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

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 ofa at a particular parse-tree node labeled X.

Inherited and Synthesized At tributes

There are two kinds of attributes for nonterminals:

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.
- 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.

Inherited Attribute :

- An inherited attribute for a nonterminal B at a parse-tree nodeN is defined by a semantic rule associated with the productionat 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.

Example:

	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

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.

  - ✓ For each parse-tree node, say a node labeled by grammar symbol X, the dependency graph has a node for each attribute associated with X.
  - ✓ Suppose that a semantic rule associated with a production p defines the value of synthesized attribute A.bin terms of the value of X.C. Then, the dependency graph has an edge from X.C to A.b.
  - ✓ Suppose that a semantic rule associated with a production p defines the value of inherited attribute B.cin terms of the value of X.a. Then, the dependency graph has an edge from X.a to B.c.

Example:

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