Semantic Analysis

In Text: Chapter 3

Outline

- Static semantics
  - Attribute grammars
- Dynamic semantics
  - Operational semantics
  - Denotational semantics

Syntax vs. Semantics

- Syntax concerns the form of a valid program
- Semantics concerns its **meaning**
- Meaning of a program is important
  - It allows us to enforce rules, such as type consistency, which go beyond the form
  - It provides the information needed to generate an equivalent output program

Two types of semantic rules

- Static semantics
- Dynamic semantics

Static Semantics

- There are some characteristics of the structure of programming languages that are difficult or impossible to describe with BNF
  - E.g., type compatibility: a floating-point value cannot be assigned to an integer type variable, but the opposite is legal

Static Semantics

- The static semantics of a language is only indirectly related to the meaning of programs during execution; rather, it has to do with the legal forms of programs
  - Syntax rather than semantics
- Many static semantic rules of a language state its type constraints

Dynamic semantics

- It describes the meaning of expressions, statements, and program units
- Programmers need dynamic semantics to know precisely what statements of a language do
- Compiler writers need define the semantics of the languages for which they are writing compilers

Role of Semantic Analysis

- Following parsing, the next two phases of the "typical" compiler are
  - **semantic analysis**
  - (intermediate) code generation

Role of Semantic Analysis

- The principal job of the semantic analyzer is to enforce static semantics
  - Constructs a syntax tree (usually first)
  - Performs analysis of information that is gathered
  - Uses that information later during code generation

Conventional Semantic Analysis

- Compile-time analysis and run-time "actions" that enforce language-defined semantics
  - Static semantic rules are enforced at compile time by the compiler
    - Type checking
  - Dynamic semantic rules are enforced at runtime by the compiler-generated code
    - Bounds checking