Parul® University

NAAC A++

Compiler design
Chapter-2: Introduction to syntax analysis

Asst.Prof. Vaibhavi Parikh
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

INDEX

Techniques for writing grammars for programming languages (removal left recursion, left Factor)

A context-free grammar is said to be left recursive if it contains a production rule where the non-terminal on the left-hand side of the rule also appears as the first symbol on the right-hand side.

In other words, the grammar is trying to define a non-terminal in terms of itself, creating a recursive loop.
This can be represented formally as −

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

Where −
A is a non-terminal symbol.
α represents a sequence of terminals and/or non-terminals.
β represents another sequence of terminals and/or non-terminals.

Eliminating Left Recursion

Let's illustrate this with our previous example −

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

We can eliminate the left recursion by introducing a new non-terminal 'A'` and rewriting the rule as follows −

$$A \rightarrow \beta A'$$
$$A' \rightarrow \alpha A' \mid \varepsilon$$

Left Factoring

After the left recursion, let us see the idea of left factoring.
While left recursion presents a parsing challenge, left factoring is a desirable property for top-down parsing.
It involves restructuring the grammar to eliminate common prefixes in production rules.

Importance of left factoring:

When a grammar has multiple production rules for a non-terminal with a common prefix, the parser faces ambiguity during the parsing process.
It has to choose between these rules without knowing which one will ultimately lead to a successful parse.
Left factoring helps in resolving this ambiguity by postponing the decision point, making the parsing process more efficient.

Performing Left Factoring

The process of left factoring involves identifying the longest common prefix (α) in the alternatives for a non-terminal and rewriting the rules to factor out this common prefix.

Consider a grammar with the following rules −

$A \rightarrow \alpha\beta1 | \alpha\beta2 | \ldots | \alpha\beta n | \gamma$

Here, 'α' is the longest common prefix in the first 'n' alternatives for non-terminal 'A'. We can left factor this as follows −

$A \rightarrow \alpha A' | \gamma$

$A \rightarrow \beta1 | \beta2 | \ldots | \beta n$

PPT Content Resources Reference Sample:

1. **Book Reference**

   Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Pearson Education.

2. **Journal Article**

   Muchnick, S. S., & Hecht, M. S. (2018). Advances in compiler optimization: Modern approaches for language processing. *Journal of Computer Science and Engineering*, 12(4), 233–245

3. **Website Reference**

   GeeksforGeeks. (2024). *Structure of a compiler*. Retrieved November 7, 2025, from https://www.geeksforgeeks.org/structure-of-compiler/ .

   1. **Conference Presentation**

      Sharma, R., & Patel, D. (2022). *Applications of compiler technology in modern software development*. Paper presented at the International Conference on Advanced Computing and Communication Systems (ICACCS), Chennai, India.

4. **Report**

   IEEE Computer Society. (2021). *Trends in programming language translation and compiler design*.

5. **Sources**

   TutorialsPoint. (2024). *Lexical Analysis in Compiler Design.*

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/