Context Free Grammar (CFG) and a Context Free Language (CFL)

A context free Grammar is a quadruple G = (N, ∑, P, S) where

N = a finite set of non-terminal symbols (variables)

∑ = alphabet of terminal symbols

S = Start Symbol Є N

P = a finite set of productions or rewrite rules. Each rule is a pair (A, α) (written as A → α where A Є N (head) and α Є (N U ∑)* (body). A Grammar is specified by giving the productions with the first production having the start symbol as the head.

Ex 1 S → 0S1, S → ε. This can also given as S → 0S1 | ε.

Ex 2 S →0S1|01.

Ex 3 $E \to a|E+E|E*E|(E)$.

Conventions :

A,B,C,D,E,S,T $\in$ N

a,b,c,d,e,s,t,0,1,…,(,),special symbols $\in \Sigma$

x,y,z,u,v,w $\in \Sigma^*$

X,Y,Z,U,V,W $\in (\Sigma \cup N)$

α,β,γ…. $\in (\Sigma \cup N)^*$

Derivations : We say that α derives β in a single step ($\alpha \to \beta$) if $\alpha = \alpha_1 A \alpha_2$, $\beta = \alpha_1 \alpha' \alpha_2$ and $A \to \alpha'$ $\in$ P ie β is obtained from α by replacing (rewriting) the head of a production by its body.

Thus in Ex 1, $S \to 0S1$, $00S11 \to 000S111$ and $0S1 \to 01$.

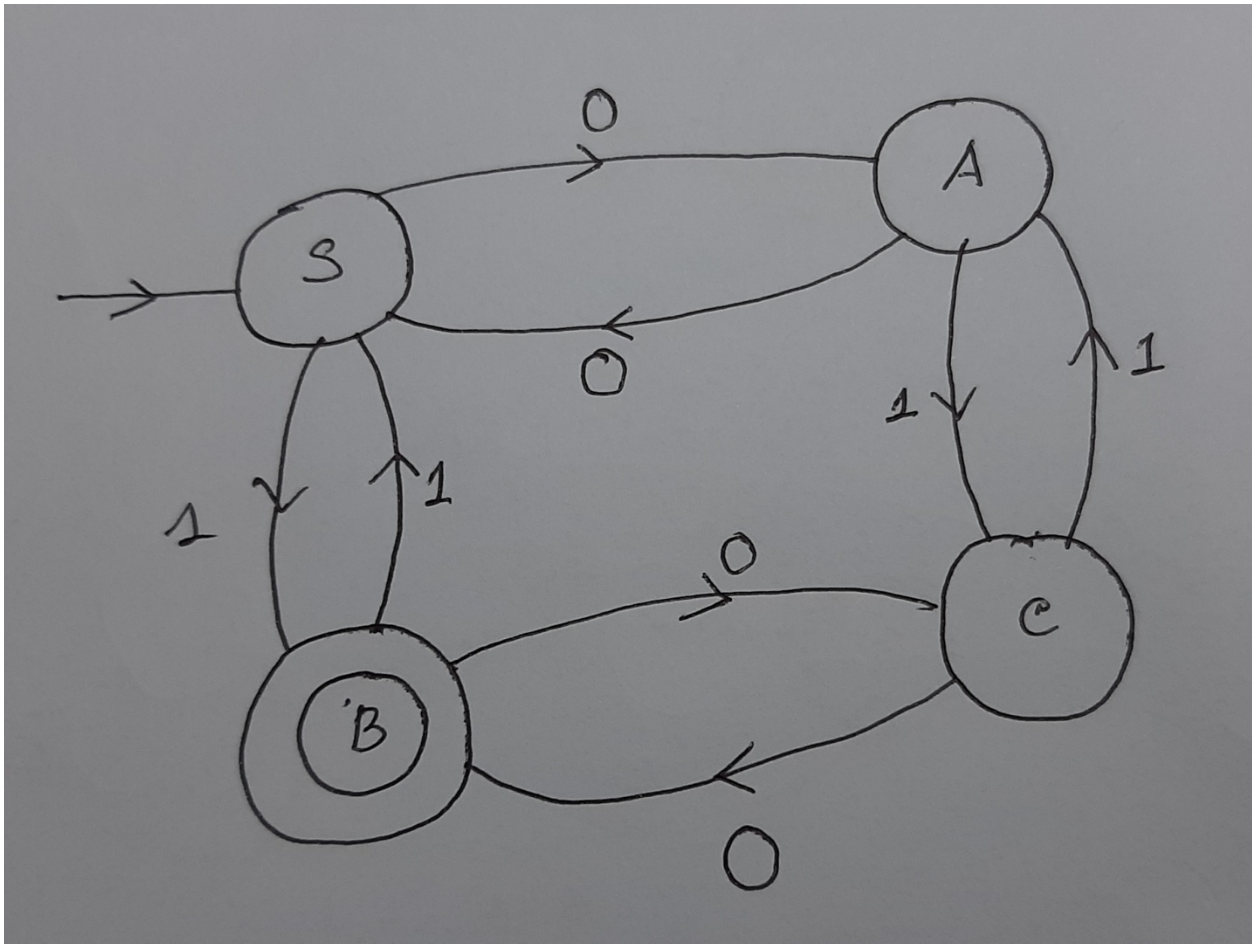
We say that α derives β ($\alpha \to^* \beta$) if either $\alpha=\beta$ or $\alpha=\alpha_1 \to \alpha_2 \ldots\ldots \to \alpha_k = \beta$. The language L(G) generated by a CFG G is $L(G)=\{w \in \sum^* | S \rightarrow^* w\}$. Thus in Ex 1, $L(G)=\{0^n1^n | n \geq 0\}$, in Ex 2, $L(G)= \{0^n1^n | n \geq 1\}$ and in Ex 3, L(G) = simple arithmetic expressions with variable a, operators + and *, and parantheses (,).

A language L is called a context-free language (CFL) if it is generated by a CFG G ie if L=L(G) for a CFG G.

Theorem : A regular language is a CFL.

Let $M = (Q,\sum,\delta,q_0,F)$ be an ε-NFA for a regular language L. Define $G = (Q,\sum,P,S=q_0)$. If there is a transition $A \rightarrow^a B$ for $a \in \sum U\{\varepsilon\}$ ie if $B \in \delta(A,a)$ take a production $A \rightarrow aB$ in P. If $A \in F$ take a production $A \rightarrow \varepsilon$ in P. Then it can be easily proved that L=L(G).

Ex 1 : Language of even 0’s and odd 1’s



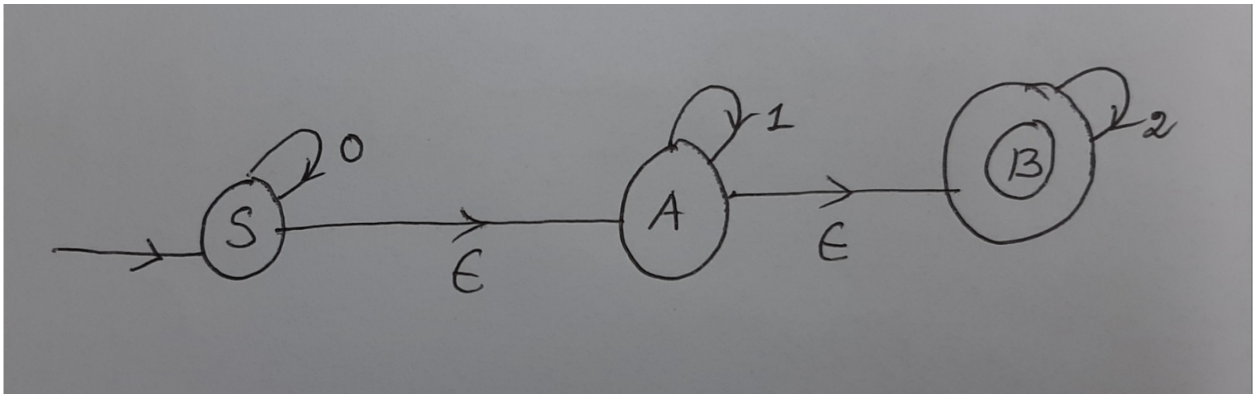
S→0A|1B, A→0S|1C, B→0C|1S|ε, C→0B|1A.

For 01011

S→0A→01C→010B→0101S→01011B→01011

For 0110 S→0A→01C→011A→0110S stuck

Ex 2 0*1*2*

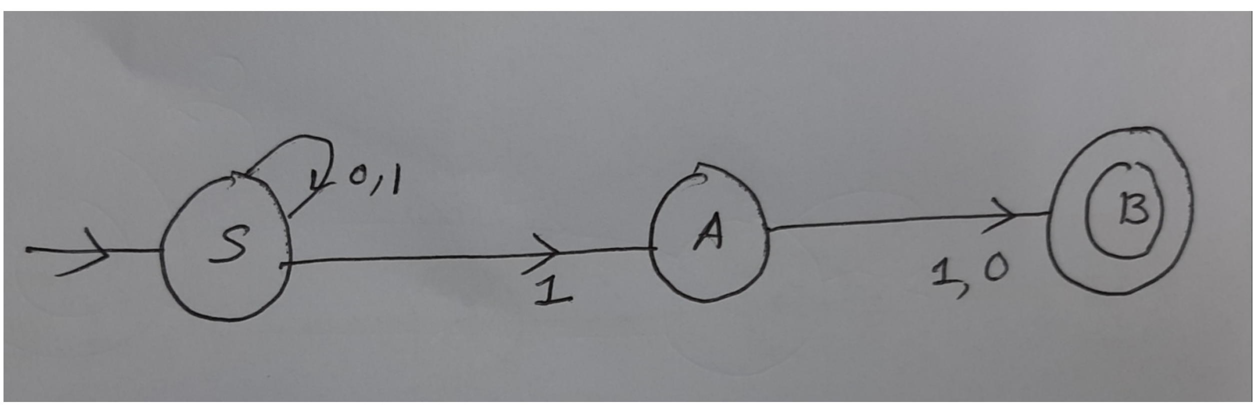


S→0S|A, A→1A|B, B→2B|ε

For 002 S→0S→00S→00A→00B→002B→002

For 021 S→0S→0A→0B→02B stuck

Ex 3 : strings with 1 as the second symbol from the end.



S→0S|1S|1A, A→0B|1B, B→ε

For 010 $S \to 0S \to 01A \to 010B \to 010$ ok

$01S \to 010S$ x

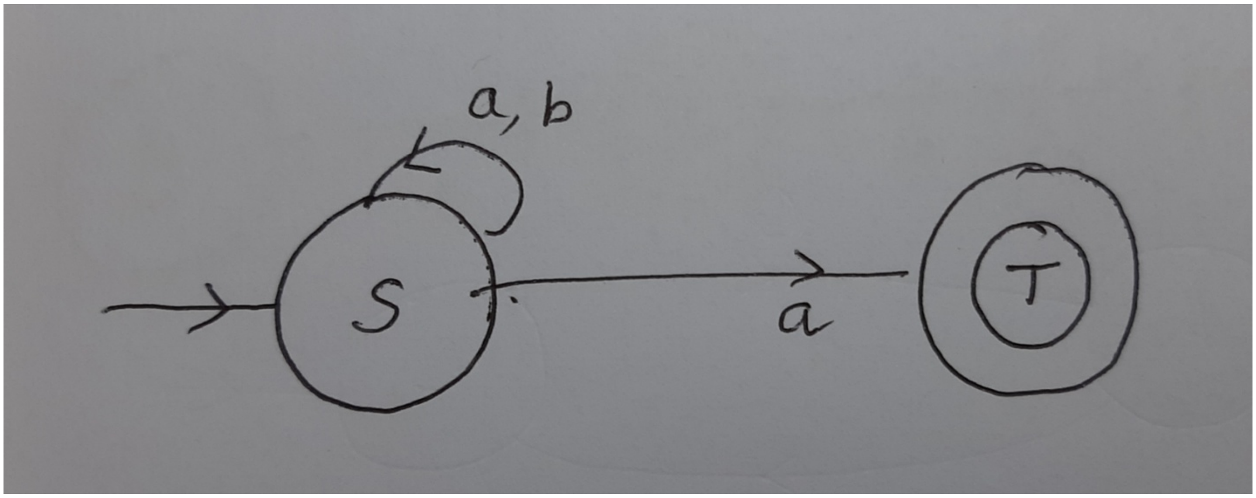
For 001 $S \to 0S \to 00S \to 001S$ x

$001A$ x

Linear Grammar and Linear Language : A CFG is linear if every production has at most one variable in its body. A language is linear if it is generated by a linear Grammar. We have already seen that regular languages are linear. However $S \to 0S1 \mid \varepsilon$ is also linear but generates $\{0^n1^n \mid n \geq 0\}$ which is not regular. Thus the class of regular languages is a proper subclass of linear languages. L_{par}, the language of balanced parantheses is generated by the Grammar $S \to (S) \mid () \mid SS$ and hence is context free. But using the Pumping Lemma for Linear Languages it can be proved that L_{par} is not linear. Thus the class of linear languages is a proper subclass of the CFL's.

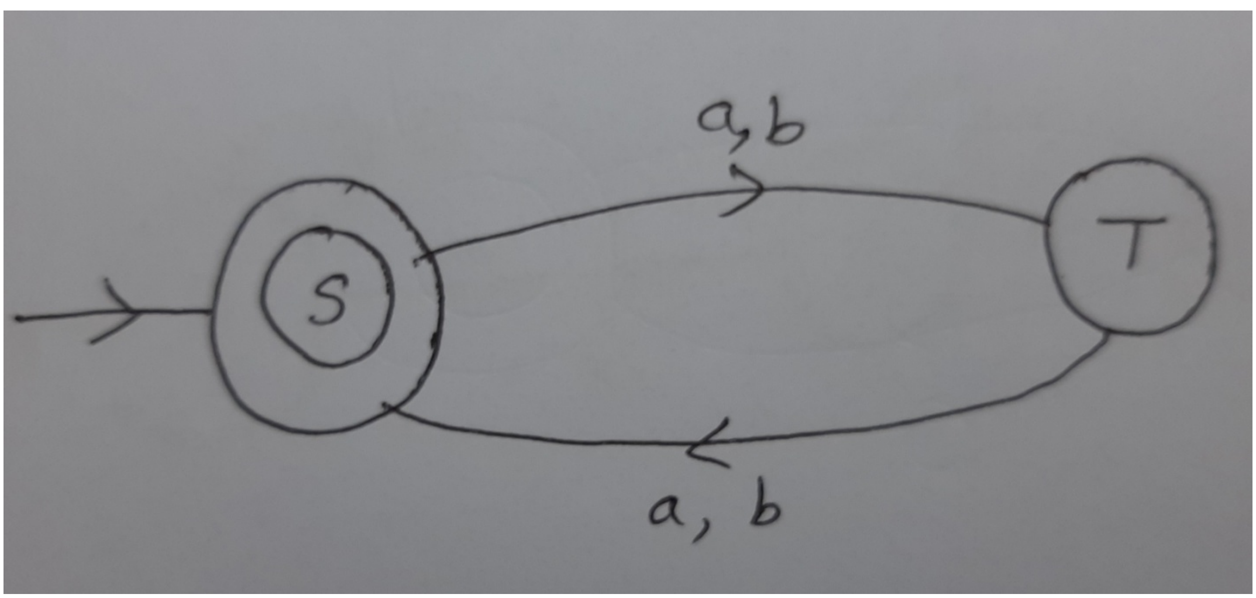
Right Linear Grammar : is a Linear Grammar where if the body of any production has a variable then the variable is the rightmost symbol. We have seen that a regular language is generated by a Right Linear Grammar. Conversely given any Right Linear Grammar G, we can replace $A \rightarrow a_1 a_2 \ldots a_k B$ by $A \rightarrow a_1 B_1$, $B_1 \rightarrow a_2 B_2$, …, $B_{k-1} \rightarrow a_k B$. We can also replace $A \rightarrow a$ by $A \rightarrow aB$, $B \rightarrow \varepsilon$. Then all the productions are of the form $A \rightarrow aB$ or $A \rightarrow C$ or $D \rightarrow \varepsilon$. Now consider the variables as states, the start symbol as the start state and for a production $A \rightarrow aB$ for $a \in \sum \cup \{\varepsilon\}$ take a transition from A to B labeled by a. Finally for all productions $C \rightarrow \varepsilon$, take C as a final state. Then it is easy to prove that the resulting ε-NFA accepts L(G). Thus a language is regular iff it is generated by a Right Linear Grammar.

Ex 1 S→aS|bS|a which gives S→aS|bS|aT,T→ε.



Strings that end with a.

Ex 2 S→aT|bT|ε, T→aS|bS.



Even length strings

Ex 3 $S \to aA|bB, A \to abC, C \to d|D|\varepsilon$. This is same as $S \to aA|bB, A \to aA', A' \to bC, C \to dE|D|\varepsilon, E \to \varepsilon$. The generated language is accepted by the ε-NFA $M=(\{S,A,B,A',C,D,E\},\{a,b,c,d\},\delta,S,\{C,E\})$

where $\delta(S,a)=\{A\}, \delta(S,b)=\{B\}, \delta(A,a)=\{A'\}$,

$\delta(A',b)=\{C\}$, $\delta(C,d)=\{E\}$, $\delta(C,\varepsilon)=\{D\}$.

Left Linear Grammar : is a Linear Grammar where if the body of a production has a variable then the variable is the leftmost symbol. Given a Left Linear Grammar G, if we replace every production $A \to \alpha$ by $A \to \alpha^R$ we get a Right Linear Grammar G^R and obviously $L(G^R) = L(G)^R$. Hence $L(G)^R$ is regular and therefore L(G) is regular. Conversely given a regular language L, L^R is regular and we get a Right Linear Grammar G' generating L^R. Then the left linear grammar $G = G'^R$ will generate L. Thus a language is regular iff it is generated by a left linear grammar. A Grammar which is either Left Linear or Right Linear is called regular and a language is regular iff it is generated by a regular grammar.

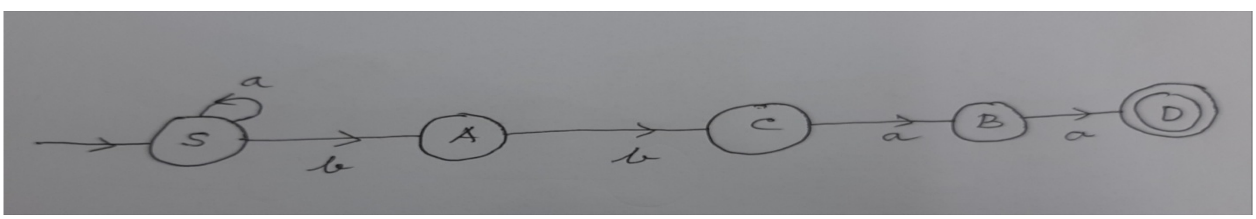
Examples of Conversions :

Convert to an equivalent ε-NFA

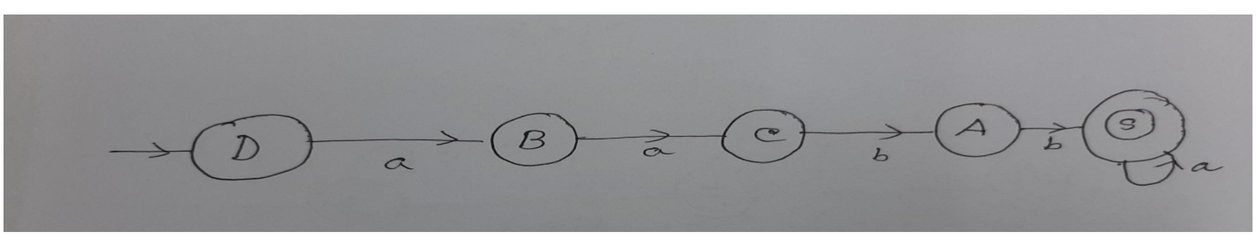
L : S→Sa|Ab,A→Bab,B→a. R.L.G. for L^R : S→

aS|bA,A→baB,B→a. which is equivalent to S→ aS|bA,A→bC,C→aB,B→aD,D→ε

ε-NFA for L^R



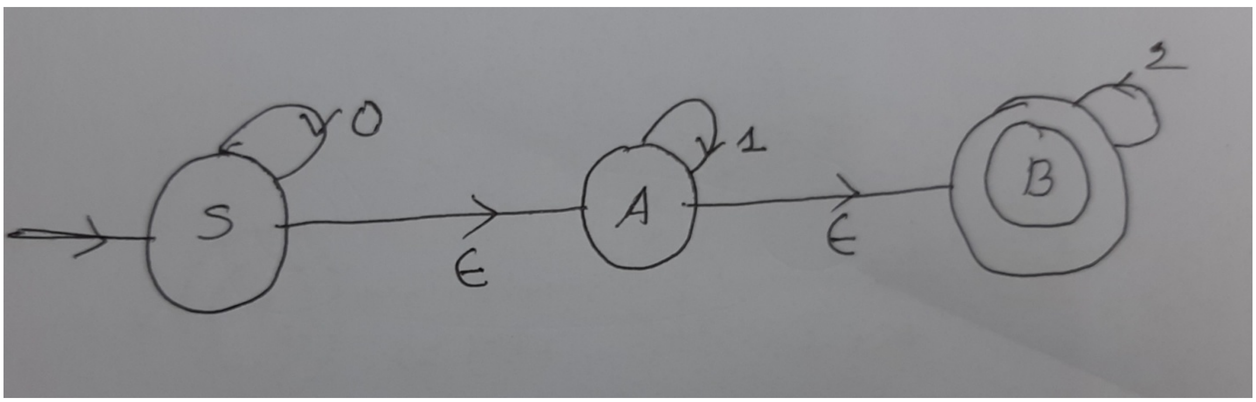
ε-NFA for L



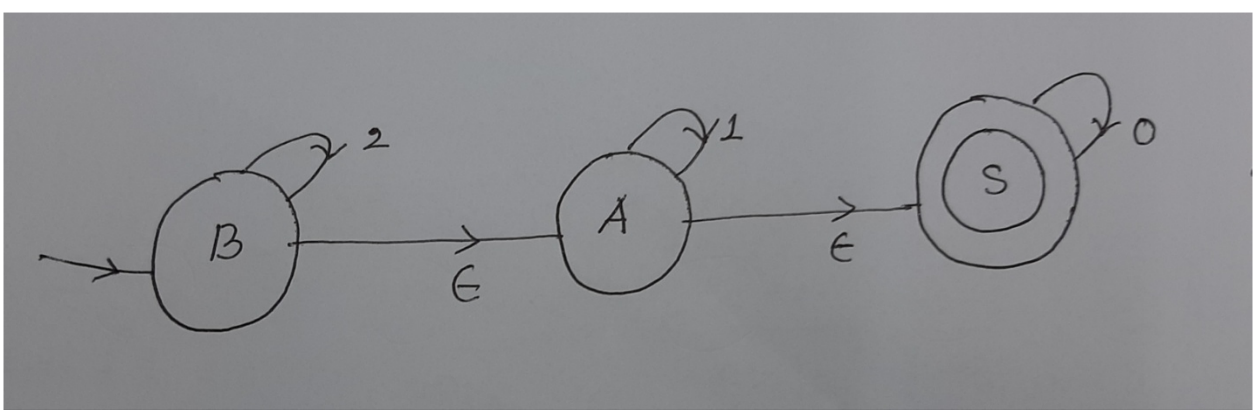
We can also get an equivalent R.L.G. : D→aB, B→aC,C→bA,A→bS,S→aS|ε.

Convert the following ε-NFA to an equivalent Left Linear Grammar.

ε-NFA for L



ε-NFA for L^R



R.L.G. for L^R : B→2B|A,A→1A|S,S→0S|ε

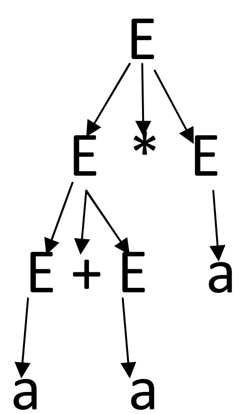
L.L.G. for L : B→B2|A,A→A1|S,S→S0|ε

A regular language is generated by some regular Grammar. But a grammar which is not regular may also generate a regular language. For example convince yourself that the Grammar S→SS|1S|0 which is not regular generates the regular language (0+1)*0.

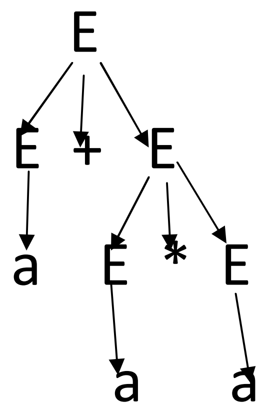
Derivations in a CFL L(G) for a CFG G : For a string w Є L(G), w can be derived from the starting symbol thru a sequence of generated strings. This is called a derivation of w. For example in the Grammar E→E*E|E+E|a, we have the following derivation for a+a*a : E→ E*E→E+E*E→a+E*E→a+a*E→a+a*a. In this derivation at every step the leftmost variable was handled. Such a derivation is called a leftmost derivation. Similarly we can have a rightmost derivation for the same string. E→ E*E→E*a→E+E*a→E+a*a→a+a*a.

Every derivation of w Є L(G) has a unique leftmost and a unique rightmost derivation. Another derivation of a+a*a is E→E+E→E+E*E → a+E*E→a+a*E→a+a*a. The equivalent leftmost and rightmost derivations are E→E+E →a+E→a+E*E→a+a*E→a+a*a and E→E+E→ E+E*E→E+E*a→E+a*a→a+a*a.

A derivation can be associated with a parse tree where the leaves from the left to right gives the string. The parse tree for the first derivation is

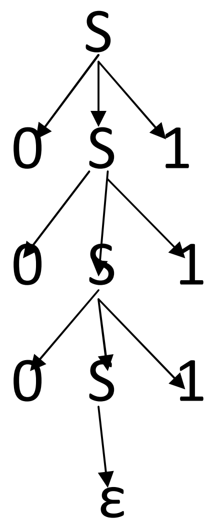


and for the second derivation

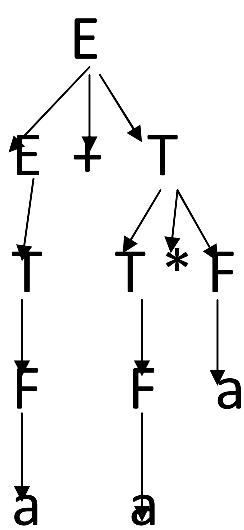


The parse trees convey different meanings.

Ambiguous and unambiguous Grammar : A CFG G is called ambiguous if in L(G) there is a string w with different parse trees. Our example $E \rightarrow E*E|E+E|a$, is ambiguous since the string a+a*a has two parse trees. A Grammar where every $w \in L(G)$ has a unique parse tree is called unambiguous eg $S \rightarrow 0S1|\varepsilon$. Here for example $0^3 1^3$ has a unique parse tree :

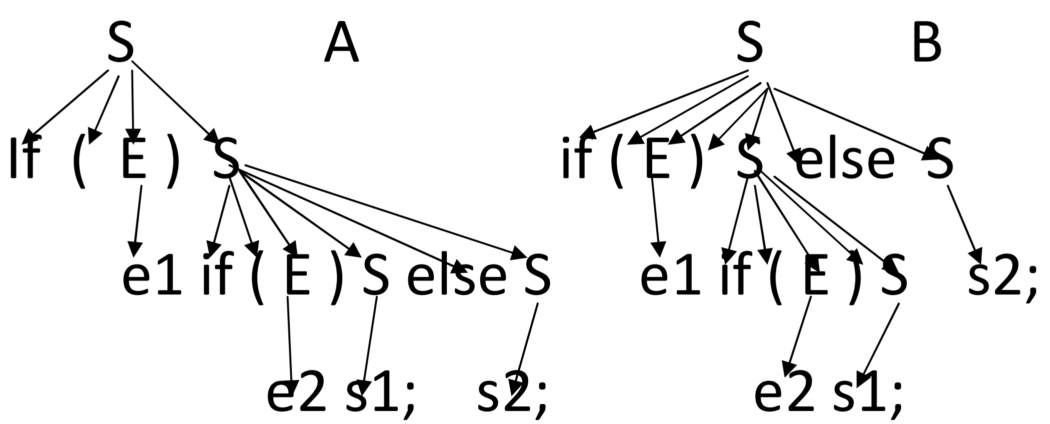


Sometimes the language generated by an ambiguous Grammar G may also be generated by some other unambiguous Grammar. For example L(G) for G : E→E*E|E+E|a can be generated by the unambiguous Grammar : E→ E+T|T,T→T*F|F,F→a. Here for a +a*a we have :



which is the unique parse tree.

Consider the following Grammar for if/else statement used in most programming languages. : S → if (E) S|if (E) S else S|s1;s2;,E→ e1|e2. Take the statement if (e1) if (e2) s1; else s2; This is ambiguous since there are two parse trees



This is called the ambiguity of the dangling else. Where will the else go ? Actually it is assumed to go to the nearest if (in this case A is assumed). The compilers use this ambiguous Grammar and A gets forced during parsing.

A CFL is called inherently ambiguous if it has no unambiguous Grammar. The existence of inherently ambiguous language was proved by Rohit Parikh in 1961 (a MIT research report). This was an existential proof. The first concrete example was provided by Hopcroft and Ullman in their book “Introduction to Automata Theory Languages and computation” (1969 – our text-book)

$L_U=\{a^n b^m c^m d^n | m,n>0\} \cup \{a^n b^n c^m d^m | m,n>0\}$

A CFG is $S \to S1|S2, S1 \to aS1d|aS3d, S3 \to bS3c|bc,$ $S2 \to S4\ S5, S4 \to aS4b|ab, S5 \to cS5d\ |cd$

It can be proved that this language is not linear.

A linear inherently ambiguous language is given by $L_S = \{a^i b^j c^k \mid i=j \text{ or } j=k\}$. A linear Grammar is given by $S \to S1|S2, S1 \to S1c|S3, S3 \to aS3b|\varepsilon, S2 \to aS2|S4, S4 \to bS4c|\varepsilon$. (Problem 2.41 in Introduction to Theory of Computation by M.Sipser, Reference Book in our Syllabus)

Construct CFG

1) Odd length strings with middle symbol 0 $S \rightarrow 0S0|0S1|1S0|1S1|0$
2) Even length strings with two middle symbols same $S \rightarrow 0S0|0S1|1S0|1S1|00|11$.
3) Odd length strings with first middle and last symbol same. Let A be with middle symbol 0 and B be with 1. $S \rightarrow 0A0|1B1$, $A \rightarrow 0A0|0A1|1A0|1A1|0$, $B \rightarrow 0B0|0B1|1B0|1B1|1$.
4) Odd length strings with first last same but different from middle HW
5) $L=\{x \in \{0,1\}^* | n_0(x)=n_1(x)\}$ $S \rightarrow 0S1|1S0|SS|\varepsilon$ Language not linear.
6) Palindromes over $\{0,1\}$ L_{pal} $S \rightarrow 0S0|1S1|0|1|\varepsilon$.
7) Not a palindrome $S \rightarrow 0S0|1S1|0A1|1A0$, $A \rightarrow 0A|1A|\varepsilon$.

8) Balanced parantheses L_{par} $S \to (S) | SS | \varepsilon$.

9) $L=\{0^i1^j0^k | j=i+k\}$ $S \to AB, A \to 0A1 | \varepsilon, B \to 1B0 | \varepsilon$

10) $L=\{0^i1^j0^k | j>i+k\}$ HW

11) $L=\{a^ib^jc^k | i=j+k\}$ $S \to aSc | B, B \to aBb | \varepsilon$.

12) $L=\{a^ib^jc^k | i<j+k\}$ $S \to aSc | T, T \to Tc | c | A,$
$A \to aAb | B, B \to Bb | b$.

13) $L=\{a^ib^jc^k | i>j+k\}$ HW

14) $L=\{a^ib^j | i \le 2j\}$ $S \to aaSb | B, B \to Bb | ab | \varepsilon$.

15) $L=\{a^ib^j | i<2j\}$ HW

16) $L=\{a^ib^j | i>j\}$ $S \to aSb | A, A \to aA | a$.

17) $L=\{a^ib^j | i \ne j\}$ HW

18) $L=\{a^ib^j | i \le j \le 2i\}$ $S \to aSb | aaSb | \varepsilon$.