Context Free Grammar Examples

15.12.2025

Example 1

The grammar is given with the following products.

E→ I

E→ E+E | E*E

E→ Ia | Ib

E→ I0 |I1

I → a|b |0|1

i)Is the grammar G, a CFG?

ii) Test whether the string "a ∗ b + a1" can be derived by this grammar.

In other words, generate a derivation of that string)

Example 2

The problems in this section are all concerned with the following grammar:

P→ S

S →s; | {L}

L→ SL|S

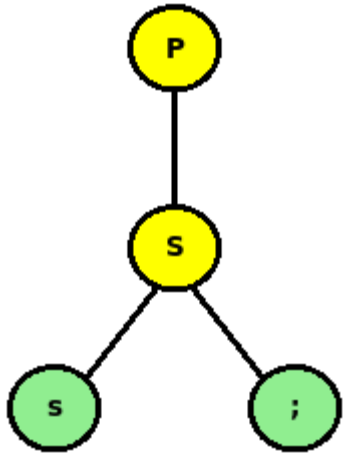
What this grammar “means” and how to read it

Recursion in context-free grammars to describe lists of things is a very common pattern.

P→ S
S →s; | {L}
L→ SL|S

i) Give the derivation for the string s;

P → S → s ;



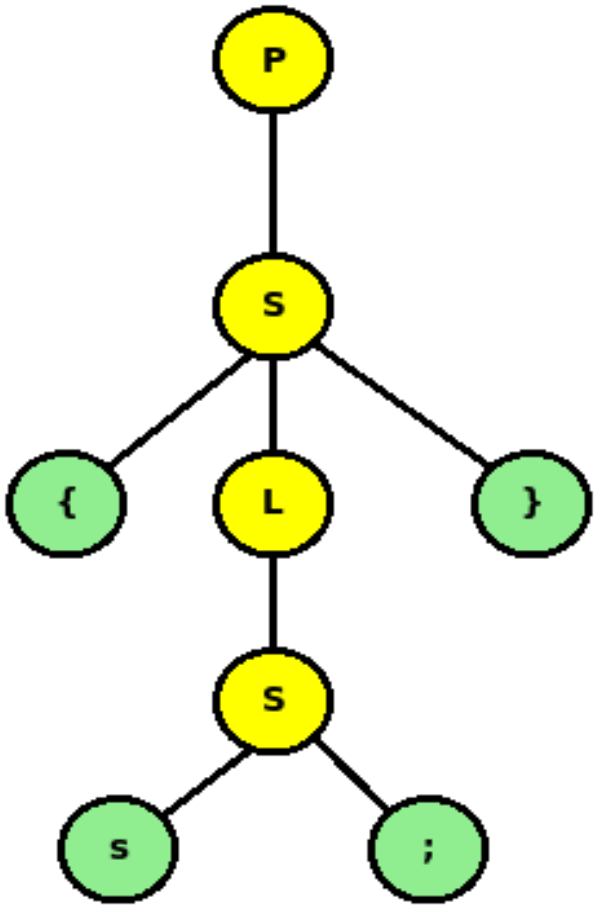
P→ S

S →s; | {L}

L→ SL|S

ii) Give a derivation for the string {s;}

P → S → { L } → { S } → { s ; }



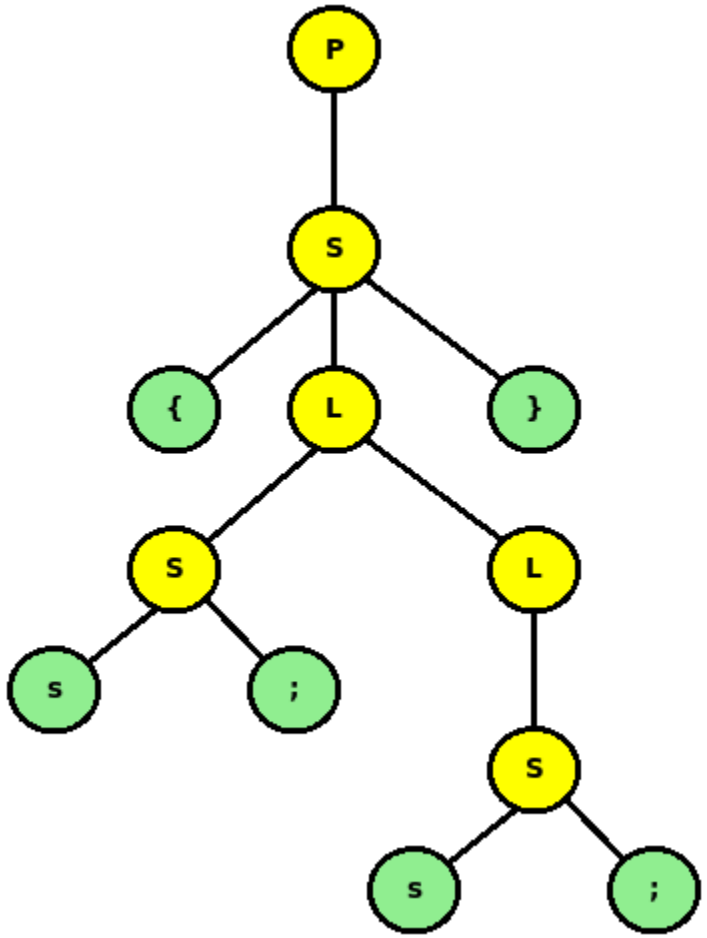
P→ S

S →s; | {L}

L→ SL|S

iii) Give a **left-most derivation** for the string {s;s;}

P → S → { L } → { S L } → { s ; L } → { s ; S } → { s ; s ; }



P→ S
S →s; | {L}
L→ SL|S

iv) Give a **right-most derivation** for the string {s;s;}

P → S → { L } → { S L } → { S S } → { S s ; } → { s ; s ; }

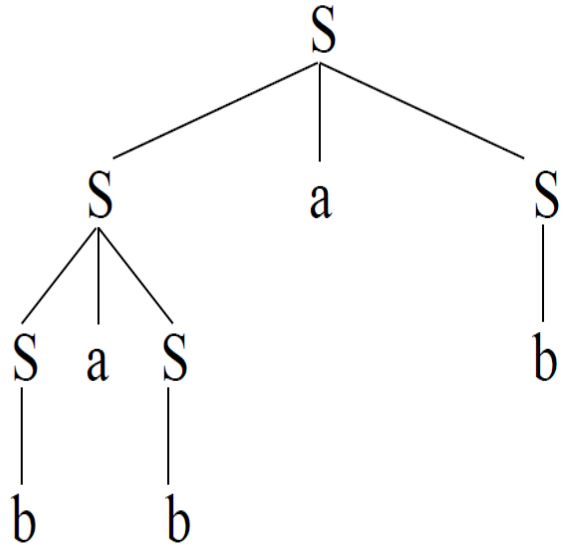
Ambiguous Grammars

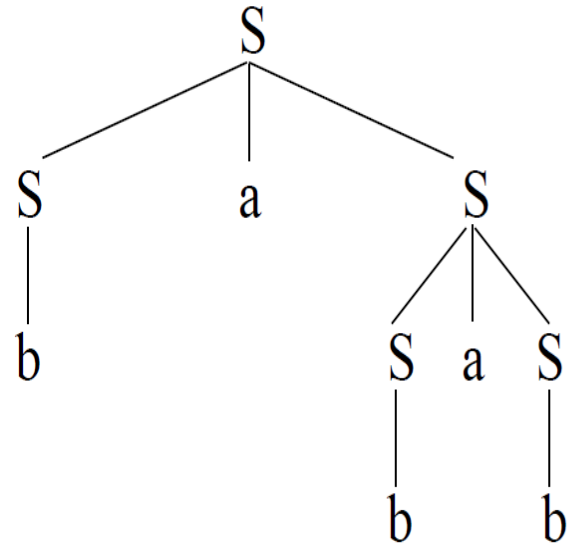
- A CFG is ambiguous if it produces more than one parse tree for a string in the language.
  - there is a string in the language that is the yield of two or more parse trees.

$S \rightarrow SaS \mid b$

is an ambiguous grammar.

There are two parse trees for the string babab





Ambiguity, Leftmost and Rightmost Derivations

- If there are two different parse trees for a string in the language, they must produce two different leftmost derivations for that string.
- Conversely, two different leftmost derivations of a string produce two different parse trees for that string.
- Likewise for rightmost derivations
- Thus, equivalent definitions of ambiguous grammar are:
  - A CFG is ambiguous if there is a string in the language that has two different leftmost derivations.
  - A CFG is ambiguous if there is a string in the language that has two different rightmost derivations.

$S \rightarrow SaS \mid b$

- There are two **leftmost** derivation sequences for the string **babab**

1. $\mathbf{S \Rightarrow_{lm} SaS \Rightarrow_{lm} SaSaS \Rightarrow_{lm} baSaS \Rightarrow_{lm} babaS \Rightarrow_{lm} babab}$
2. $\mathbf{S \Rightarrow_{lm} SaS \Rightarrow_{lm} baS \Rightarrow_{lm} baSaS \Rightarrow_{lm} babaS \Rightarrow_{lm} babab}$

- There are two **rightmost** derivation sequences for the string **babab**

1. $\mathbf{S \Rightarrow_{rm} SaS \Rightarrow_{rm} Sab \Rightarrow_{rm} SaSab \Rightarrow_{rm} Sabab \Rightarrow_{rm} babab}$
2. $\mathbf{S \Rightarrow_{rm} SaS \Rightarrow_{rm} SaSaS \Rightarrow_{rm} SaSab \Rightarrow_{rm} Sabab \Rightarrow_{rm} babab}$

Ambiguity

- We can create an equivalent CFG which produces the same

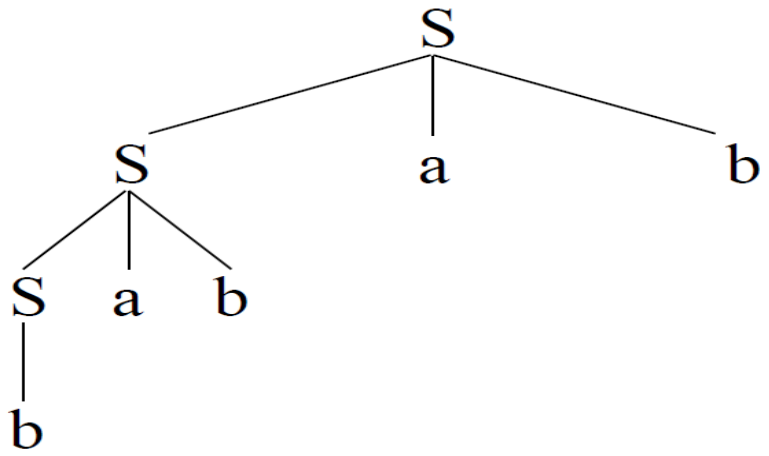
Language by eliminating the ambiguity from the following ambiguous CFG.

S → SaS | b

In the following unambiguous CFG, we prefer left groupings.

S→ Sab | b

- Now, there is only one parse tree for the string «babab»



Example 3

The following SFG grammar rules are given

P → Stmt

StmtList → Stmt StmtList | Stmt

Stmt → Expr=Expr; | { StmtList } | if (Expr) Stmt | if (Expr) Stmt else Stmt

Expr → x | y

This grammar can generate the possible sequences of **if** and ***else*** (abbreviated by **i** and e). **Stmt** is abbreviated by **S** and **Expr** is abbreviated by **E**

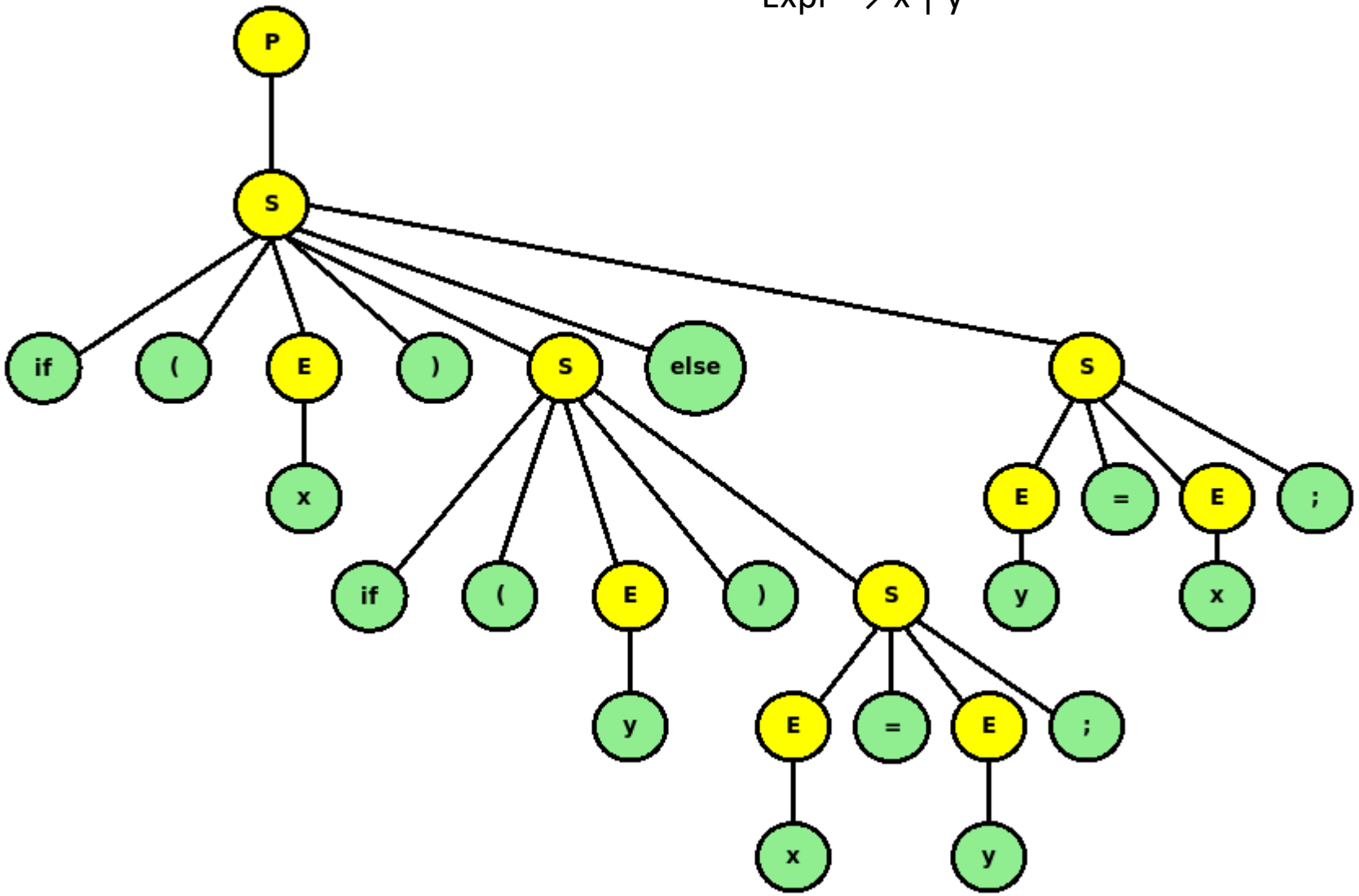
Show that the string ‘ *if(x) if(y) x=y; else y=x;* ’ is in the language described by this grammar.

P → Stmt

StmtList → Stmt StmtList | Stmt

Stmt → Expr=Expr; | { StmtList } | if (Expr) Stmt | if (Expr) Stmt else Stmt

Expr → x | y



P → S

P→ i(E)SeS

P→ i(x)SeS

P→ i(x) i(E) SeS

P→ i(x) i(y) SeS

P→ i(x) i(y) E= E; eS

P→ i(x) i(y) x = E; eS

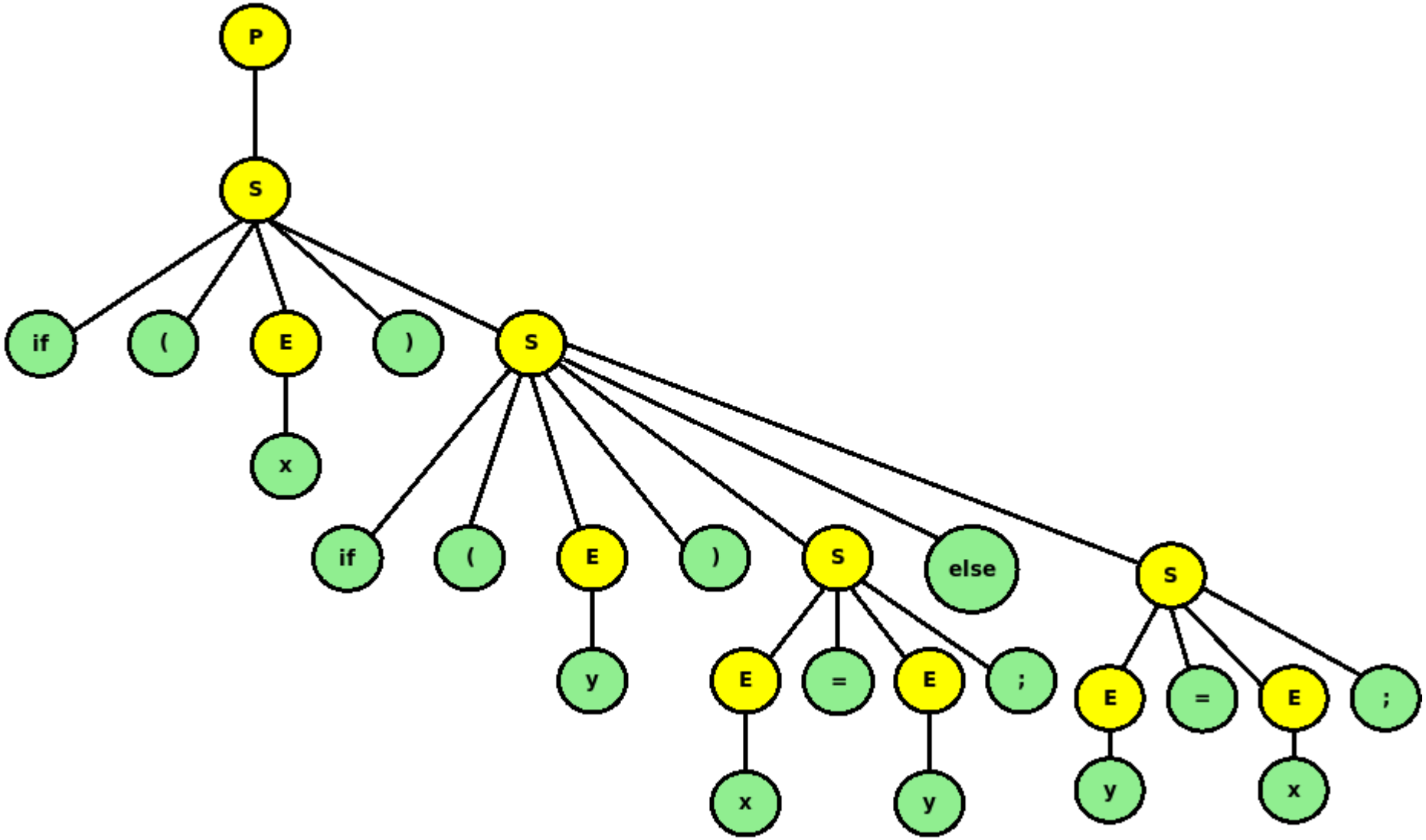
P→ i(x) i(y) x = y; eS

P→ i(x) i(y) x = y; e E = E;

P→ i(x) i(y) x=y; e y = E;

P→ i(x) i(y) x=y; e y = x;

- Using that same string, we show that the grammar is **ambiguous.**
- To show ambiguity, it suffices to show two different parse trees for some string, or two different left derivations for it, or two different right derivations.



Context Free Grammar /CFG in Computer Science

1 Compiler Design

- ❑CFGs define the syntax of programming languages
- ❑Converts source code into parse trees
- ❑Tools like Yacc, Bison use CFG rules; they are parser generators.
  - ❖Java, C, Python syntax is defined using CFG

Expression → Expression + Term | Term

Term → Term * Factor | Factor

Factor → (Expression) | number

CFG in Computer Science

2. Programming Language Design

- ❑Define what valid programs look like
- ❑Grammar rules ensure code correctness and they are syntactic rules.
- ❑Invalid syntax caught during parsing as error detection
- ❑BNF (Backus-Naur Form) notation uses CFG

CFG in Computer Science

3. Natural Language Processing (NLP)

- ❑Analyze grammatical structure giving sentence parsing
- ❑Build hierarchical sentence representations with syntax trees
- ❑Understanding source language structure as machine translation
- ❑Parse user queries for question answering

CFG in Computer Science

4. XML and Markup Languages

Document Structure: Define valid XML/HTML documents

Schema Validation: DTD (Document Type Definition) uses CFG

Parsing: XML parsers use CFG-based rules

Example: HTML tags must follow proper nesting rules

CFG in Computer Science

5. Database Query Languages

SQL Parsing: SQL queries follow CFG rules

Query Validation: Ensure syntactically correct queries

Query Optimization: Parse tree helps in optimization

Example: SELECT statements have specific grammar rules

CFG in Computer Science

6. Configuration Files

- ❑Define valid configuration syntax with JSON/YAML parsing
- ❑Ensure proper structure for file format validation
- ❑Parse and generate config files giving serialization

JSON object structure follows CFG rules is an example

CFG in Computer Science

7. Code Generation

- ❑Convert parse trees to code as intermediate representation
- ❑Analyze and improve generated code as optimization
- ❑Convert between programming languages
- ❑Example: TypeScript → JavaScript compilation

CFG in Computer Science

8. Domain-Specific Languages (DSLs)

Custom Languages: Create specialized languages for specific domains

Scripting Languages: Define syntax for domain tools

Configuration Languages: YAML, Terraform use CFG

Example: SQL, GraphQL, Terraform HCL

CFG in Computer Science

9. Formal Verification

- ❑Verify programs follow grammar rules to show program correctness
- ❑Ensure systems behave correctly as model checking
- ❑Detect errors before runtime constituting static analysis

CFG in Computer Science

10. Artificial Intelligence

- ❑Understanding the meaning of parsed structures is semantic analysis
- ❑ Organizing information hierarchically is knowledge representation
- ❑Defining rule-based knowledge gives expert systems
- ❑Real-World Example: C Language Parsing