Regular Expression (RE)

أ.م.د. عبير طارق مولود

أ.م. علاء نوري

النظرية الاحتسابية/الكورس الاول/كل الافرع

قسم علوم الحاسوب/الجامعة التكنولوجية

2020-2021

Definition:-

The language-defining symbols we are about to create are called regular expressions. The languages that are associated with these regular expressions are called regular languages.

Set notation :

Set is simply a collection of objects without repetition

- Each object in a set is called an element of that set .
- If the number of elements in the set is not too large than the set can be specified by listing its elements .

Ex: A={ 1,2,3,4,…………..}

A={X : X is integers , X > 0 }

Note : If an element X is a member of a set ,then X ϵ A and if X is not an element of A , X∄A

- **A set A is a subset of a set B, A C B if every element of A is a member of the set B .**

Example:

{ 1,2,4,} C { 1,2,3,4,5}

Note : A C U

Ø C A

Note : A=B ACB and BCS

{ 3,5,7,6} ={5,3,6,7}

Basic operations sets are :

1. Unary operation , Complement (´)
2. Binary operation , union (U)
3. Intersection (∩) and difference (/)

EX// : U = { 0,1,2,3,4,5,6,7,8,9} where, A={0,1,3,5} , B = {2,3,5}

À ={ 2,4,6,7,8,9} AUB= {0,1,2,3,5}

A∩B = {3,5} A/B = {0,1} B/A ={2}

EX//: If L1= {01,0} and L2={ ∈,0,01}

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

- Length of sting numbers of symbols in The sequence

EX// :

S = 1100111 |S| = |1100111| = 7

Examples (RE):-

Example1 consider the language L

where L={Λ ,x ,xx, xxx, …} by using star notation we may
write L=language(x*).

Since x* is any string of x's (including Λ).

Example2 if we have the alphabet ∑={a,b}

And L={a ,ab, abb, abbb, abbbb, …}

Then L=language(ab*)

Example3 (ab)*= Λ or ab or abab or ababab or abababab or ….

Example4 L1=language(xx*)

The language L1 can be defined by any of the expressions:

xx* or x^+ or xx*x* or x*xx* or x^+x* or x*x^+ or x*x*x*xx* …

Remember x* can always be Λ.

Example5 language(ab*a)={aa, aba, abba ,abbba, abbbba, …}

Example6 language(a*b*)={ Λ, a b, aa bb, aaa bbb, aaaa bbb b… } ba and aba are not in this language so a*b* ≠ (ab)*

Example7 the following expressions both define the language

L2={x^{odd}}: x(xx)* or (xx)*x

L=[X,XXX,XXXXX,XXXXXXX,…

But the expression x*xx* does not since it includes the word (xx)x(x).

Example8 consider the language T defined over the alphabet ∑={a,b,c}

T={a ,c ,ab, cb, abb, cbb ,abbb, cbbb, abbbb, cbbbb, …}

Then T=language((a+c)b*)

T=language(either a or c then some b's)

Example9 consider a finite language L that contains all the strings of a's and b's of length exactly three.

L={aaa ,aab ,aba, abb, baa ,bab, bba, bbb}

L=language((a+b)(a+b)(a+b))

L=language($(a+b)^3$)

Note from the alphabet ∑={a,b} , if we want to refer to the set of all possible strings of a's and b's of any length (including Λ) We could write

(a+b)*

Example10 Another expressions that denote all the words with at least two a's are:

RE= (a+b)*a(a+b)*a(a+b)*

L=[aa,aaaaa,aabab,babab…]

= (some beginning)(the first important a)(some middle)(the second important a)(some end)

Note: we say that two regular expressions are equivalent if they describe the same language.

Example11 if we want all the words with exactly two a's, we could use the expression: L= (a+b)* a (a+b)* a (a+b)*

*b*ab*ab** L =[aa,babab,bbabbabb

which describe such words as aab, baba, bbbabbabbbb,…

Example12 the language of all words that have at least one a and at least one b is: *(a+b)*a(a+b)*b(a+b)*+(a+b)*b(a+b)*a(a+b)**

Note: usually when we employ the star operation we are defining an infinite language. We can represent a finite language by using the plus alone.

* تعني التكرار من 0 الى مالا نهاية

+ تعني التكرار من 1 الى مالانهاية

Exmple13 L={X}

L^*={ ∈, X, XX ,XXX ,XXXX ,…….}

= {x^n,n= 0,1,2,3,…………..}

L^+={ X, XX ,XXX ,XXXX ,…….}

= {x^n,n= 1,2,3,…………..}

Example14 **L1=language(xx*)**

The language L1 can be defined by any of the expressions:

xx* or x^+ or xx*x* or x*xx* or x^+x* or x*x^+ or x*x*x*xx* … Remember

x* can always be Λ.

Example15 L={ Λ ,a ,aa, bbb}

L=language(Λ + a + aa + bbb)

Example16 L={Λ ,a ,b ,ab ,bb, abb, bbb ,abbb, bbbb, …} We can define L by using the expression $b^* + ab^*$

Note: $L^* = L^{**}$

Example17 $(ab)^*$ = { ^ ,ab,abab,ababab , ……….}

Example18 $(a^* b^*)$ ={ ^ ,ab,aabb,aaabbb,………..}

Note (ab)* ≠ (a* b*)

Example19 The language defined by the expression ab^*a

The set of all strings of a's and b's that have at least two letters , that begin and end with a's and that have nothing but b's inside L={aa, aba, abba, abbba, abbbba,………}

Example20 L = {a,b,c}

T= {a,c,ab,cb,abb,cbb,abbb,cbbb,abbbb,cbbbb, ……..}

T=language ((a+c)b^*)

Example 21 L= { aaa,aab,aba,abb,baa,bab,bba ,bbb}

-The first letter of each word in L is either a or b

-The second letter of each word in L is either a or b

-The third letter of each word in L is either a or b

Then the L= Language (a+b)(a+b)(a+b))

= Language ((a+$b)^3$

Exmple22 All words that begin with an a and end with b

$a(a+b)^*b$

L={ab,aab,abb,aaab,abbb,…..}

Example23 All words over the alphabet L = {a,b} that have an <u>a</u> in somewhere

(a+b)*a (a+b)* =L{a,aaa,aab,baa,…}

Example24 All words with letters a and b that begin with letter a

a(a+b)*

L=[a,aa,ab,aaa,abb,

Example25 All the words with exactly two a's

$b^*ab^*\ ab^*$ L={ aa,babab,bbabbabb,…..}

Example26 The language of all words that have at least two a's

(a+b)*a(a+b)*a(a+b)* L = { aa,aaaaa, … }

Example27 The language of all words over the alphabet L={a,b} that have at least one a and at least b:

$$(a+b)^{*}\text{a }(a+b)^{*}\text{b }(a+b)^{*}$$

Example28 All words over the alphabet L ={a,b} with at least two consecutive a's

(a+b)*aa (a+b)*

Example29 All string of a's and b's beginning with b and not having two consecutive a's :

$$(b+ba)^{*}$$

Example30 All string of a's and b's ending in abb :

(a+b)*abb

Example31 The set of strings of a's and b's that of some point contain a double letter :

$$(a+b)^{*}(\text{aa+bb })(a+b)^{*}$$

Example32 L= { All strings that contain exactly three b's

$a^* ba^* ba^* ba^*$ L={ bbb,...}

Note	Language	RE
	{ }	∅
	{2}	2
	{a}	a
	{ a,b}	{ a+b}
	{ aa,bb,01}	{ aa+bb+01}
	{∧,a, aa,aaa,…}	a^*
	{ a,aa,aaa,...}	a^+

Note $(0+1)^*$ = { ∧ ,0,1,00,11,01,10,000,111,001,010,…}

$(1+10)^*$ ={∧,1,10,11,110,1010,101,1110,111,1101,…..}

Note:

$(a+b^*)^*=(a+b)^*$

$(a^*)^*=a^*$

$(a^*b^*)^*=(a+b)^*$

Example33

$E=[aa+bb+(ab+ba)(aa+bb)^*(ab+ba)]^*$

Even-even={Λ, aa, bb, aabb ,abab, abba, baab baba bbaa aaaabb aaabab………..}

Thank you for Listening

Finite Automata (FA)

أ.م.د. عبير طارق مولود

أ.م. علاء نوري

النظرية الاحتسابية/الكورس الاول/كل الافرع

قسم علوم الحاسوب/الجامعة التكنولوجية

2020-2021

Definition:-

A finite automata is a collection of five things:

1. A finite set of states
2. An alphabet ∑ of possible input letters from which are formed strings that are to be read one letter at a time.
3. A finite set of transitions that tell for each state and for each letter of the input alphabet which state to go to next.
4. The initial state;
5. The set of final states.

Therefore formally a finite automata is a five-tuple:

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

where:

Q is a set of states of the finite automata,

Σ is a set of input symbols, and

δ specifies the transitions in the automata.

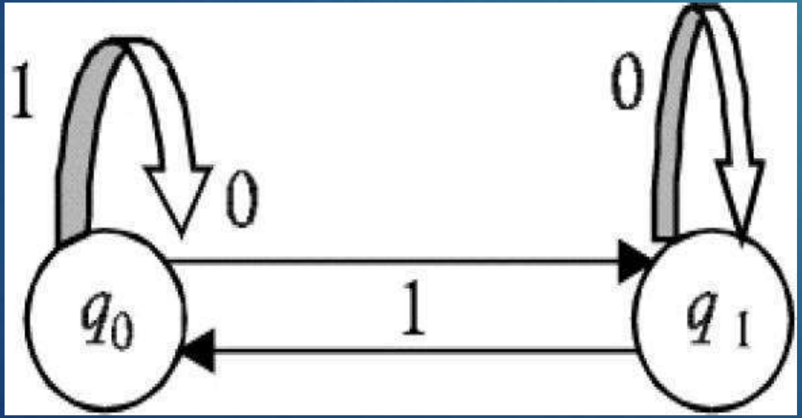
Example:

$$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

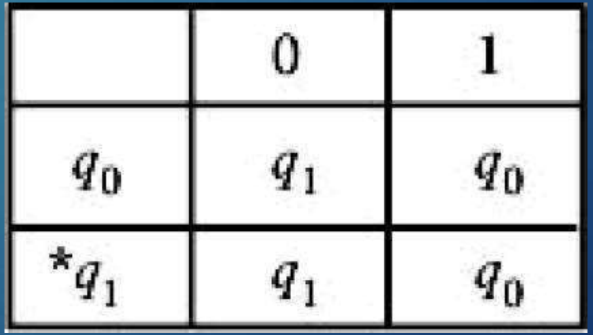
where

$$\delta(q_0, 0) = q_1, \delta(q_0, 1) = q_0$$
$$\delta(q_1, 0) = q_1, \delta(q_1, 1) = q_0$$

The transition diagram of this automata is:



Transition Diagram

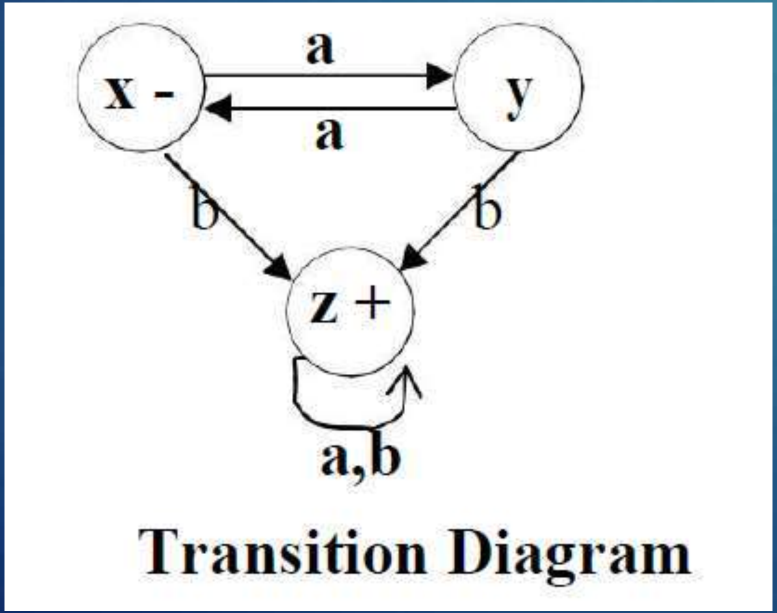


	0	1
q_0	q_1	q_0
$^*q_1$	q_1	q_0

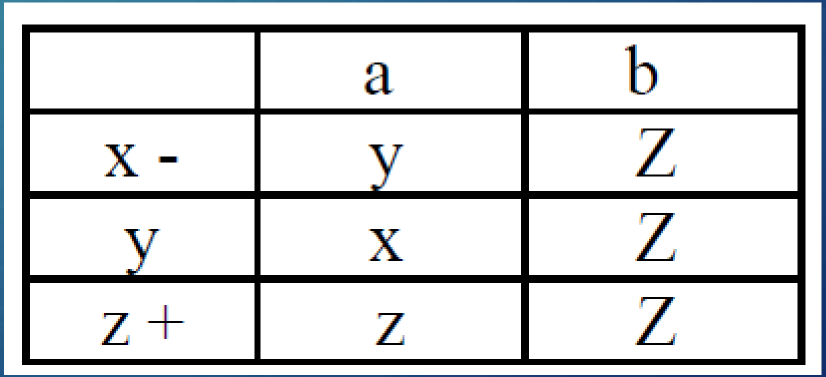
Transition Table

Example 2

if Σ={a,b}, states={x,y,z}
Rules of transition:
1. From state x and input a go to state y.
2. From state x and input b go to state z.
3. From state y and input a go to state x.
4. From state y and input b go to state z.
5. From state z and any input stay at the state z. Let x be the start state and z be the final state.



Transition Diagram



	a	b
x -	y	Z
y	x	Z
z +	z	Z

Note

The FA above will accept all strings that have the letter b in them and no other strings. The language associated with(or accepted by) this FA is the one defined by the regular expression: a*b(a+b)*

The set of all strings that do leave us in a final state is called the language defined by the FA. The word abb is accepted by this FA, but
The word aaa is not.

There are two types of finite automata:

- **DFA(deterministic finite automata)**
- **NFA(non-deterministic finite automata)**

1. DFA :
DFA refers to Deterministic Finite Automaton. A Finite Automata(FA) is said to be deterministic, if corresponding to an input symbol, there is single resultant state i.e. there is only one transition.

2. NFA :
NFA refers to Nondeterministic Finite Automaton. A Finite Automata(FA) is said to be non deterministic, if there is more than one possible transition from one state on the same input symbol.

Both of them are set of five tuples and represented as,

Q: A set of non empty finite states.
Σ: A set of non empty finite input symbols.
δ: It is a transition function that takes a state from Q and an input symbol from and returns a subset of Q.
qo: Initial state of NFA and member of Q.
F: A non-empty set of final states and member of Q.

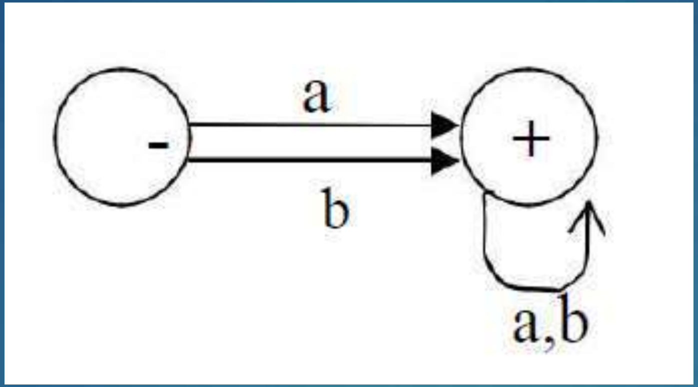
Difference between DFA and NFA :

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.

Example 3: L= [a,b,aa,bb,aaa,bbb….

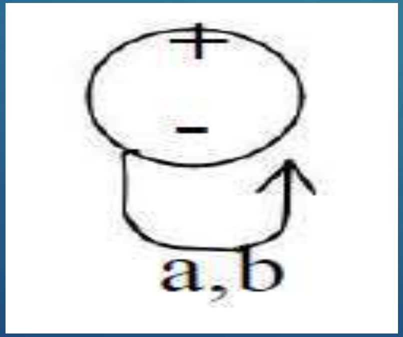
Find DFA for The following regular expression $(a+b)(a+b)^*=(a+b)^+$



Example 4:

Find DFA for The following regular expression: (a+b)*

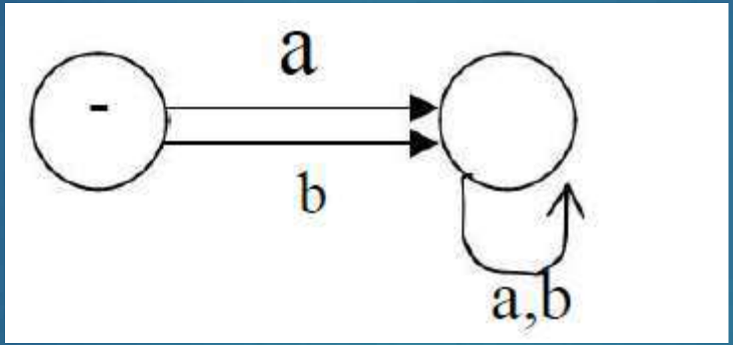
L=[^,a,b,aa,bb,



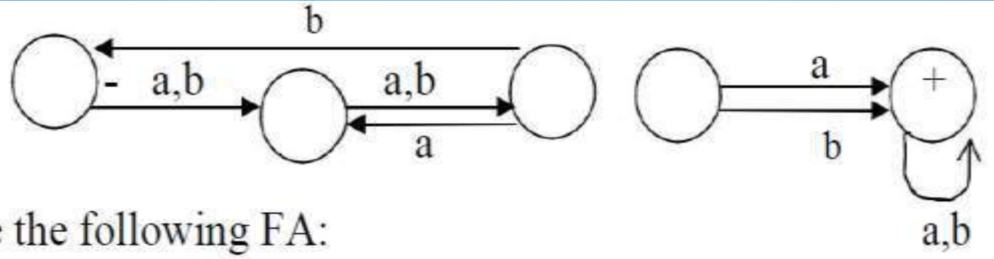
Note: every language that can be accepted by an FA can be defined by a regular expression and every language that can be defined by a regular expression can be accepted by some FA.

FA that accepts no language will be one of the two types:

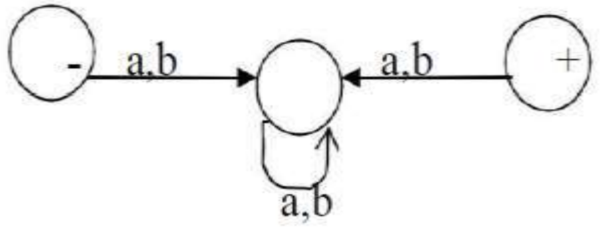
1. FA that have no final states. Like the following FA:



2. FA in which the final states cannot be reached. Like the following FA:

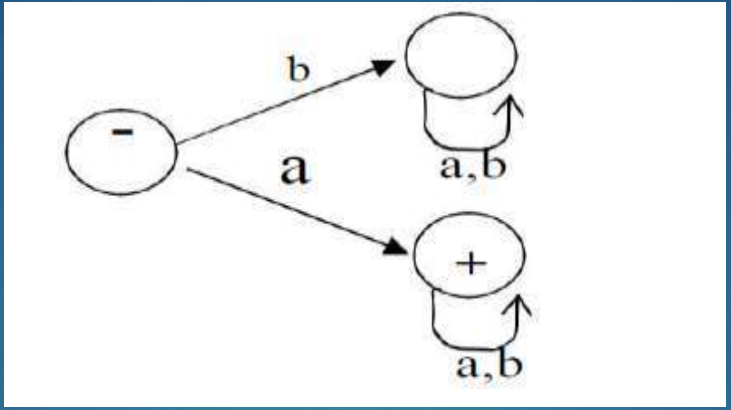


Or Like the following FA:



Example 5:

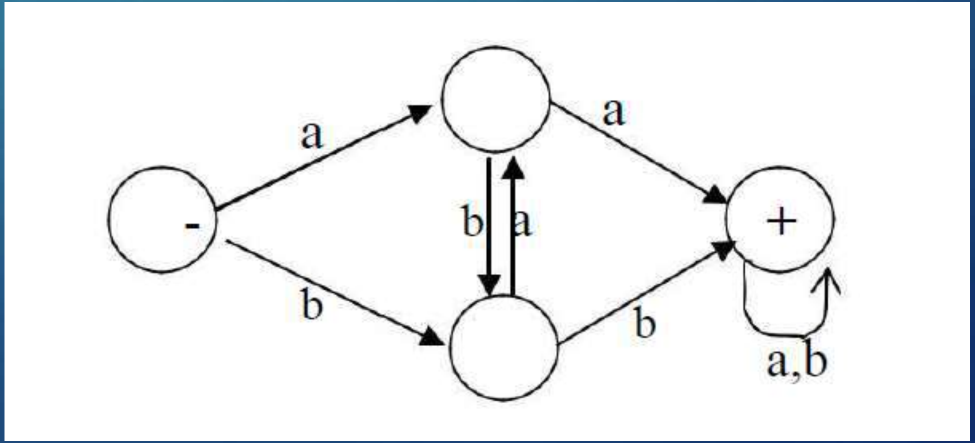
The following DFA accept all strings from the alphabet {a,b} that start with a.
L=[a,aa,ab,aaa,abb,…



The regular expression is: a(a+b)*

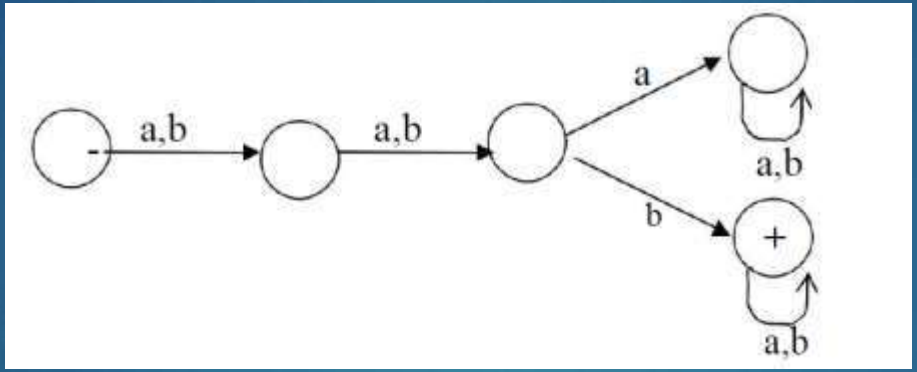
Example 6:

The following DFA accept all strings from the alphabet {a,b} with double letter. The regular expression is: (a+b)*(aa+bb) (a+b)*



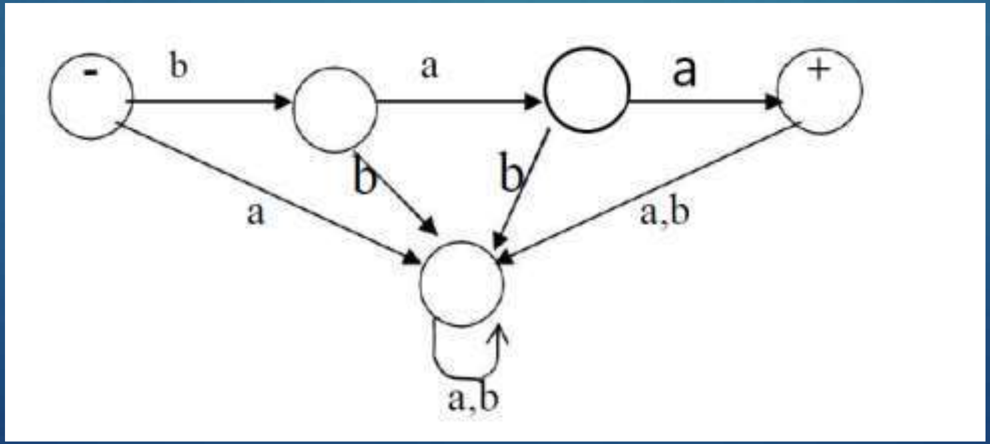
Example7:

The following DFA accepts the language defined by the regular expression: (aab+abb+bab+bbb)(a+b)*



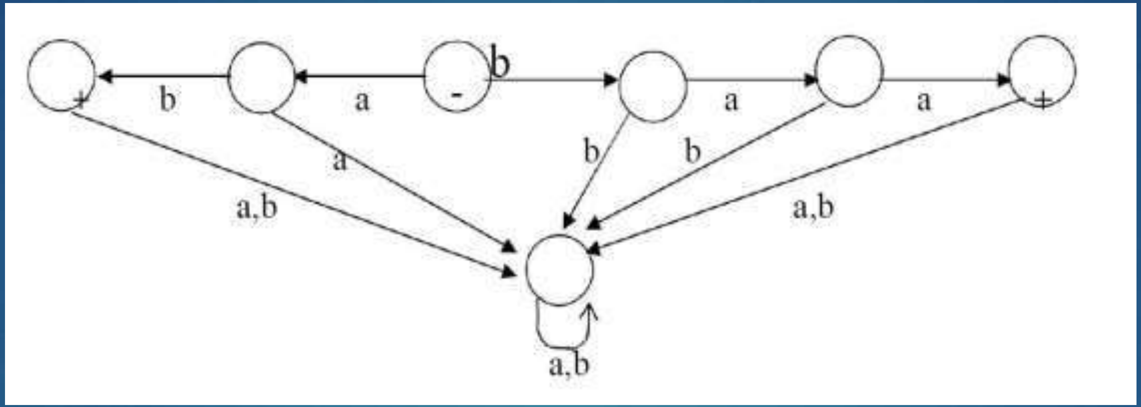
في حالة بناء FA لكلمة معينة يجب استخدام الطرق الوهمية لتحقيق مبدأ او شرط ال DFA

Example 8: the following DFA accepts only the word baa.

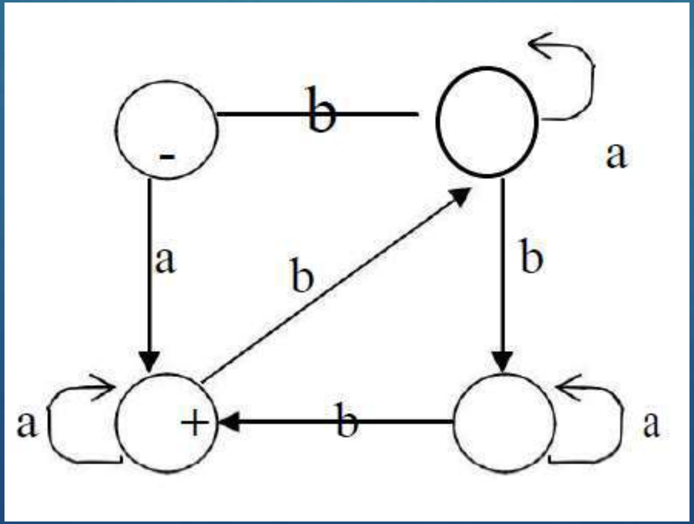


Example 9:

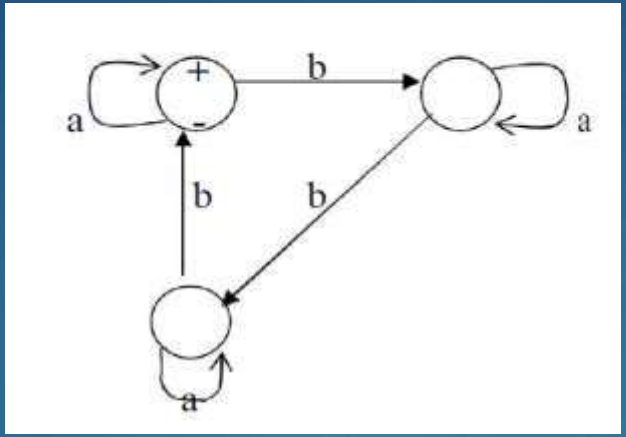
The following DFA accepts the words baa and ab.



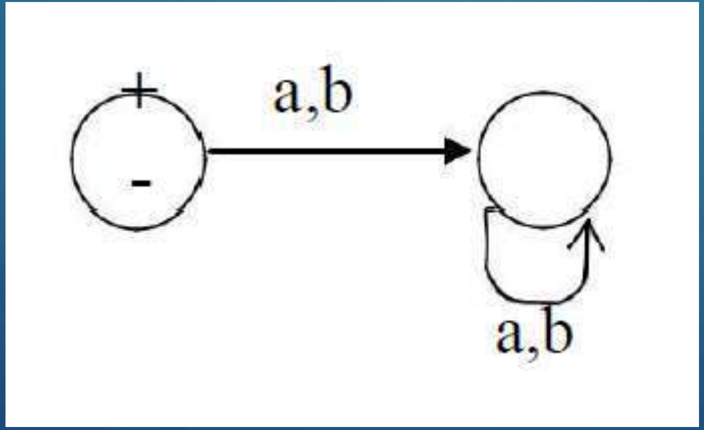
Example 10: the following FA accepts the language defined by the regular expression: $(a+ba^*ba^*b)^+$



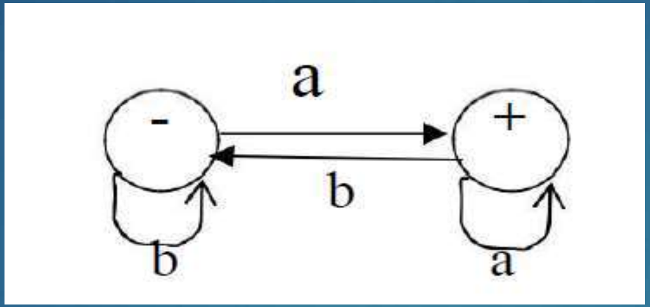
Example 11: the following DFA accepts the language defined by the regular expression: (a+ba*ba*b)*



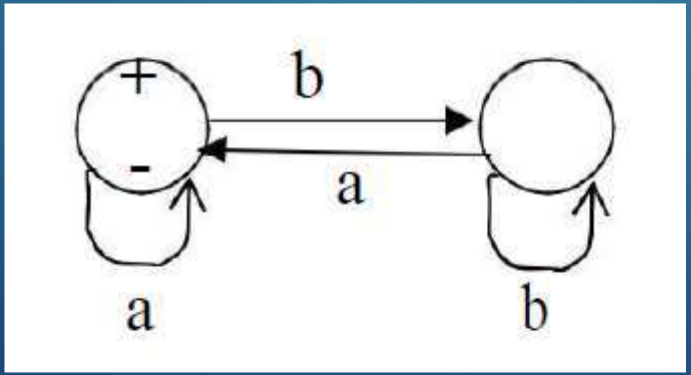
Example 12: the following DFA accepts only the word Λ.يوجد طرق وهمية



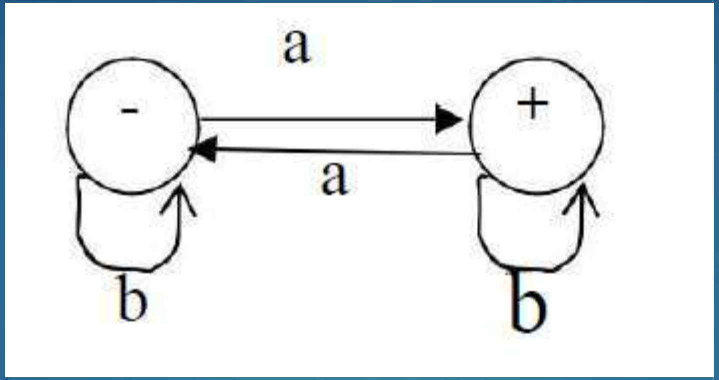
Example 13: the following DFA accepts all words from the alphabet {a,b} that end with a. The regular expression for this language is: (a+b)*a
L=[a,aa,ba,bba.aaa,bbba…



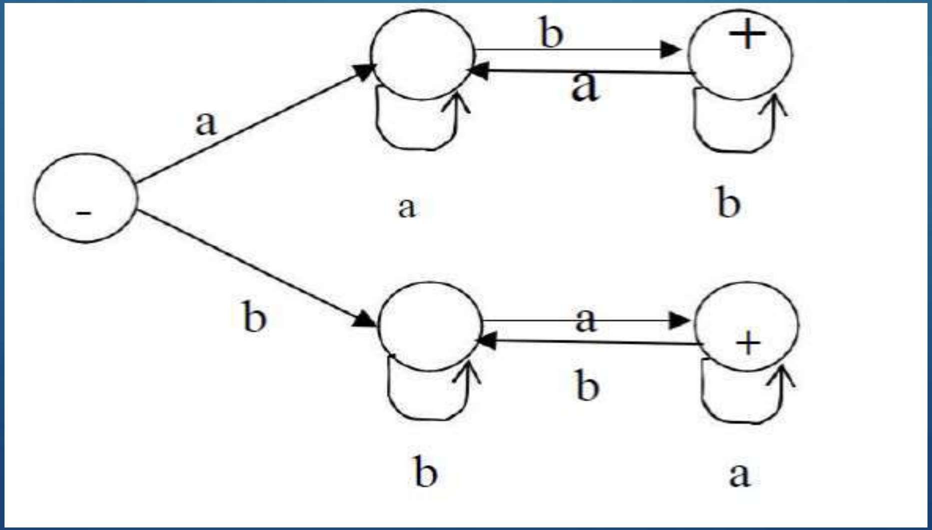
Example 14: the following DFA accepts all words from the alphabet {a,b} that do not end in b and accept Λ. The regular expression for this language is: (a+b)*a + Λ



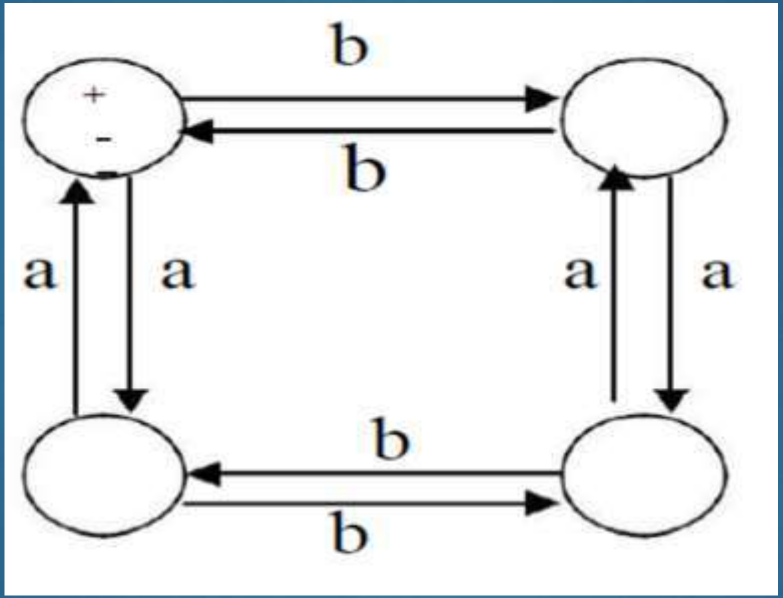
Example 15: the following FA accepts all words from the alphabet {a,b} with an odd number of a's. The regular expression for this language is: b*a(b*ab*ab*)*



Example 16: the following FA accepts all words from the alphabet {a,b} that have different first and last letters. The regular expression for this language is: a(a+b)*b + b(a+b)*a



Example 17: the following FA accepts the language defined by the regular expression (even-even): [aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*



H.W :-Find the DFA that accepts the language defined by the regular expression (odd-odd)

Thank you for Listening

Introduction to Grammar PSG , CSG , CFG

أ.م.د. سهاد مال الله

أ.م.د. عبير طارق مولود

أ.م. علاء نوري

النظرية الاحتسابية المرحله الثالثه/الكورس الاول/كل الافرع

قسم علوم الحاسوب/الجامعة التكنولوجية

2020-2019

Grammar in Automata-

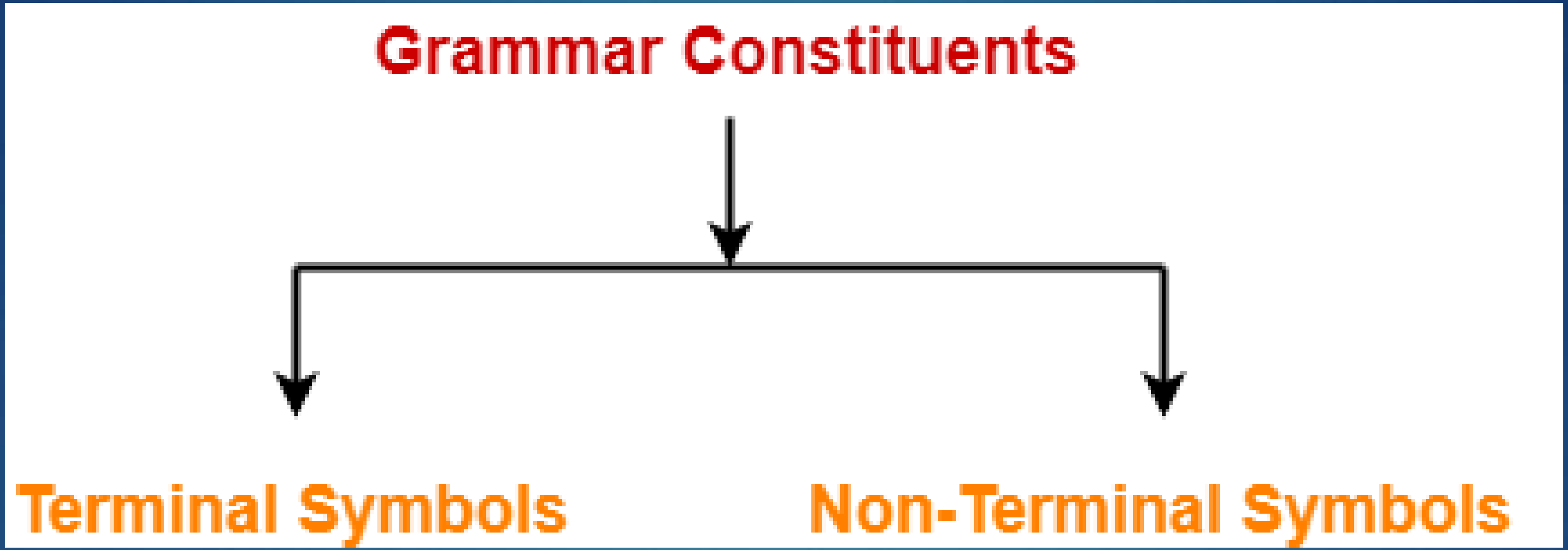
A grammar is a 4-tuple such that-

G = (N , T , P , S)

Where:-

- N = Finite non-empty set of non-terminal symbols
- T = Finite set of terminal symbols
- P = Finite non-empty set of production rules
- S = Start symbol

A Grammar is mainly composed of two basic elements-



1. **Terminal symbols**
2. **Non-terminal symbols**

1. Terminal Symbols-

- Terminal symbols are those which are the constituents of the sentence generated using a grammar.
- Terminal symbols are denoted by using small case letters such as a, b, c etc.

2. Non-Terminal Symbols-

- Non-Terminal symbols are those which take part in the generation of the sentence but are not part of it.
- Non-Terminal symbols are also called as auxiliary symbols or variables.
- Non-Terminal symbols are denoted by using capital letters such as A, B, C etc.

Example:-

Consider a grammar G = (N , T , P , S) where:-

- N = { S } // Set of Non-Terminal symbols
- T = { a , b } // Set of Terminal symbols
- P = { S → aSbS , S → bSaS , S → ∈ } // Set of production rules
- S = { S } // Start symbol

$$S \rightarrow aSbS$$
$$S \rightarrow bSaS$$
$$S \rightarrow \in$$

Example:

Consider a grammar G = (N , T , P , S) where-

- N= { S , A , B } // Set of Non-Terminal symbols
- T = { a , b } // Set of Terminal symbols
- P = { S → ABa , A → BB , B → ab , AA → b } // Set of production rules
- S = { S } // Start symbol

S → ABa
A → BB
B → ab
AA → b

Equivalent Grammars-

Two grammars are said to be equivalent if they generate the same languages.

Example-

Consider the following two grammars-

Grammar G1-

$S \rightarrow aSb \mid \epsilon$

Grammar G2-

$S \rightarrow aAb \mid \epsilon$

$A \rightarrow aAb \mid \epsilon$

Both these grammars generate the same language given as-

$L = \{ a^n b^n , n>=0 \}$

Example:-

Let us consider the grammar –

G2 = ({S, A}, {a, b}, S, {S → aAb, aA → aaAb, A → ε })

Some of the strings that can be derived are –

$S \Rightarrow \underline{aA}b$	using production S → aAb
$\Rightarrow a\underline{aA}bb$	using production aA → aAb
$\Rightarrow aaa\underline{A}bbb$	using production aA → aAb
$\Rightarrow aaabbb$	using production A → ε

Some Types of Grammar:

1. Context Free Grammar (CFG)
2. phrase-Structure Grammar (PSG)
3. context-sensitive grammar (CSG)

Context Free Grammar:-

A *context free grammar*, called CFG, is a collection of three things:

1. An alphabet ∑ of letters called terminals from which we are going to make strings that will be the words of a language.

A set of symbols called nonterminals, one of which is the symbol S, standing for "start here".

2. A finite set of production of the form:

3. One nonterminal → finite string of terminals and/ or nonterminals Where the strings of terminals and nonterminals can consist of only terminals or of any nonterminals, or any mixture of terminals and nonterminals or even the empty string. We require that at least one production has the nonterminal S as its left side.

Definition:-

The language generated by the CFG is the set of all strings of terminals that can be produced from the start symbol S using the production as substitutions. A language generated by the CFG is called a context free language (CFL).

Example

Let the only terminal be a.

Let the only nonterminal be S.

Let the production be:

$S \rightarrow aS$

$S \rightarrow \Lambda$

The language generated by this CFG is exactly a*.

In this language we can have the following derivation:

$S \rightarrow aS \rightarrow aaS \rightarrow aaaS \rightarrow aaaaS \rightarrow aaaaaS \rightarrow aaaaa\Lambda = aaaaa$

Example

Let the terminals be a, b. And the only nonterminal be S.

Let the production be:

$S \rightarrow aS$

$S \rightarrow bS$

$S \rightarrow \Lambda$

The language generated by this CFG is $(a+b)^*$.

In this language we can have the following derivation:

$S \rightarrow bS \rightarrow baS \rightarrow baaS \rightarrow baa\Lambda = baa$

Example

Let the terminals be a, b.

Let the nonterminals be S,X.

Let the production be:

$S \rightarrow XaaX$

$X \rightarrow aX$

$X \rightarrow bX$

$X \rightarrow \Lambda$

The language generated by this CFG is (a+b)* aa(a+b)*.

To generate baabaab we can proceed as follows:

$S \rightarrow XaaX \rightarrow bXaaX \rightarrow baXaaX \rightarrow baaXaaX \rightarrow baabXaaX \rightarrow baab\Lambda aaX = baabaaX$

$\rightarrow baabaabX \rightarrow baabaab\Lambda = baabaab$

1- phrase-Structure Grammar:-

A *phrase-Structure Grammar*, called PSG, is a collection of three things:

1. **An alphabet ∑ of letters called terminals.**
2. **A set of symbols called nonterminals that includes the start symbol S.**
3. **A finite set of productions of the form:**

$$\text{String } 1 \rightarrow \text{String } 2$$

Where string 1 can be any string of terminals and nonterminals that contains at least one nonterminals and where string 2 is any string of terminals and nonterminals whatsoever.

Definition:

The language generated by the PSG is the set of all strings of terminals that can be derived starting at S.

Example: the following is a phrase-structure grammar over Σ={a,b} with nonterminals S, A and B:

$$S \longrightarrow aSBA$$
$$S \longrightarrow abA$$
$$AB \longrightarrow BA$$
$$BB \longrightarrow bb$$
$$bA \longrightarrow ba$$
$$aA \longrightarrow aa$$

2- Context-Sensitive Grammar (CSG)

A context-sensitive grammar (CSG) is a formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by a context of terminal and nonterminal symbols.

Context Sensitive Grammar(Type1 Grammar)

- A context-sensitive grammar (CSG) is an unrestricted grammar in which every production has the form $\alpha \to \beta$ with $|\beta| \geq |\alpha|$ (where α and β are strings of nonterminals and terminals).

- The concept of context-sensitive grammar was introduced by Noam Chomsky in the 1950.

- In every derivation the length of the string never decreases.

- The term ”context-sensitive” comes from a normal form for these grammars , where each production is of the form $\alpha 1 A \alpha 2 \to \alpha 1 \beta \alpha 2$,with β not equal to empty.

- They permit replacement of variable A by string β only in the ”context ” $\alpha 1$ - $\alpha 2$.

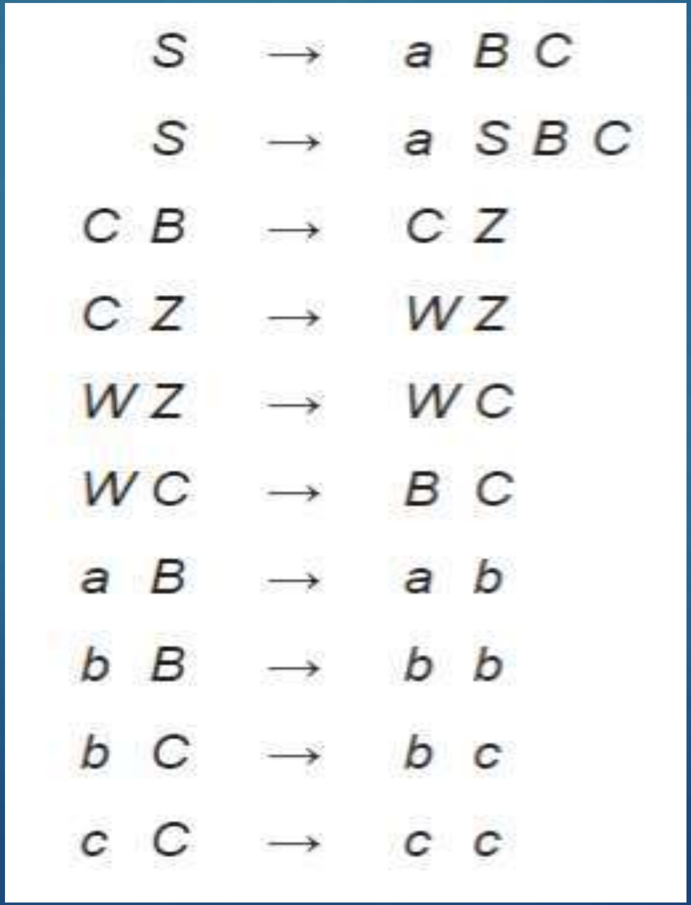
- a rule of the form

$$S \to \lambda$$

where λ represents the empty string and S does not appear on the right-hand side of any rule is permitted.

Example:

The following context-sensitive grammar, with start symbol S, generates the canonical non-context-free language $\{ a^n b^n c^n : n \geq 1 \}$:



S	$\rightarrow$	$a\ B\ C$
S	$\rightarrow$	$a\ S\ B\ C$
$C\ B$	$\rightarrow$	$C\ Z$
$C\ Z$	$\rightarrow$	$W\ Z$
$W\ Z$	$\rightarrow$	$W\ C$
$W\ C$	$\rightarrow$	$B\ C$
$a\ B$	$\rightarrow$	$a\ b$
$b\ B$	$\rightarrow$	$b\ b$
$b\ C$	$\rightarrow$	$b\ c$
$c\ C$	$\rightarrow$	$c\ c$

Thank you for Listening

CHOMSKY NORMAL FORM (CNF)

أ.م.د. سهاد مال الله
أ.م.د. عبير طارق مولود
أ.م. علاء نوري

النظرية الاحتسابية/الكورس الاول/كل الافرع

قسم علوم الحاسوب/الجامعة التكنولوجية

٢٠١٩-٢٠٢٠

Definition

If a CFG has only productions of the form:

Nonterminal → string of two Nonterminals

Or of the form:

Nonterminal → One Terminal

It is said to be in **Chomsky Normal Form (CNF)**.

Definition

In a given CFG, we call a nonterminal N **nullable** if:

There is a production: **N→ Λ**

Or

There is a derivation that start at N and leads to Λ: **N→…→ Λ**

Example

Consider the following CFG:

$S \rightarrow a \mid Xb \mid aYa$

$X \rightarrow Y \mid \Lambda$

$Y \rightarrow b \mid X$

X and Y are nullable.

The new CFG is:

$S \rightarrow a \mid Xb \mid aYa \mid b \mid aa$

$X \rightarrow Y$

$Y \rightarrow b \mid X$

Example

Consider the following CFG:

$S \rightarrow Xa$

$X \rightarrow aX \mid bX \mid \Lambda$

X is the only nullable nonterminal.

The new CFG is:

$S \rightarrow Xa \mid a$

$X \rightarrow aX \mid bX \mid a \mid b$

Example

Consider the CFG:

$S \rightarrow XY$

$X \rightarrow Zb$

$Y \rightarrow bW$

$Z \rightarrow AB$

$W \rightarrow Z$

$A \rightarrow aA \mid bA \mid \Lambda$

$B \rightarrow Ba \mid Bb \mid \Lambda$

A, B, W and Z are nullable.

The new CFG is:

$S \rightarrow XY$

$X \rightarrow Zb \mid b$

$Y \rightarrow bW \mid b$

$Z \rightarrow AB \mid A \mid B$

$W \rightarrow Z$

$A \rightarrow aA \mid bA \mid a \mid b$

$B \rightarrow Ba \mid Bb \mid a \mid b$

Definition

A production of the form: One Nonterminal → One Nonterminal is called a **unit** production.

Theorem

If there is a CFG for the language L that has no Λ-production, then there is also a CFG for L with no Λ-production and no unit production.

Example

Consider the CFG:

$S \to A \mid bb$
$A \to B \mid b$
$B \to S \mid a$

Sol:

$S \to A$	gives	$S \to b$
$S \to A \to B$	gives	$S \to a$
$A \to B$	gives	$A \to a$
$A \to B \to S$	gives	$A \to bb$
$B \to S$	gives	$B \to bb$
$B \to S \to A$	gives	$B \to b$

The new CFG for this language is:

$S \to bb \mid b \mid a$
$A \to b \mid a \mid bb$
$B \to a \mid bb \mid b$

Theorem

If L is a language generated by some CFG then there is another CFG that generates all the non- Λ words of L, all of these productions are of one of two basic forms:

Nonterminal → string of only Nonterminals

Or

Nonterminal → One Terminal

Example

Consider the CFG:

$S \to X_1 \mid X_2aX_2 \mid aSb \mid b$

$X_1 \to X_2X_2 \mid b$

$X_2 \to aX_2 \mid aaX_1$

Becomes:

$S \to X_1$

$S \to X_2AX_2$

$S \to ASB$

$S \to B$

$X_1 \to X_2X_2$

$X_1 \to B$

$X_2 \to AX_2$

$X_2 \to AAX_1$

$A \to a$

$B \to b$

Theorem

For any CFL the non- Λ words of L can be generated by a grammar in which all productions are of one of two forms:

Nonterminal $\rightarrow$ string of exactly two Nonterminals

Or

Nonterminal $\rightarrow$ One Terminal

Definition

If a CFG has only productions of the form:

Nonterminal → string of two Nonterminals

Or of the form:

Nonterminal → One Terminal

It is said to be in **Chomsky Normal Form (CNF)**.

Example

Convert the following CFG into CNF:

$S \rightarrow aSa \mid bSb \mid a \mid b \mid aa \mid bb$

$S \rightarrow ASA$

$S \rightarrow BSB$

$S \rightarrow AA$

$S \rightarrow BB$

$S \rightarrow a$

$S \rightarrow b$

$A \rightarrow a$

$B \rightarrow b$

The CNF:

$S \rightarrow AR_1$

$R_1 \rightarrow SA$

$S \rightarrow BR_2$

$R_2 \rightarrow SB$

$S \rightarrow AA$

$S \rightarrow BB$

$S \rightarrow a$

$S \rightarrow b$

$A \rightarrow a$

$B \rightarrow b$

Example

Convert the following CFG into CNF:

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

The CNF:

$S \rightarrow YA \mid XB$

$A \rightarrow YR_1 \mid XS \mid a$

$B \rightarrow XR_2 \mid YS \mid b$

$X \rightarrow a$

$Y \rightarrow b$

$R_1 \rightarrow AA$

$R_2 \rightarrow BB$

Example

Convert the following CFG into CNF:

$S \rightarrow AAAAS$

$S \rightarrow AAAA$

$A \rightarrow a$

The CNF:

$S \rightarrow AR_1$

$R_1 \rightarrow AR2$

$R_2 \rightarrow AR_3$

$R_3 \rightarrow AS$

$S \rightarrow AR_4$

$R_4 \rightarrow AR_5$

$R_5 \rightarrow AA$

$A \rightarrow a$