CSE331 Automata and Computability

Lecture 15: Leftmost/Rightmost Derivation And Parse Tree

Presenter: Azwad Anjum Islam *Scribe: Warida Rashid*

A derivation tree or a parse tree of a string given a Context Free Grammar (CFG) is a graphical representation of how a string can be generated from the grammar and the semantic information (More on this will be discussed in Compiler Design).

Example

E → E+E

E → E-E

E → E*E

E → E/E

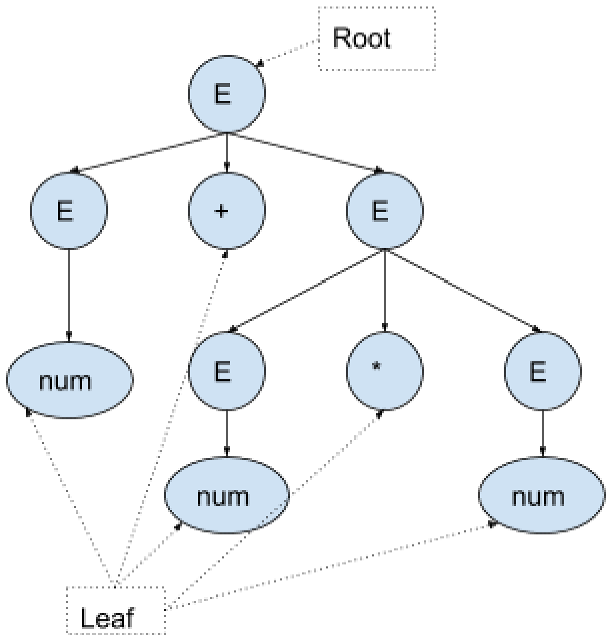
E → (E)

E → num

String derivation:

Consider the string **num +num * num**

To derive a string, we start from the designated start symbol and keep expanding it until the string is generated. The parse tree for the string would be:



Representation Technique:

Root Vertex: Must be labeled by the start symbol.

Vertex: Labeled by the non-terminal symbol/variables.

Leaves: Labeled by a terminal symbol or ε.

Leftmost Derivation:

A leftmost derivation is obtained by taking the production of the leftmost variable in each step.

The leftmost derivation of the example string is as follows:

E → E+E

E → num+E

E → num+E*E

E → num+num*E

E → num+num*num

Rightmost Derivation:

A rigthmost derivation is obtained by taking the production of the rightmost variable in each step.

The leftmost derivation of the example string is as follows:

E → E+E

E → E+E*E

E → E+E*num

E → E+num*num

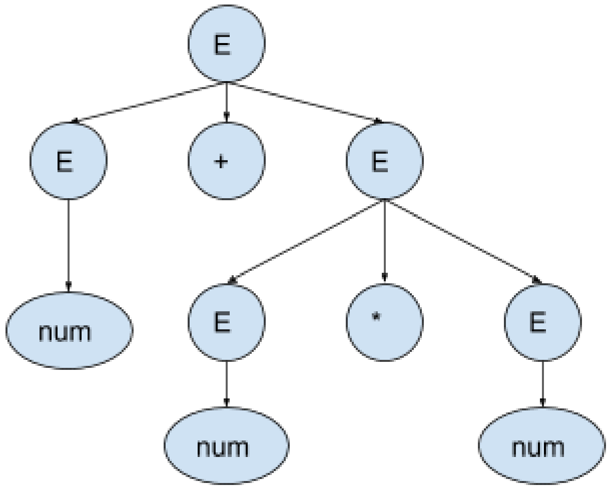
E → num+num*num

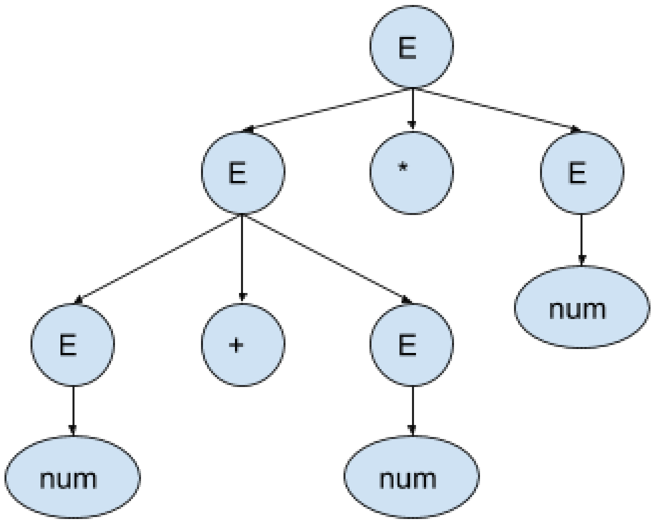
Ambiguity:

A Context Free Grammar is said to be ambiguous, if for some string w that belongs to the language of the grammar there exists multiple parse trees or multiple leftmost derivations or multiple rightmost derivations.

Example 1:

The Grammar in the example is ambiguous. The string **num+num*num** have two different leftmost derivations:

1.	E → E+E E → num+E E → num+E*E E → num+num*E E → num+num*num	 Fig. Parse tree from the leftmost derivation

2.	E → E*E E → E+E*E E → E+E*E E → num+E*E E → num+num*E E → num+num*num		 Fig. Parse tree from the rightmost derivation

Example 2:

The following grammar generates balanced parenthesis:

S → SS

S → (S)

S → ()

Two leftmost derivation of the string **()()():**

1. S → SS

   S → ()S

   S → ()SS

   S → ()()S

   S → ()()()
2. S → SS

   S → SSS

   S → ()SS

   S → ()()S

   S → ()()()

We can design a grammar for the same language (Balanced parenthesis) that is not ambiguous.

$S \rightarrow (RS$

$S \rightarrow \varepsilon$

$S \rightarrow RR$

$R \rightarrow)$

Therefore, **“Ambiguity”** is a property of the grammar, not the language itself.

Inherent Ambiguity:

inherently ambiguous language A context-free language that has no unambiguous grammar. The languages that can only be defined with an ambiguous grammar is said to be inherently ambiguous.

Example:

$L = \{0^i 1^j 2^k$, where $i = j$ or $j = k$ and $i,j,k > 0\}$

This language can only be defined with an ambiguous grammar:

$S \rightarrow AB \mid CD$

$A \rightarrow 0A1 \mid 01$

$B \rightarrow 2B \mid 2$

$C \rightarrow 0C \mid 0$

$D \rightarrow 1D2 \mid 12$

For example, the string 001122, can be derived by taking either of the two productions of the start symbol S.

Limitations

1. **Unaware of the Context**

   A Context Free Grammar, as implied by the name, is unaware of the context.

   A simplified version of English language

   For example, The following image contains a CFG for a simplified English Language and the parse tree for the string “the man read this book”.

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

S → NP VP
→ Det NOM VP
→ *The* NOM VP
→ *The* Noun VP
→ *The man* VP
→ *The man* Verb NP
→ *The man read* NP
→ *The man read* Det NOM
→ *The man read this* NOM
→ *The man read this* Noun
→ *The man read this book*

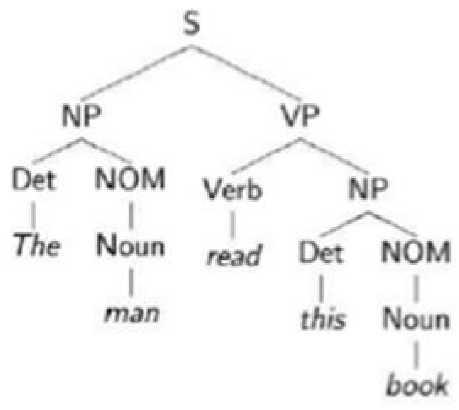


Figure: Parse Tree

However, this grammar can also generate strings like "This book man read that flight". Because, any production rule in the grammar can be applied at any point regardless of the context and the subparts of the tree cannot communicate among themselves.

2. **Limited Memory**

   A CFG has limited memory. It can match two things and no more than that.