Grammar

Definition of a Grammar

Definition 4.1 A phrase-structure grammar (or simply a grammar) is $(V_N, \sum, P, S)$where

(i) V_N is a finite non empty set whose elements are called variables,

(ii) $\sum$ is a finite non empty set whose elements are called terminals,

(iii) $V_N \cap \sum = \emptyset$

(iv) S is a special variable (i.e, an element of V_N) called the start symbol

(v) P is a finite set whose elements are $\alpha \rightarrow \beta$, where α and β are strings on $V_N \cup \sum$. α has at least one symbol from V_N. Elements of P are called productions or production rules or rewriting rules.

Definition of a Grammar

- The production rules.

1) Reverse substitution is not permitted.

For example, if *S* ->*AB* is a production,
then we can replace *S* by *AB*,
but we cannot replace *AB* by *S*.

2) No inversion operation is permitted.

For example, if *S* -> *AB is* a production.
It is not necessary that *AB* -> *S* is a production.

Grammar : Example 1

- G = (V_N ,∑, *P*, *S)* is a grammar
- where
- V_N = {<sentence>, <noun>, <verb>, <adverb>}
- ∑ = [Ram, Sam, ate, sang, well]
- S =<sentence>
- *P* consists of the following productions:

<sentence> -> <noun> <verb>
<sentence> -> <noun> <verb> <adverb>
<noun> -> Ram
<noun> -> Sam
<verb> -> ate
<verb> -> sang
<adverb> -> well

Grammar : Example 2

If $G = (\{S\}, \{0, 1\}, \{S \to 0S1,\ S \to \Lambda\}, S)$, find $L(G)$.

$S \to 0S1$

$S \to \Lambda$ so $\Lambda \in L(G)$

$S \to 0S1$

$0S1 \to 0^2S1^2 \to 0^2 1^2 \to 0011$

$000S111 \to 0^3S1^3 \to 0^3 1^3 \to 000111$

$\to 0^n S 1^n$

$0^n \Lambda 1^n$

$\Rightarrow 0^n 1^n$

Grammar : Example 2

If $G = (\{S\}, \{0, 1\}, \{S \to 0S1, S \to \Lambda\}, S)$, find $L(G)$.

Solution

As $S \to \Lambda$ is a production, $S \underset{G}{\Rightarrow} \Lambda$. So Λ is in $L(G)$. Also, for all $n \geq 1$,

$$S \underset{G}{\Rightarrow} 0S1 \underset{G}{\Rightarrow} 0^2S1^2 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} 0^nS1^n \underset{G}{\Rightarrow} 0^n1^n$$

Therefore,

$$0^n1^n \in L(G) \text{ for } n \geq 0$$

In the above derivation, *S* ->0S1 is applied at every step except

in the last step. In the last step, we apply S -> ^).

L(G) = {0^n1^n , *n>=0*}

Grammar : Example 3

Σ

If G = ({ S}, {*a*}, {S ->SS}, S), find the language generated by **G.**

S → SS

variables but Σ, no production

empty lang

L(G) = ϕ

Grammar : Example 3

If G = ({ S}, *{a}*, {S ->SS}, S), find the language generated by **G.**

Solution

- *L(G)* = ø
- Since the only production S -> SS in G has no terminal on the right-hand side.

Regular Grammar

- **Regular grammar is a formal grammar in which every production is restricted to have one of the following forms:-**

A->aB

A->a

- **where A and B are the non-terminals, a is a terminal symbol**

- **With** ε productions permitted as special case when L(G) contains ε

Regular Grammar

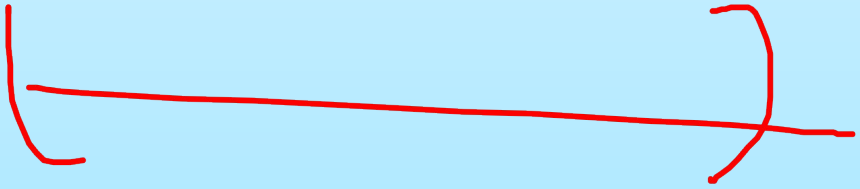
- It is called as regular grammar, because if the format of every production is limited to A->aB, A->a
- then the grammar can specify only regular sets and hence there exists a finite automata accepting L(G), if G is regular

Equivalence of Regular Grammar and Finite Automata

- Given a regular grammar G, a finite automata accepting L(G) can be obtained as follows:-

- **The number of states of the automata will be equal to the number of non-terminals of the grammar plus one, i.e.**

- **there will be a state corresponding to every non-terminal of the grammar and one more state will be there, which will be the final state of the automata.**



Equivalence of Regular Grammar and Finite Automata

- **The state corresponding to the start symbol of the grammar will be initial state of the automata**
- **If L(G) contains** ε then make start state also as the final state

Equivalence of Regular Grammar and Finite Automata

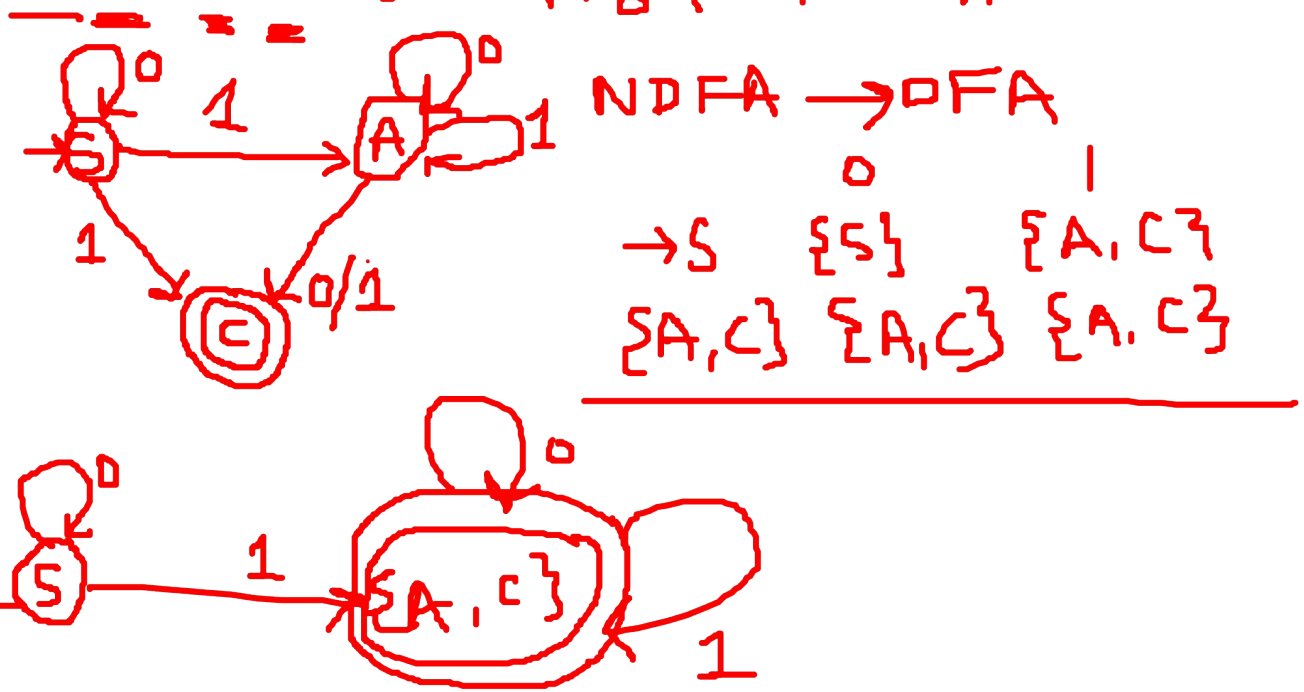
- **The transitions in the automata can be obtained as follows:-**
- **For every production A->aB do make**

  $\delta(A,a)=B$
- **For every production of the form A->a do make**

  $\delta(A,a)=$final state

Equivalence of Regular Grammar and Finite Automata-Example 1

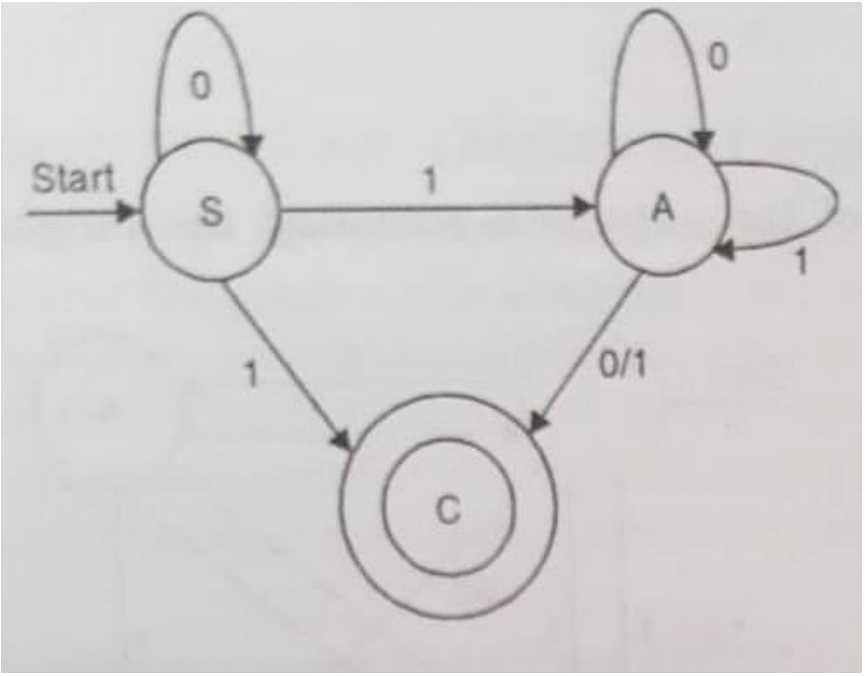
- **Consider a regular grammar given below:-**
- **S->0S|1A|1**
- **A->0A|1A|0|1**



Equivalence of Regular Grammar and Finite Automata-Example 1

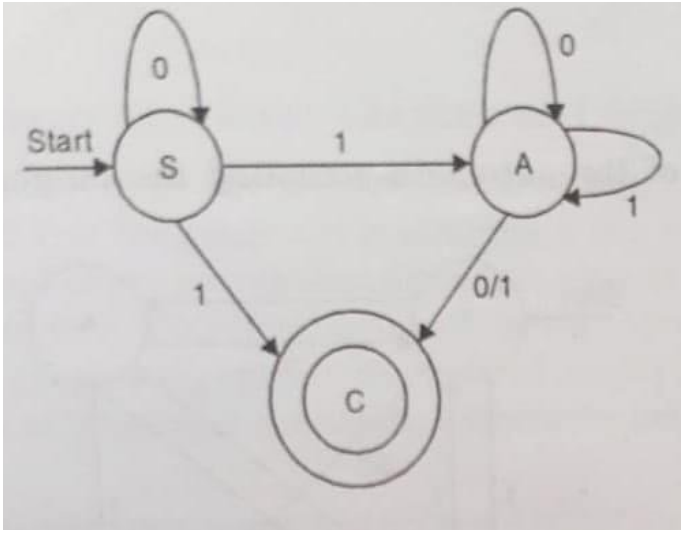
- **Consider a regular grammar given below:-**
- **S->0S|1A|1**
- **A->0A|1A|0|1**

- **Transition Diagram for the finite automata:-**



Equivalence of Regular Grammar and Finite Automata-Example 1

- **Its NDFA:**
- **Corresponding DFA transition table:-**

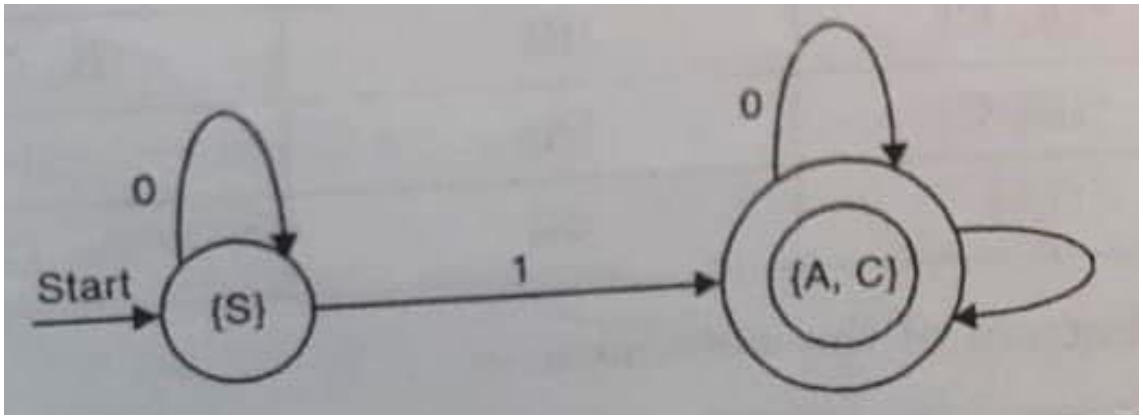


	0	1
{S}	{S}	{A, C}
*{A, C}	{A, C}	{A, C}

Equivalence of Regular Grammar and Finite Automata-Example 1

- **Its NDFA:**
- **Corresponding DFA transition diagram:-**

	0	1
{S}	{S}	{A, C}
*{A, C}	{A, C}	{A, C}



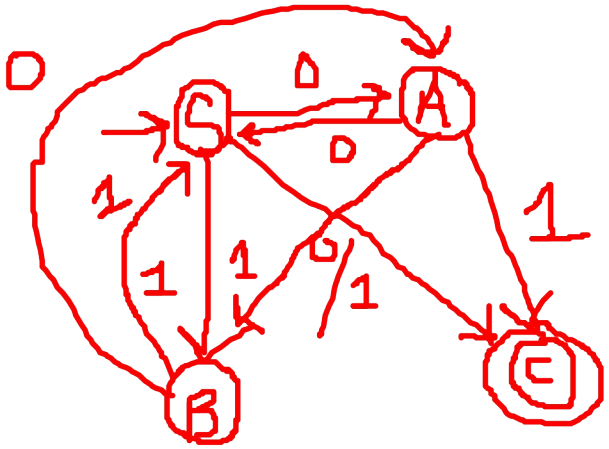
Equivalence of Regular Grammar and Finite Automata-Example 2

- **Consider a regular grammar given below:-**

S->0A|1B|0|1

A->0S|1B|1

B->0A|1S



	0	1
→ S	{A, C}	{B, C}
{A, C}	{S}	{B, C}
{B, C}	{A}	{S}
{A}	{S}	{B, C}

Equivalence of Regular Grammar and Finite Automata-Example 2

- **Consider a regular grammar given below:-**

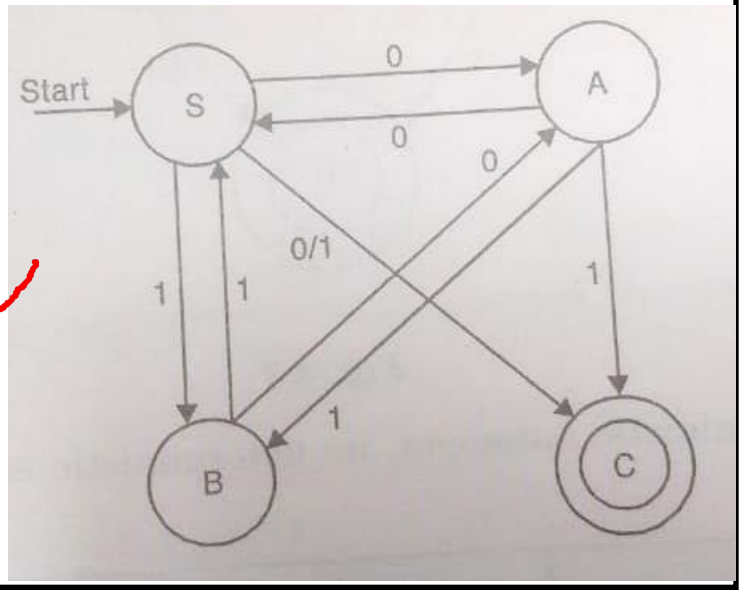
S->0A|1B|0|1

A->0S|1B|1

B->0A|1S

Transition Diagram for the Automata:-

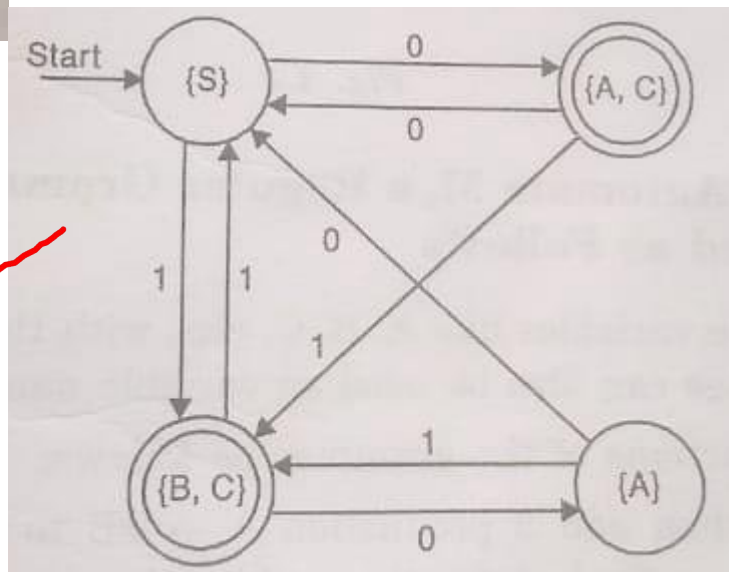
It's a NDFA



Equivalence of Regular Grammar and Finite Automata-Example 2

- **The transition diagram for the Equivalent DFA is :**

	0	1
{S}	{A, C}	{B, C}
*{A, C}	{S}	{B, C}
*{B, C}	{A}	{S}
{A}	{S}	{B, C}

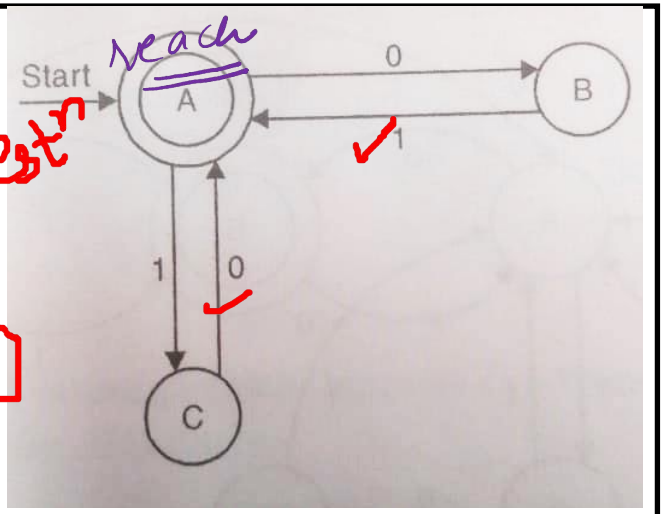


Generating Regular Grammar from Given Finite Automata

1) Associate suitable variables like A,B,C with the states of the automata. The labels of the states can be used as variable names
2) Obtain the productions of the grammar as follows:-
   - If δ(A,a)=B, then add a production A->aB to the list of productions of the grammar.
   - If B is a final state, then add either A->a or B-> ε to the list of productions
3) The variable associated with the initial state of the automata is the start symbol of the grammar

Generating Regular Grammar from Given Finite Automata –Example 1

- **Consider a FA given below:-**



$A \rightarrow 0B \mid 1C$

$B \rightarrow 1A \mid 1$

$C \rightarrow 0A \mid 0$

$\in$ free

$A \rightarrow a$ or $B \rightarrow \in$

$\delta(A, a) = B$ + final

OR

$A \rightarrow 0B \mid 1C \mid \in$

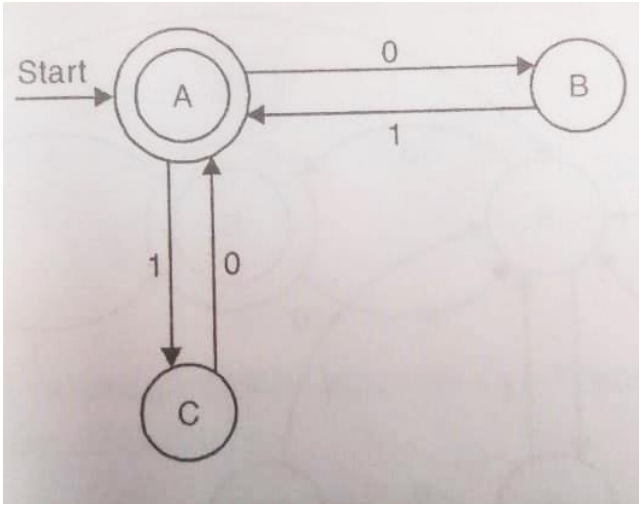
$B \rightarrow 1A$

$C \rightarrow 0A$

with $\in$

Generating Regular Grammar from Given Finite Automata –Example 1

- Consider a FA given below:-



A->0B|1C|ε
B->1A
C->A

or

A->0B|1C
B->1A|1
C->0A|0

- Both the grammars are same,
- The first one contains ε transitions,
- The second one is ε free.

Generating Regular Expression from Regular Grammar

1) Replace -> symbol in the production by "=" symbol to get a set of equations
2) Solve the the set of equations obtained above to get the value of the variable S, where S is the start symbol of the grammar, The result is the regular expression specifying L(G)

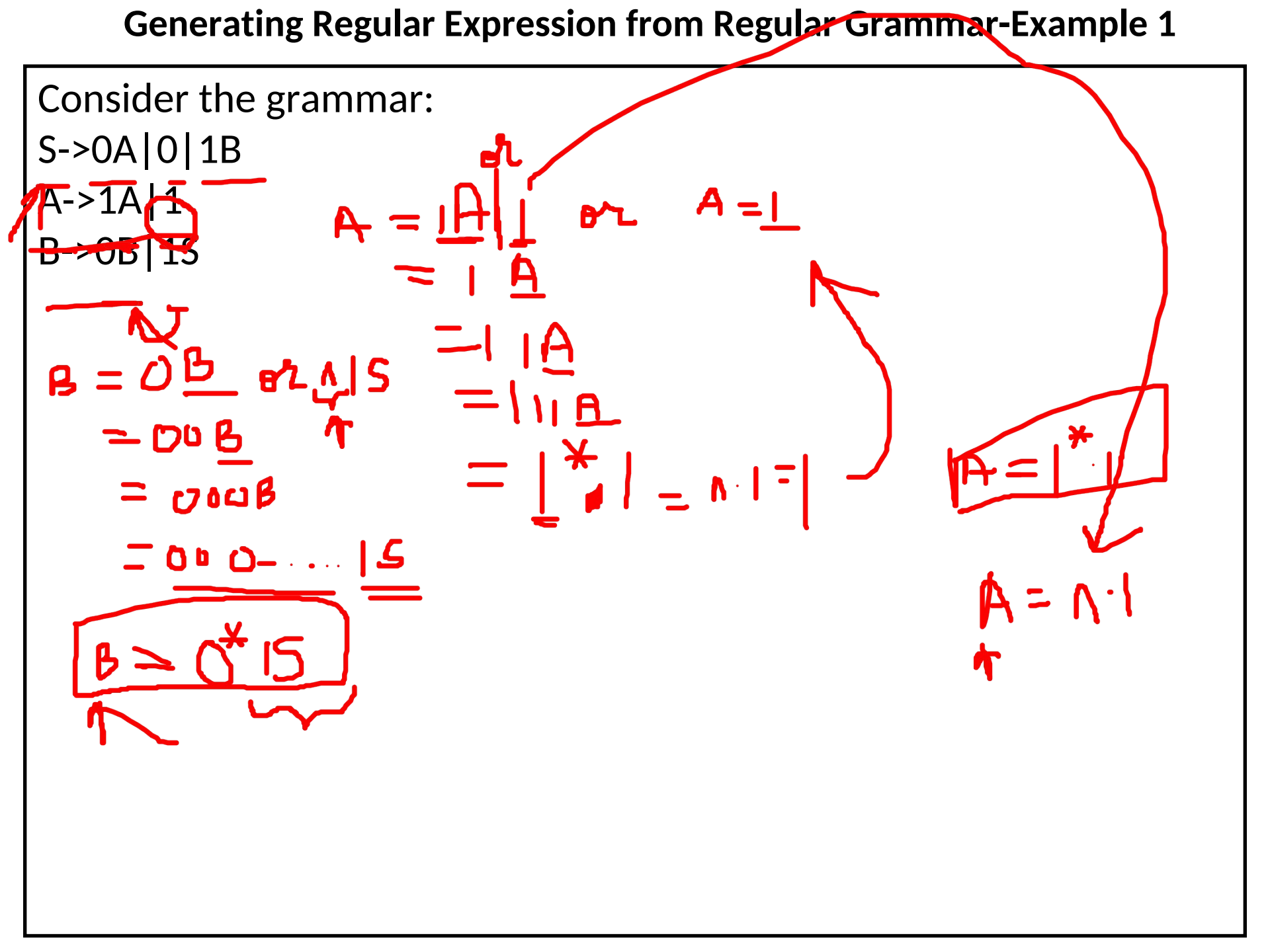
Generating Regular Expression from Regular Grammar-Example 1

Consider the grammar:

S->0A|0|1B

A->1A|1

B->0B|1S



Generating Regular Expression from Regular Grammar-Example 1

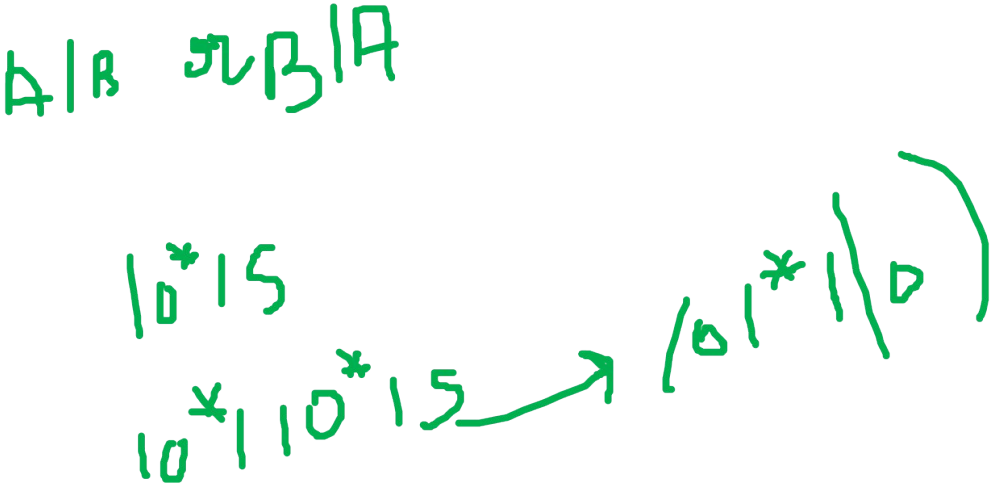
Consider the grammar:
S->0A|0|1B
A->1A|1
B->0B|1S

S=0A|0|1B
A=1A|1
B=0B|1S

B=0*1S
A=1*1

S=01*1|0|10*1S
S= (01*1|0) |(10*1S)
S=(10*1)*(01*1|0)
S=(10*1)*(01*1|0)

Regular Expression is
(10*1)*(01*1|0)



Regular Grammar

- **Regular grammar is a formal grammar in which every production is restricted to have one of the following forms:-**

A->aB

A->a

- **where A and B are the non-terminals, a is a terminal symbol**

- **With** ε productions permitted as special case when L(G) contains ε

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

Today's

Regular Grammar

Regular grammar is a type of grammar that describes a regular language.

This grammar can be of two forms:

- Right Linear Regular Grammar
- Left Linear Regular Grammar

Right Regular Grammar

Right Linear Regular Grammar

- In this type of regular grammar, all the non-terminals on the right-hand side exist at the rightmost place, i.e; right ends.

Example 1 :

- A ⇢ a,
- A ⇢ a**B**,
- A ⇢ ∈
- where, A and B are non-terminals,
- a is terminal, and ∈ is empty string

Right Regular Grammar

Example 2 :

- S ⇢ 00**B** | 11**S**
- B ⇢ 0**B** | 1**B** | 0 | 1

- where, S and B are non-terminals, and
- 0 and 1 are terminals

Left Regular Grammar

Left Linear Regular Grammar

- In this type of regular grammar, all the non-terminals on the right-hand side exist at the leftmost place, i.e; left ends.

Example 1 :

- A ⇢ a,
- A ⇢ **B**a,
- A ⇢ ∈
- where, A and B are non-terminals,
- a is terminal, and ∈ is empty string

Left Regular Grammar

Example 2 :

- S ⇢ **S**00 | **S**11
- B ⇢ **B**0 | **B**1 | 0 | 1
- where S and B are non-terminals, and
- 0 and 1 are terminals

Conversion of Right Linear Grammar to Left Linear Grammar

RLG LLG

1) Obtain a regular expression for the language generated by the given grammar
2) Reverse the regular expression obtained in Step 1
3) Obtain the regular/right linear grammar for the regular expression obtained in step 2 by drawing FA
4) Reverse the right side of every production of the grammar obtained in step 3

The resultant grammar will be an equivalent left linear grammar

① RE
② RRE
③ FA (Draw) & RG (write/derive)
④ RHS → Rev

Conversion of Right Linear Grammar to Left Linear Grammar

- Consider Right Linear grammar:-

S->01B|0

B->1B|11

① RE

$S = 01B \mid 0$

$B = 1B \mid 11$

Find RE for B

$B = 1B \mid 11$

Case II

$B = 1B$

$1\,1B$

$1.1\,1B$

$1.1.1\,1B$

Case I

$B = 11$

RE

String 0 = (11)

$\Lambda \cdot (11)$

Str = $1\,\underline{B} \Rightarrow \underline{1}.(11)$

$1.1B \Rightarrow \underline{1.1}.(11)$

$1.1.1.B \Rightarrow \underline{1.1.1}.(11)$

$B = 1^*(11)$

$S = 01B \mid 0$

$RE = 011^*(11) \mid 0$

for the grammar/system

② RRE

$(F+G)^R = F^R + G^R = G^R + F^R$

Concat $(F.G)^R = G^R.F^R$

$(F^*)^R = (F^R)^*$

$RRE = (011^*(11) \mid 0)$

$RRE = (1^R.1^R.(1^*)^R.1^R.0^R \mid 0^R)$

$= 1.1.(1^R)^*.1.0 \mid 0$

$= (1.1.1^*.1.0 \mid 0)$

Conversion of Right Linear Grammar to Left Linear Grammar

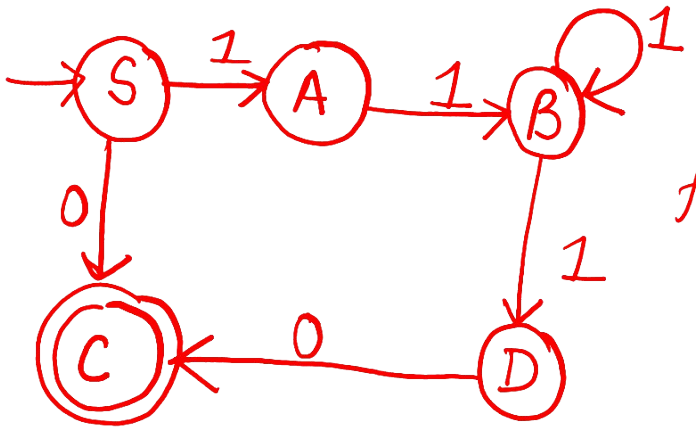
- Consider Right Linear grammar:-

S->01B|0

B->1B|11

② $(111^*10|0)$ Reverse RE

③ Draw FA & write Grammar



$S \to 1A \mid 0C \mid 0$ (0C: rem)

$A \to 1B$

$B \to 1B \mid 1D$

$D \to 0C \mid 0$ (0C: rem)

finite automata

C is a useless state, Remove C from productions

$S \to 1A \mid 0$

$A \to 1B$

$B \to 1B \mid 1D$

$D \to 0$

Replace the value of D

$S \to 1A \mid 0$

$A \to 1B$

$B \to 1B \mid 10$

Repl Substitute value of A in S

RHS

$S \to 1B \mid 0$

$B \to 1B \mid 10$

Reduced Grammer

④ $S \to B11 \mid 0$

$B \to B1 \mid 01$

left

RHS

Conversion of Right Linear Grammar to Left Linear Grammar

- Consider Right Linear grammar:-

S->01B|0

B->1B|11

Obtain a regular expression

S=01B|0

B=1B|11

B=1*(11)

S=011*(11)|0

Regular Expression=(011*(11)|0)

Reversal of a Regular Expression

- Basis: If E is a symbol a, ε, or ∅, then $E^R = E$.
- Induction: If E is
  - E=F+G, then $E^R = F^R + G^R$.
  - E=FG, then $E^R = G^R F^R$
  - $E=F^*$, then $E^R = (F^R)^*$.

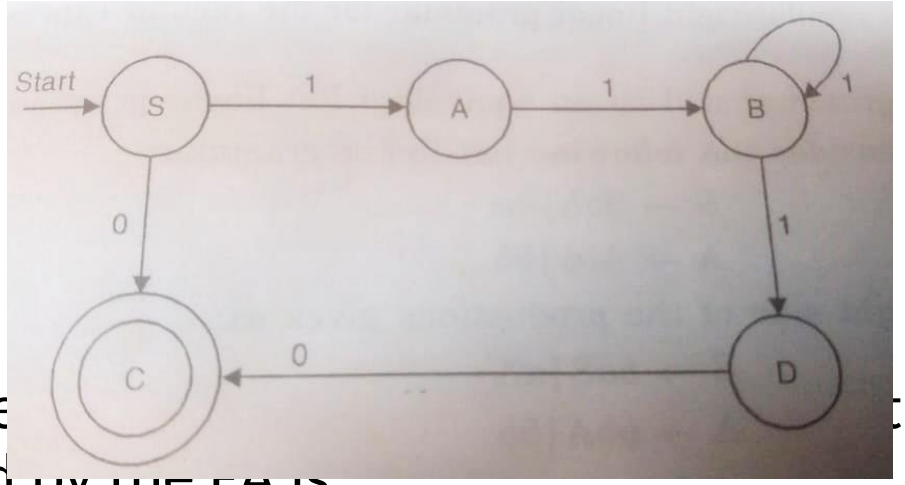
Conversion of Right Linear Grammar to Left Linear Grammar

Regular Expression=(011*(11)|0)

Reverse Regular Expression=(0|(11)1*10)=(0|111*10)

Conversion of Right Linear Grammar to Left Linear Grammar

- Reverse Regular Expression=(0|(11)1*10)=(0|111*10)
- Finite Automata accepting the language specified by the RE



- Thus, the ... the language accepted by the FA is:
- S->1A|0C|0
- A->1B
- B->1D|1B
- D->0C|0

Conversion of Right Linear Grammar to Left Linear Grammar

- Thus, the right linear grammar generating the language accepted by the FA is:

S->1A|0C|0
A->1B
B->1D|1B
D->0C|0

- Since C is not useful eliminating C gives:-

S->1A|0
A->1B
B->1D|1B
D->0

Conversion of Right Linear Grammar to Left Linear Grammar

S->1A|0
A->1B
B->1D|1B
D->0
This can be further simplified
S->1A|0
A->1B
B->1B|10
Further simplified as
S->11B|0
B->1B|10

Conversion of Right Linear Grammar to Left Linear Grammar

S->11B|0
B->1B|10
Reversing the right side of the productions give:
S->B11|0
B->B1|01
Which is equivalent Left Linear grammar

Obtaining Right Linear Grammar Equivalent to a Given Left Linear Grammar

LLG → RLG

1) Reverse the right side of every production of the given grammar
2) Obtain a regular expression for the language generated by the grammar obtained in step 1
3) Reverse the regular expression obtained in Step 2
4) Obtain the regular/right linear grammar for the regular expression obtained in step 3

The resultant grammar will be an equivalent left linear grammar

① RHS Reversing
② RE
③ RRE
④ FA (Draw) & Write/Derive (Grammar)

Conversion of Left Linear Grammar to Right Linear Grammar

Consider the following Left Linear Grammar:-

S->Sab|Aa

A->Abb|bb

RHS

① RHS reversal

$S \rightarrow baS \mid aA$

$A \rightarrow bbA \mid bb$

② RE

Case II

$S1 = b.b(bb)b.b.A$

$S2 = b.b.b.b(bb)b.b.bA$

2 occurs b.b b.b bbA

$S3 = b.b\ b.b\ b.b(bb)$

3 occurences

$A = (bb)^*.(bb)$

Case I

$A = (bb)$

RE

$S0 = bb$

$1.(bb)$

$S = baS \mid a(bb)^*(bb)$

term var terminals

RE

$S = (ba)^* a (bb)^* (bb)$

$(F^*)^R = (F^R)^*$

baS ✓

$ba\ baS$ ✓

$baba\ baS$ ✓

③ RRE

$S = ((ba)^* a (bb)^* (bb))^R$

$(F.G)^R = G^R.F^R$

$= (bb)^R.((bb)^*)^R.a^R.((ba)^*)^R$

$= (bb).(bb)^*.a.(ab)^*$

RRE

Conversion of Left Linear Grammar to Right Linear Grammar

Consider the following Left Linear Grammar:-

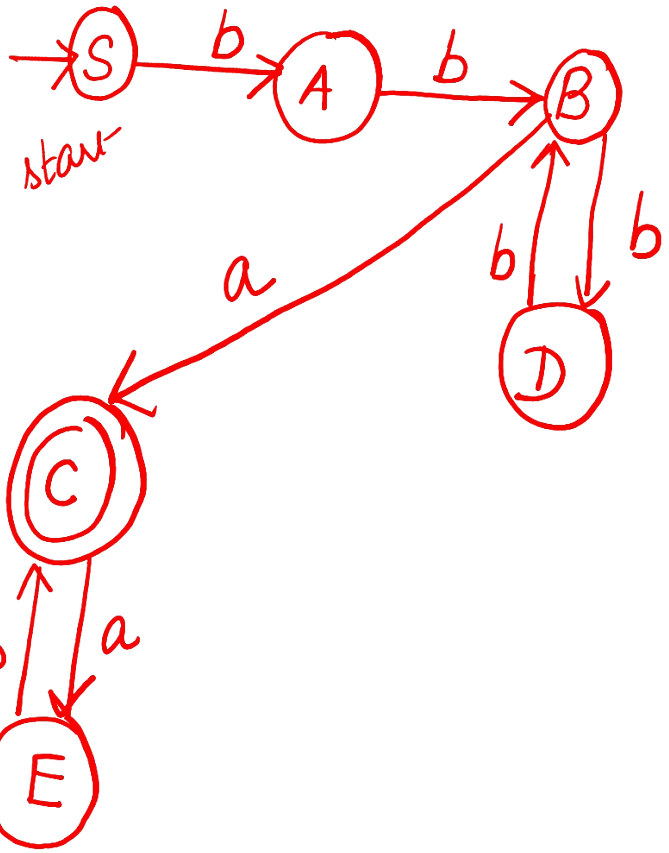
S->Sab|Aa

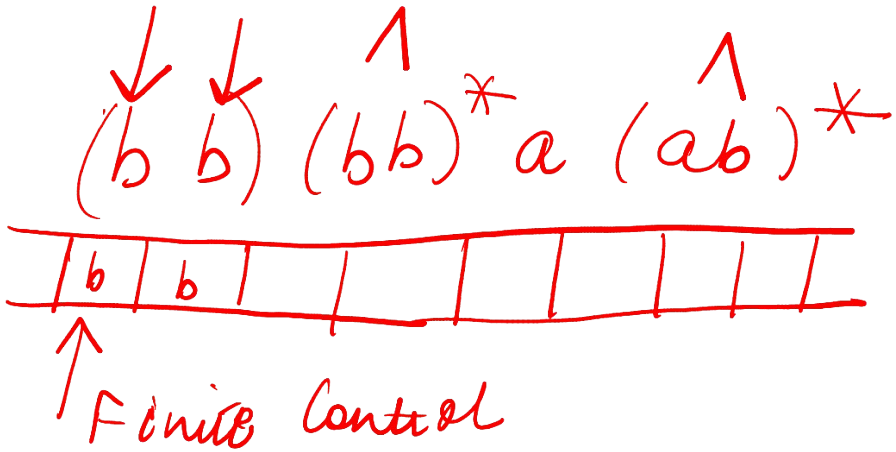
A->Abb|bb

RRE ⇒ $(bb)(bb)^*a(ab)^*$

(2) FA (draw) & derive Grammar

bba ⇒ final stat





Conversion of Left Linear Grammar to Right Linear Grammar

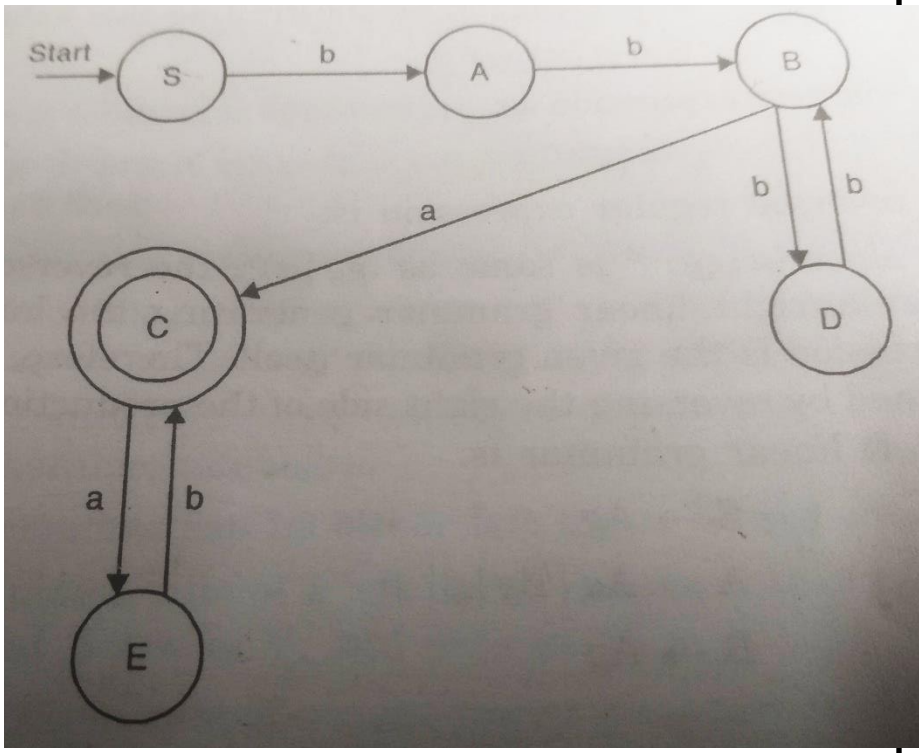
Consider the following Left Linear Grammar:-
S->Sab|Aa
A->Abb|bb
Reversing the right side of the productions gives us:
S->baS|aA
A->bbA|bb

Obtain the Regular expression specifying the language –
S=baS|aA
A=bbA|bb

We get A=(bb)*bb
S=baS|(a(bb)*bb)
S=(ba)*(a(bb)*bb)
Reversed RE=(bb (bb)*a)(ab)*

Conversion of Left Linear Grammar to Right Linear Grammar

RE=(bb (bb)*a)(ab)*
FA for the Reversed RE



Regular Grammar for the FA
S->bA
A->bB
B->cA|bD|a
D->bB
C->aE
E->bC|b

B→aC/bD/a
B→aaE/bD/a
E→baE/b
baE

Conversion of Left Linear Grammar to Right Linear Grammar

Regular Grammar for the FA
S->bA
A->bB
B->cA|bD|a
D->bB
C->aE
E->bC|b
can be reduced to:
S->bbB
B->aaE|bbB|a
E->baE|b
Required Right Linear Grammar

Conversion of Left Linear Grammar to Right Linear Grammar

Regular Grammar for the FA

S->bA

A->bB

B->cA|bD|a

D->bB

C->aE

E->bC|b

can be reduced to:

S->bbB

B->aaE|bbB|a

E->baE|b

Required Right Linear Grammar

Conversion of Left Linear Grammar to Right Linear Grammar Exercise

Consider the following grammar:
S->gA
A->aA|gB|g
B->gA
Obtain an equivalent left linear grammar

Obtain RE for above grammar:-
S=gA
A=aA|gB|g
B=gA

Substituting the value of B in A
S=gA
A=aA|ggA|g
A=(a|gg)A|g
A=(a|gg)*g
S=g(a|gg)*g
Regular Expression=g(a|gg)*g

Conversion of Left Linear Grammar to Right Linear Grammar Exercise

Regular Expression=g(a|gg)*g
Reversed Regular Expression=g(gg|a)*g
But since (a|gg)* is same as (gg|a)*
The reversed regular expression is same
Hence the regular/right linear grammar generated by the reverse RE is the given grammar itself
An equivalent left linear grammar can be obtained by reversing the right side of the production of the given grammar
Thus, Equivalent left linear grammar is
S->Ag
A->Aa|Bg|g
B->Ag