Comsats University Islamabad, Attock Campus



BACHELOR OF COMPUTER SCIENCE

***Name:***

Muhammad Osman

***Registration no.***

FA21-BCS-029

***Course:***

Compiler Construction

***Submitted To:***

Sir Bilal Bukhari

Qno.5

**Semantic Analysis in a Mini Compiler**

**1. Overview**  
Semantic analysis ensures that a program complies with the logical rules of the language after passing through lexical analysis and parsing. While lexical analysis identifies tokens (such as keywords, variables, and operators), and parsing ensures the program adheres to syntactic rules (grammar), semantic analysis verifies correct use of variables, type compatibility, and function correctness.

Key tasks of semantic analysis include:

* **Type Checking**: Verifies that operations involving variables are valid according to their types.
* **Scope Checking**: Ensures variables and functions are declared before they are used.
* **Function Call Checking**: Ensures functions are called with the correct number and types of arguments.
* **Declaration Checking**: Verifies variables and functions are declared before use.

**2. Components of the Semantic Analyzer**

The following classes play key roles in semantic analysis:

**SymbolTable Class:**  
The SymbolTable class maintains information about declared variables and functions, enabling semantic analysis to look up variable types, function declarations, and argument types.

**SemanticAnalyzer Class:**  
The SemanticAnalyzer class conducts the core semantic analysis, checking for variable declarations, usage, expression types, and function call correctness.

**3. Integration with the Lexer and Parser**  
In a typical mini compiler, the Lexer breaks the source code into tokens, and the Parser organizes those tokens into an Abstract Syntax Tree (AST) based on the language's syntax rules. After parsing, semantic analysis checks the meaning and correctness of the program using the SymbolTable and SemanticAnalyzer classes.

**Integration with Lexer and Parser**

* **Lexer**: The Lexer generates a sequence of tokens, identifying keywords, operators, identifiers, and other elements.
* **Parser**: The Parser converts these tokens into an Abstract Syntax Tree (AST), representing the program structure.

Semantic analysis is an essential phase in the compiler pipeline that ensures the logical correctness of the program. By checking for type mismatches, undeclared variables, and incorrect function calls, the SemanticAnalyzer helps identify errors early in the compilation process. When integrated with the Lexer and Parser, it provides a thorough validation of program correctness before code generation.