Computation Theory

Context-Free Grammar (CFG)

Lecturer: Baida Al kinzawi

Context-Free Grammar (CFG)

CFG stands for context-free grammar. It is a formal grammar which is used to generate all possible patterns of strings in a given formal language. Context-free grammar G can be defined by four tuples as:

G = (N, T, P, S) Where:

1- G is the grammar, which consists of a set of the production rule. It is used to generate the string of a language.

2- T is the final set of a terminal symbol. It is denoted by lower case letters.

3- N is the final set of a non-terminal symbol. It is denoted by capital letters.

4- P is a set of production rules, which is used for replacing non-terminals symbols (on the left side of the production) in a string with other terminal or non-terminal symbols (on the right side of the production).

5- S is the start symbol which is used to derive the string. We can derive the string by repeatedly replacing a non-terminal by the right-hand side of the production until all non-terminal have been replaced by terminal symbols.

Example: Let G(L) = ({S}, {a}, P, S), where P is:

S → aS rule 1

S →Λ rule 2

If we apply production (S → aS) four times and then apply production (S →Λ) we generate the following string: a^4

S → aS using rule 1

→ aaS using rule 1

→ aaaS using rule 1

→ aaaaS using rule 1

→ aaaaΛ using rule 2

The RE = a* can generate a set of string {Λ, a, aa, aaa,.....}. We can have a null string because S is a start symbol and rule 2 gives S →Λ .

نلاحظ في هذه القواعد بأنه يمكن التكرار بأي عدد من الخطوات والتوقف في اي مرحلة من االاشتقاق، والصيغة العامة للكلمات الناتجة من هذه القواعد:

$$\{a^n, n \geq 0 \text{ by } n \text{ steps}\}$$

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

Solution:

The CFG can be given by,

Production rule (P):

$S \rightarrow 0S \mid 1S$

$S \rightarrow \wedge$

The rules are in the combination of 0's and 1's with the start symbol. Since (0+1)* indicates {∧, 0, 1, 01, 10, 00, 11, …}.

Example: Construct a CFG for a language $L = \{wcw^R : w \in (a, b)^*\}$.

Solution:

The string that can be generated for a given language is {aacaa, bcb, abcba, bacab, abbcbba, ...}

The grammar could be:

$S \rightarrow aSa$ rule 1

$S \rightarrow bSb$ rule 2

$S \rightarrow c$ rule 3

Now if we want to derive a string "abbcbba", we can start with start symbols.

$S \rightarrow aSa$

$\rightarrow abSba$ using rule 2

$\rightarrow abbSbba$ using rule 2

$\rightarrow abbcbba$ using rule 3

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

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

Solution:

The string that can be generated for a given language is {abb, aabbbb, aaabbbbbb, ...}.

The grammar could be: **there is another grammar (H.W)**

$S \rightarrow aSbb \mid abb$

Now if we want to derive a string "aabbbb", we can start with start symbols.

$S \rightarrow aSbb$

$\rightarrow aabbbb$

Homework: Construct a CFG for the language $\{a^m b^n \mid m >= n\}$

Derivation Trees

Derivation tree 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 a given set of production rules.

The derivation tree is also called parse tree or syntax tree or parse tree or generation tree or production tree or derivation tree.

The properties of parse tree are:

1- Root: The root node is always a node indicating start symbols.

2- PSG: The interior nodes are always the non-terminal nodes.

3- Leaves: The leaf node is always terminal nodes.

4- Links: collection of connections.

Phrase Structure Grammar (PSG) وهي القواعد التي ظهرت في الشكل الشجري بشكل العقد الوسطية التي تتحول في النهاية الى الرموز النهائية Terminal، وتتكون ايضاً من اربعة مجاميع .PSG = (N, T, P, S)

Derivation Derivation is a sequence of production rules. It is used to get the input string through these production rules. During parsing, we have to take two decisions.

These are as follows:

- We have to decide the non-terminal which is to be replaced.
- We have to decide the production rule by which the non-terminal will be replaced.

We have two options to decide which non-terminal to be placed with production rule.

1. **Leftmost Derivation:** In the leftmost derivation, the input is scanned and replaced with the production rule from left to right. So in leftmost derivation, we read the input string from left to right.

$N \longrightarrow t \mid Nt$ - Leftmost Derivation

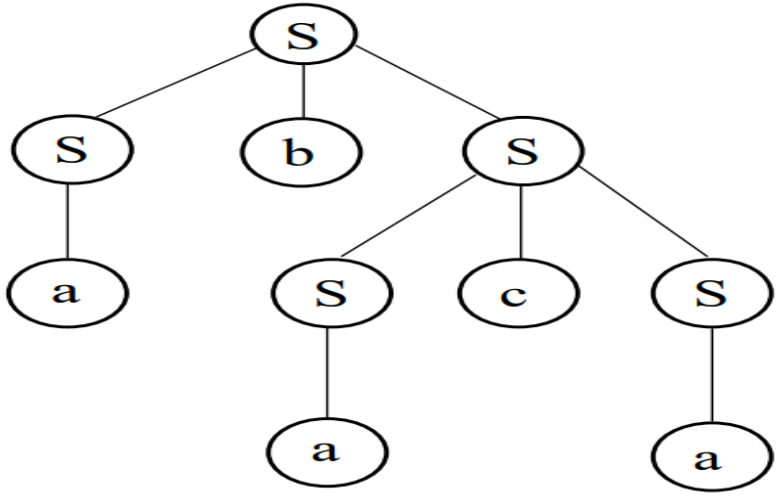
Example: Let G(L) = ({S}, {a, b, c}, P, S), where P is:

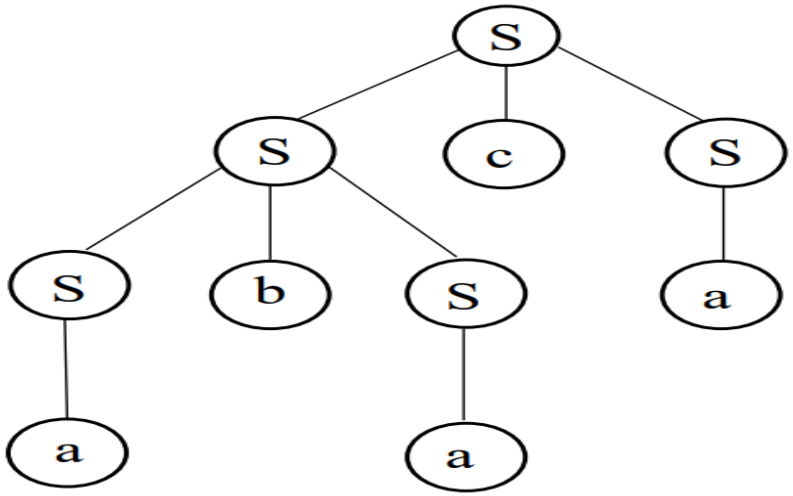
S → SbS | ScS | a

Find the string "abaca"

S → SbS
→ abS
→ abScS
→ abacS
→ abaca Accept

S → ScS
→ SbScS
→ abScS
→ abacS
→ abaca Accept

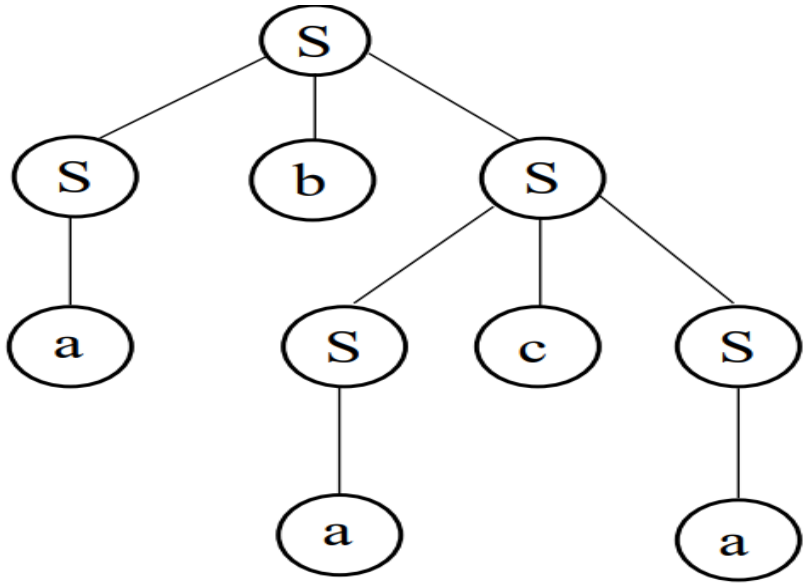




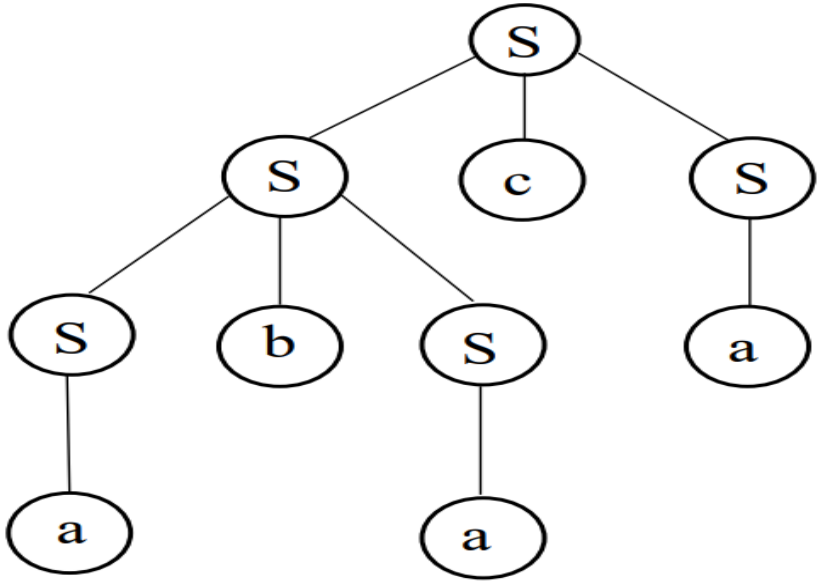
2. **Rightmost Derivation**: In rightmost derivation, the input is scanned and replaced with the production rule from right to left. So in rightmost derivation, we read the input string from right to left.

N ⟶ t | t<u>N</u> - Rightmost Derivation

S → Sb<u>S</u>
→ SbScS
→ SbSca
→ Sbaca
→ abaca Accept



S → Sc<u>S</u>
→ Sca
→ SbSca
→ Sbaca
→ abaca Accept



Examples of Derivation:

Example 1:

Derive the string "abb" for leftmost derivation and rightmost derivation using a CFG given by,

$S \rightarrow AB \mid \wedge$

$A \rightarrow aB$

$B \rightarrow Sb$

Solution:

Leftmost derivation:

$S \rightarrow AB$

$\rightarrow aBB$

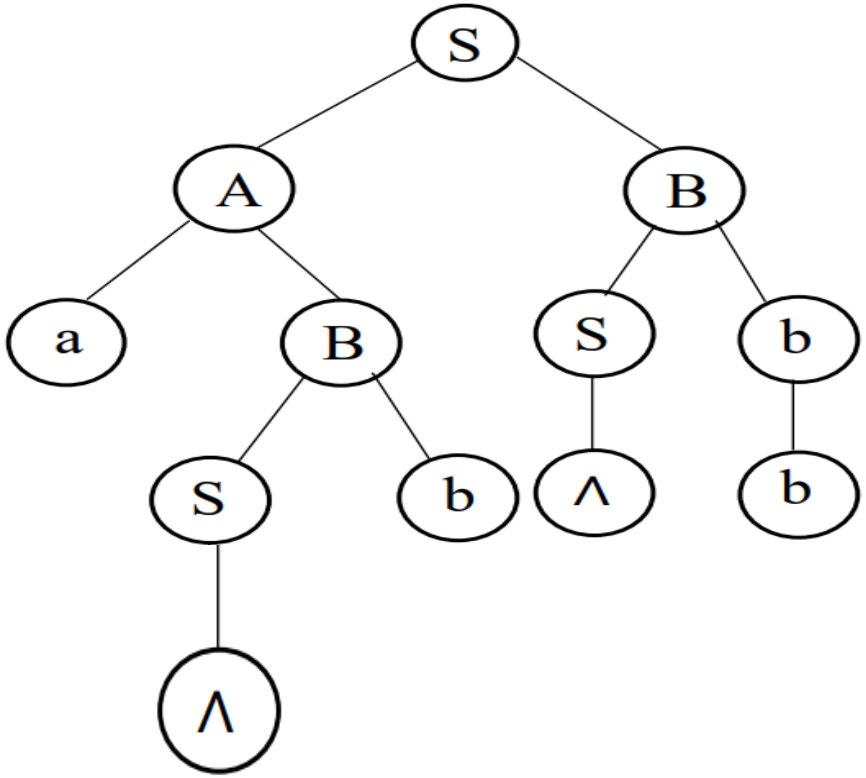
$\rightarrow aSbB$

$\rightarrow a\wedge bB$

$\rightarrow abSb$

$\rightarrow ab\wedge b$

$\rightarrow abb$



Rightmost derivation:

$S \rightarrow AB$

$\rightarrow ASb$

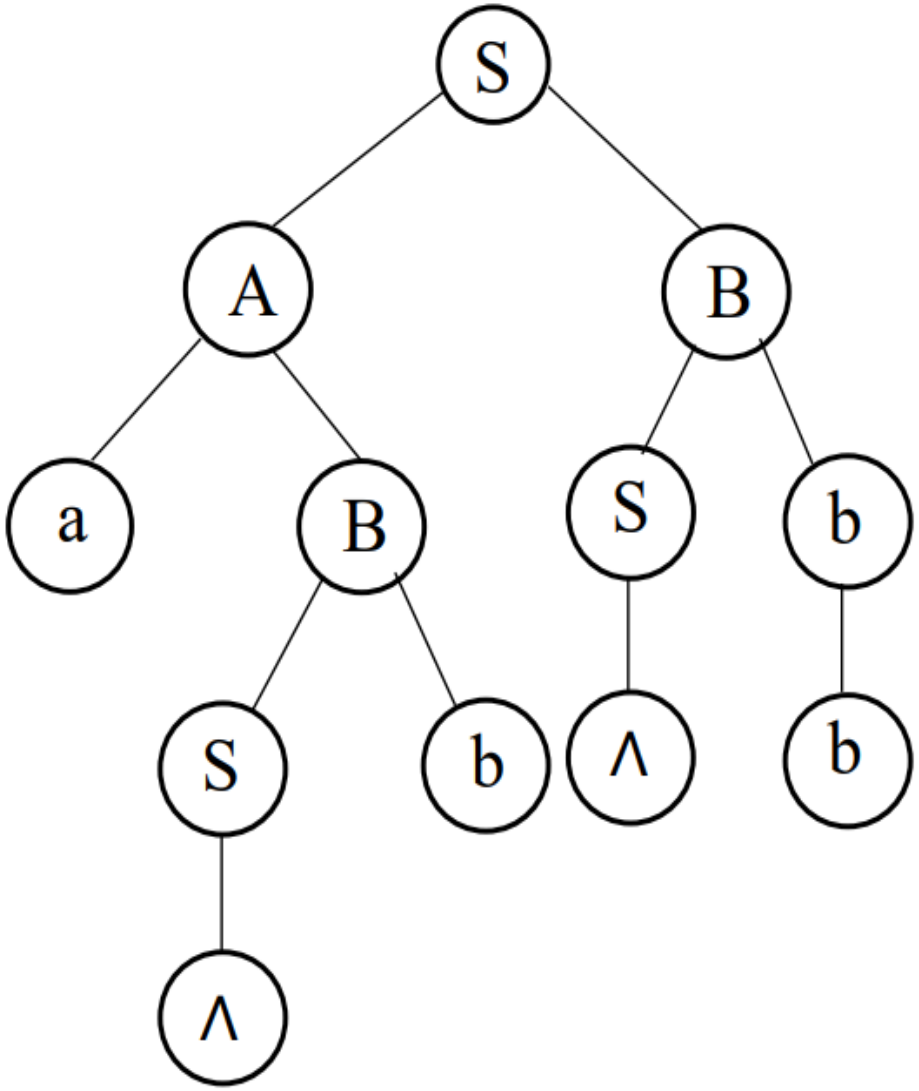
$\rightarrow A\wedge b$

$\rightarrow aBb$

$\rightarrow aSbb$

$\rightarrow a\wedge bb$

$\rightarrow abb$



Example 2:

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

$S \rightarrow aB \mid bA$

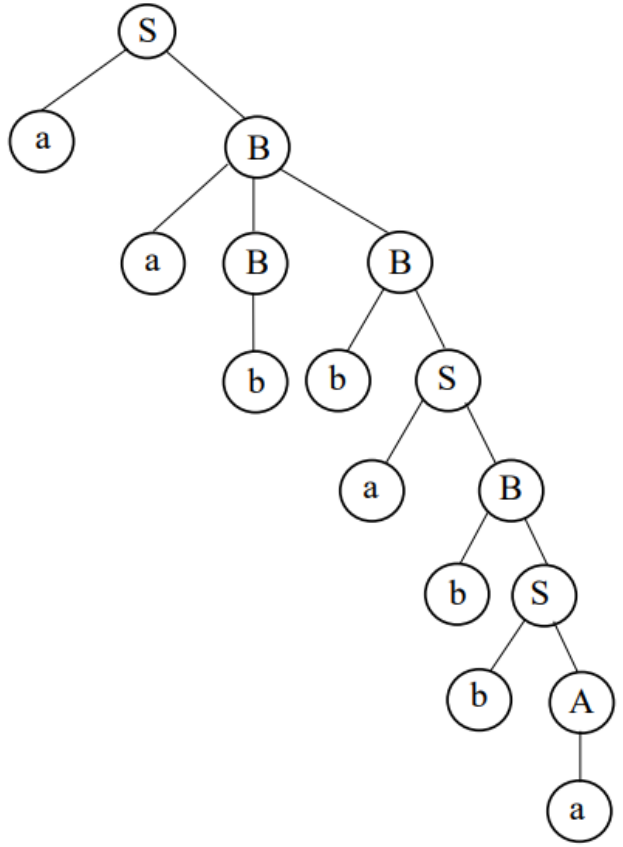
$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid aS \mid aBB$

Solution:

Leftmost derivation:

$S \rightarrow aB$
$\rightarrow aaBB$
$\rightarrow aabB$
$\rightarrow aabbS$
$\rightarrow aabbaB$
$\rightarrow aabbabS$
$\rightarrow aabbabbA$
$\rightarrow aabbabba$



Rightmost derivation:

$S \rightarrow aB$

$\rightarrow aaBB$

$\rightarrow aaBbS$

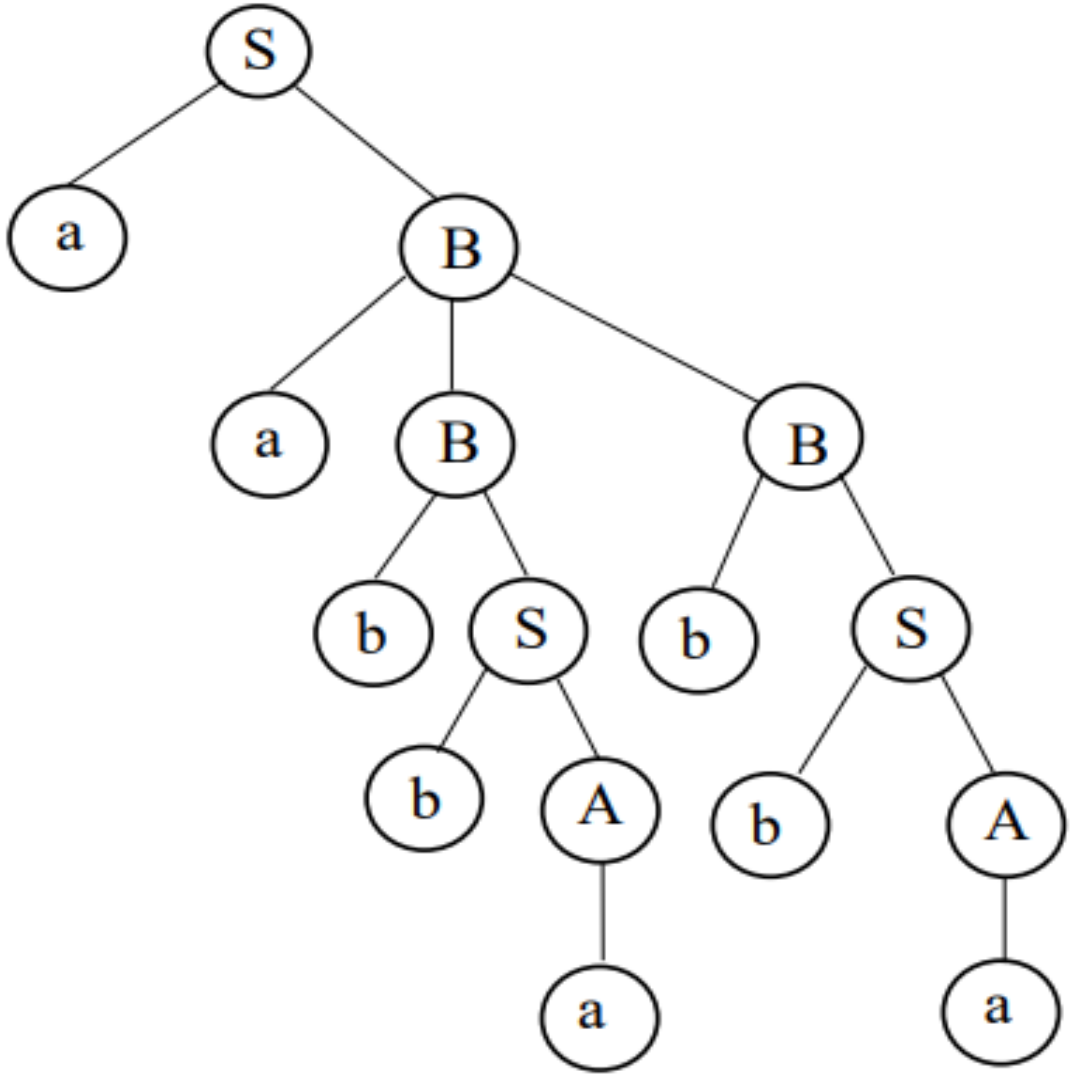
$\rightarrow aaBbbA$

$\rightarrow aaBbba$

$\rightarrow aabSbba$

$\rightarrow aabbAbba$

$\rightarrow aabbabba$



Thanks for listening