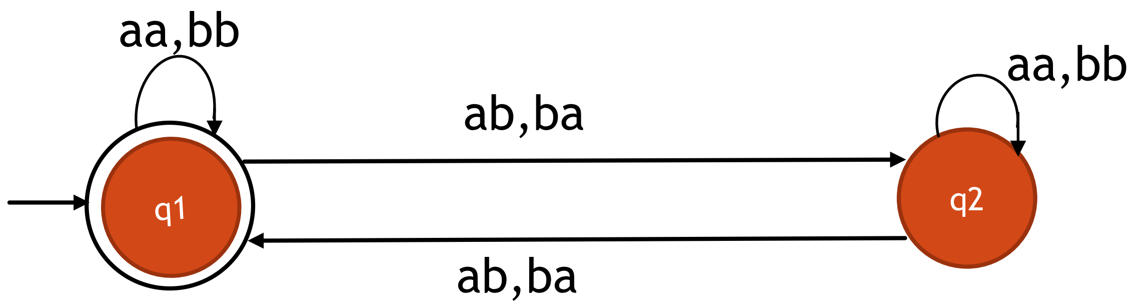


Example:-TG which accepts all words that begin and end with double letters



Re=(aa+bb)(a+b)*aa + (aa+bb)(a+b)*bb = (aa+bb)(a+b)* (aa+bb)

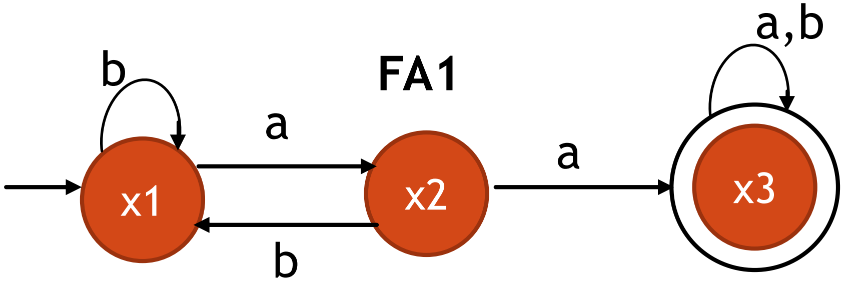
homework:-The following TG which accepts all words with an even number of a’s and even number of b’s

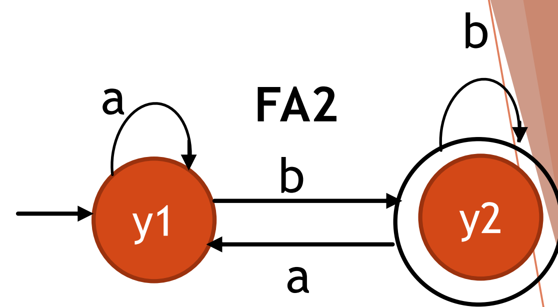


What is the regular expression of the above diagram ?

Part 3:- Every language that can be defined by a regular expression can also be defined by a finite automata.
If there is an FA called FA1 that accepts the language defined by regular expression r1 and there is an FA called FA2 that accepts the language defined by the regular expression r2, then there is an FA called FA3 that accepts the language defined by regular expression (r1+r2).

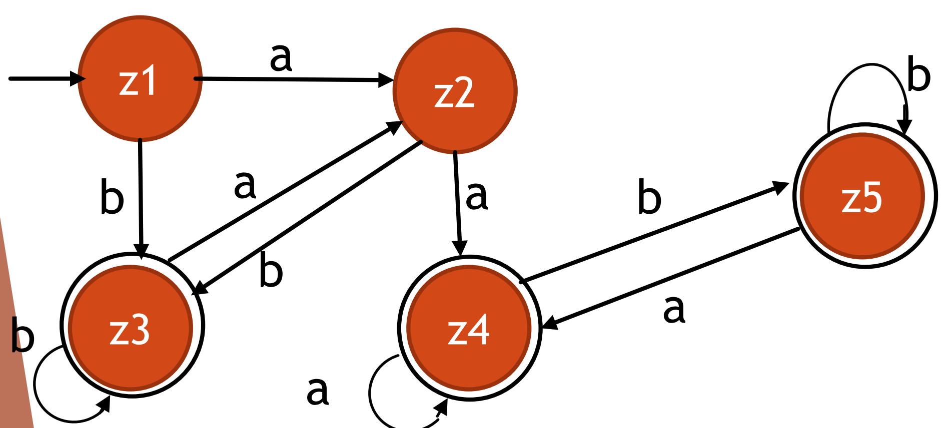
Example:Find FA1+FA2





FA1 =the machine that accepts only strings with a double a in them
FA2 =the machine that accepts all words that end in the letter b

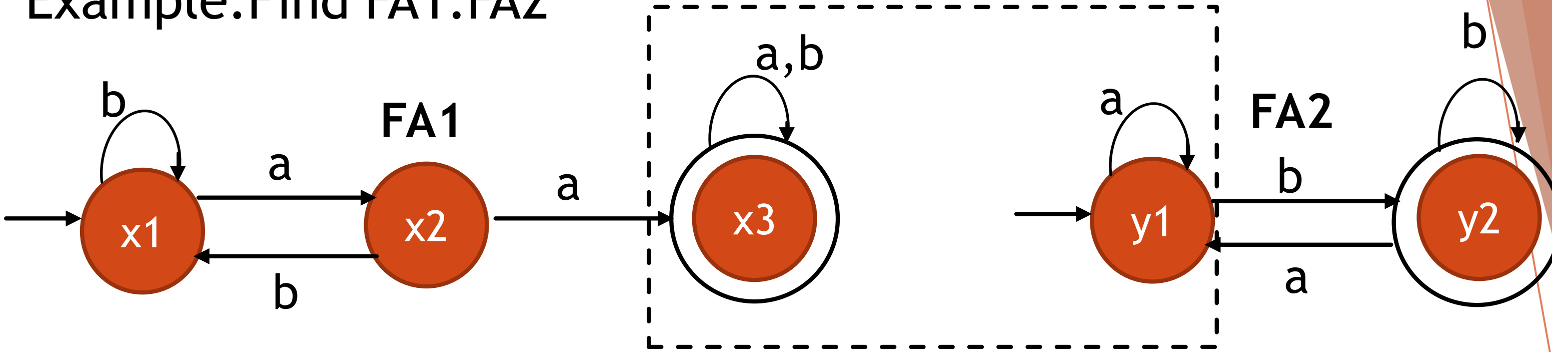
	a	b
-Z1=[x1,y1]	z2=[x2,y1]	Z3=[x1,y2]
Z2=[x2,y1]	Z4=[x3,y1]	Z3=[x1,y2]
+Z3=[x1,y2]	Z2=[x2,y1]	Z3=[x1,y2]
+Z4=[x3,y1]	Z4=[x3,y1]	Z5=[x3,y2]
+Z5=[x3,y2]	Z4=[x3,y1]	Z5=[x3,y2]



FA1+FA2

-If there is an FA called FA1 that accepts the language defined by regular expression r1 and there is an FA called FA2 that accepts the language defined by the regular expression r2, then there is an FA called FA3 that accepts the language defined by regular expression (r1.r2).

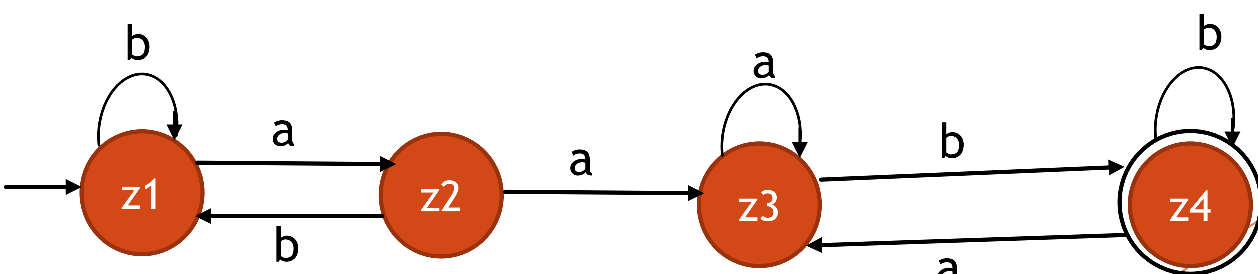
Example:Find FA1.FA2



FA1 =the machine that accepts only strings with a double a in them

FA2 =the machine that accepts all words that end in the letter b

solution	a	b
-Z1=[x1]	z2=[x2]	Z1=[x1]
Z2=[x2]	Z3=[x3,y1]	Z1=[x1]
Z3=[x3,y1]	Z3=[x3,y1]	Z4=[x3,y1,y2]
+Z4=[x3,y1,y2]	Z3=[x3,y1]	Z4=[x3,y1,y2]



FA1.FA2

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 9

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Grammar

The formal definition of Grammar

G=(V,T,P,S) Where

V is a Finite set of Variable(non-terminal)(represented by upper case letters)

T is a Finite set of terminal (represent by lower case letter)

P is a Finite set of production

S is called the start symbol (non-terminal)

Example:

G=({S,A,B},{a,b},P,S} consist of production

$$\left.\begin{array}{l} S \to A \mid B \\ A \to aA \mid a \\ B \to bB \mid b \end{array}\right\} \text{Production}$$

<u>Derivation</u>:- is used to generate(determine)sentence of the given language,(and it is sequence of application the rule to produce the finished string of terminal from S.

$\rightarrow$ ***denotes derivation***

<u>Definition</u> :- The language generated by G [denoted L(G)] is {w | w in T* and S $\overset{*}{\underset{G}{\rightarrow}}$ w} that is, a string is L(G) if:

1) The string consists solely of terminals.
2) The string can be derived from S.

<u>Example</u>: Consider the following grammar G=({S},{a},P,S) where P consist of

$S \rightarrow aS|a \quad \Big] P$

$S \overset{*}{\rightarrow} aaa$

$S \rightarrow a$

$S \rightarrow a\underline{S} \rightarrow aa$

$S \rightarrow a\underline{S} \rightarrow aa\underline{S} \rightarrow aaa$

Example:

G=({S,A,B},{a,b},P,S) consist of production

$$\left.\begin{array}{l} \mathbf{S} \rightarrow \boldsymbol{A} \mid \boldsymbol{B} \\ \mathbf{A} \rightarrow \mathbf{aA} \mid \mathbf{a} \\ \mathbf{B} \rightarrow \mathbf{bB} \mid \mathbf{b} \end{array}\right\} P$$

$S \rightarrow \underline{A} \rightarrow a$

$S \rightarrow \underline{A} \rightarrow a\underline{A} \rightarrow aa$

$S \rightarrow \underline{A} \rightarrow a\underline{A} \rightarrow aa\underline{A} \rightarrow aaa$

$S \rightarrow \underline{A} \rightarrow a\underline{A} \rightarrow aa\underline{A} \rightarrow aaa\underline{A} \rightarrow aaaa$

$S \rightarrow \underline{B} \rightarrow b$

$S \rightarrow \underline{B} \rightarrow b\underline{B} \rightarrow bb$

$S \rightarrow \underline{B} \rightarrow b\underline{B} \rightarrow bb\underline{B} \rightarrow bbb$

$S \rightarrow \underline{B} \rightarrow b\underline{B} \rightarrow bb\underline{B} \rightarrow bbb\underline{B} \rightarrow bbbb$

$\boldsymbol{L(G)=\{a^n \mid n>=1\} \cup \{b^n \mid n>=1\}}$

<u>Leftmost and rightmost derivations:-</u>

If at each step in a derivation a production is applied to the left most variable, then the derivation is said to be leftmost.

Similarity a derivation in which rightmost variable is replaced at each step is said to be rightmost.

<u>Example:-</u>

Consider the grammar $G=(\{S,A\},\{a,b\},P,S)$, where P consist of

$S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid SS \mid ba$

the word aabbaa

Leftmost derivation

$S \rightarrow a\underline{A}S \rightarrow a\underline{S}bAS \rightarrow aab\underline{A}S \rightarrow aabba\underline{S} \rightarrow aabbaa$

Rightmost derivation

$S \rightarrow aA\underline{S} \rightarrow a\underline{A}a \rightarrow aSb\underline{A}a \rightarrow a\underline{S}bbaa \rightarrow aabbaa$

Derivation Trees:-

If is useful to display derivations as trees. These pictures, called derivation(or generation or production or syntax)trees.

Leaf: a vertex which has no sons, usually represent a terminal.

Interior vertex: a vertex which has one or more sons usually $\in V$.

Yield of the derivation tree: if we read the label of the leaves from left to right, we have a sentential form, we call this string the yield.

Example:-

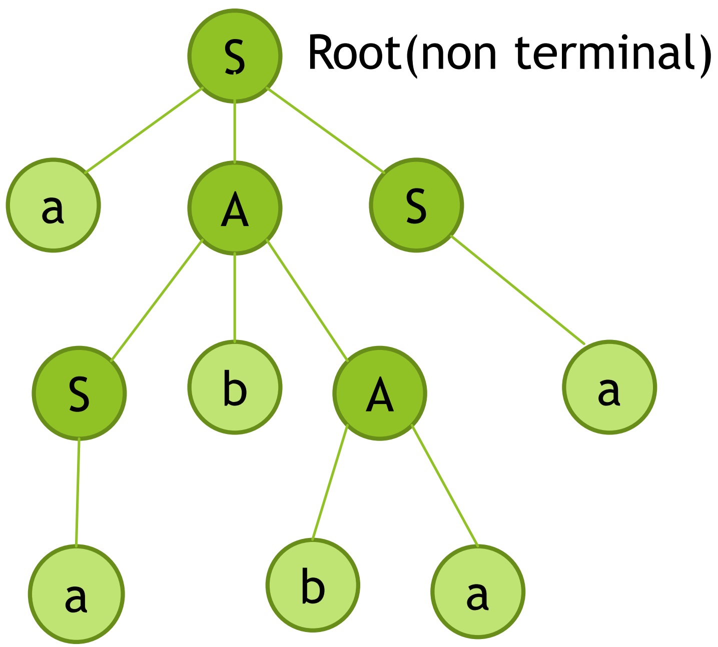
Consider the grammar G=({S,A},{a,b},P,S}, where P consist of

$S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid SS \mid ba$

Draw the derivation tree of the string(aabbaa)

$S \rightarrow a\underline{A}S \rightarrow a\underline{S}bAS \rightarrow aab\underline{A}S \rightarrow aabba\underline{S} \rightarrow aabbaa$



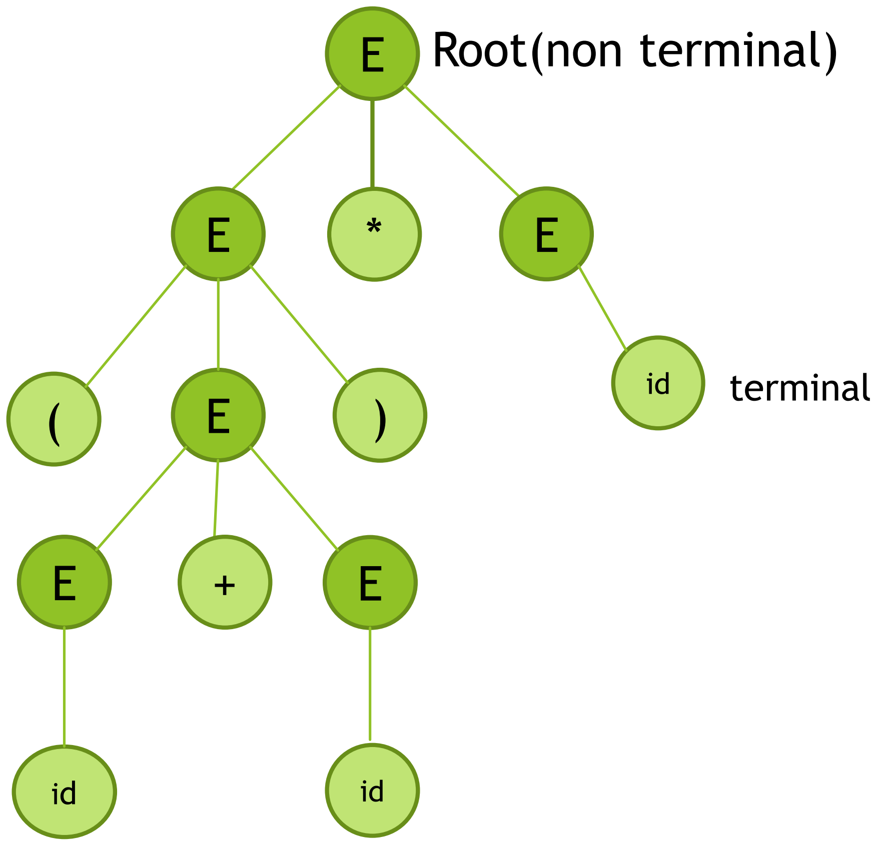
Example:-

Consider the grammar G=({E},{id,(,),*,+},P,E}, where P consist of

E → E+E | E*E |(E) |id

Draw the derivation tree of the string((id+id)*id)

E →<u>E</u>*E →(<u>E</u>)*E →(<u>E</u>+E)*E →(id+<u>E</u>)*E →(id+id)*<u>E</u> →(id+id)*id)



Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory

2^{nd} Class/1^{st} Sem

Lecture 10

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Type of Grammar

A phrase Structure Grammar(PSG)

Is a 4 tuple (V,T,P,S) Where

V : Finite set of non-terminals.

T :is a Finite set of terminals such that $V \cap T = \emptyset$.

P :is a Finite set of production of the form $\alpha \to \beta$,where α the string on the left hand side of the production, is such that $\alpha \in (V \cup T)^{+}$ and β the string on the right hand side of the production, is such $\beta \in (V \cup T)^{*}$.

S $\in$ V is symbol designed as the start symbol of the grammar.

Example1:

Consider the PSG, $G1=(\{S,A,B\},\{a,b\},P,S)$ with production

$S\rightarrow A \qquad S\rightarrow B$

$A \rightarrow aA \qquad A \rightarrow a$

$B \rightarrow bB \qquad B \rightarrow b$

The above productions can be abbreviated

$$\left.\begin{array}{l} S\rightarrow A|B \\ A \rightarrow aA|a \\ B \rightarrow bB|b \end{array}\right\} p$$

$$\boxed{L(G1)=\{a^n \mid n>=1\} \cup \{b^n \mid n>=1\}}$$

Example2:

Consider the PSG,G2 with production

$$\left.\begin{array}{l} S \to aSBC \mid aBC \\ CB \to BC \\ aB \to ab \\ bB \to bb \\ bC \to bc \\ cC \to cc \end{array}\right\} p$$

In this example the left side of the production are not all single non terminal.

$S \xrightarrow{*} aabbcc$

$S \to \underline{aB}C \to a\underline{bC} \to \boxed{abc}$

$S \to a\underline{S}BC \to a\underline{aB}CBC \to aab\underline{CB}C \to aa\underline{bB}CC \to aab\underline{bC}C \to aabb\underline{cC} \to \boxed{aabbcc}$

$L(G2)=\{a^n b^n c^n \mid n \geq 1\}$

Some time ,it ma be that two different grammar G and G` generate the same language L(G)=L(G `).

In this case the grammars are said to be equivalent .

An example of a grammar equivalent to G1 is G3 with productions

$$\left.\begin{array}{l} S \rightarrow aA|bB|a|b \\ A \rightarrow aA|a \\ B \rightarrow bB|b \end{array}\right\} P$$

PSG is also known as **unrestricted grammar.**

Context Sensitive Grammar(CSG)

Suppose a restriction is placed on productions
$\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ $|\boldsymbol{\alpha}| \leq |\boldsymbol{\beta}|$ (that β *be at least as long as* α).
Then the resulting grammar is called Context-Sensitive grammar(**CSG**)and the language a Context Sensitive language(**CSL**).

The term “Context-Sensitive” comes from a normal form for these grammar, where each production is of the form $\alpha 1 A \alpha 2 \rightarrow \alpha 1 \mathrm{B} \alpha 2$, with $\mathrm{B} \neq \epsilon$, they permit replacement of variable A by string B only in the "context"$\alpha 1 \alpha 2$.

Example:

Consider the CSG,G=({S,B,C},{x,y,z},P,S} with production

$$\left.\begin{array}{l} \mathbf{S \rightarrow Byz} \\ \mathbf{B \rightarrow x \mid xBc} \\ \mathbf{cy \rightarrow yc} \\ \mathbf{cz \rightarrow yzz} \end{array}\right\} \text{p}$$

$S \xrightarrow{*} xxxyyyzzz$

$S \rightarrow \underline{B}yz \rightarrow xyz$

$S \rightarrow \underline{B}yz \rightarrow x\underline{B}cyz \rightarrow xx\underline{B}ccyz \rightarrow xxxc\underline{cy}z \rightarrow xxx\underline{cy}cz \rightarrow xxxyc\underline{cz} \rightarrow xxxy\underline{cy}zz \rightarrow xxxyy\underline{cz}z \rightarrow xxxyyyzzz$

$L(G)=\{x^n y^n z^n \mid n \geq 1\}$

Context Free Grammar(CFG)

A limiting to the left-hand sides of each production $\alpha \rightarrow \beta$ in a CSG to be a single nonterminal A → B where A ∈ V and $\beta \in (V \cup T)^*$.

Example1: Consider a grammar G=({S},{a,b},P,S)and P is the following set

$\left.\textbf{S} \rightarrow \textbf{aSb|ab}\right\}$ p

S →ab

S → a$\underline{\text{S}}$b →aabb

S → a$\underline{\text{S}}$b →aa$\underline{\text{S}}$bb →aaabbb

S → a$\underline{\text{S}}$b →aa$\underline{\text{S}}$bb →aaa$\underline{\text{S}}$bbb →aaaabbbb

L(G)={$a^n b^n$ |n>=1}

Example2: Consider a grammar $G=(\{S\},\{a,b\},P,S)$ and P is the following set

S → aSb | ϵ } P

$S \rightarrow \epsilon$

$S \rightarrow a\underline{S}b \rightarrow ab$

$S \rightarrow a\underline{S}b \rightarrow aa\underline{S}bb \rightarrow aabb$

$S \rightarrow a\underline{S}b \rightarrow aa\underline{S}bb \rightarrow aaa\underline{S}bbb \rightarrow aaabbb$

$L(G)=\{a^n b^n \mid n>=0\}$

Example3: Consider a grammar $G=(\{S\},\{a,b\},P,S)$ and P is the following set

$$\left.\begin{array}{l} \mathbf{S \rightarrow aB \mid bA} \\ \mathbf{A \rightarrow aS \mid a \mid bAA} \\ \mathbf{B \rightarrow bS \mid b \mid aBB} \end{array}\right\} \mathbf{P}$$

$S \xrightarrow{*} abba$

$S \rightarrow a\underline{B} \rightarrow ab\underline{S} \rightarrow abb\underline{A} \rightarrow abba$

$S \xrightarrow{*} bababa$

$S \rightarrow b\underline{A} \rightarrow ba\underline{S} \rightarrow bab\underline{A} \rightarrow baba\underline{S} \rightarrow babab\underline{A} \rightarrow bababa$

$S \xrightarrow{*} baabba$

$S \rightarrow b\underline{A} \rightarrow ba\underline{S} \rightarrow baa\underline{B} \rightarrow baab\underline{S} \rightarrow baabb\underline{A} \rightarrow baabba$

The language L(G) is the set of all words in T^+ consisting of equal number of a’s and b’s.

Example 4: Consider a grammar

G=({S,A},{a,b,c,d},P,S)and P is the following set

$$\left.\begin{array}{l}\mathbf{S \rightarrow aSd|aAd} \\ \mathbf{A \rightarrow bAc|bc}\end{array}\right\} p$$

$S \rightarrow a\underline{S}d \rightarrow aa\underline{A}dd \rightarrow \boxed{aabcdd}$

$S \rightarrow a\underline{S}d \rightarrow aa\underline{A}dd \rightarrow aab\underline{A}caa \rightarrow \boxed{aabbccdd}$

$S \rightarrow a\underline{S}d \rightarrow aa\underline{S}dd \rightarrow aaa\underline{S}ddd \rightarrow aaaa\underline{A}dddd \rightarrow \boxed{aaaabcdddd}$

$S \rightarrow a\underline{S}d \rightarrow aa\underline{S}dd \rightarrow aaa\underline{S}ddd \rightarrow aaaa\underline{A}dddd \rightarrow aaaab\underline{A}cdddd$
$\rightarrow \boxed{aaaabbccdddd}$

The language L(G) $=\{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

❖ Palindrome

$S \to aSa \mid bSb \mid b \mid a \mid \epsilon$ p

$S \to b$

$S \to a$

$S \to \epsilon$

$S \to a\underline{S}a \to aba$

$S \to a\underline{S}a \to aaa$

$S \to a\underline{S}a \to a\epsilon a \to aa$

$S \to a\underline{S}a \to ab\underline{S}ba \to abbba$

$S \to a\underline{S}a \to ab\underline{S}ba \to ababa$

$S \to a\underline{S}a \to ab\underline{S}ba \to ab\epsilon ba \to abba$

$S \to b\underline{S}b \to bbb$

$S \to b\underline{S}b \to bab$

$S \to b\underline{S}b \to b\epsilon b \to bb$

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 11

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Properties of Grammar

Properties of grammar

1-If G1 and G2 are CFG then the union (CFG1+CFG2)is also CFG.

2-If G1 and G2 are CFG then their concatenation (CFG1.CFG2) is also CFG.

3- If G1 is a CFG then the closure of CFG1$(CFG1)^*$ is also CFG.

Properties of Context Free Language(CFL)

1-If L1 and L2 are CFL then the union (CFL1+CFL2)is also a CFL.

2-If L1 and L2 are CFL then their concatenation (CFL1.CFL2) is also CFL .

3- If L1 is a CFL then the closure of CFL1 $(CFL1)^*$ is also CFL.

Example

Let L1={$a^n b^{2n}$ |n>=1}be a CFL for the following CFG1

S→ $\boldsymbol{aSbb|abb}$

And

L2=palindrome on {a,b} be a CFL with the following CFG2

S → aSa|bSb|a|b|$\boldsymbol{\epsilon}$

Then L1+L2 (CFG1+CFG2)

S → S1|S2

S1 → $aS1bb|abb$

S2 → aS2a|bS2b|a|b|ϵ

And L1.L2 (CFG1.CFG2)

S → S1.S2

S1 → $aS1bb|abb$

S2 → aS2a|bS2b|a|b|ϵ

and L1* (CFG1)*

S → S1S| ϵ

S1→ $aS1bb|abb$

Ambiguity:

a CFG is said to be **ambiguous grammar** if there exist some word w with two parse tree or equivalently has more than one leftmost or rightmost derivation for a particular word w.

A CFL for which every CFG is ambiguous is said to be inherently ambiguous CFL.

Example

(Non ambiguous CFL)

$S \to aAS \mid a$

$A \to SbA \mid SS \mid ba$

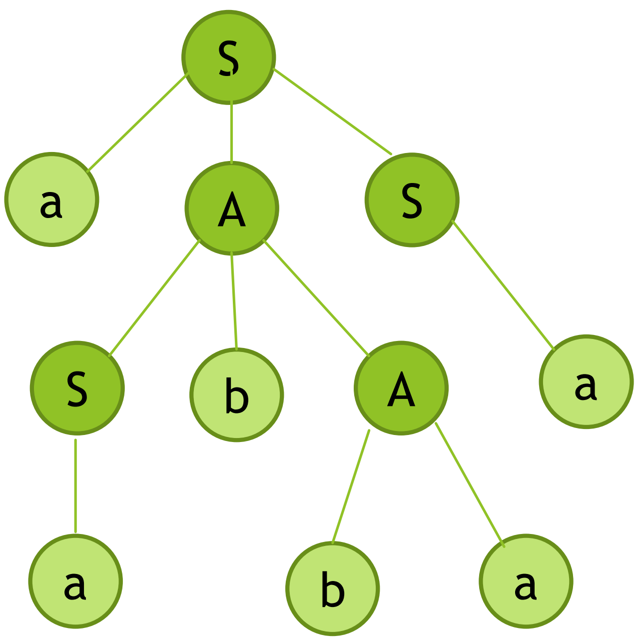
Find $S \xrightarrow{*} aabbaa$

Leftmost derivation

$S \to a\underline{A}S \to a\underline{S}bAS \to aab\underline{A}S \to aabba\underline{S} \to aabbaa$

Rightmost derivation

$S \to aA\underline{S} \to a\underline{A}a \to aSb\underline{A}a \to a\underline{S}bbaa \to aabbaa$



<u>Example</u>(ambiguous CFL)

$S \rightarrow SbS \mid ScS \mid a$

Find $S \overset{*}{\rightarrow} abaca$

Leftmost derivation

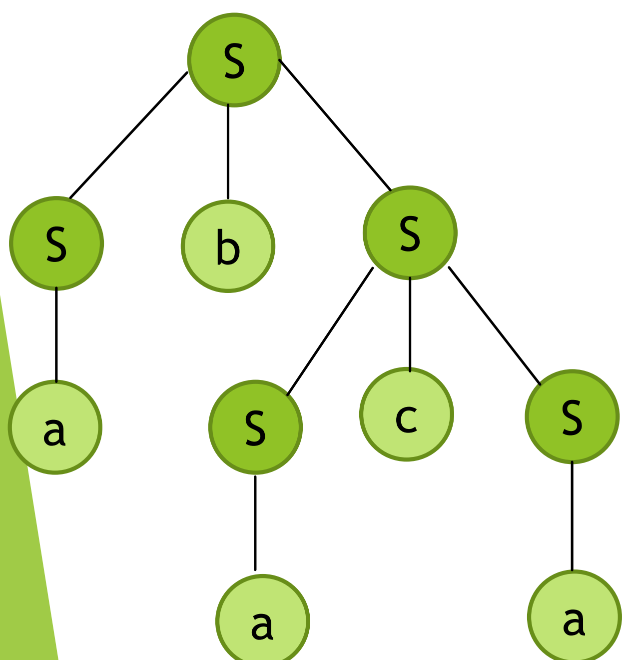
$S \rightarrow \underline{S}bS \rightarrow ab\underline{S} \rightarrow ab\underline{S}cS \rightarrow abac\underline{S} \rightarrow abaca$

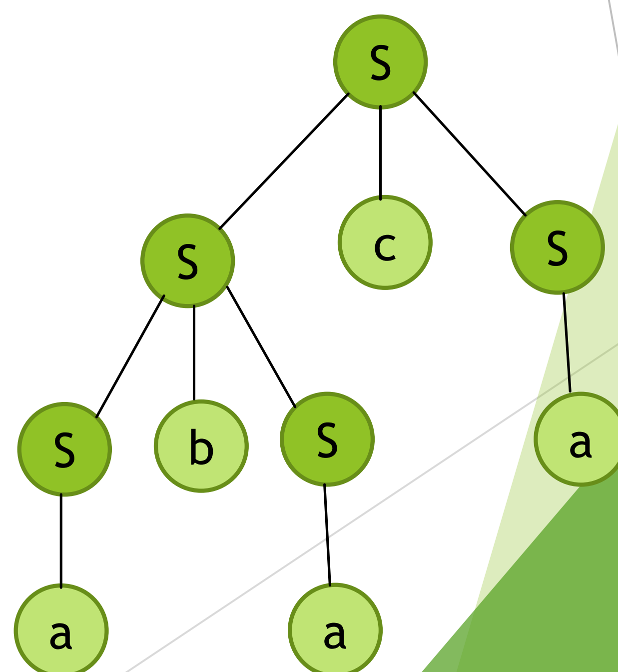
$S \rightarrow \underline{S}cS \rightarrow \underline{S}bScS \rightarrow ab\underline{S}cS \rightarrow abac\underline{S} \rightarrow abaca$

Rightmost derivation

$S \rightarrow Sb\underline{S} \rightarrow SbSc\underline{S} \rightarrow Sb\underline{S}ca \rightarrow \underline{S}baca \rightarrow abaca$

$S \rightarrow Sc\underline{S} \rightarrow \underline{S}ca \rightarrow Sb\underline{S}ca \rightarrow \underline{S}baca \rightarrow abaca$





Simplification of Context Free Grammar

If L is a nonempty CFL then it can be generated by a CFG G with the following properties

1-Each variable and terminal of G appears in the derivation of some word in L.

2-There are no production of the form $A \rightarrow B$ where A and B are variables.

3-If ϵ is not in L, there need no production of the form $A \rightarrow \epsilon$.

Useless Symbols:-

Let G=(V,T,P,S) be a grammar. A symbol X is useful if there is a derivation S $\xrightarrow{*}$ $\alpha X \beta$ $\xrightarrow{*}$w for some α, β And w, where w $\in$ T^{*} *otherwise* X is useless.

There are two aspect to usefulness.

First some terminal string must be derivable from X.

Second X, must occur in some string derived from S.

Useless Symbols:-

Lemma1:-Given a CFG G=(V,T,P,S) with L(G)$\neq \emptyset$ we can fined an equivalent CFG $\tilde{G}$ ($\tilde{V}$,T,$\tilde{P}$,S) such that for each A in V there is some w $\in T^*$ for which A $\xrightarrow{*}$ w.

Lemma2:-Given a CFG G=(V,T,P,S)we can find an equivalent CFG $\tilde{G}$=($\tilde{V}$,$\tilde{T}$,$\tilde{P}$,S) such that for each X in ($\tilde{V}\cup\tilde{T}$) there exist $\alpha\ and\ \beta$ in $(\tilde{V}\cup\tilde{T})^*$ For which S $\xrightarrow{*}$ $\alpha\ X\ \beta$.

Note:- you first have to applying lemma1 then Lemma2.

Example:-

$G=(\{S,A,B\},\{a\},P,S)$

$$\left.\begin{aligned} &\mathbf{S \rightarrow AB \mid a} \\ &\mathbf{A \rightarrow a} \end{aligned}\right\} P$$

By apply Lemma1

$$\left.\begin{aligned} &S \rightarrow a \\ &A \rightarrow a \end{aligned}\right\} P1$$

$G1=(\{S,A\},\{a\},P1,S)$

By apply Lemma2

$G2=(\{S\},\{a\},P2,S)$

$$S \rightarrow a \}\ P2$$

Example2: G=({S,X,C,A},{a,b},P,S)

$$\left.\begin{array}{l} \mathbf{S} \rightarrow \mathbf{AX} \mid \boldsymbol{BA} \\ \mathbf{X} \rightarrow \boldsymbol{XB} \mid \boldsymbol{AX} \\ \mathbf{B} \rightarrow \mathbf{aXe} \mid \mathbf{b} \\ \mathbf{C} \rightarrow \mathbf{a} \mid \mathbf{abx} \end{array}\right\} P$$

$\mathbf{A} \rightarrow \mathbf{a}$

By applying Lemma1 G1=({S,B,C,A},{a,b},P1,S)

$$\left.\begin{array}{l} S \rightarrow BA \\ B \rightarrow b \\ C \rightarrow a \\ A \rightarrow a \end{array}\right\} P1$$

By applying Lemma2 G2=({S,A,B},{a,b},P2,S)

$$\left.\begin{array}{l} S \rightarrow BA \\ B \rightarrow b \\ A \rightarrow a \end{array}\right\} P2$$

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

VIEDEO LECTURES

Computation Theory
2nd Class/1st Sem
Lecture 12

ا.م وجدان عبد الامير حسن

Computation Theory

Simplification of Context Free Grammar

Simplification of Context Free Grammar

If L is a nonempty CFL then it can be generated by a CFG G with the following properties

1-Each variable and terminal of G appears in the derivation of some word in L.

2-There are no production of the form $A \rightarrow B$ where A and B are variables.

3-If ϵ is not in L, there need no production of the form $A \rightarrow \epsilon$.

<u>Useless Symbols:-</u>

Let G=(V,T,P,S) be a grammar. A symbol X is useful if there is a derivation S $\xrightarrow{*} \alpha X \beta \xrightarrow{*}$ w for some α, β And w, where w $\in T^{*}$ $otherwise$ X is useless.

There are two aspect to usefulness.

<u>**First**</u> some terminal string must be derivable from X.

<u>**Second**</u> X, must occur in some string derived from S.

Useless Symbols:-

Lemma1:-Given a CFG G=(V,T,P,S) with L(G)$\neq \emptyset$ we can fined an equivalent CFG G′ (V′,T,P′,S) such that for each A in V there is some w $\in T^*$ for which A $\xrightarrow{*}$ w.

Lemma2:-Given a CFG G=(V,T,P,S)we can find an equivalent CFG G′=(V′,T′,P′,S) such that for each X in (V′ ∪T′) there exist α *and* β in (V′ ∪T′) For which S $\xrightarrow{*} \alpha\, X\, \beta$.

Note:- you first have to applying lemma1 then Lemma2.

Example1:-

G=({S,A,B},{a},P,S)

$$\left.\begin{aligned} &\mathbf{S \rightarrow AB|a} \\ &\mathbf{A \rightarrow a} \end{aligned}\right\} P$$

By apply Lemma1

$$\left.\begin{aligned} &S \rightarrow a \\ &A \rightarrow a \end{aligned}\right\} P1$$

G1=({S,A},{a},P1,S)

By apply Lemma2

G2=({S},{a},P2,S)

$$S \rightarrow a \;\} P2$$

Example2: G=({S,X,C,A},{a,b},P,S)

$$\left.\begin{array}{l} \mathbf{S} \rightarrow \mathbf{AX} \mid \boldsymbol{BA} \\ \mathbf{X} \rightarrow \boldsymbol{XB} \mid \boldsymbol{AX} \\ \mathbf{B} \rightarrow \mathbf{aXe} \mid \mathbf{b} \\ \mathbf{C} \rightarrow \mathbf{a} \mid \mathbf{abx} \end{array}\right\} P$$

$\mathbf{A} \rightarrow \mathbf{a}$

By applying Lemma1 G1=({S,B,C,A},{a,b},P1,S)

$$\left.\begin{array}{l} S \rightarrow BA \\ B \rightarrow b \\ C \rightarrow a \\ A \rightarrow a \end{array}\right\} P1$$

By applying Lemma2 G2=({S,A,B},{a,b},P2,S)

$$\left.\begin{array}{l} S \rightarrow BA \\ B \rightarrow b \\ A \rightarrow a \end{array}\right\} P2$$

$\in -$**production:-**

If L=L(G) for some CFG, G=(V,T,P,S) then L- $\{\in\}$ is L(G´)for a CFG G´ with no useless symbols or $\in$ productions.

We can determine the **nullable symbols** of G by the following:-

If $A \rightarrow \in$ is a production then A is nullable symbols. If $\beta \rightarrow \alpha$ is a production and all symbols of α have been found nullable then β is nullable.

Example1: for the following grammar remove $\in -production$

Let G have productions

S $\rightarrow (\boldsymbol{E})|\boldsymbol{E}$

E →T|E+T|E-T

T → F|T*F|T/F

F →a|b|c|$\in$

Nullable=(F,T,E,S)

S →(E)|E|()

E →T|E+T|E-T|E+|E-|+T|-T|+|-

T →F|T*F|T/F|T*|T/|*F|/F|*|/

F →a|b|c

<u>Example2</u>: for the following grammar remove $\in -production$

A $\rightarrow$ $\boldsymbol{E|CE}$ **| BC**

B $\rightarrow$**CD**

C $\rightarrow$ **c |** $\in$

D $\rightarrow$**d |** $\in$

E $\rightarrow$**eE | e**

Nullable=(A,B,C,D)

A $\rightarrow$E|CE|BC|B|C

B $\rightarrow$CD|C|D

C $\rightarrow$ c

D $\rightarrow$ d

E $\rightarrow$ eE|e

Unit production:-

If a grammar have production of the form **$A \rightarrow B$** whose right hand side consist of a single variable we call these production by **unit production.**

All other production of the form $A \rightarrow a$ and $\in$ -productions are non unit productions.

For CFG defined by G=(V,T,P,S) where G has no $\in$ -productions construct a new set of production P' from P by first including all non unit production of P.

Then $A \xrightarrow{*} B$ for A and B in V add to P' all productions of the form $A \rightarrow \alpha$ is a non unit production.

Example1: for the following grammar remove unit production

$A \rightarrow E|CE|B|C|BC$

$B \rightarrow C|D|CD$

$C \rightarrow c$

$D \rightarrow d$

$E \rightarrow Ee|e$

Solution

$A \rightarrow Ee|e|CE|c|d|CD|BC$

$B \rightarrow c|d|CD$

$C \rightarrow c$

$D \rightarrow d$

$E \rightarrow Ee|e$

Example2: for the following grammar remove unit production

$S \rightarrow A|ABA$

$A \rightarrow aA|a|B$

$B \rightarrow bB|b$

Solution

$S \rightarrow aA|a|bB|b|ABA$

$A \rightarrow aA|a|bB|b$

$B \rightarrow bB|b$

H.W:-Remove unit production from the following grammar

$S \rightarrow D$

$D \rightarrow AB$

$A \rightarrow c$

$C \rightarrow ac|a$

$B \rightarrow M$

$M \rightarrow bM|b$

Thanks for lessening

قسم علوم الحاسوب
Computer Dep.

التعليم الالكتروني
E-Learning

جامعة بغداد
University of Baghdad

كلية العلوم
College of Science

Computation Theory
2nd Class/1st Sem
Lecture 14

ا.م وجدان عبد الامير حسن

VIEDEO LECTURES

Computation Theory

Canonical Form of Context Free grammar

Canonical form of Context Free Grammar

In the Context Free Grammar(CFG) there are two canonical form

1-Chomsky Normal Form(**CNF**).

2-Greibach Normal Form(**GNF**).

This formal form prove that all CFG are equivalent to grammar with restrictions on the forms of productions.

Greibach Normal Form(GNF):-

Every Context Free Language without $\in$ can be generated by grammar for every productions is of the form **A** $\rightarrow$ $\mathbf{a}\boldsymbol{\alpha}$ where **A** is a variable and $\boldsymbol{\alpha}$ is a(possibly empty)string of variables.

Lemma1:

Define an A-production to be a production with variable A on the left. Let G=(V,T,P,S)be a CFG.

Let $A \rightarrow \alpha 1 B \alpha 2$ *be* a production in p and

$B \rightarrow B1|B2|B3|..|B_r$ be the set of all B-productions.

Let G1=(V,T,P1,S)be obtained from G by deleting the production $A \rightarrow \alpha 1 B \alpha 2$ and adding the production

$A \rightarrow \alpha 1 B1 \alpha 2 | \alpha 1 B2 \alpha 2 | \alpha 1 B3 \alpha 2 | .. \alpha 1 B_r \alpha 2$ then

L(G)=L(G1).

Greibach Normal Form(GNF):-

Lemma2:

Let G=(V,T,P,S) be a CFG. Let A $\rightarrow A\alpha 1|A\alpha 2|..A\alpha r$ be the set of A-productions. For which A is the leftmost symbol of the right-hand side. Let A → B1|B2|..Bs be the remaining A-production.

Let G1=(V∪{B},T,P1,S)be the CFG formed by adding the variable B to V and replacing all the A-productions by the productions

1) $\left.\begin{array}{l} A \rightarrow Bi \\ A \rightarrow BiB \end{array}\right]$ 1<=i<=s 2) $\left.\begin{array}{l} B \rightarrow \alpha i \\ B \rightarrow \alpha i B \end{array}\right]$ <=i<=r

Then L(G1)=L(G)

<u>Example</u>: Convert to Greibach normal form the grammar

G=({A1,A2,A3},{a,b},P,A1)

Where P consist of following $\mathbf{A \rightarrow a\alpha}$

A1 → A2A3

A2 → A3A1|b

A3 → A1A2|a

<u>Solution</u>

A3 →A2A3A2|a

A3 →A3<u>A1A3A2</u>|bA3A2|a B3 →A1A3A2|A1A3A2B3

A3 →bA3A2B3|aB3|bA3A2|a

B3 →A1A3A2|A1A3A2B3

A2 →bA3A2B3A1|aB3A1|bA3A2A1|aA1|b

A1 →bA3A2B3A1A3|aB3A1A3|bA3A2A1A3|aA1A3|bA3

B3 →A1A3A2|A1A3A2B3

B3 → bA3A2B3A1A3A3A2|aB3A1A3A3A2|bA3A2A1A3A3A2|aA1A3|bA3A3A2|
bA3A2B3A1A3A3A2B3|aB3A1A3A3A2B3|bA3A2A1A3A3A2B3|aA1A3A3A2B3|b
A3A3A2B3

Thanks for lessening