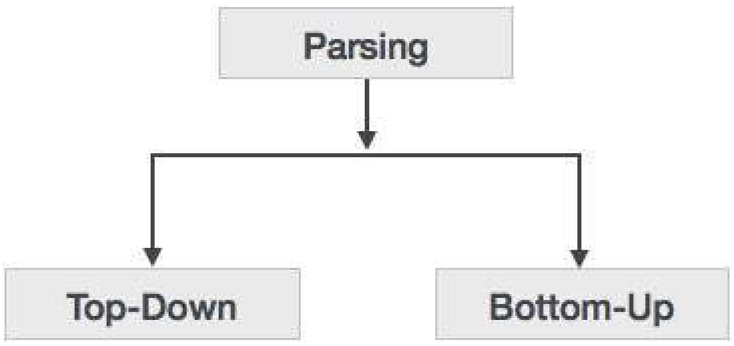
COMPILER DESIGN - TYPES OF PARSING

Syntax analyzers follow production rules defined by means of context-free grammar. The way the production rules are implemented *derivation* divides parsing into two types : top-down parsing and bottom-up parsing.



Top-down Parsing

When the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input, it is called top-down parsing.

- **Recursive descent parsing** : It is a common form of top-down parsing. It is called recursive as it uses recursive procedures to process the input. Recursive descent parsing suffers from backtracking.
- **Backtracking** : It means, if one derivation of a production fails, the syntax analyzer restarts the process using different rules of same production. This technique may process the input string more than once to determine the right production.

Bottom-up Parsing

As the name suggests, bottom-up parsing starts with the input symbols and tries to construct the parse tree up to the start symbol.

Example:

Input string : a + b * c

Production rules:

```
S → E
E → E + T
E → E * T
E → T
T → id
```

Let us start bottom-up parsing

```
a + b * c
```

Read the input and check if any production matches with the input:

```
a + b * c
T + b * c
E + b * c
E + T * c
E * c
E * T
E
S
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js