📘 Compiler Design – UNIT II

🔥 Syntax Analysis & Syntax Directed Translation

🎯 RGPV Exam Oriented Notes (Easy + Detailed + 14 Marks Ready)

🌟 UNIT II SYLLABUS

📌 Syntax Analysis

- CFG (Context Free Grammar)
- Top Down Parsing
- Brute Force Approach
- Recursive Descent Parsing
- Grammar Transformation
- Predictive Parsing
- Bottom Up Parsing
- Operator Precedence Parsing
- LR Parsers (SLR, LALR, LR)
- Parser Generation

📌 Syntax Directed Translation

- Syntax Directed Definition
- Syntax Tree Construction
- S-attributed Definition

- L-attributed Definition
- Top Down Translation
- Bottom Up Evaluation
- Recursive Evaluation
- Analysis of SDD

🔥 MOST IMPORTANT TOPICS

✅ CFG
✅ Recursive Descent Parsing
✅ Predictive Parsing
✅ Bottom Up Parsing
✅ Operator Precedence Parsing
✅ LR Parser
✅ Syntax Directed Definition
✅ Syntax Trees

🌟 1. SYNTAX ANALYSIS

📌 Definition

Syntax Analysis compiler ka second phase hota hai.

Ye:

- Grammar check karta hai
- Tokens ka structure verify karta hai

🎯 Main Work

✔ Program grammar check karna

✔ Parse tree banana

✔ Syntax errors detect karna

🌟 Syntax Analyzer is also called:

```
Parser
```

🎯 Diagram

```
Tokens
   ↓
Syntax Analyzer
   ↓
Parse Tree
```

🌟 Example

Program:

```
a = b + c;
```

Correct syntax hai ✔

Wrong:

```
= a b +
```

Syntax error ❌

🚨 PYQ ALERT

🔥 Define Syntax Analysis

🔥 Explain parser with diagram

🌟 2. CONTEXT FREE GRAMMAR (CFG)

🔥 VERY IMPORTANT

📌 Definition

CFG ek grammar hoti hai jo language ka syntax define karti hai.

🎯 Components of CFG

CFG me 4 components hote hain:

Symbol	Meaning
V	Variables
T	Terminals
P	Productions

Symbol	Meaning
S	Start Symbol

🎯 Example

```
E → E + T
E → T
T → id
```

🌟 Explanation

Symbol	Meaning
E	Expression
T	Term
id	Identifier

🌟 Parse Tree

```
      E
     /|\
    E + T
    |   |
    T  id
    |
   id
```

🌟 Advantages of CFG

✅ Program structure define karta hai

✅ Parsing easy banata hai

✅ Syntax errors detect karta hai

🚨 PYQ ALERT

🔥 Define CFG with example

🔥 Construct parse tree using CFG

🌟 3. TOP DOWN PARSING 🔥

IMPORTANT

📌 Definition

Top Down Parsing:

- Parse tree ko root se leaf tak construct karta hai.

🎯 Working

Start symbol se parsing start hoti hai.

```
Start Symbol
      ↓
Production Rules
```

```
      ↓
Input String
```

🌟 Example

Grammar:

```
S → aAB
A → b
B → c
```

Input:

```
abc
```

🌟 Advantages

✅ Simple implementation

✅ Easy to understand

🌟 Disadvantages

❌ Backtracking ho sakta hai

❌ Left recursion handle nahi karta

🌟 4. BRUTE FORCE APPROACH

📌 Definition

Sab possible parse trees try karta hai.

🎯 Problem

```
Very Slow
```

because:

- Multiple combinations try karta hai.

🌟 Drawback

❌ Time complexity high
❌ Efficient nahi hai

🌟 5. RECURSIVE DESCENT PARSING 🔥 VERY IMPORTANT

📌 Definition

Top down parsing technique hai.

Har non-terminal ke liye:

```
Separate recursive function
```

banaya jata hai.

🎯 Example Grammar

```
E → T + E
E → T
T → id
```

🌟 Recursive Functions

```
void E()
{
   T();
}
```

🌟 Advantages

✅ Easy coding

✅ Small grammar ke liye useful

🌟 Disadvantages

❌ Left recursion support nahi karta

❌ Backtracking issue

🚨 PYQ ALERT

🔥 Explain Recursive Descent Parsing

🔥 Advantages and disadvantages

🌟 6. TRANSFORMATION ON GRAMMAR

🔥 IMPORTANT

Grammar ko parser friendly banane ke liye transformations use hote hain.

📌 Left Recursion

Grammar:

$$A \rightarrow A\alpha \mid \beta$$

🎯 Problem

Infinite recursion create karta hai.

📌 Removing Left Recursion

Convert into:

```
A → βA'
A' → αA' | ϵ
```

🌟 Left Factoring

📌 Definition

Common prefix remove karna.

🎯 Example

Before:

```
A → ab | ac
```

After:

```
A → aA'
A' → b | c
```

🚨 PYQ ALERT

🔥 Remove left recursion

🔥 Explain left factoring

🌟 7. PREDICTIVE PARSING 🔥 VERY IMPORTANT

📌 Definition

Top down parser jo:

```
Backtracking use nahi karta
```

🎯 Uses:

- FIRST set
- FOLLOW set
- Parsing table

🌟 Predictive Parsing Table

Non Terminal	Input Symbol	Production
E	id	E → TE'

🌟 Advantages

✅ Fast parsing

✅ No backtracking

✅ Efficient

🌟 Disadvantages

❌ Left recursive grammar support nahi karta

🚨 PYQ ALERT

🔥 Explain Predictive Parsing

🔥 FIRST and FOLLOW based parsing

🌟 8. BOTTOM UP PARSING 🔥 VERY IMPORTANT

📌 Definition

Leaf se root ki taraf parse tree construct karta hai.

🎯 Working

```
Input String
      ↓
Reduce Operations
      ↓
Start Symbol
```

🌟 Shift Reduce Parsing

📌 Operations

Operation	Meaning
Shift	Input stack me push
Reduce	Grammar rule apply
Accept	Successful parsing
Error	Invalid string

🎯 Example

```
id + id
```

🚨 PYQ ALERT

🔥 Explain Bottom Up Parsing

🔥 Explain Shift Reduce Parser

🌟 9. OPERATOR PRECEDENCE PARSING

📌 Definition

Operators ki priority ke according parsing karta hai.

🎯 Example

```
a + b * c
```

First:

```
b * c
```

Then:

```
a + result
```

🌟 Advantages

✅ Fast parsing

✅ Expression parsing me useful

🌟 Disadvantages

❌ Complex grammar support nahi karta

🌟 10. LR PARSER 🔥 VERY VERY IMPORTANT

📌 Definition

Most powerful bottom up parser.

🌟 Meaning of LR

Symbol	Meaning
L	Left to Right scan
R	Rightmost derivation

🌟 Types of LR Parser

Parser	Full Form
SLR	Simple LR
LALR	Look Ahead LR
CLR	Canonical LR

🌟 SLR Parser

Simple version of LR parser.

🌟 LALR Parser

Most commonly used parser.

🌟 CLR Parser

Most powerful but large parsing table.

🌟 Advantages

✅ Efficient parsing
✅ Handles large grammar
✅ Better error detection

🌟 Disadvantages

❌ Complex implementation

🚨 PYQ ALERT

🔥 Explain LR parser
🔥 Difference between SLR, LALR and CLR

🌟 11. PARSER GENERATION

Automatic parser generation tools use hote hain.

Example:

```
YACC
BISON
```

🌟 12. SYNTAX DIRECTED DEFINITION (SDD) 🔥 IMPORTANT

📌 Definition

Grammar + Semantic Rules:

```
Syntax Directed Definition
```

🎯 Uses

✅ Syntax tree generation
✅ Type checking
✅ Translation

🌟 Syntax Directed Translation

Translation semantic rules ki help se hota hai.

🌟 13. SYNTAX TREE 🔥 IMPORTANT

📌 Definition

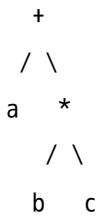
Program structure ka tree representation.

🎯 Example

Expression:

```
a + b * c
```

Tree:



🌟 14. S-ATTRIBUTED DEFINITION

📌 Definition

Sirf synthesized attributes use karta hai.

🎯 Bottom Up Parsing me use hota hai.

🌟 15. L-ATTRIBUTED DEFINITION

📌 Definition

Inherited + synthesized attributes use karta hai.

🎯 Top down parsing me useful.

🌟 16. TOP DOWN TRANSLATION

Parsing root se leaf ki taraf hoti hai.

🌟 17. BOTTOM UP EVALUATION

Leaf se root ki taraf evaluation hoti hai.

🌟 18. RECURSIVE EVALUATION

Attributes recursively calculate hote hain.

🌟 19. ANALYSIS OF SDD

Semantic rules ki correctness analyze ki jati hai.

🔥 MOST IMPORTANT 14-MARK QUESTIONS

1. Explain syntax analysis with diagram.
2. Explain CFG with parse tree.
3. Explain top down parsing.
4. Explain recursive descent parsing.
5. Explain predictive parsing.
6. Explain bottom up parsing.
7. Explain operator precedence parser.
8. Explain LR parser and its types.
9. Explain syntax directed definition.
10. Explain S-attributed and L-attributed definition.
11. Explain syntax tree construction.

⚡ One Night Revision

✔ Parser checks grammar
✔ CFG defines syntax rules
✔ Top Down = Root → Leaf
✔ Bottom Up = Leaf → Root
✔ Predictive parser uses FIRST/FOLLOW
✔ LR parser = most powerful parser
✔ Shift Reduce parser uses stack
✔ Syntax tree represents expressions
✔ S-attributed uses synthesized attributes
✔ L-attributed uses inherited + synthesized attributes