1 picture

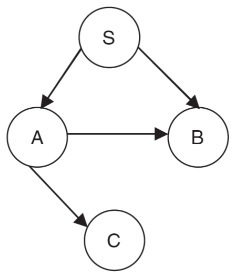

Fig. 6.16

The graph does not contain any loop. So, the language generated by the CFG is finite. The derivation from the grammar is $S \rightarrow AB \rightarrow BCB \rightarrow aaa$.

Thus, the length of the longest string is 3.

b) In the grammar, there is a unit production $A \rightarrow B$. By removing the unit production, the grammar becomes

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow SC/a \\ B &\rightarrow SC/a \\ C &\rightarrow AB/b \end{aligned}$$

The grammar is in CNF. The non-terminal transitional graph for the grammar is shown in Fig. 6.17.

2 picture

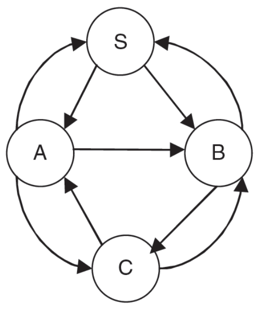

Fig. 6.17

The graph contains loops. So, the language generated by the CFG is infinite.

6.11.3 Membership Problem

Membership problem decides whether a string w is generated by a given CFG.

This is proved by the CYK algorithm. This algorithm was proposed by John Cocke, Daniel Younger, and Tadao Kasami. According to the algorithm, the string of terminals is generated from length 1 to length of w, which is also the string to be checked for membership. If w $\in$ the set of string generated, then w is a member of the strings generated by the CFG. For this, we need to convert the given grammar into CNF. With the help of an example we are showing this.

Example 6.50 Let the grammar converted to CNF is

$$\begin{aligned} S &\rightarrow C_bA/C_aB \\ A &\rightarrow C_bD/C_aS/a \\ B &\rightarrow C_aE/C_bS/a \\ D &\rightarrow AA \\ \Sigma &\rightarrow BB \\ C_a &\rightarrow a \\ C_b &\rightarrow b \end{aligned}$$

Check whether baba is a member of the CFL generated by the CFG.

Solution:Start producing strings of length 1.

String	Producing NT
A	A
A	B
A	C$_a$
B	C$_b$

Produce strings of length 2

String	Producing NT
ba	$S(S \rightarrow C_bA)$
aa	$S(S \rightarrow C_aB)$
aa	$D(D \rightarrow AA)$
aa	$E(E \rightarrow BB)$

Produce strings of length 3. This may be the first one produced by an NT and the last two produced by another NT or vice versa.

String	Producing NT
baa	$A(A \to C_bD)$
aba	$A(A \to C_aS)$
aaa	$A(A \to C_aS)$
aaa	$B(B \to C_aE)$
bba	$B(B \to C_bS)$
baa	$B(B \to C_bS)$

Produce strings of length 4. This may be the First one produced by an NT and the last three produced by another NT or the first two produced by an NT and the last two produced by another NT or the first three produced by an NT and the last one produced by another NT. Two–two combination is not available for this grammar as there is no production combining two of S, D, and E.

String	Producing NT
abaa	$D(D \rightarrow AA, A \rightarrow a, A \rightarrow CbD)$
baaa	$D(D \rightarrow AA, A \rightarrow CbD, A \rightarrow a)$
aaba	$D(D \rightarrow AA, A \rightarrow a, A \rightarrow CaS)$
abaa	$D(D \rightarrow AA, A \rightarrow CaS, A \rightarrow a)$
aaaa	$D(D \rightarrow AA, A \rightarrow CaS, A \rightarrow a)$
aaaa	$D(D \rightarrow AA, A \rightarrow a, A \rightarrow CaS)$
aaaa	$E(E \rightarrow BB)$
aaaa	$E(E \rightarrow BB)$
abba	$E(E \rightarrow BB)$
bbaa	$E(E \rightarrow BB)$
abaa	$E(E \rightarrow BB)$
baaa	$E(E \rightarrow BB)$
aaaa	$S(S \rightarrow CaB)$
abba	$S(S \rightarrow CaB)$
abaa	$S(S \rightarrow CaB)$
bbaa	$S(S \rightarrow CbA)$
baba	$S(S \rightarrow CbA)$
baaa	$S(S \rightarrow CbA)$

The string in bold is 'baba'. Thus, baba is a member of the CFL generated by the CFG.

6.12 CFG and Regular Language

According to the Chomsky hierarchy, we know that a regular grammar is a subset of CFG. Thus, for every regular language, there exists a CFG. In this section, we shall discuss two theorems related to this

$Theorem 1$: Every regular language is generated by a CFG.

Proof: Assume that L is regular. Thus, there exists a DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ accepting L. From the DFA M, we can generate a CFG $G = \{V_N, \Sigma, P, S\}$ using the following rules.

1. Every state $\in Q$ of M is treated as a non-terminal $\in V_N$. The start state $S = q_0$ (initial state of DFA).

2. For a transitional function $\delta(q_1, a) \to q_2$, where $q_1, q_2 \in Q$ and a $\in \Sigma$, add a production $q_1 \to aq_2$ in P.

3. If q_2 is a fi nal state, add $q_1 \to aq_2$ and $q_1 \to a$ in P.

All there productions have a single non-terminal at the LHS. Thus, it is context free.

$Theorem 2$**:** The language generated by a regular CFG is a regular language. Or every regular grammar generates a regular language.

Proof: Let $G = \{V_N, \Sigma, P, S\}$ be a regular CFG. From this grammar, construct a NDFA $M = \{Q, \Sigma, \delta, q_0, F\}$ using the following rules.

1. For each production in the form $< NT_1 > \rightarrow < i/p >< NT_2 >$, add a transitional function

$$\delta(< NT_1 >, < i/p >) \rightarrow < NT_2 >$$

2. For each production in the form $< NT_1 > \rightarrow < i/p >$, add a transitional function

$$\delta(< NT_1 >, < i/p >) \rightarrow \text{final state}$$

We know that a language accepted by an NDFA is regular expression. Thus, it is proved.

Every regular grammar is some right linear grammar. Already it is given that a right linear grammar can be converted to a left linear grammar. Thus, it can be proved that if L is a regular set, then L is generated by some left linear or some by some right linear grammar.

6.13 Applications of Context-free Grammar

The compiler is a program that takes a program written is the source language as input and translates it into an equivalent program in the target language. Syntax analysis in an important phase in the compiler design. In this phase, mainly grammatical errors called syntax errors are checked. The syntax analyzer (parser) checks whether a given source program satisfies the rules implied by a context-free grammar or not. If it satisfies, the parser creates the parse tree of that program. Otherwise, the parser gives the error messages.

In C language, an identifier is described by the following CFG.

The definition of an identifier in a programming language is

$$letter \rightarrow A|B|...|Z|a|b|...|z$$
$$digit \rightarrow 0|1|...|9$$

$$id \rightarrow letter(letter|digit)_*$$

Let a programmer declare the following variables

```
    int capital;
int r_o_i;
int year;
int 1st_year_interest;
```

then the syntax error will be shown in the fourth line as it does not match with the language produced by the grammar.

For the iteration statements (loop) also, there are CFGs.

```
<iteration statement>→ while(<logical expression>) <statement>
    /do <statement> while (<logical expression>)
    /for(<expression>; ¡expression>; ¡expression>) <statement>
```

The CFG is used to develop extensive markup language (XML). XML is a markup language much like HTML. XML is used as a database and can be applicable to share and store data. It can even be used to construct new languages such as WML.