Modeling Computation

Introduction to Formal Languages and Automata
Parsing
Parsing

The first job of every compiler is to check syntactical correctness of a program.

CFGs provide compilers with the opportunity to parse programs in order to check for syntactical validity.
\[ S \rightarrow I = E; S \]
\[ S \rightarrow S \text{ while}(C)\{S\} S \]
\[ S \rightarrow \epsilon \]
\[ I \rightarrow a \mid b \mid \ldots \mid z \]
\[ C \rightarrow E > E \]
\[ E \rightarrow (E) \mid -(E) \mid (E + E) \mid (E \ast E) \mid (E/E) \mid I \]
Production rules can be applied in any order.
Thus, a string \( w \) can have many derivations.

Ex: derivation of \((a + (b * c))\)

Leftmost derivation:
\[
\varepsilon \Rightarrow (\varepsilon + \varepsilon) \Rightarrow (a + \varepsilon) \Rightarrow (a + (\varepsilon * \varepsilon)) \Rightarrow (a + (1 * \varepsilon))
\]

Rightmost derivation:
\[
E \Rightarrow (E + E) \Rightarrow (E + (E * E)) \Rightarrow (E + (E * \varepsilon)) \Rightarrow \ldots
\]
Visualize derivations with parse trees. $(a + (b \times c))$

All derivations with the same parse tree are equivalent.
Different parse trees for the same word means that the grammar is ambiguous.
Example

$S \rightarrow A \mid B \mid AB$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

\[ a \rightarrow aA \rightarrow aa \]

\[ S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaB \rightarrow aa \]
“Dangling Else” Example

\[
\text{if } \overline{A} \text{ then } \left( \text{if } \overline{B} \left( \text{then } \overline{C} \right) \text{else } \overline{D} \right)
\]