4

Context Free Grammar

Objectives

- Concept of regular grammar.
- Conversion of regular grammar to FA.
- Concept of context free grammar.
- Derivation trees.
- Ambiguity.
- Simplification of CFG.

4.1 Introduction

(In previous chapter) we have learnt the effective use of regular expressions for defining the regular languages. In fact we have learnt three ways of characterising regular languages : i) Regular expressions ii) Finite automata and iii) Construction of natural languages using simple operations. There is yet another way of expressing the regular languages; that is by something called grammar. The grammar is basically the set of rules used to define the language. These rules can be rewritten which are used to generate the desired string. For example, consider the language represented by 1^* which can be represented by a set {1, 11, 111, ...}. To generate the strings of this language we will use one simple method : Let S be a symbol to start the process with. Now we can write a rule with symbol S as $S \rightarrow 1$ and $S \rightarrow 1S$. These rules mean that S is rewritten as 1 or 1S. Now to generate a string 111, start with S then replace S by 1S, then again apply the second rule to replace S by 1S which gives 11S. Now apply first rule $S \rightarrow 1$ to replace S by 1. Ultimately we will get the string 111. Thus grammar is an effective way to represent the regular language.

The use of context free grammar is in compilation of the program. That is, when we compile the program we scan it completely and the syntax of programming constructs is checked. For checking the syntax of the program certain rules should be defined. And this task can be carried out by context free grammar or CFG. In this chapter we will understand the concept of CFG, then we will discuss the derivation of grammar and ambiguity in grammars and languages.

4.2 Regular Grammar

A regular grammar is defined as

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

V is set of symbols called non terminals which are used to define the rules.

T is a set of symbols called terminals.

P is a set of production rules.

S is a start symbol which ∈ v.

The production rules P are of the form.

$$A \rightarrow aB$$

$$A \rightarrow a$$

Where A and B are non terminal symbols, and a is terminal symbol.

For example,

Consider G = (V, T, P, S) with

V = {S, A}

T = {0, 1}

S is a start symbol and production rules are as given below.

$$S \rightarrow 0S$$

$$S \rightarrow 1B$$

$$B \rightarrow \varepsilon$$

This is called a regular language and it can be represented by some DFA. Thus, when a grammar can be represented by some finite automata then it, is a regular grammar.

4.2.1 Construction of Regular Grammar from Regular Expression

We can convert the regular expression into its equivalent regular grammar by using following method.

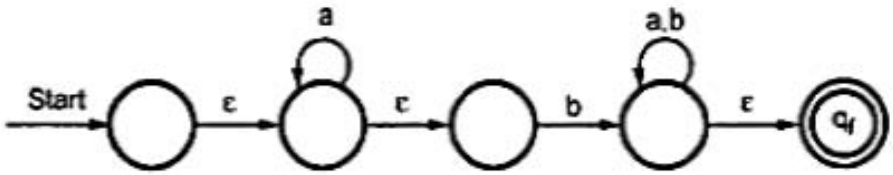
1. Construct a NFA with ε from given regular expression.
2. Eliminate ε transitions and convert it to equivalent DFA.
3. From constructed DFA, the corresponding states become non terminal symbols and transitions made are equivalent to production rules.

We will understand this method with the help of some examples.

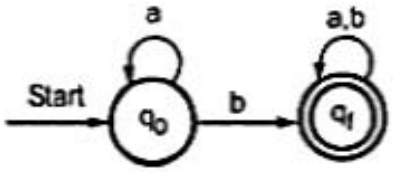
Example 4.1 : *Construct a regular grammar for the regular expression* $a^*b(a+b)^*$.

Solution : We will first construct a DFA in a straightforward manner for given regular expression.

Now we will convert it to NFA with ε transitions.



Now eliminate ε moves,



We can write the regular grammar as

$G = (V, T, P, A_0)$ where

$V = \{A_0, A_1\}$

$T = \{a, b\}$

$P = \{A_0 \rightarrow a A_0$

$A_0 \rightarrow b A_1$

$A_0 \rightarrow b$

$A_1 \rightarrow a A_1$

$A_1 \rightarrow b A_1$

$A_1 \rightarrow a$

$A_1 \rightarrow b$

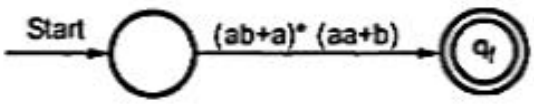
$\}$

and A_0 is a start symbol.

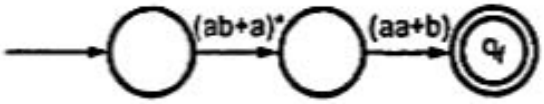
Thus G is required regular grammar.

➡ **Example 4.2 :** *Construct a regular grammar for* $(ab + a)^* (aa + b)$.

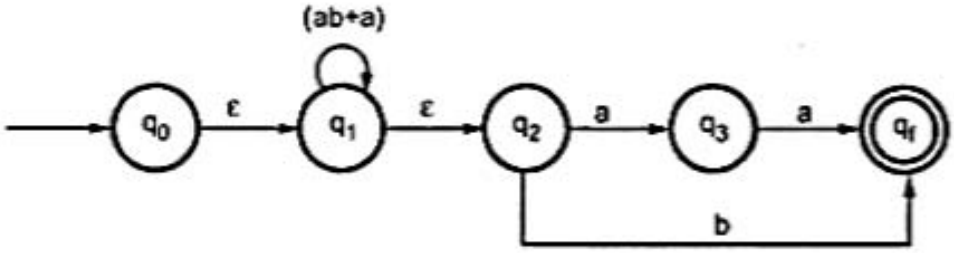
Solution : We will first construct FA for given r.e.



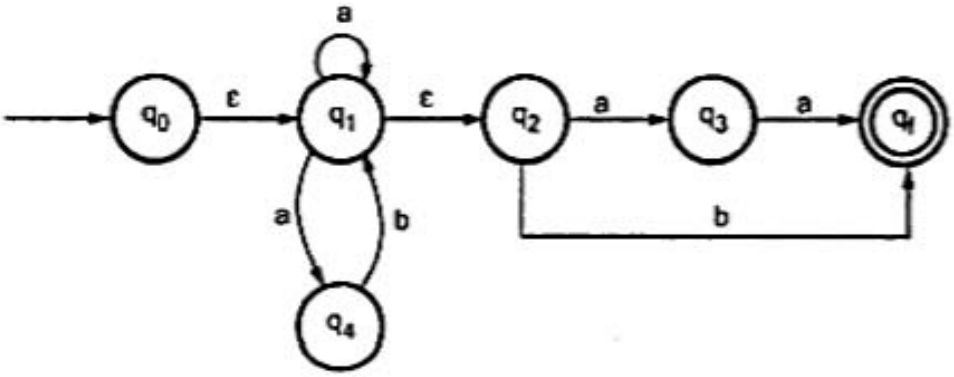
i.e.



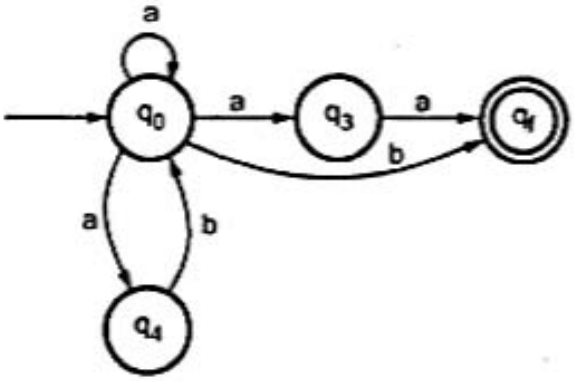
This FA can be constructed using ε transitions as



i.e.



If we eliminate ε transitions then the NFA will be -



The transition table can be -

State \ Input	a	b
q_0	$\{q_0, q_3, q_4\}$	q_f
q_3	q_f	ϕ
q_4	ϕ	q_0
q_f	ϕ	ϕ

We can convert this NFA to equivalent DFA as -

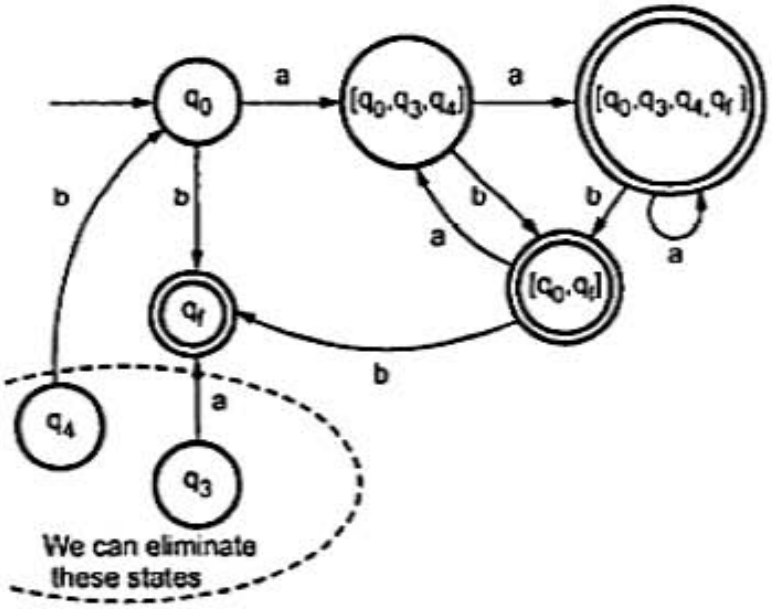
$$\delta'(q_0, a) = [q_0, q_3, q_4]$$
$$= [q_0, q_3, q_4] \rightarrow \text{new state generated}$$
$$\delta'(q_0, b) = \delta(q_0, b)$$
$$= [q_f]$$

We will find input transitions on $[q_0, q_3, q_4]$

$$\delta'([q_0, q_3, q_4], a) = \delta(q_0, a) \cup \delta(q_3, a) \cup \delta(q_4, a)$$
$$= \{q_0, q_3, q_4\} \cup \{q_f\} \cup \phi$$
$$= \{q_0, q_3, q_4, q_f\}$$
$$= [q_0, q_3, q_4, q_f] \rightarrow \text{new state}$$
$$\delta'([q_0, q_3, q_4], b) = \delta(q_0, b) \cup \delta(q_3, b) \cup \delta(q_4, b)$$
$$= \{q_f\} \cup \phi \cup \{q_0\}$$
$$= \{q_0, q_f\}$$
$$= [q_0, q_f] \rightarrow \text{new state}$$
$$\therefore \delta'([q_0, q_3, q_4, q_f], a) = \delta(q_0, a) \cup \delta(q_3, a) \cup \delta(q_4, a) \cup \delta(q_f, a)$$
$$= \{q_0, q_3, q_4\} \cup \{q_f\} \cup \phi \cup \phi$$
$$= \{q_0, q_3, q_4, q_f\}$$
$$= [q_0, q_3, q_4, q_f]$$
$$\delta'([q_0, q_3, q_4, q_f], b) = \delta(q_0, b) \cup \delta(q_3, b) \cup \delta(q_4, b) \cup \delta(q_f, b)$$
$$= \{q_f\} \cup \phi \cup \{q_0\} \cup \phi$$
$$= \{q_0, q_f\}$$
$$= [q_0, q_f]$$
$$\therefore \quad \delta'([q_0, q_f], a) = \delta(q_0, a) \cup \delta(q_f, a)$$
$$= \{q_0, q_3, q_4\} \cup \phi$$
$$= [q_0, q_3, q_4]$$
$$\delta'([q_0, q_f], b) = \delta(q_0, b) \cup \delta(q_f, b)$$
$$= \{q_f\} \cup \phi$$
$$= [q_f]$$

The transition table for this DFA will be -

Input / State	a	b
→ $[q_0]$	$[q_0, q_3, q_4]$	$[q_f]$
$[q_3]$	$[q_f]$	ϕ
$[q_4]$	ϕ	$[q_0]$
$([q_f])$	ϕ	ϕ
$[q_0, q_3, q_4]$	$[q_0, q_3, q_4, q_f]$	$[q_0, q_f]$
$([q_0, q_3, q_4, q_f])$	$[q_0, q_3, q_4, q_f]$	$[q_0, q_f]$
$([q_0, q_f])$	$[q_0, q_3, q_4]$	$[q_f]$



Now we can rename the states as :

$[q_0] = S$

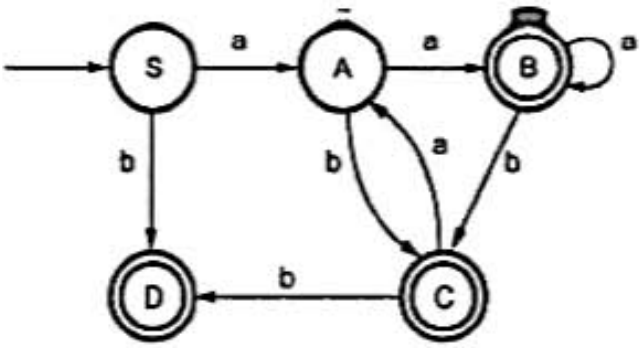
$[q_0, q_3, q_4] = A$

$[q_0, q_3, q_3, q_f] = B$

$[q_0, q_f] = C$

$[q_f] = D$

The DFA with renamed states will be -



The regular grammar can be

$S \rightarrow aA$

$S \rightarrow bD$

$S \rightarrow b$

$A \rightarrow aB$

$A \rightarrow a$

$A \rightarrow bC$

$A \rightarrow b$

$B \rightarrow aB$

$B \rightarrow a$

$B \rightarrow bC$

$B \rightarrow b$

$C \rightarrow aA$

$C \rightarrow a$

$C \rightarrow bD$

$C \rightarrow b$

$D \rightarrow \varepsilon$

4.2.2 Right-linear and Left-linear Grammar

If the non terminal symbol appears as a rightmost symbol in each production of regular grammar then it is called right linear grammar. The right linear grammar is of following form

$A \rightarrow aB$

$$A \rightarrow a$$

$$A \rightarrow \varepsilon$$

Where A and B are non terminal symbols and 'a' is a terminal symbol.

If the non terminal symbol appears as a leftmost symbol in each production of regular grammar then it is called left linear grammar.

The left linear grammar is of following form.

$$A \rightarrow Ba$$

$$A \rightarrow a$$

$$A \rightarrow \varepsilon$$

Where A and B are non terminal symbols and 'a' is a terminal symbol.

The language is called regular if it is accepted by either left linear or right linear grammar.

4.3 Equivalence between Regular Grammar and FA

Let $M = (\{q_0, q_1, ..., q_n\}\ \Sigma, \delta, q_0, F)$ be a DFA. The equivalent grammar G can be constructed from this DFA such that productions should correspond to transitions. The derivations can be terminated by a production rule giving terminals. For such production rule, the transitions terminating at some final state is encountered.

Let,

$$G = (\{A_0, A_1, ... A_n\}, \Sigma, P, A_0)$$

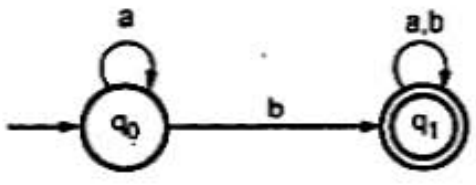
Where P the set of production rules can be defined by following rules.

1. $A_i \rightarrow a A_j$ is a production rule if $\delta(q_i, a) = q_j$, where $q_j \notin F$
2. $A_i \rightarrow a A_j$ and $A_i \rightarrow a$ are production rules if $\delta(q_i, a) = a_j$ where $q_j \in F$.

Thus the given grammar is accepted by DFA M.

Problems based on FA to regular grammar

➡ **Example 4.3 :** *Construct regular grammar for given DFA.*



Solution : The grammar equivalent to given DFA will be -

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

$$V = \{A_0, A_1\}$$

$$T = \{a, b\}$$

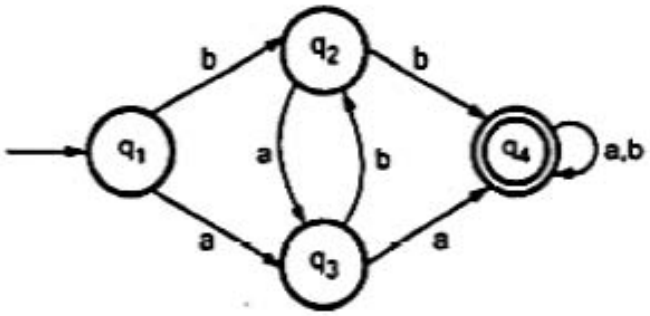
$$A_0 \rightarrow a A_0 \quad A_1 \rightarrow a A_1 | a$$

$$A_0 \rightarrow b A_1 \quad A_1 \rightarrow b A_1 | b$$

$$A_0 \rightarrow b$$

Thus the two states correspond to two non terminals. The state q_1 is final state. Hence the transition terminating to final state q_1 are giving the terminal symbols. Note that we are considering outgoing edges for production rule.

Example 4.4 : *Construct a regular grammar for given FA.*



Solution : Let $G = (\{A_1, A_2, A_3, A_4\}, \{a, b\}, P, A_1)$

The regular grammar can be

$$A_1 \rightarrow b A_2$$

$$A_1 \rightarrow a A_3$$

$$A_2 \rightarrow b A_4$$

$$A_2 \rightarrow b$$

$$A_2 \rightarrow a A_3$$

$$A_3 \rightarrow b A_2$$

$$A_3 \rightarrow a A_4$$

$$A_3 \rightarrow a$$

$$A_4 \rightarrow a A_4$$

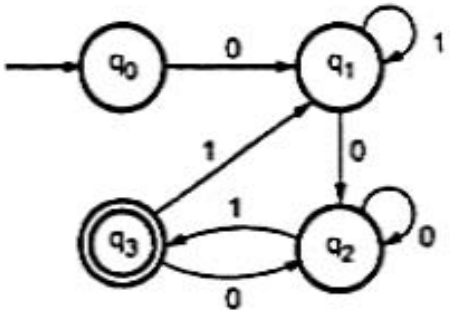
$A_4 \rightarrow a$

$A_4 \rightarrow bA_4$

$A_4 \rightarrow b$

As state q_4 is a final state the corresponding non terminal A_4 is used. Now all the edges leading to q_4 gives terminal productions as well.

➠ **Example 4.5 :** *Construct a regular grammar for*



Solution : The equivalent regular grammar can be denoted by G = (V, T, P, S) where

$$V = \{A_0, A_1, A_2, A_3\}$$

The production rules can be

$A_0 \rightarrow 0A_1$

$A_1 \rightarrow 1A_1$

$A_1 \rightarrow 0A_2$

$A_2 \rightarrow 0A_2$

$A_2 \rightarrow 1A_3$

$A_2 \rightarrow 1$

$A_3 \rightarrow 1A_1$

$A_3 \rightarrow 0A_2$

4.3.1 Construction of FA from Regular Grammar

Let $G = (V, \Sigma, P, A_0)$ be a regular grammar. We can construct DFA M whose

i) States correspond to variables.

ii) Initial state correspond to A_0.

iii) Transitions in M correspond to productions in P.

If there is a production of the form $A_i \rightarrow a$, the corresponding transition terminates at a new state. This is the unique final state. Thus the DFA M can be

$$M = (\{q_0, q_1, \ldots, q_n, q_f\}, \Sigma, \delta, q_0, \{q_f\})$$

The δ is defined as

i) Each production $A_i \rightarrow a A_j$ induces a transition from q_i to q_j with label a.

ii) Each production $A_k \rightarrow a$ induces a transition from q_k to q_f with label a.

Therefore L(G) can be given by corresponding FA M.

Problems based on regular grammar to FA

Example 4.6 : *Construct a finite automata recognizing L(G) where G is the grammar*

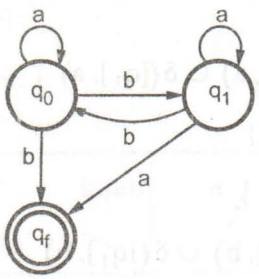
$S \rightarrow aS \mid bA \mid b$

$A \rightarrow aA \mid bS \mid a$

Solution : We can have the FA M as

$$M = (\{q_0, q_1, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$$

Where q_0 and q_1 correspond to S and A and q_f is a final state.



This is basically NFA because from q_0 with b there are two next states q_1 and q_f.

Example 4.7 : *Construct a deterministic finite automaton equivalent to the grammar*

$S \rightarrow aS \mid bS \mid aA$

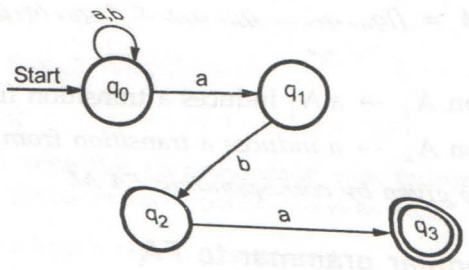
$A \rightarrow bB$

$B \rightarrow aC$

$C \rightarrow A$

Solution : The DFA is denoted by,

$$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, q_3)$$



Now this NFA can be converted to equivalent DFA

$$\delta'([q_0], a) = \{q_0, q_1\} = [q_0, q_1]$$

$$\delta'([q_0], b) = \{q_0\} = [q_0]$$

$$\delta'([q_1], a) = \phi$$

$$\delta'([q_1], b) = \{q_2\} = [q_2]$$

$$\delta'([q_2], a) = \{q_3\} = [q_3]$$

$$\delta'([q_2], b) = \phi$$

$$\delta'([q_0, q_1], a) = \delta([q_0], a) \cup \delta([q_1], a)$$
$$= \{q_0, q_1\}$$
$$= [q_0, q_1]$$

$$\delta'([q_0, q_1], b) = \delta([q_0], b) \cup \delta([q_1], b)$$
$$= \{q_0, q_2\}$$

$$\delta'([q_0, q_2], b) = \delta([q_0], a) \cup \delta([q_2], a)$$
$$= [q_0, q_1] \cup [q_3]$$
$$= [q_0, q_1, q_3]$$

$$\delta'([q_0, q_2], b) = \delta(q_0, b) \cup \delta(q_2, b)$$
$$= \{q_0\} \cup \phi$$
$$= [q_0]$$

$$\delta'([q_0, q_1, q_3], a) = \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_3, a)$$
$$= \{q_0, q_1\} \cup \phi \cup \phi$$

$= \{q_0, q_1\} = [q_0, q_1]$

$\delta'([q_0, q_1, q_3], b) = \delta\{q_0, b\} \cup \delta(q_1, b) \cup \delta(q_3, b)$

$= \{q_0, q_1\} \cup \{q_2\} \cup \phi$

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

$= [q_0, q_1, q_2]$

$\delta'([q_0, q_1, q_2], a) = \delta([q_0], a) \cup \delta([q_1], a) \cup \delta([q_2], a)$

$= \{q_0, q_1\} \cup \phi \cup \{q_3\}$

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

$= [q_0, q_1, q_3]$

$\delta'([q_0, q_1, q_2], b) = \delta([q_0], b) \cup \delta([q_1], b) \cup \delta([q_2], b)$

$= \{q_0\} \cup \{q_2\} \cup \phi$

$= \{q_0, q_2\}$

$= [q_0, q_2]$

The transition table will be -

	Input / State	a	b
→	$[q_0]$	$[q_0, q_1]$	$[q_0, q_2]$
	$[q_1]$	ϕ	$[q_2]$
	$[q_2]$	$[q_3]$	ϕ
*	$[q_3]$	ϕ	ϕ
	$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
	$[q_0, q_2]$	$[q_0, q_1, q_3]$	$[q_0]$
*	$[q_0, q_1, q_3]$	$[q_0, q_1]$	$[q_0, q_1, q_2]$
	$[q_0, q_1, q_2]$	$[q_0, q_1, q_3]$	$[q_0, q_2]$

The final states are marked by *.

4.4 Context Free Grammar

The context free grammar can be formally defined as a set denoted by G = (V, T, P, S) where V and T are set of non terminals and terminals respectively. P is set of production rules, where each production rule is in the form of

non terminal → non terminals

or non terminal → terminals

S is a start symbol.

For example,

$$P = \{ S \rightarrow S + S$$
$$S \rightarrow S * S$$
$$S \rightarrow (S)$$
$$S \rightarrow 4 \}$$

If the language is 4 + 4 * 4 then we can use the production rules given by P. The start symbol is S. The number of non terminals in the rules P is one and the only non terminal i.e. S. The terminals are +, *, (,) and 4.

We are using following conventions.

1. The capital letters are used to denote the non terminals.
2. The lower case letters are used to denote the terminals.

Example : The formation of production rules for checking syntax of any English statement is

SENTENCE → NOUN VERB

NOUN → Rama / Seeta / Gopal

VERB → goes / writes / sings

Thus if want to derive a string "Rama sings" then we can follow the above rules.

4.5 Derivation and Languages

The production rules are used to derive certain strings. We will now formally define the language generated by grammar G = (V, T, P, S). The generation of language using specific rules is called derivation.

Definition :

Let G = (V, T, P, S) be the context free grammar. If $A \rightarrow \beta$ is a production of P and α and γ are any strings from non terminals or terminals i.e. in (VUT)* then $\alpha A \gamma \rightarrow \alpha \beta \gamma$.

Suppose $a_1, a_2, a_3, \ldots a_m$ are strings in (VUT)*, m ≥ 1.

$$a_1 \Rightarrow a_2$$

$$a_2 \Rightarrow a_3$$

$$\vdots$$

$$a_{m-1} \Rightarrow a_m$$

Then we can say that $a_1 \overset{*}{\underset{G}{\Rightarrow}} a_m$

The language generated by G is denoted by L(G). The language L is called **context free language** CFL if L(G) is for CFG.

For example

G = ({S, B} {*a*, b}, {S → *a* B b, S)

B → bbb }

is a grammar. Then we can derive a string *a*bbbb as -

i) We will first start from start symbol S

S ⇒ *a* B b

ii) Then we will replace B by bbb

S ⇒ *a* B b ⇒ *a* bbbb

Thus we can obtain the desired language L by certain rules. The language L can be described as a language which starts with letter *a* and having 4 b's following. Let us solve some examples for derivation of CFG.

Problems on context free grammar

Example 4.8 : *Construct the CFG for the language having any number of a's over the set* $\Sigma = \{a\}$.

Solution : As we know the regular expression for above mentioned language is

r.e. = a^*

Let us build the production rules for the same,

S → *a* S rule 1

S → ε rule 1

Now if want "*aaaaa*" string to be derived we can start with start symbols.

S

a S

a a S ∵ rule 1

a a a S rule 1

a a a a S rule 1

a a a a a S rule 1

a a a a a ε rule 2

= *a a a a a*

The r.e. = a^* suggest a set of {ε, *a*, *aa*, *aaa*, ...}. We can have a null string simply because S is a start symbol, and rule 2 gives S → ε.

Example 4.9 : *Try to recognize the language L for given CFG.*

$$G = [\{S\}, \{a,b\}, P, \{S\}]$$

$$\text{where } P = \begin{Bmatrix} S \to aSb \\ S \to ab \end{Bmatrix}$$

Solution : Since S → *a* S b | *a* b is a rule. | indicates the 'or' operator.

S → *a* S b

If this rule can be recursively applied then,

S

a S b

↓

a a S b b

↓

a a a S b b b

and if finally we can put S → *ab* then it becomes *aaaa* bbbb. Thus we can have any number of *a*'s first then equal number of b's following it. Hence we can guess the language as $\{L = a^n b^n \text{ where } n \geq 1\}$. The only way to recognize the language is to try out various strings from the given production rules. Simply by observing the derived strings, one can find out the language getting generated from given CFG.

Example 4.10 : *Construct the CFG for the regular expression* (0+1)*.

Solution : The CFG can be given by,

P = {S → 0S | 1S

S → ε }

The rules are in combination of 0's and 1's with the start symbol. Since $(0+1)^*$ indicates {ε, 0, 1, 01, 10, 00, 11, ...} in this set ε is a string. So in the rules we can set the rule $S \rightarrow \varepsilon$.

Example 4.11 : *Construct a grammar for the language containing strings of atleast two a's.*

Solution : Let G = (V, T, P, S)

where V = {S, A}

T = {*a*, b}

P = {S → A*a* A*a* A

A → *a*A | bA | ε }

The rule S → A*a*A*a*A is something in which the two *a*'s are maintained since at least two *a*'s should be there in the strings. And A → *a*A |bA|ε gives any combination of a's and b's i.e. this rules gives the strings of (*a* + b)*.

Thus the logic for this example will be

(any thing) *a* (any thing) *a* (any thing)

So before *a* or after *a* there could be any combination of *a*'s and b's.

Example 4.12 : *Construct a grammar generating*

$L = wcw^T$ *where* $w \in \{a, b\}^*$

Solution : The strings which can be generated for given L is {*aacaa*, bcb, *abcba*, b*aca*b ...}

The grammar could be

S → *a* S *a*

S → b S b

S → c

Since the language L = w c w^T where w ∈ (*a* + b)*

Hence S → *a* S *a* or S → b S b. The string αbcbα can be generated from given production rules as

S

a S *a*

a b S b

a b c b *a*

a	b	c	b	*a*

a	b	c	b	*a*

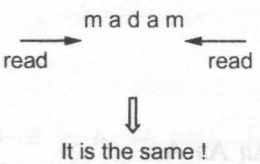
a	b	c	b	*a*

Thus any of this kind of string could be derived from the given production rules.

Example 4.13 : *Construct CFG for the language L which has all the strings which are all palindrome over* $\Sigma = \{a, b\}$.

Solution : As we know the strings are palindrome if they posses same alphabets from forward as well as from backward.

For example, the string "madam" is a palindrome because



Since the language L is over Σ = {*a*, b}. We want the production rules to be build *a*'s and b's. As ε can be the palindrome, *a* can be palindrome even b can be palindrome. So we can write the production rules as

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

P can be

$S \rightarrow a\ S\ a$

$S \rightarrow b\ S\ b$

$S \rightarrow a$

$S \rightarrow b$

$S \rightarrow \varepsilon$

The string abaaba can be derived as

S

$a\ S\ a$ $\quad a\ b\ a\ a\ b\ a$

$a\ b\ S\ b\ a$ $\quad a\ b\ a\ a\ b\ a$

$a\ b\ a\ S\ a\ b\ a$ $\quad a\ b\ a\ a\ b\ a$

$a\ b\ a\ \varepsilon\ a\ b\ a$ $\quad a\ b\ a\ a\ b\ a$

$a\ b\ a\ a\ b\ a$ $\quad a\ b\ a\ a\ b\ a$

which is a palindrome.

Example 4.14 : *Construct CFG which consists of all the strings having atleast one occurrence of 000.*

Solution : The CFG for this language can be equivalent to r.e. (any thing) (000) (anything)

Thus r.e. = $(0+1)^*\ 000\ (0+1)^*$

Let us build the production rules as

$$S \rightarrow ATA$$

$$A \rightarrow 0A \mid 1A \mid \varepsilon$$

$$T \rightarrow 000$$

Example 4.15 : *Construct CFG for the language in which there are no consecutive b's, the strings may or may not have consecutive a's.*

Solution : $S \rightarrow a\,S \mid bA \mid a \mid b \mid \varepsilon$

$A \rightarrow a\,S \mid a \mid \varepsilon$

In the above rules, there is no condition on occurence of *a*'s. But no consecutive b's are allowed. Note that in the rule $S \rightarrow bA$, and A gives all the strings which are starting with letter *a*.

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

Let us derive the string

Derivation	Input	Productions
S		
a S	*a a* b *a* b	S → *a* S
a a S	*a a* b *a* b	S → *a* S
a a b A	*a a* b *a* b	S → b A
a a b *a* S	*a a* b *a* b	A → *a* S
a a b *a* b	*a a* b *a* b	S → b

Example 4.16 : *Recognize the context free language for the given CFG.*

$$S \rightarrow a\,B \mid b\,A$$

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

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

Solution : To find the language denoted by given CFG we try to derive the rules and get various strings. Then after observing those strings we come to know, which language it is denoting. Let us rewrite all those rules and number them

$S \rightarrow a\,B$ rule 1

$S \rightarrow b\,A$ rule 2

$A \rightarrow a$ rule 3

A → *a* S	rule 4
A → b A A	rule 5
B → b	rule 6
B → b S	rule 7
B → *a* B B	rule 8

Now let us apply the rules randomly starting with start symbol.

S	
a B	rule 1
a a B B	rule 8
a a b S B	rule 7
a a b b A B	rule 2
a a b b *a* B	rule 3
a a b b *a* b S	rule 7
a a b b *a* b b A	rule 2
a a b b *a* b b *a*	rule 3

We have got some string as "*aa*bb*a*bb*a*". Let us try out something else.

S	
b A	rule 2
b b A A	rule 5
b b *a* S A	rule 4
b b *a a* B A	rule 1
b b *a a* b A	rule 6
b b *a a* b *a*	rule 3

Now we have got "bb*aa*b*a*" such a string !

Thus by obtaining more and more strings by applying more and more rules randomly will help us to find out what language it indicates. Observe the strings generated carefully, we can draw a conclusion as these strings contain equal number of *a*'s and equal number of b's. Even you can try out various rules to obtain some more strings. And see that it is a language L containing all the strings having equal number of *a*'s and b's.

Example 4.17 : *Construct CFG for the language containing atleast one occurrence of double a.*

Solution : The CFG can be built with a logic as : one rule we will built for double *a* i.e.

$A \rightarrow aa$

And the other rule we will built for any number of *a*'s and b's in any combination.

i.e. $B \rightarrow aB \mid bB \mid \varepsilon$ which is always equivalent to $(a + b)^*$ {you can note it as golden rule !}

If we combine both of these rules we can get the desired context free grammar.

$$S \rightarrow BAB \Rightarrow (\text{anything}) \begin{pmatrix} \text{one occurrence} \\ \text{of double } a \end{pmatrix} (\text{anything})$$

$A \rightarrow aa$

$B \rightarrow aB \mid bB \mid \varepsilon$

Thus the CFG contains V = {A, B, S} T = {*a*, b}. The start symbol is S.

Let us derive "*abaa*b"

S		
BAB		
*a*BAB	*abaa*b	$B \rightarrow a\ B$
*ab*BAB	*abaa*b	$B \rightarrow bB$
*ab*AB	*abaa*b	$B \rightarrow \varepsilon$
*abaa*B	*abaa*b	$A \rightarrow a\,a$
*abaa*bB	*abaab*	$B \rightarrow bB$
*abaab*ε	*abaab*	$B \rightarrow \varepsilon$
= *abaab*	*abaab*	

Example 4.18 : *Construct CFG for the language containing all the strings of different first and last symbols over* $\Sigma = \{0, 1\}$.

Solution : Since the problem statement says as if the string starts with 0 it should end with 1 or if the string starts with 1 it should end with 0.

$$S \rightarrow 0\ A\ 1 \mid 1\ A\ 0$$

$$A \rightarrow 0\ A \mid 1\ A \mid \varepsilon$$

Thus clearly, in the above CFG different start and end symbols are maintained. The non terminal $A \rightarrow 0A \mid 1A \mid \varepsilon$ indicates $(0 + 1)^*$. Thus the given CFG is equivalent to the regular expression $[0(0+1)^*1+1(0+1)^*0]$.

Example 4.19 : *Construct the CFG for the language $L = a^n b^{2n}$ where $n \geq 1$.*

Solution : As in some previous example we have seen the case of $a^n b^n$. Now the number of b's are doubled. So we can write

$$S \rightarrow aSbb \mid abb$$

It is as simple as this !

Example 4.20 : *Construct the production rules for defining a language* $L = \{a^x b^y \mid x \neq y\}$.

Solution : This is the language in which all the a's must appear before all the b's. But total number of a's must not be equal to total number of b's. Either number of a's must be equal to number of b's or number of b's must be equal to number of a's. The rule

$$S \rightarrow aSb$$

gives us equal number of a's must be followed by equal number of b's. But according to problem statement we need more number of a's either or more number of b's. Hence the required production rules can be -

$$s \rightarrow aSb \mid R1 \mid R2$$

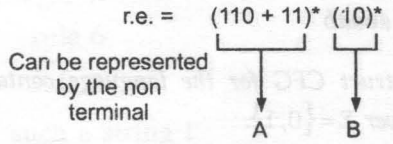
$$R1 \rightarrow aR1 \mid a$$

$$R2 \rightarrow bR2 \mid b$$

where S is a start symbol.

Example 4.21 : *Find the CFG for the regular expression (110 + 11)*(10)*.*

Solution : Let



If we assume S as start symbol then,

$$S \rightarrow AB$$

Now for each non terminal A and B we can define production rules as

$A \rightarrow 110A \mid 11A \mid \varepsilon$

$B \rightarrow 10B \mid \varepsilon$

Hence, finally the CFG for given r.e. can be

$S \rightarrow AB$

$A \rightarrow 110A \mid 11A \mid \varepsilon$

$B \rightarrow 10B \mid \varepsilon$

Example 4.22 : *Build a CFG for generating the integers.*

Solution : The integer is any numerical value without decimal place which can be either signed or unsigned. Hence the CFG can be -

$$S \rightarrow GI$$

$$G \rightarrow + \mid -$$

$$I \rightarrow DI \mid D$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \ldots \mid 9$$

Here G is for denoting sign, I denotes integer and D denotes digits from 0 to 9. To understand the rules lets derive these rules for some integer.

Let,- 21 can be derived as

$S \rightarrow GI$

$- I \qquad G \rightarrow -$

$- DI \qquad I \rightarrow DI$

$- 2I \qquad D \rightarrow 2$

$- 2D \qquad I \rightarrow D$

$- 21 \qquad D \rightarrow 1$

Example 4.23 : *Build a CFG for the language* $L = \{0^i\ 1^j\ 2^k \mid j > i+k\}$.

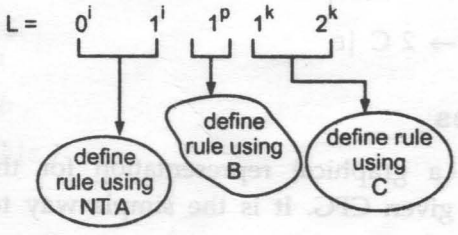
Solution : As the language $L = 0^i\ 1^j 2^k$ such that $j > i + k$.

Hence we can rewrite L as

$$L = 0^i\ 1^i\ 1^k\ 1^p\ 2^k$$

This will be greater than i+k

We can again rewrite L for simplification as -



$$A \rightarrow 0\ A\ 1 \mid \varepsilon$$

$$B \rightarrow 1\ B \mid 1$$

$$C \rightarrow 1\ C\ 2 \mid \varepsilon$$

Hence the CFG can be

$$S \rightarrow A\ B\ C$$

$$A \rightarrow 0\ A\ 1 \mid \varepsilon$$

$$B \rightarrow \mid B \mid 1$$

$$C \rightarrow 1\ C\ 2 \mid \varepsilon$$

Example 4.24 : *Build a CFG for the language* $L = \{0^i\ 1^j\ 2^k \mid i=j\}$

Solution : As i = j and there is no condition on k. We can say that k is an arbitrary value. Then we rewrite L as

$$L = \underbrace{0^i\ 1^j}_{A}\ \underbrace{2^k}_{B}$$

$\therefore$ $$S \rightarrow A\ B$$

$$A \rightarrow 0\ A\ 1 \mid \varepsilon$$

$$B \rightarrow 2\ B \mid \varepsilon$$

Example 4.25 : *Obtain CFG for the language* $L = \{0^i\ 1^j\ 2^k \mid j \leq k\}$.

Solution : Here $j \leq k$. That means the language L has two variations.

$$L_1 = 0^i\ 1^j\ 2^j$$

$$L_2 = 0^i\ 1^j\ 2^k \text{ where } k > j$$

Hence the required CFG can be

$$S \rightarrow A\ B$$

$$A \rightarrow 0\ A \mid \varepsilon$$

$$B \rightarrow 1\ B\ 2 \mid C$$

$$C \rightarrow 2\ C \mid \varepsilon$$

4.6 Derivation Trees

Derivation trees is a graphical representation for the derivation of the given production rules for a given CFG. It is the simple way to show how the derivation

can be done to obtain some string from given set of production rules. The derivation tree is also called parse tree.

Following are properties of any derivation tree -

1. The root node is always a node indicating start symbol.
2. The derivation is read from left to right.
3. The leaf nodes are always terminal nodes.
4. The interior nodes are always the non terminal nodes.

For example,

$S \rightarrow bSb \mid a \mid b$ is a production rule. The S is a start symbol.

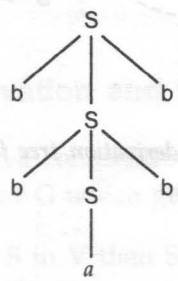
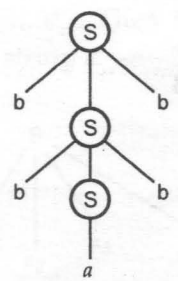


Fig. 4.1 Derivation tree

The above tree is a derivation tree drawn for deriving a string bbabb. By simply reading the leaf nodes we can obtain the desired string. The same tree can also be denoted by,



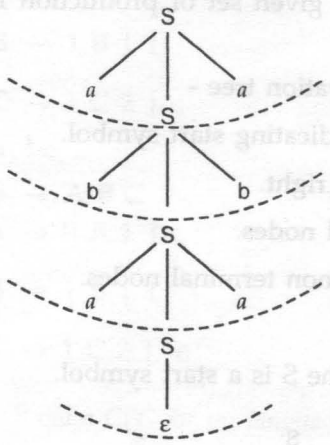
Problems on derivation tree

Example 4.26 : *Draw a derivation tree for the string abaaba for the CFG given by,*

G where $P = \{S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow a \mid b \mid \varepsilon \}$



Example 4.27 : *Construct the derivation tree for the string aabbabba from the CFG given by*

$$S \rightarrow aB \mid bA$$

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

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

Solution : To draw a tree we will first try to obtain derivation for the string *aabbabba*

S

*a*B $S \rightarrow aB$

a *a*BB $S \rightarrow aBB$

aa bS B $B \rightarrow bS$

aab bA B $S \rightarrow bA$

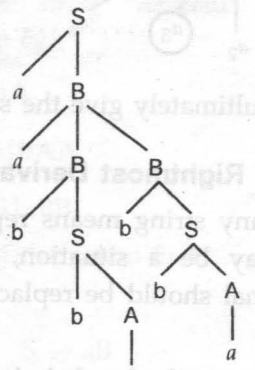
aabb *a* B $A \rightarrow a$

aabba bS $B \rightarrow bS$

aabbab bA $S \rightarrow bA$

aabbabb *a* $A \rightarrow a$

Now let us draw a tree.



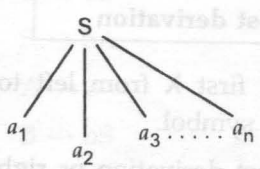
Relationship between Derivation and Derivation Tree

Theorem : Let G = (V, T, P, S) be a context free grammar. Then $S \overset{*}{\Rightarrow} a$ if and only if there is a derivation tree in grammar G which gives the string a.

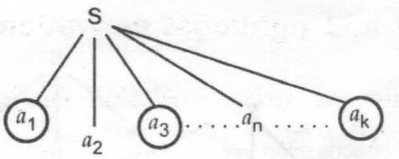
Proof : If there is a non terminal S in V then $S \overset{*}{\Rightarrow} a$ if and only if there is a S-tree which gives the string a. We can easily prove this using induction theorem. We can apply induction on number of interior nodes.

Basis of induction :

Let us assume S is the only one interior which gives the string a which can be obtained by deriving $S \rightarrow a_1, a_2, a_3, \ldots a_n$. Then there exists a derivation tree which derives $a_1, a_2, a_3 \ldots a_n$ and gives the string a.



Induction step : We assume, that the derivation will enable us to draw the tree with K-1 interior nodes. We have to prove, that it is possible to draw the derivation tree for K interior nodes which gives the string a. In the derivation tree not all the nodes are leaves or interior nodes. Some are leaves where as others are interior nodes considering for them S is a parent node.



Thus this derivation tree will ultimately give the string a.

4.7.1 Leftmost Derivation and Rightmost Derivation

As we know, derivation for any string means replacement of non terminal by its appropriate definition. There may be a situation, in which there are many non terminals. Then which non terminal should be replaced by its definition is sometimes confusing to decide.

Hence we normally apply two methods of deriving. The leftmost derivation is a derivation in which the leftmost non terminal is replaced first from the sentential form.

The rightmost derivation is a derivation in which rightmost non terminal is replaced first from the sentential form.

Let us see how it works.

For example

$S \rightarrow XYX$	$S \rightarrow XYX$
$S \rightarrow aYX$	$S \rightarrow XYa$
$S \rightarrow abX$	$S \rightarrow Xba$
$S \rightarrow aba$	$S \rightarrow aba$
Leftmost derivation	**Rightmost derivation**

Note that we have replaced first X from left to right in leftmost derivation and XYX the last X i.e. the rightmost symbol.

Actually, we may use leftmost derivation or rightmost derivation we get the same string. The type of derivation does not affect on getting of a string.

Let us solve few problems based on this topic.

Problems on leftmost and rightmost derivations

Example 4.28 : *Derive the string "aabbabba" for leftmost derivation and rightmost derivation using a CFG given by,*

$$S \rightarrow aB \mid bA$$

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

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

Solution : Let us see the leftmost derivation first,

S	
aB	$S \rightarrow aB$
aaBB	$B \rightarrow aBB$
aabB	$B \rightarrow b$
$aabb$S	$B \rightarrow bS$
$aabba$B	$S \rightarrow aB$
$aabbab$S	$B \rightarrow bS$
$aabbab$bA	$S \rightarrow bA$
$aabbabba$	$A \rightarrow a$

is derived. Now let us solve using rightmost derivation.

S	
aB	$S \rightarrow aB$
aaBB	$B \rightarrow aBB$
aaBbS	$B \rightarrow bS$
aaBbbA	$S \rightarrow bA$
aaBbb$a$	$A \rightarrow a$
aabSbb$a$	$B \rightarrow bS$
$aabb$Abb$a$	$S \rightarrow bA$
$aabbabba$	$A \rightarrow a$

is a derivation.

Example 4.29 : *Derive the string 1000111 for leftmost and rightmost derivation using CFG.*

$$G = (V, T, P, S) \text{ where}$$

$$V = \{S, T\}$$

$T = \{0, 1\}$

$P = \{ S \rightarrow T00T$

$T \rightarrow 0T \mid 1T \mid \varepsilon \}$

Solution : The leftmost derivation can be

S	
T00T	$S \rightarrow T00T$
1T00T	$T \rightarrow 1T$
10T00T	$T \rightarrow 0T$
10 ε 00T	$T \rightarrow \varepsilon$
1000T	
10001T	$T \rightarrow 1T$
100011T	$T \rightarrow 1T$
1000111T	$T \rightarrow 1T$
1000111 ε	$T \rightarrow \varepsilon$
1000111	

Right most derivation -

S	
T00T	
T001T	$T \rightarrow 1T$
T0011T	$T \rightarrow 1T$
T00111T	$T \rightarrow 1T$
T00111 ε	$T \rightarrow \varepsilon$
T00111	
1T00111	$T \rightarrow 1T$
10T00111	$T \rightarrow 0T$
10 ε 00111	$T \rightarrow \varepsilon$
1000111	

4.8 Ambiguity

The grammar can be derived in either leftmost derivation or rightmost derivation. We can draw a derivation tree called as parse tree or syntax tree based on these derivations. The parse tree has to be unique even though the derivation is leftmost or rightmost.

But if there exists more than one parse trees for a given grammar, that means there could be more than one leftmost or rightmost derivation possible and then that grammar is said to be ambiguous grammar.

For example - the CFG given by G = (V, T, P, S)

where

$$V = \{E\}$$

$$T = \{id\}$$

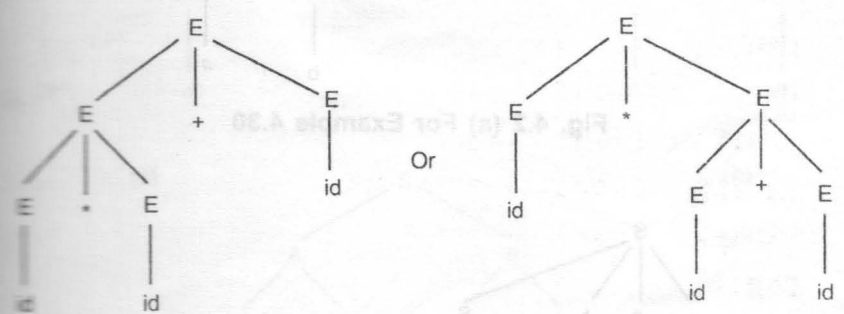
$$P = \{E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow id \}$$

$$S = \{E\}$$

Now if the string is id * id + id then we can draw the two different parse trees generating our id * id + id.



Thus the above grammar is an ambiguous grammar. Let us solve some exercise on

Problems on ambiguity of grammar

Example 4.30 : *Check whether the given grammar is ambiguous or not.*

$S \rightarrow iC\ t\ S$

$S \rightarrow iC\ t\ S\ e\ S$

$S \rightarrow a$

$C \rightarrow b$

Solution : To check whether given grammar is ambiguous or not we will have som string for derivation tree such as ibtibtibt*a*e*a*.

Let us draw the derivation tree.

ibtibtibt*a*e*a*

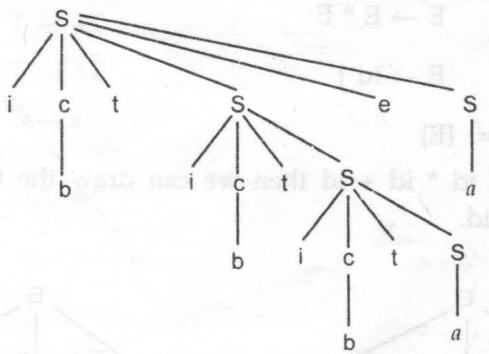


Fig. 4.2 (a) For Example 4.30

or

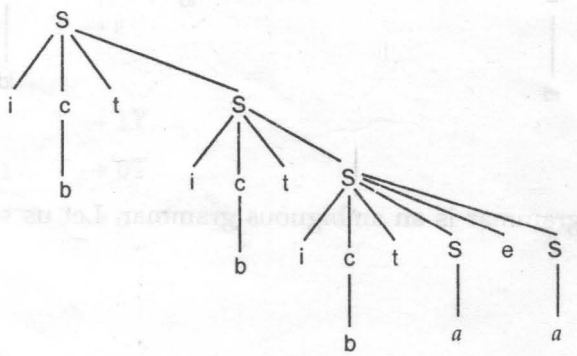


Fig. 4.2 (b) For Example 4.30

Thus we have got more than two parse trees. Hence the given grammar ambiguous.

Example 4.31 : *Consider the grammar* $G=(V, \Sigma, R, S)$.

where V = {S, A}

Σ = {*a*, b}

R = { S → AA

A → AAA

A → *a*

A → bA

A → Ab }

Show that this is an ambiguous grammar

Solution : Consider a string b*a*bb*a*b.

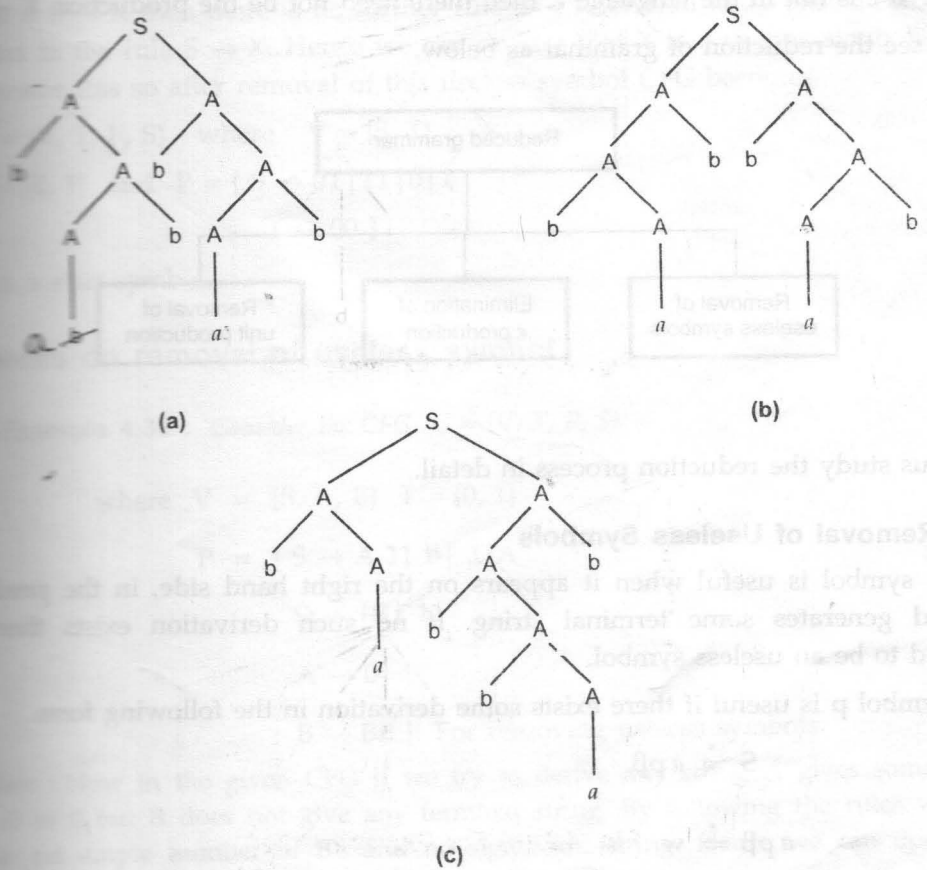


Fig. 4.3 For Example 4.31

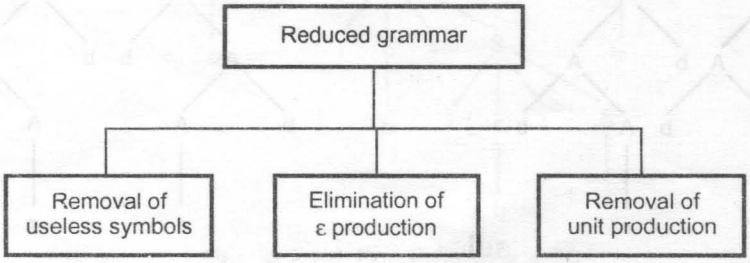
Thus there are more than one parse tree getting generated. Hence the grammar is ambiguous.

4.9 Simplification of CFG

As we have seen various languages can effectively be represented by context tree grammar. All the grammars are not always optimized. That means grammar may consists of some extra symbols (non terminals). Having extra symbols unnecessary increases the length of grammar. Simplification of grammar means reduction of grammar by removing useless symbols. The properties of reduced grammar are given below.

1. Each variable (i.e. non terminal) and each terminal of G appears in the derivation of some word in L.
2. There should not be any production as $X \rightarrow Y$ where X and Y are non terminals.
3. If ε is not in the language L then there need not be the production $X \rightarrow \varepsilon$.

We see the reduction of grammar as below.



Let us study the reduction process in detail.

4.9.1 Removal of Useless Symbols

Any symbol is useful when it appears on the right hand side, in the production rule and generates some terminal string. If no such derivation exists then it is supposed to be an useless symbol.

A symbol p is useful if there exists some derivation in the following form.

$$S \overset{*}{\Rightarrow} \alpha p \beta$$

and $$\alpha p \beta \overset{*}{\Rightarrow} w$$

Then p is said to be useful symbol.

Where α and β may be some terminal or non terminal symbol and will help us to derive certain string w in combination with P. Let us see what exactly means the useless symbol with some example.

For example, G = (V, T, P, S) where V = {S, T, X}, T = {0, 1}

$S \rightarrow 0T|1T|X|0|1$ rule 1

$T \rightarrow 00$ rule 2

Now, in the above CFG, the non terminals are S, T and X.

To derive some string we have to start from start symbol S.

S

0T $S \rightarrow 0T$

000 $T \rightarrow 00$

Thus we can reach to certain string after following these rules.

But if $S \rightarrow X$ then there is no further rule as a definition to X. That means there is no point in the rule $S \rightarrow X$. Hence we can declare that X is a useless symbol. And we can remove this so after removal of this useless symbol CFG becomes

G = (V, T, P, S) where V = {S, T}

T = {0, 1} and P = { $S \rightarrow 0T|1T|0|1$

$T \rightarrow 00$ }

S is a start symbol.

Problems on removal of useless symbol

Example 4.32 : *Consider the CFG G = (V, T, P, S)*

where V = {S, A, B} T = {0, 1}

P = { $S \rightarrow A\ 11\ B|\ 11A$

$S \rightarrow |B|\ 11$

$A \rightarrow 0$

$B \rightarrow BB$ } For removing useless symbols

Solution : Now in the given CFG if we try to derive any string A gives some terminal symbol as 0 but B does not give any terminal string. By following the rules with B we simply get ample number of B's and no significant string. Hence we can declare B as

useless symbol and can remove the rules associated with it. Hence after removal of useless symbols we get,

$$S \rightarrow 11\ A \mid 11$$

$$A \rightarrow 0$$

Example 4.33 : *Find CFG with no useless symbols equivalent to*

$S \rightarrow AB \mid CA \qquad B \rightarrow BC \mid AB$

$A \rightarrow a \qquad C \rightarrow aB \mid b$

Solution : Consider, the rule

$$S \rightarrow AB \qquad 1$$

$$S \rightarrow CA \qquad 2$$

In the rule 2, C and A can be replaced by some terminating string but in rule 1, B cannot be replaced by terminating string. It tends to form a never ending loop.

e.g. : $S \rightarrow AB \rightarrow aAB \rightarrow aaAB \rightarrow aaaAB$ and so on.

Thus we come to know as B is an useless symbol. Removing B the rules are now,

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$C \rightarrow b$$

Example 4.34 : *Consider the following CFG*

$G = (V, \Sigma, R, S)$

where $V = \{S, X, Y\}$

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

$R = \{S \rightarrow XY \mid 0$

$X \rightarrow 1\ \}$

Remove the useless symbols from it.

Solution : As

$$S \rightarrow XY$$

$$S \rightarrow 0$$

$$X \rightarrow 1$$

It is easy to recognize that there is no derivation for Y. Hence we can eliminate the production with Y, and the rules are

$$S \rightarrow 0$$

$$X \rightarrow 1$$

Example 4.35 : *Remove useless symbols from the following grammar.*

$$S \rightarrow aA \mid bB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bB$$

$$D \rightarrow ab \mid Ea$$

$$E \rightarrow aC \mid d$$

Solution : We will first find all the productions which are giving terminal symbols.

$$A \rightarrow a$$

$$D \rightarrow ab$$

$$E \rightarrow d$$

Now consider start symbol

$$S \rightarrow aA \mid bB$$

Now since $B \rightarrow bB$ we will be continuously in a loop with B productions. Thus B is a useless symbol. So we will eliminate it. Then the rules are

$$S \rightarrow aA$$
$$A \rightarrow a|aA$$
$$D \rightarrow ab|Ea$$
$$E \rightarrow aC|d$$

Again if we look at E productions there is a production $E \rightarrow aC$. But there is no rule for C. Hence we will remove the production $E \rightarrow aC$.

Now,

$$S \rightarrow aA$$
$$A \rightarrow aA|a$$
$$D \rightarrow ab|Ea$$
$$E \rightarrow d$$

Now the rules S and A indicates that there is no D and E in the derivation. Again we get D and E as useless symbols. So we will eliminate D and E. Finally after removal of all useless symbols the production rules are ,

$$S \rightarrow aA$$

$$A \rightarrow aA \mid a$$

Example 4.36 : *Eliminate the useless symbols from the following grammar.*

$$S \rightarrow aS \mid A \mid C$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

Solution : We will first write all the productions that are giving terminal symbols.

$$A \rightarrow a$$

$$B \rightarrow aa$$

But if we look at start symbol S then,

$$S \rightarrow aS \mid A \mid C$$

There is no production for B from start symbol. Thus B becomes a useless symbo

Similarly,

$$S \rightarrow C \rightarrow aCb \rightarrow aaCbb \rightarrow aaaCbbb \rightarrow \ldots$$

There is no terminating symbol for C. Therefore we will eliminate C. Then set rules are

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

Thus we obtain the grammar having these two rules after removal of usele symbols B, C.

Example 4.37 : *Eliminate the useless symbols from following grammar.*

$$S \rightarrow aA \mid a \mid Bb \mid cC$$

$$A \rightarrow aB$$

$$B \rightarrow a \mid Aa$$

$$C \rightarrow cCD$$

$$D \rightarrow ddd$$

Solution : Consider all the productions that are giving terminal symbols.

$$S \rightarrow a$$

$$B \rightarrow a$$

$$D \rightarrow ddd$$

Now consider

$$S \rightarrow cC$$

$$C \rightarrow cCD$$

$$D \rightarrow ddd.$$

When we try to derive the string using production rule for C we get,

$$S \rightarrow cC \rightarrow ccCD \rightarrow ccCddd \rightarrow cccCDddd \rightarrow cccCddddd \rightarrow \ldots$$

We will not get any terminal symbol for C. Thus we get a useless symbol C. To reach to D the only rule available is by using C. But as C gets eliminated, there is no point in keeping D. Hence D also will be removed.

$$S \rightarrow aA \mid a \mid Bb$$

$$A \rightarrow aB$$

$$B \rightarrow a \mid Aa$$

is the reduced grammar.

Elimination of ε Productions from Grammar

As we have seen in finite automata and regular expression that ε or a null string indicates a string with no value. We have also seen that even though some NFA contains ε moves we can convert that NFA without ε moves. Even in context free grammar, if at all there is ε production we can remove it, without changing the meaning of the grammar. Thus ε productions are not necessary in a grammar.

For example,

$$S \rightarrow 0S|1S|\varepsilon$$

Then we can remove ε production. But we have to take a care of meaning of CFG. The meaning of CFG should not get changed if we place $S \rightarrow \varepsilon$ in other rules we get $S \rightarrow 0$ when $S \rightarrow 0\,S$ and $S \rightarrow \varepsilon$

as well as $S \rightarrow 1$ when $S \rightarrow 1\,S$ and $S \rightarrow \varepsilon$

Hence we can rewrite the rules as

$$S \rightarrow 0S|1S|0|1$$

Thus ε production is removed.

Problems on Removal of ε Production

Example 4.38 : *Remove the ε production from following CFG by preserving meaning it.*

$$S \rightarrow XYX$$

$$X \rightarrow 0X|\varepsilon$$

$$Y \rightarrow 1\ Y|\varepsilon$$

Solution : Now, while removing ε production we are deleting the rules $X \rightarrow \varepsilon$ and Y - To preserve the meaning of CFG we are actually placing ε at right hand side whereeve and Y have appeared.

Let us take

$$S \rightarrow XYX$$

if first X at right hand side is ε.

Then $S \rightarrow YX$

Similarly if last X in R.H.S. = ε.

Then $S \rightarrow XY$

If $Y = \varepsilon$ then

$$S \rightarrow XX$$

If Y and X are ε then,

$$S \rightarrow X \text{ also}$$

$$S \rightarrow Y \text{ when both X are replaced by } \varepsilon$$

$$\therefore \quad S \rightarrow XY|YX|XX|X|Y$$

Now let us consider

$$X \rightarrow 0X$$

If we place ε at right hand side for X then,

$$X \rightarrow 0$$

$$X \rightarrow 0X|0$$

Similarly $Y \rightarrow 1\ Y|1$

Collectively we can rewrite the CFG with removed ε productions as

$$S \rightarrow XY|YX|XX|X|Y$$

$X \rightarrow 0X|0$

$Y \rightarrow 1Y|1$

Example 4.39 : *For the CFG given below remove the ε production*

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow \varepsilon$

Solution : According to the replacement procedure, we will place ε at S in the sentential form.

$S \rightarrow a\,S\,a$ if $S = \varepsilon$

$S \rightarrow a\,a$

Similarly if $S \rightarrow b\,S\,b$ if $S = \varepsilon$

$S \rightarrow b\,b$

Thus finally the rules are

$S \rightarrow a\,S\,a|b\,S\,b|a\,a|b\,b$

Example 4.40 : *Eliminate the ε productions from the CFG given below.*

$A \rightarrow 0\,B\,1 \mid 1\,B\,1$

$B \rightarrow 0\,B|1\,B|\varepsilon$

Solution : Now the ε production is $B \rightarrow \varepsilon$ we will delete this production. And then add the productions having B replaced by ε.

$A \rightarrow 0\,B\,1$

if $B = \varepsilon$

$A \rightarrow 0\,1$

Similarly $A \rightarrow 1\,B\,1 \rightarrow 11$

$\therefore$ $A \rightarrow 0\,B\,1|1\,B\,1|01|11$

Similarly, $B \rightarrow 0\,B$

if $B = \varepsilon$

$B \rightarrow 0$

as well as $B \rightarrow 1$

$B \rightarrow 0\ B \mid 1\ B \mid 0 \mid 1$

Collectively, we can write

$A \rightarrow 0\ B\ 1 \mid 1\ B\ 1 \mid 01 \mid 11$

$B \rightarrow 0\ B \mid 1\ B \mid 0 \mid 1$

Example 4.41 : *Construct CFG without ε production from the one which is given below*

$S \rightarrow a \mid Ab \mid aBa$

$A \rightarrow b \mid \varepsilon$

$B \rightarrow b \mid A$

Solution : If you observe carefully, then not only A have ε production but even B al indicates ε production i.e. $A \rightarrow \varepsilon$ straight forward but $B \rightarrow A \rightarrow \varepsilon$. Let us apply the meth of replacement.

$S \rightarrow A\ b$

if $A = \varepsilon$ then

$S \rightarrow b$

if $B = \varepsilon$

$S \rightarrow a\ a$

$S \rightarrow a \mid A\ b \mid b \mid a\ a \mid a\ B\ a$

$A \rightarrow b$

$B \rightarrow b$

Finally the rules are

$S \rightarrow a \mid A\ b \mid b \mid a\ a \mid a\ B\ a$

$A \rightarrow b$

$B \rightarrow b$

Example 4.42 : *Consider the CFG given below.*

$S \rightarrow XY$

$X \rightarrow Zb$

$Y \rightarrow BW$

$Z \rightarrow AB$

$W \rightarrow Z$

$A \rightarrow aA \mid bA \mid \varepsilon$

$B \rightarrow Ba \mid Bb \mid \varepsilon$

In this example only A and B shows ε productions.

If we put $A = \varepsilon$ in A production then

$A \rightarrow a \mid b$

$B \rightarrow a \mid b$

$Z \rightarrow A\,B$ Putting $A = \varepsilon$, $B = \varepsilon$

$Z \rightarrow \varepsilon\varepsilon \rightarrow \varepsilon$

So whenever Z appears we place ε

$X \rightarrow b$

Since $W \rightarrow Z \rightarrow \varepsilon$ we can straightaway delete it.

$Y \rightarrow b$

Collectively let us rewrite the rules

$S \rightarrow XY$

$X \rightarrow b$

$Y \rightarrow b$

$A \rightarrow a\,A \mid b\,A \mid a \mid b$

$B \rightarrow B\,a \mid B\,b \mid a \mid b$

Since S is a start symbol and it defines XY only whereas X and Y yields b as a terminal symbol. There is no chance for producing rules by A or B. Hence A and B are useless symbols and we can remove it. Finally rules are

$$S \rightarrow XY$$
$$X \rightarrow b$$
$$Y \rightarrow b$$

which ultimately

$$S \rightarrow bb$$

Example 4.43 : *Consider the CFG given below,*

For eliminating ε productions

$$S \rightarrow P\,0\,Q \mid Q\,Q \mid 0\,R \mid R\,Q\,P$$
$$P \rightarrow R\,0 \mid 1\,R \mid R\,R \mid R\,Q\,P$$
$$Q \rightarrow Q\,0 \mid P\,Q \mid \varepsilon$$
$$R \rightarrow 0\,P \mid Q\,Q\,Q$$

Solution : As we can see that $Q \rightarrow \varepsilon$.

Similarly $R \rightarrow QQQ \rightarrow \varepsilon$.

Hence we will replace Q and R by ε and accordingly add the production rules.

Similarly $P \rightarrow R\,R \rightarrow \varepsilon$

Let us modify the rules if P or Q or R becomes ε.

$$S \rightarrow P\,0\,Q \mid 0 \mid 0\,R \mid Q\,Q \mid R\,Q\,P$$
$$P \rightarrow R\,0 \mid 0 \mid 1 \mid 1\,R \mid RR \mid RQP$$
$$Q \rightarrow Q\,0 \mid 0 \mid PQ$$
$$R \rightarrow 0\,P \mid 0 \mid Q\,Q\,Q$$

Note that with P, Q, R other meanings are associated, they are not purely giving to us. Therefore we cannot remove P, Q and R symbols.

4.9.3 Removing Unit Productions

The unit productions are the productions in which one non terminal gives anothe non terminal.

For example if

$$X \rightarrow Y$$
$$Y \rightarrow Z \qquad Z \rightarrow X$$

Then X, Y and Z are unit productions. To optimize the grammar we need t remove the unit productions.

If $A \Rightarrow B$ is a unit production and $B \rightarrow X_1 X_2 X_3 \ldots X_n$ then while removing $A \rightarrow B$ production we should add a rule $A \rightarrow X_1 X_2 X_3 \ldots X_n$.

Let us solve something.

Example 4.44 : *If the CFG is as below.*

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

$$A \rightarrow 0S \mid 00$$

$$B \rightarrow 1 \mid A$$

$$C \rightarrow 01$$

then remove the unit productions.

Solution : Clearly $S \rightarrow C$ is a unit production. But while removing $S \rightarrow C$ we have to consider what C gives. So, we can add a rule to S.

$$S \rightarrow 0A \mid 1B \mid 01$$

Similarly $B \rightarrow A$ is also a unit production so we can modify it as

$$B \rightarrow 1 \mid 0S \mid 00$$

Thus finally we can write CFG without unit production as

$$S \rightarrow 0A \mid 1B \mid 01$$

$$A \rightarrow 0S \mid 00$$

$$B \rightarrow 1 \mid 0S \mid 00$$

$$C \rightarrow 01$$

Example 4.45 : *Optimize the CFG given below by reducing the grammar. S is a start symbol.*

$$S \rightarrow A \mid 0C1$$

$$A \rightarrow B \mid 01 \mid 10$$

$$C \rightarrow \varepsilon \mid CD$$

Solution : $S \rightarrow A \rightarrow B$ is a unit production

$C \rightarrow \varepsilon$ is a null production.

$C \rightarrow CD$ B and D are useless symbol.

Reducing a grammar we have to avoid all the above conditions.

Let $S \rightarrow A$

i.e. $A \rightarrow B$ is a useless symbol because B is not defined further more.

$S \rightarrow 01 \mid 10$

i.e. $S \rightarrow 01 \mid 10 \mid 0C1$

But $C \rightarrow \varepsilon$

Hence ultimately $S \rightarrow 01 \mid 10$

$A \rightarrow B$ but we can remove this production since B is a useless symbol.

Hence $A \rightarrow 01 \mid 10$

But the start symbol $S \rightarrow 01 \mid 10$

There is no A in the derivation of A so by considering A also as a useless symb we get final CFG as

$S \rightarrow 01 \mid 10$

Example 4.46 : *Eliminate the unit productions from following grammar*

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C \mid b$

$C \rightarrow D$

$D \rightarrow E \mid bC$

$E \rightarrow d \mid Ab$

Solution : As,

$S \rightarrow AB$ and

$A \rightarrow a$

There is only one rule with A and that too giving terminal symbol. Hence there i no question of getting unit production with A. Now consider

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow E$

It is clear that B, C and D are unit productions. As E → d | Ab, we will replace value of D. Then,

$$D \rightarrow d \mid Ab \mid bC$$

Similarly as C → D we can write

$$C \rightarrow d \mid Ab \mid bC$$

Hence B becomes

$$B \rightarrow d \mid Ab \mid bC \mid b$$

Thus the grammar after removing unit productions

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow d \mid Ab \mid bC \mid b$$

$$C \rightarrow d \mid Ab \mid bC$$

$$D \rightarrow d \mid Ab \mid bC$$

$$E \rightarrow d \mid Ab.$$

Now there is no path for D and E. From start state we can remove them by considering useless symbols. The optimized grammar will be

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow d \mid Ab \mid bC \mid b$$

$$C \rightarrow d \mid Ab \mid bC$$

Review Questions

1. *Design a context free grammar to check well formedness of parenthesis.*
2. *Find the language generated by following grammar*

   $S \rightarrow 0S\ 1 \mid 0A\ 1$

   $A \rightarrow 0A \mid 1S \mid 0 \mid 1$

   $A \rightarrow 1A \mid 1S \mid 1$
3. *Construct the CFG for set of all strings over {a, b} consisting of equal number of a's and b's.*
4. *Show that the following grammar is ambiguous.*

   $S \rightarrow a$

   $S \rightarrow abSb$

   $S \rightarrow aAb$

   $A \rightarrow bS$

   $A \rightarrow aAAb$