SYNTAX DIRECTED DEFINITION

- 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.

PRODUCTION	SEMANTIC RULE
$E \to E_1 + T$	$E.code = E_1.code \parallel T.code \parallel '+'$

- This production has two non-terminals, 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 ' +'.
- A syntax-directed translation scheme embeds program fragments called semantic actions within production bodies, as in

$$E \to E_1 + T \quad \{ \text{print } '+' \}$$

- 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.

SDD:

- 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 particular parse-tree node labeled X.

Inherited and Synthesized Attributes

1. Synthesized attribute:

- 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.
- A synthesized attribute at node N is defined only in terms of attribute values at the children of N and at N itself.

2. Inherited Attribute:

- 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.
- An inherited attribute at node N is defined only in terms of attribute values at N's parent, N itself, and N's siblings.

Example:

The SDD in Figure below is based on arithmetic expressions with operators + and *. It evaluates expressions terminated by an end marker 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.

	PRODUCTION	SEMANTIC RULES
1)	$L \rightarrow E$ **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)	$F \rightarrow$ **digit**	$F.val =$ **digit**.lexval

- The rule for production 1, L -> E n, 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 parse tree 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.
- An SDD that involves only synthesized attributes is called S-attributed;
- In an S-attributed SDD, each rule computes an attribute for the non-terminal at the head of a production from attributes taken from the body of the production.
- An SDD without side effects is sometimes called an attribute grammar. The rules in an attribute grammar define the value of an attribute purely in terms of the values of other attributes and constants.

Evaluating an SDD at the Nodes of a Parse Tree:

- The rules of an SDD are applied by first constructing a parse tree and then using the rules to evaluate all of the attributes at each of the nodes of the parse tree.
- A parse tree, showing the value(s) of its attribute(s) is called an annotated parse tree. Before we can evaluate an attribute at a node of a parse tree, we must evaluate all the attributes upon which its value depends.
- For example, if all attributes are synthesized, then we must evaluate the attributes at all of the children of a node before we can evaluate the attribute at the node itself.

Evaluation Orders for Syntax Directed Definitions:

- Dependency graphs are a useful tool for determining an evaluation order for the attribute instances in a given parse tree.
- While an annotated parse tree shows the values of attributes, a dependency graph helps us determine how those values can be computed.
- The two important classes of SDD are the "S-attributed" and "L-attributed" SDD's

Dependency Graphs:

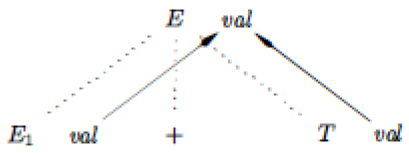
- A dependency graph depicts the flow of information among the attribute instances in a particular parse tree; an edge from one attribute instance to another means that the value of the first is needed to compute the second.
- Edges express constraints implied by the semantic rules.

Example:

Consider the following production and rule

PRODUCTION	SEMANTIC RULE
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$

- At every node N labeled E, with children corresponding to the body of this production, the synthesized attribute val at N is computed using the values of val at the two children, labeled E and T.
- As a convention, we shall show the parse tree edges as dotted lines, while the edges of the dependency graph are solid.



Ordering the Evaluation of Attributes:

- The dependency graph characterizes the possible orders in which we can evaluate the attributes at the various nodes of a parse tree. If the dependency graph has an edge from node M to node N, then the attribute corresponding to M must be evaluated before the attribute of N.
- Thus, the only allowable orders of evaluation are those sequences of nodes N_1 , N_2,…, N_k such that if there is an edge of the dependency graph from N_i to N_j , then $i<j$. Such an ordering embeds a directed graph into a linear order, and is called a topological sort of the graph.