Semantic Analysis

Zakia Zinat Choudhury
Lecturer
Department of Computer Science & Engineering
University of Rajshahi

1

Introduction

Semantics of a language provide meaning to its constructs, like tokens and syntax structure. Semantics **help interpret symbols, their types, and their relations with each other.** Semantic analysis judges whether the syntax structure constructed in the source program derives any meaning or not.

Semantic Analysis computes **additional information related** to the meaning of the program once the syntactic structure is known.

2

Introduction

In typed languages as C, semantic analysis involves

- ❑adding information to the symbol table and
- ❑performing type checking.

As for Lexical and Syntax analysis, also for Semantic Analysis we need both a Representation Formalism and an Implementation Mechanism.

3

Semantic Errors

There are some semantics errors that the semantic analyzer is expected to recognize:

- × **Type mismatch**
- × **Undeclared variable**
- × **Reserved identifier misuse.**
- × **Multiple declaration of variable in a scope.**
- × **Accessing an out-of-scope variable.**
- × **Actual and formal parameter mismatch.**

4

Syntax-Directed Translation

The Principle of Syntax Directed Translation states that **the meaning of an input sentence is related to its syntactic structure**, i.e., to its Parse-Tree.

By Syntax Directed Translations we indicate those formalisms for specifying translations for programming language constructs guided by context-free grammars.

We associate **Attributes** to the grammar symbols representing the language constructs.

Values for attributes are computed by **Semantic Rules** associated with grammar productions.

5

Syntax-Directed Translation

Evaluation of Semantic Rules may:

- Generate Code
- Insert information into the Symbol Table
- Perform Semantic Check
- Issue error messages

There are two notations for attaching semantic rules:

1. **Syntax Directed Definitions:** High-level specification hiding many implementation details. It is also called Attribute Grammars.

2. **Translation Schemes:** More implementation oriented. Indicate the order in which semantic rules are to be evaluated.

6

Syntax-Directed Definitions

A syntax-directed definition (SDD) is a context-free grammar together with, attributes and rules.

CFG + semantic rules = Syntax Directed Definitions

For example:

int a = “value”;

should not issue an error in lexical and syntax analysis phase, as it is lexically and structurally correct, but it should generate a semantic error as the type of the assignment differs. These rules are set by the grammar of the language and evaluated in semantic analysis.

The following tasks should be performed in semantic analysis:

- Scope resolution
- Type checking
- Array-bound checking

7

Attribute Grammar

Attribute grammar is a **special form of context-free grammar where some additional information** (attributes) **are added to one or more of its non-terminals in order to provide context-sensitive information.** Each attribute has well-defined domain of values, such as integer, float, character, string, and expressions.

Attribute grammar is a medium to provide semantics to the context-free grammar and it can help specify the syntax and semantics of a programming language. Attribute grammar (when viewed as a parse-tree) can pass **values or information among the nodes of a tree**.

8

Attribute Grammar

$$E \rightarrow E + T \ \{ E.value = E.value + T.value \}$$

The right part of the CFG contains the semantic rules that specify how the grammar should be interpreted. Here, the values of non-terminals E and T are added together, and the result is copied to the non-terminal E.

9

Attribute Grammar

Semantic attributes may be assigned to their values from their domain at the time of parsing and evaluated at the time of assignment or conditions.

Based on the way the attributes get their values, they can be broadly divided into two categories : **synthesized attributes and inherited attributes.**

10

SYNTHESIZED ATTRIBUTES

A synthesized attribute for a nonterminal A at a parse-tree node N is defined by a semantic rule associated with the production at N. These attributes get values from the attribute values of their child nodes.

To illustrate, assume the following production:

$$S \rightarrow ABC$$

If S is taking values from its child nodes (A,B,C), then it is said to be a synthesized attribute, as the values of ABC are synthesized to S.

11

INHERITED ATTRIBUTES

An inherited attribute for a nonterminal B at a parse-tree node N is defined by a semantic rule associated with the production at the parent of N. Inherited attributes can take values from parent and/or siblings.

As in the following production,

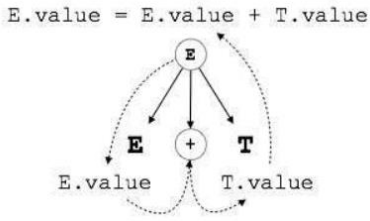
$$S \rightarrow ABC$$

A can get values from S, B and C. B can take values from S, A, and C. Likewise, C can take values from S, A, and B.

12

S-attributed SDT

If an SDT uses only synthesized attributes, it is called as S-attributed SDT. These attributes are evaluated using S-attributed SDTs that have their semantic actions written after the production (right hand side).



As depicted above, attributes in S-attributed SDTs are evaluated in bottom-up parsing, as the values of the parent nodes depend upon the values of the child nodes.

13

L-attributed SDT

This form of SDT uses both synthesized and inherited attributes with restriction of not taking values from right siblings.

In L-attributed SDTs, a non-terminal can get values from its parent, child, and sibling nodes.

As in the following production

$$S \rightarrow ABC$$

S can take values from A, B, and C (synthesized). A can take values from S only. B can take values from S and A. C can get values from S, A, and B. No non-terminal can get values from the sibling to its right.

Attributes in L-attributed SDTs are evaluated by depth-first and left-to-right parsing manner.

We may conclude that if a definition is S-attributed, then it is also L-attributed as L-attributed definition encloses S-attributed definitions.

14

Example

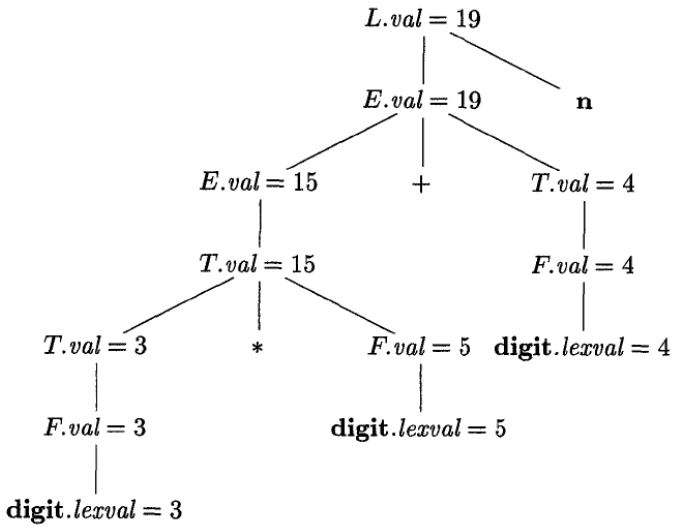
In the SDD, each of the non terminals has a single synthesized attribute, called **val**. We also suppose that the terminal digit has a synthesized attribute **lexval**, which is an integer value returned by the lexical analyzer.

The rule for production 1, $L \rightarrow E\ \mathbf{n}$, sets $L.val$ to $E.val$,

	PRODUCTION	SEMANTIC RULES
1)	$L \rightarrow E\ \mathbf{n}$	$L.val = E.val$
2)	$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3)	$E \rightarrow T$	$E.val = T.val$
4)	$T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
5)	$T \rightarrow F$	$T.val = F.val$
6)	$F \rightarrow (E)$	$F.val = E.val$
7)	$F \rightarrow \mathbf{digit}$	$F.val = \mathbf{digit}.\text{lexval}$

15

Annotated Parse Tree for $3 * 5 + 4\ \mathbf{n}$



Example

16