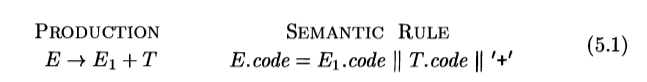
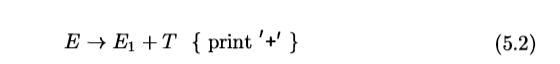
**Chapter 5 Syntax-Directed Translation**

We associate information with a language construct by attaching attributes to the grammar symbol(s) representing the construct, as discussed in Section 2.3.2. A syntax-directed definition specifies the values of attributes by associating semantic rules with the grammar productions. For example, an infix-to-postfix translator might have a production and rule



This production has two nonterminals, E and T; the subscript in El distinguishes the occurrence of E in the production body from the occurrence of E as the head; Both E and T have a string-valued attribute code. The semantic rule specifies that the string E. code is formed by concatenating E1. code, T. code, and the character ' +'. While the rule makes it explicit that the translation of E is built up from the translations of E1, T, and '+', it may be inefficient to implement the translation directly by manipulating strings.

From Section 2.3.5, a syntax-directed translation scheme embeds program fragments called semantic actions within production bodies, as in



By convention, semantic actions are enclosed within curly braces. (If curly braces occur as grammar symbols, we enclose them within single quotes, as in

'{' and '}'.) The position of a semantic action in a production body determines the order in which the action is executed. In production (5.2), the action occurs at the end, after all the grammar symbbls; in general, semantic actions may occur at any position in a production body.

Between the two notations, syntax-directed definitions can be more readable, and hence more useful for specifications. However, translation schemes can be more efficient, and hence more useful for implementations.

The most general approach to syntax-directed translation is to construct a parse tree or a syntax tree, and then to compute the values of attributes at the nodes of the tree by visiting the nodes of the tree. In many cases, translation can be done during parsing, without building an explicit tree. We shall therefore study a class of syntax-directed translations called "L-attributed translations" (L for lef t-to-right), which encompass virtually all translations that can be performed during parsing. We also study a smaller class, called "S-attributed tran 'slations" (8 for synthesized), which can be performed easily in connection with a bottom-up parse.

**5.1 Syntax-Directed Definitions**

A syntax-directed definition (SDD) is a context-free grammar together with attributes and rules. Attributes are associated with grammar symbols and rules are associated with productions. If X is a symbol and a is one of its attributes, then we write X.a to denote the value of a at a particular parse-tree node labeled X. If we implement the nodes of the parse tree by records or objects, then the attributes of X can be implemented by data fields in the records that represent the nodes for X. Attributes may be of any kind: numbers, types, table references, or strings, for instance. The strings may even be long sequences of code, say code in the intermediate language used by a compiler.

**5.1.1 Inherited and Synthesized Attributes**

We shall deal with two kinds of attributes for nonterminals:

1. 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. Note that the production must have A as its head. A synthesized attribute at node N is defined only in terms of attribute values at the children of N and at N itself.

2. 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. Note that the production must have B as a symbol in its body. An inherited attribute at node N is defined only in terms of attribute values at N's parent, N itself, and N's siblings.

An Alternative Definition of Inherited Attributes No additional translations are enabled if we allow an inherited attribute B.c at a node N to be defined in terms of attribute values at the children of N, as well as at N itself, at its parent, and at its siblings. Such rules can be "simulated" by creating additional attributes of B, say B ,Cl, B .C2, . .. • These are synthesized attributes that copy the needed attributes of the children of the node labeled B. We then compute B.c as an inherited attribute, using the attributes B ,Cl, B .C2, ..• in place of attributes at the children. Such attributes are rarely needed in practice.

While we do not allow an inherited attribute at node N to be defined in terms of attribute values at the children of node N, we do allow a synthesized attribute at node N to be defined in terms of inherited attribute values at node N itself. Terminals can have synthesized attributes, but not inherited attributes. Attributes for terminals have lexical values that are supplied by the lexical analyzer; there are no semantic rules in the SDD itself for computing the value of an attribute for a terminal.

**Example 5.1**: The SDD in Fig. 5.1 is based on our familiar grammar for arithmetic expressions with operators + and \*. It evaluates expressions terminated by an endmarker n. In the SDD, each of the nonterminals 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.

Figure 5.1: Syntax-directed definition of a simple desk calculator

The rule for production 1, L -+ En, sets L.val to E.val, which we shall see is the numerical value of the entire expression.

Production 2, E -+ El + T, also has one rule, which computes the val attribute for the head E as the sum of the values at El and T. At any parsetree node N labeled E, the value of val for E is the sum of the values of val at the children of node N labeled E and T.

be the same as the value of val at the child for T. Production 4 is similar to the second production; its rule multiplies the values at the children instead of adding them. The rules for produ.ctions 5 and 6 copy values at a child, like that for the third production. Production 7 gives F. val th� value of a digit, that is, the numerical value of the token digit that the lexical analyzer returned.