BRAJ KISHOR PRASAD, brajki@rediffmail.com,
Department of MCA, 2nd Semester
MCA CS2T07: *Automata Theory*

Elimination of Left Recursion and Left Factoring

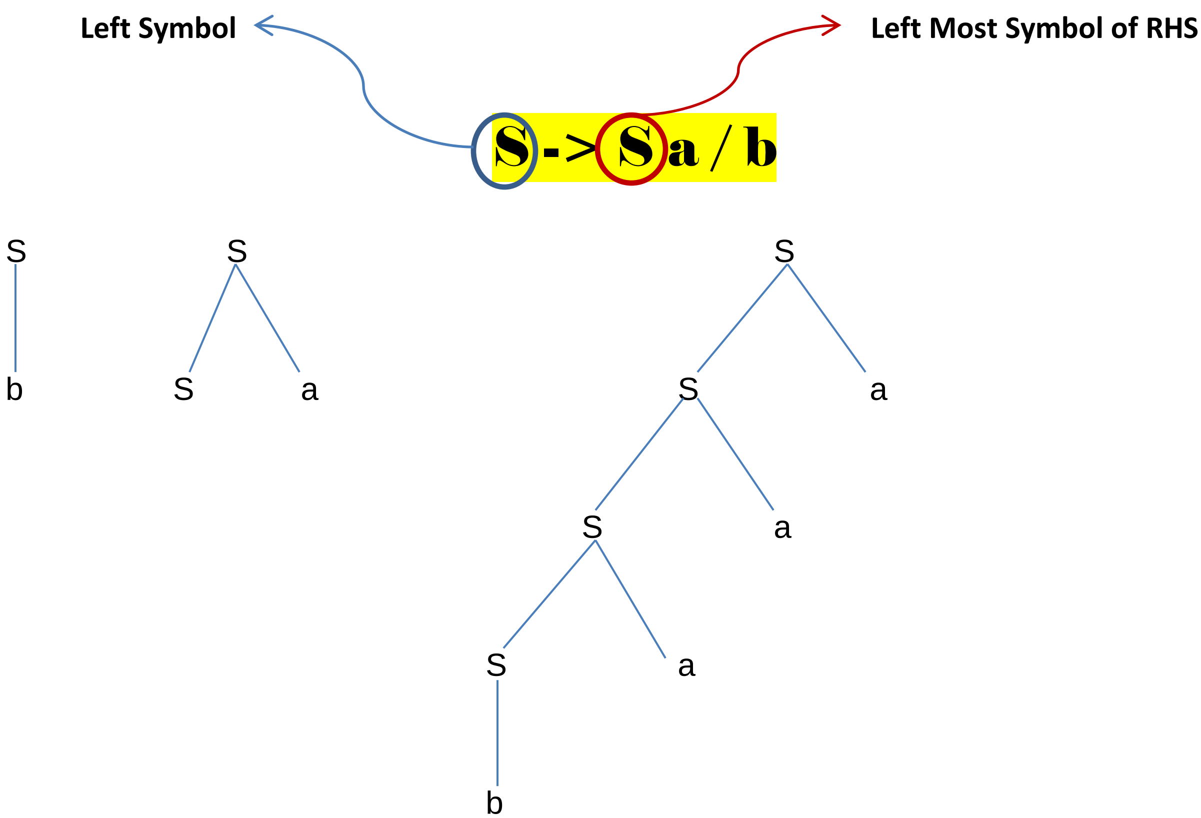
Recursion

There are ***two*** types of recursion:

- Left Recursion (LR)
- Right Recursion (RR)

Left Recursion (LR): When ***left most symbols of RHS*** (Right Hand Side) is same as ***left symbol*** in the production then we say that the production is in Left Recursion.

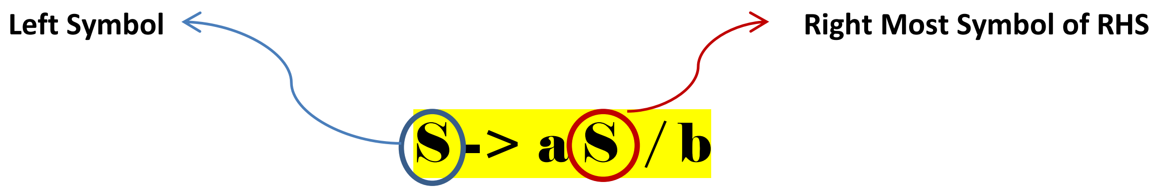
Example:

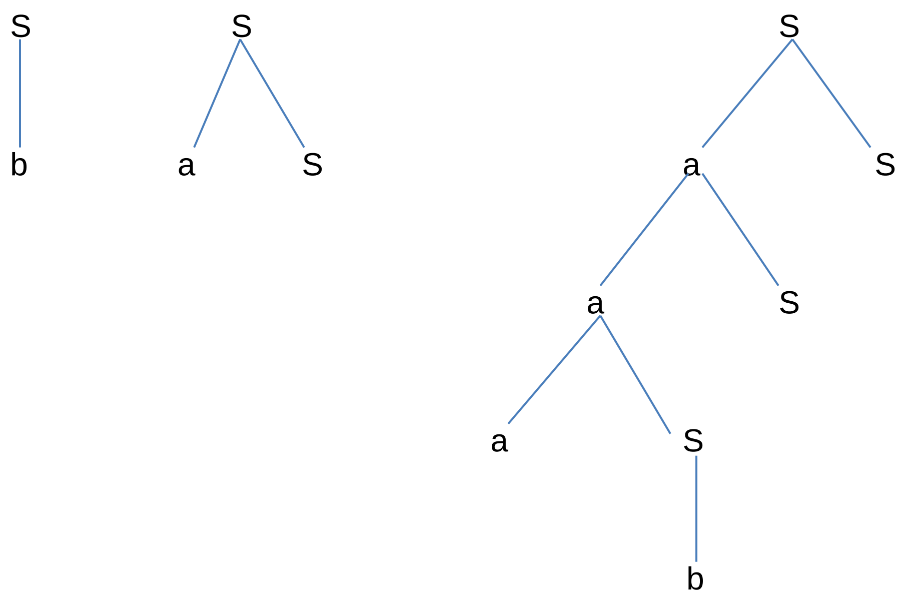


The above figure or concept can be represented as **ba*** in notational form.

Right Recursion (**RR**): When ***right most symbols of RHS*** (Right Hand Side) is same as ***left symbol*** in the production then we say that the production is in Right Recursion.

Example:





The above figure or concept can be represented as **a*b** in notational form.

How to eliminate Left Recursion from the production?

Example 1:

E -> E + T / T

Left Recursive Production

S -> S a / b

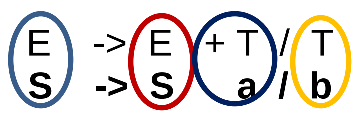
S -> ba*

S' -> a*   (Let)

Therefore,

S -> bS'

S' -> aS' / ∈



S -> S a/b

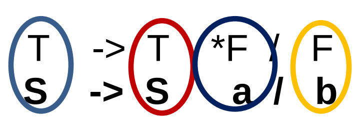
S -> bs'
S' -> aS' / ∈

E -> E +T/T

E -> TE'
E' -> +TE' / ∈

Example 2:

T -> T * F | F



S -> S a/b

S -> bs'
S' -> aS' / ∈

T -> T *F/F

T -> FT'
T' -> *FT' / ∈

Left Factoring:

Left Factoring converts ***non-deterministic grammar*** into ***deterministic grammar***.

Example 1:

S -> $aB_1/aB_2/aB_3/aB_4$

S -> aS'
S -> $B_1/B_2/B_3/B_4$

Example 2:

S -> aSSdS
/aSaSd
/add
/d

S -> aS'/d
S' -> SSdS
/SaSd
/dd

S -> aS'/d
S'' -> SS"
S" -> SdS
/aSd
/dd

Example 2: Find the First() and Follow() of the following grammar.

E -> E + T | T
T -> T * F | F
F -> (E) / id

For LL (1) parsing, the grammar should be free of left recursion and should be left factored. So, we eliminate them then we get the following:

E -> TE'
E' -> +TE' | ∈
T -> FT'
T' -> *FT' | ∈
F -> (E) | id

First() and Follow() Table

	First()	Follow()
E -> TE'	{(, id}	{), $}
E' -> +TE' | ∈	{+, ∈ }	{), $}
T -> FT'	{(, id}	{+,), $}
T' -> *FT' | ∈	{*, ∈ }	{+,), $}
F -> (E) | id	{(, id}	{*,+,), $}

	First()	Follow()
E -> TE'	{ }	{ }
E' -> +TE' | ∈	{ }	{ }
T -> FT'	{ }	{ }
T' -> *FT' | ∈	{ }	{ }
F -> (E) | id	{ }	{ }