Define parse tree. Why parse tree construction is only possible for CFG ?

A parse tree is like a family tree for a sentence in a programming language or any other formal language. It shows how the sentence is constructed according to the rules of a Context-Free Grammar (CFG). Here's why parse tree construction is only possible for CFG:

1. Clear Structure: A parse tree follows a specific structure where each node represents a part of the sentence, either a terminal (like a word or symbol) or a non-terminal (like a rule or production). This structure aligns well with the rules of CFG.
2. Start Symbol as Root: In a parse tree, the root node always represents the start symbol of the grammar (usually denoted as 'S'). This aligns perfectly with CFG, where the start symbol is the initial point for constructing valid sentences.
3. Terminal and Non-terminal Labels: Every node in a parse tree is labeled either as a terminal (actual word or symbol in the language) or a non-terminal (a rule or production of the grammar). This labeling system is inherent to CFG as well.
4. Production Rules as Branches: The branches of the parse tree correspond to the production rules of the CFG. Each non-terminal node expands into its production rules, showing how smaller parts combine to form larger structures.
5. Terminal Symbols at Leaves: Terminal symbols, which are the actual words or symbols in the language, are always found at the leaves of the parse tree. This corresponds to the fact that in CFG, terminal symbols are the ultimate building blocks of sentences.

Related posts:

1. What are the types of passes in compiler ?
2. Discuss the role of compiler writing tools. Describe various compiler writing tools.
3. What do you mean by regular expression ? Write the formal recursive definition of a regular expression.
4. How does finite automata useful for lexical analysis ?
5. Explain the implementation of lexical analyzer.

6. Write short notes on lexical analyzer generator.
7. Explain the automatic generation of lexical analyzer.
8. Explain the term token, lexeme and pattern.
9. What are the various LEX actions that are used in LEX programming ?
10. Describe grammar.
11. Explain formal grammar and its application to syntax analyzer.
12. Define parse tree. What are the conditions for constructing a parse tree from a CFG ?
13. Describe the capabilities of CFG.
14. What is parser ? Write the role of parser. What are the most popular parsing techniques ? OR Explain about basic parsing techniques. What is top-down parsing ? Explain in detail.
15. What are the common conflicts that can be encountered in shift-reduce parser ?
16. Differentiate between top-down and bottom-up parser.Under which conditions predictive parsing can be constructed for a grammar ?
17. Differentiate between recursive descent parsing and predictive parsing.
18. What is the difference between S-attributed and L-attributed definitions ?
19. What is intermediate code generation and discuss benefits of intermediate code ?
20. Discuss symbol table with its capabilities ?
21. What are the symbol table requirements ? What are the demerits in the uniform structure of symbol table ?