**TABLE-DRIVEN LL PARSING**

Table-driven LL parsing involves feeding the input string to a parser as a series of tokens. This process can be thought of as a multi-step process, where the parser keeps track of the position in the input string and the current non-terminal in the parse tree. Each time a non-terminal is reduced, the parser outputs the corresponding syntactic structure.

- **Top-down parsing** reads the input string from left to right and builds the parse tree from the root to the leaves.

- **Bottom-up parsing** reads the input string from right to left and builds the parse tree from the leaves to the root.

In both cases, the parser uses a table that contains all possible grammar rules and their associated actions. The parser looks up the current non-terminal and input symbol to determine the next action.

---

**Examples**

Consider the following grammar:

```
E -> T * F  
F -> (E) | ID  
T -> E + T  | T * F  |
```

These rules define the syntax of arithmetic expressions. The grammar rules are as follows:

- `E` can be `T * F` or `T + F`.
- `F` can be `(E)` or `ID`.
- `T` can be `E + T` or `T * F`.

To parse an expression, the parser starts with the `E` production and then applies the appropriate rule based on the lookahead symbol.

---

**Error Handling**

Table-driven parsers handle errors by looking up the next rule that can be applied and then reducing the parse tree accordingly. This process is called **error recovery**.

- **Look-ahead** is used to decide which rule to apply next.

- **Error recovery** involves backtracking or shifting input symbols to recover from errors.

---

**Conclusion**

Table-driven LL parsing is a powerful technique for parsing context-free grammars. It is particularly useful for languages with simple grammatical structures. However, it may not be the best choice for languages with complex grammatical structures or languages where the grammar is not context-free.

---

**Notes for CSC 2/454, Sept. 2, 9, and 14, 2020**

In the next lecture, we will discuss algorithms for top-down and bottom-up parsing.

---

**References**

- [LL(1) Parsers](https://www.compilers.org/ll1-parser/)
- [Table-Driven Parsing](https://www.compilers.org/table-driven-parsing/)

---

**Questions**

1. What is the purpose of LL(1) parsers?
2. How does table-driven parsing work?
3. What is the difference between top-down and bottom-up parsing?
4. What are the advantages and disadvantages of table-driven parsing?