UNIT 4

1. Ambiguity in CFG

- A **CFG** is ambiguous if a string can have more than one leftmost or rightmost derivation, or more than one parse tree.
- Example:
  - Grammar: S→S+S ∣ S×S ∣ aS \rightarrow S + S \ | \ S \times S \ | \ aS→S+S ∣ S×S ∣ a
  - String: a+a×aa + a \times aa+a×a
    This can be derived in different ways, leading to ambiguity.

2. Leftmost and Rightmost Derivations

- **Leftmost derivation**: At each step, expand the leftmost non-terminal.
- **Rightmost derivation**: At each step, expand the rightmost non-terminal.
- Example:
  - Grammar: S→aSb ∣ ϵS \rightarrow aSb \ | \ \epsilonS→aSb ∣ ϵ
  - Leftmost: S⇒aSb⇒abS \Rightarrow aSb \Rightarrow abS⇒aSb⇒ab
  - Rightmost: S⇒aSb⇒abS \Rightarrow aSb \Rightarrow abS⇒aSb⇒ab

3. Language of a CFG

- The **language** generated by a CFG is the set of all strings that can be derived from the start symbol using the production rules.

4. Sentential Forms

- A **sentential form** is a string of terminals and non-terminals that can be derived from the start symbol of a grammar.
- Example: For S→aSb ∣ ϵS \rightarrow aSb \ | \ \epsilonS→aSb ∣ ϵ, aSbaSbaSb is a sentential form.

5. Applications of CFG

- CFGs are used in:
  - **Parsing**: Used in compilers to check the syntax of programming languages.
  - **Natural Language Processing**: Helps in parsing sentences.
  - **Programming Languages**: Describes syntax of expressions and statements.

6. Pumping Lemma for CFG

- The **Pumping Lemma** helps in proving that a language is **not** context-free.
- If a language is context-free, there exists a pumping length ppp such that any string of length ≥ ppp can be broken into parts that can be pumped (repeated) while still being in the language.

7. Derivations Generated by a Grammar

- Derivations refer to the steps to derive a string from the start symbol using production rules.
- **Leftmost** and **rightmost** derivations are the main ways to generate strings.

8. Construction of Reduced Grammars

- A grammar is **reduced** if:
  - It has no **useless symbols** (symbols that cannot derive any string).
  - It has no **unreachable symbols** (symbols that cannot be reached from the start symbol).

9. Elimination of Null and Unit Productions

- **Null production**: A production of the form A→ϵA \rightarrow \epsilonA→ϵ.
- **Unit production**: A production of the form A→BA \rightarrow BA→B, where BBB is a non-terminal.
- **Elimination** involves removing these productions while preserving the language.

10. Normal Forms for CFG

- **Chomsky Normal Form (CNF)**: Every production is of the form:
  - A→BCA \rightarrow BCA→BC (where BBB and CCC are non-terminals)
  - A→aA \rightarrow aA→a (where aaa is a terminal)
  - or A→ϵA \rightarrow \epsilonA→ϵ, if AAA is the start symbol.
- CNF helps in simplifying parsing algorithms.

11. Simplification of Context-Free Grammars

- **Greibach Normal Form (GNF)**: Every production is of the form:
  - A→aαA \rightarrow a \alphaA→aα, where aaa is a terminal and α\alphaα is a string of non-terminals.
- GNF is useful in defining recursive descent parsers.

Example of CFG simplification:

- Grammar: S→aSb|ϵS \rightarrow aSb | \epsilonS→aSb|ϵ
- To simplify:
  - Eliminate null and unit productions.
  - Convert to Chomsky or Greibach normal form.

Important MCQ

1. A CFG is said to be ambiguous if:
    - (A) A string has more than one parse tree.
    - (B) A string has exactly one parse tree.
    - (C) It generates no terminal strings.
    - (D) None of the above.
      Answer: A
2. Which of these CFGs is ambiguous?
    - (A) S→aSb|abS \rightarrow aSb | abS→aSb|ab
    - (B) S→SS|aS \rightarrow SS | aS→SS|a
    - (C) Both A and B
    - (D) Neither A nor B
      Answer: C
3. Ambiguity in CFG is:
    - (A) Always removable.
    - (B) Not always removable.
    - (C) Depends on the language.
    - (D) Both B and C.
      Answer: D
4. If a CFG is ambiguous, which of the following is true?
    - (A) Parsing is deterministic.
    - (B) A string has multiple derivations.
    - (C) CFG cannot generate any language.
    - (D) CFG becomes context-sensitive.
      Answer: B
5. A leftmost derivation always replaces:
    - (A) The rightmost non-terminal first.
    - (B) The leftmost non-terminal first.
    - (C) Any non-terminal randomly.
    - (D) Only the start symbol.
      Answer: B
6. What is the result of a rightmost derivation?
    - (A) A unique parse tree.
    - (B) Different derivation sequences.
    - (C) A string of terminals.
    - (D) Both A and C.
      Answer: D

7. Leftmost and rightmost derivations of the same string result in:
    - (A) Same parse tree.
    - (B) Different parse trees.
    - (C) Always ambiguous CFG.
    - (D) No derivation tree.
      Answer: A
8. A string derived by replacing the rightmost non-terminal first is called:
    - (A) Leftmost derivation.
    - (B) Rightmost derivation.
    - (C) Reverse derivation.
    - (D) None of these.
      Answer: B
9. The language of a CFG is a set of:
    - (A) Parse trees.
    - (B) Terminal strings.
    - (C) Non-terminal strings.
    - (D) Both B and C.
      Answer: B
10. Which of these is an example of a language generated by a CFG?
    - (A) {anbn∣n≥0}\{ a^n b^n \mid n \geq 0 \}{anbn∣n≥0}
    - (B) {anbncn∣n≥0}\{ a^n b^n c^n \mid n \geq 0 \}{anbncn∣n≥0}
    - (C) Both A and B.
    - (D) None of these.
      Answer: A
11. A sentential form contains:
    - (A) Only terminals.
    - (B) Only non-terminals.
    - (C) Both terminals and non-terminals.
    - (D) No symbols.
      Answer: C
12. S→aSb∣abS \rightarrow aSb | abS→aSb∣ab. The string aabbbaabbbaabbb is a:
    - (A) Terminal form.
    - (B) Sentential form.
    - (C) Derivation tree.
    - (D) Not derivable.
      Answer: B

13. CFG is commonly used in:
    - (A) Lexical analysis.
    - (B) Syntax analysis.
    - (C) Semantic analysis.
    - (D) Code optimization.
      Answer: B
14. Which type of parser uses CFG for analysis?
    - (A) Top-down parser.
    - (B) Bottom-up parser.
    - (C) Both A and B.
    - (D) None of these.
      Answer: C
15. CFGs are essential for defining:
    - (A) Context-free languages.
    - (B) Regular languages.
    - (C) Finite automata.
    - (D) Turing machines.
      Answer: A
16. The pumping lemma proves that a language is:
    - (A) Context-free.
    - (B) Not context-free.
    - (C) Regular.
    - (D) Deterministic.
      Answer: B
17. Pumping lemma applies to strings of length:
    - (A) Less than pumping length.
    - (B) Equal to pumping length.
    - (C) Greater than or equal to pumping length.
    - (D) None of these.
      Answer: C
18. Pumping Lemma splits a string www as:
    - (A) w=uvxyw = uvxyw=uvxy.
    - (B) w=uvxyzw = uvxyzw=uvxyz.
    - (C) w=uvwxyw = uvwxyw=uvwxy.
    - (D) w=uvxyzww = uvxyzww=uvxyzw.
      Answer: B

19. Useless symbols are removed to:
    - (A) Minimize ambiguity.
    - (B) Simplify the grammar.
    - (C) Change the language.
    - (D) None of these.
      Answer: B
20. A reduced grammar does not contain:
    - (A) Useless symbols.
    - (B) Null productions.
    - (C) Unit productions.
    - (D) All of the above.
      Answer: D

21. Null productions have:
    - (A) A→BA \rightarrow BA→B.
    - (B) A→ϵA \rightarrow \epsilonA→ϵ.
    - (C) A→aA \rightarrow aA→a.
    - (D) None of these.
      Answer: B
22. Removing unit productions ensures:
    - (A) A finite language.
    - (B) Simplified parsing.
    - (C) Deterministic parsing.
    - (D) None of these.
      Answer: B
23. A unit production is of the form:
    - (A) A→BA \rightarrow BA→B, where BBB is a non-terminal.
    - (B) A→aA \rightarrow aA→a.
    - (C) A→ABA \rightarrow ABA→AB.
    - (D) None of these.
      Answer: A
24. In Chomsky Normal Form, productions are of the form:
    - (A) A→BCA \rightarrow BCA→BC.
    - (B) A→aA \rightarrow aA→a.
    - (C) A→ϵA \rightarrow \epsilonA→ϵ.
    - (D) All of the above.
      Answer: D

25. In Greibach Normal Form, productions have the form:
    - (A) A→BCA \rightarrow BCA→BC.
    - (B) A→aαA \rightarrow a \alphaA→aα, where α\alphaα is a string of non-terminals.
    - (C) A→ϵA \rightarrow \epsilonA→ϵ.
    - (D) None of these.
      Answer: B
26. Parsing is the process of:
    - (A) Constructing a parse tree.
    - (B) Generating a derivation.
    - (C) Simplifying grammar rules.
    - (D) Both A and B.
      Answer: D
27. Which type of parser uses top-down derivations?
    - (A) Recursive descent parser.
    - (B) Shift-reduce parser.
    - (C) LR parser.
    - (D) All parsers.
      Answer: A
28. Simplification of CFG involves:
    - (A) Removing useless symbols.
    - (B) Eliminating null and unit productions.
    - (C) Reducing grammar size.
    - (D) All of the above.
      Answer: D
29. A CFG is simplified to:
    - (A) Reduce ambiguity.
    - (B) Minimize the number of productions.
    - (C) Simplify parsing.
    - (D) All of the above.
      Answer: D
30. In a reduced CFG, every symbol:
    - (A) Derives at least one terminal.
    - (B) Is reachable from the start symbol.
    - (C) Both A and B.
    - (D) None of the above.
      Answer: C

31. Eliminating null productions means:
    - (A) Removing ϵ\epsilonϵ-productions.
    - (B) Replacing ϵ\epsilonϵ with non-terminals.
    - (C) Rewriting the grammar without changing the language.
    - (D) Both A and C.
    Answer: D
32. A grammar with ϵ\epsilonϵ-productions:
    - (A) Is always ambiguous.
    - (B) Can generate an empty string.
    - (C) Cannot generate a finite language.
    - (D) All of the above.
    Answer: B
33. Unit productions are eliminated by:
    - (A) Replacing them with equivalent rules.
    - (B) Merging them into one rule.
    - (C) Removing all non-terminal rules.
    - (D) None of these.
    Answer: A
34. Chomsky Normal Form (CNF) requires all productions to be:
    - (A) A→BCA \rightarrow BCA→BC or A→aA \rightarrow aA→a.
    - (B) A→aA \rightarrow aA→a.
    - (C) A→ϵA \rightarrow \epsilonA→ϵ.
    - (D) None of these.
    Answer: A
35. Which of the following is a valid production in CNF?
    - (A) S→ABS \rightarrow ABS→AB
    - (B) A→aA \rightarrow aA→a
    - (C) Both A and B
    - (D) None of these
    Answer: C
36. Greibach Normal Form (GNF) ensures every production starts with:
    - (A) A terminal symbol.
    - (B) A non-terminal symbol.
    - (C) Both terminal and non-terminal symbols.
    - (D) ϵ\epsilonϵ.
    Answer: A

37. GNF is useful for:

- (A) Top-down parsing.
- (B) Bottom-up parsing.
- (C) Eliminating ambiguity.
- (D) None of these.
Answer: A

38. What is true about CNF and GNF?

- (A) Both simplify CFGs.
- (B) GNF is stricter than CNF.
- (C) Both are used for parser design.
- (D) All of the above.
Answer: D

39. The process of parsing includes:

- (A) Breaking input into tokens.
- (B) Constructing parse trees.
- (C) Checking syntax correctness.
- (D) Both B and C.
Answer: D

40. A bottom-up parser constructs the parse tree by:

- (A) Starting from the root.
- (B) Starting from the leaves.
- (C) Using only leftmost derivations.
- (D) None of these.
Answer: B

41. Which parsing method is more efficient for ambiguous grammars?

- (A) Top-down parsing.
- (B) Bottom-up parsing.
- (C) Recursive descent parsing.
- (D) None of these.
Answer: B

42. Recursive descent parsers work on:

- (A) Regular grammars.
- (B) Context-free grammars.
- (C) Context-sensitive grammars.
- (D) None of these.
Answer: B

43. Pumping Lemma is used to prove:

- (A) A language is regular.
- (B) A language is not regular.
- (C) A language is not context-free.
- (D) Both B and C.
  Answer: D

44. Which of the following satisfies the pumping lemma for CFL?

- (A) {anbncn|n≥0}\{ a^n b^n c^n \mid n \geq 0 \}{anbncn|n≥0}
- (B) {aibj|i,j≥0}\{ a^i b^j \mid i, j \geq 0 \}{aibj|i,j≥0}
- (C) {anbn|n≥0}\{ a^n b^n \mid n \geq 0 \}{anbn|n≥0}
- (D) {ww|w∈{a,b}∗}\{ ww \mid w \in \{a, b\}^* \}{ww|w∈{a,b}∗}
  Answer: C

45. Pumping lemma splits www into:

- (A) u,v,x,y,zu, v, x, y, zu,v,x,y,z.
- (B) uvxyzuvxyzuvxyz.
- (C) uxyzwuxyzwuxyzw.
- (D) uvwxyuvwxyuvwxy.
  Answer: B

46. CFGs are primarily used in:

- (A) Machine learning.
- (B) Compiler design.
- (C) Database indexing.
- (D) None of these.
  Answer: B

47. Syntax analysis uses CFG to:

- (A) Generate assembly code.
- (B) Identify syntax errors.
- (C) Check semantic rules.
- (D) None of these.
  Answer: B

48. A derivation tree is also known as:

- (A) Syntax tree.
- (B) Parse tree.
- (C) Abstract syntax tree.
- (D) Both A and B.
  Answer: D

49. A parse tree is constructed using:

- (A) Top-down derivation.
- (B) Bottom-up derivation.
- (C) Both A and B.
- (D) Only rightmost derivation.
  Answer: C

50. Derivation trees represent:

- (A) Syntax of the language.
- (B) Semantic meaning of the input.
- (C) The flow of program execution.
- (D) None of these.
  Answer: A