Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

Introduction to syntax analysis

Study Guide

Asst. Prof. Vaibhavi Parikh
CSE, PIT
Parul University

2.3 Parse Tree and Ambiguity

A **parse tree** (also called a **syntax tree** or **derivation tree**) is a **tree representation of the syntactic structure** of a string according to a grammar.
It shows **how a start symbol of a grammar derives (produces)** a given input string.

Structure of a Parse Tree

- **Root node** → represents the **start symbol** of the grammar.
- **Internal nodes** → represent **non-terminals**.
- **Leaf nodes** → represent **terminals (tokens)** of the input string.
- The tree visually shows the **derivation steps** according to production rules.

Example

Consider the grammar:

```
E → E + T | T
T → T * F | F
F → (E) | id
```

and the input string:

```
id + id * id
```

One possible parse tree is:

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

This shows that the expression is parsed as:

```
E → E + T → T + T → F + T → id + T → id + T * F → id + F * F → id + id * id
```

Ambiguity in Grammars

A **grammar is ambiguous** if **there exists at least one string** that can have **more than one distinct parse tree (or derivation)**.

Why It Happens

Ambiguity arises when the grammar does not clearly define **operator precedence** or **associativity** rules.

Example of Ambiguous Grammar

```
E → E + E | E * E | (E) | id
```

For the string:

```
id + id * id
```

There are **two possible parse trees**:

1. **(+ first)** → (id + id) * id
2. *(first)** → id + (id * id)

This means the grammar is **ambiguous** because the same input has **two different structures** (and hence two possible meanings).

How to Remove Ambiguity

To eliminate ambiguity, we rewrite the grammar to enforce **operator precedence and associativity**.

Example (Disambiguated Grammar)

```
E → E + T | T       // '+' has lowest precedence
T → T * F | F       // '*' has higher precedence
F → (E) | id        // parentheses and operands
```

Now:

- Multiplication `*` binds tighter than `+`
- Both operators are **left-associative**

This grammar is **unambiguous** for arithmetic expressions.

3. Summary of Key Concepts

- **Parse Tree:**

A hierarchical tree showing how a string is derived from a grammar's start symbol using production rules.

- Root → start symbol
- Internal nodes → non-terminals
- Leaves → terminals
  Example: For `id + id * id`, the parse tree shows the order of operations and structure of the expression.

- **Ambiguity:**

A grammar is **ambiguous** if a single string can have **more than one valid parse tree** (e.g., `id + id * id` can be parsed as either `(id + id) * id` or `id + (id * id)`).

- **Removing Ambiguity:**

Redefine grammar rules to enforce **operator precedence** and **associativity** (e.g., separate non-terminals for `+`, `*`, and operands).

Next Steps

- Explore other basic functions of Removal left factoring and left recursion .

References:

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 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/

4. **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.

5. **Report**

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

6. **Sources**

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

.

Parul® University
Vadodara, Gujarat

NAAC
GRADE A++

https://paruluniversity.ac.in/