Module 5
Compilers: Analysis Phase

Semantic analysis

- Syntax directed definitions

Semantic Analysis

- Parsing only verifies that the program consists of tokens arranged in a syntactically valid combination.

- Semantic analysis checks whether they form a sensible set of instructions in the programming language.

- For a program to be semantically valid, all variables, functions, classes, etc. must be properly defined, expressions and variables must be used in proper ways

Semantic Analysis

- A large part of semantic analysis consists of tracking variable/function/type declarations and type checking

- **Type checking** is the process of verifying that each operation executed in a program respects the type system of the language.

- This means that all operands in any expression are of appropriate types and number.

Semantic Analysis

- A language is considered *strongly typed* if each and every type error is detected during compilation.

- Type checking can be done compilation, during execution, or divided across both

Semantic Analysis

- **Static type checking** is done at compile time.
- The information the type checker needs is obtained via declarations and stored in a master symbol table.
- After this information is collected, the types involved in each operation are checked.
- It is very difficult for a language that only does static type checking to meet the full definition of strongly typed.

Semantic Analysis

- **Dynamic type checking** is implemented by including type information for each data location at runtime.

- Dynamic type checking clearly comes with a runtime performance penalty, but it can report errors that are not possible to detect at compile time

Semantic Analysis

- Translation techniques like **Syntax Directed Definition (SDD)** is applied to type checking and intermediate-code generation.

Syntax Directed Definitions(SDD)

- We can associate information with a language construct by attaching attributes to the grammar symbols.

- A syntax directed definition specifies the values of attributes by associating semantic rules with the grammar productions.

- Attributes are associated with grammar symbols and rules are associated with productions

- Attributes may be of many kinds: numbers, types, table references, strings, etc.

Syntax Directed Definitions(SDD)

- 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.

SDD for a simple desk calculator

Production	Semantic Rules
L → E n	Print (E.val)
L → E1 + T	E.val = E1.val + T.val
E → T	E.val = T.val
T → T1 * F	T.val = T1.val * F.val
T → F	T.val = F.val
F → (E)	F.val = E.val
F → digit	F.val = digit.lexval

Syntax Directed Definitions(SDD)

- Two Types:

  - Synthesized Attributes
  - Inherited Attributes

Synthesized Attributes

- A synthesized attribute for a non terminal A at a parse tree node N is defined by a semantic rule associated with the production at N.

- 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.

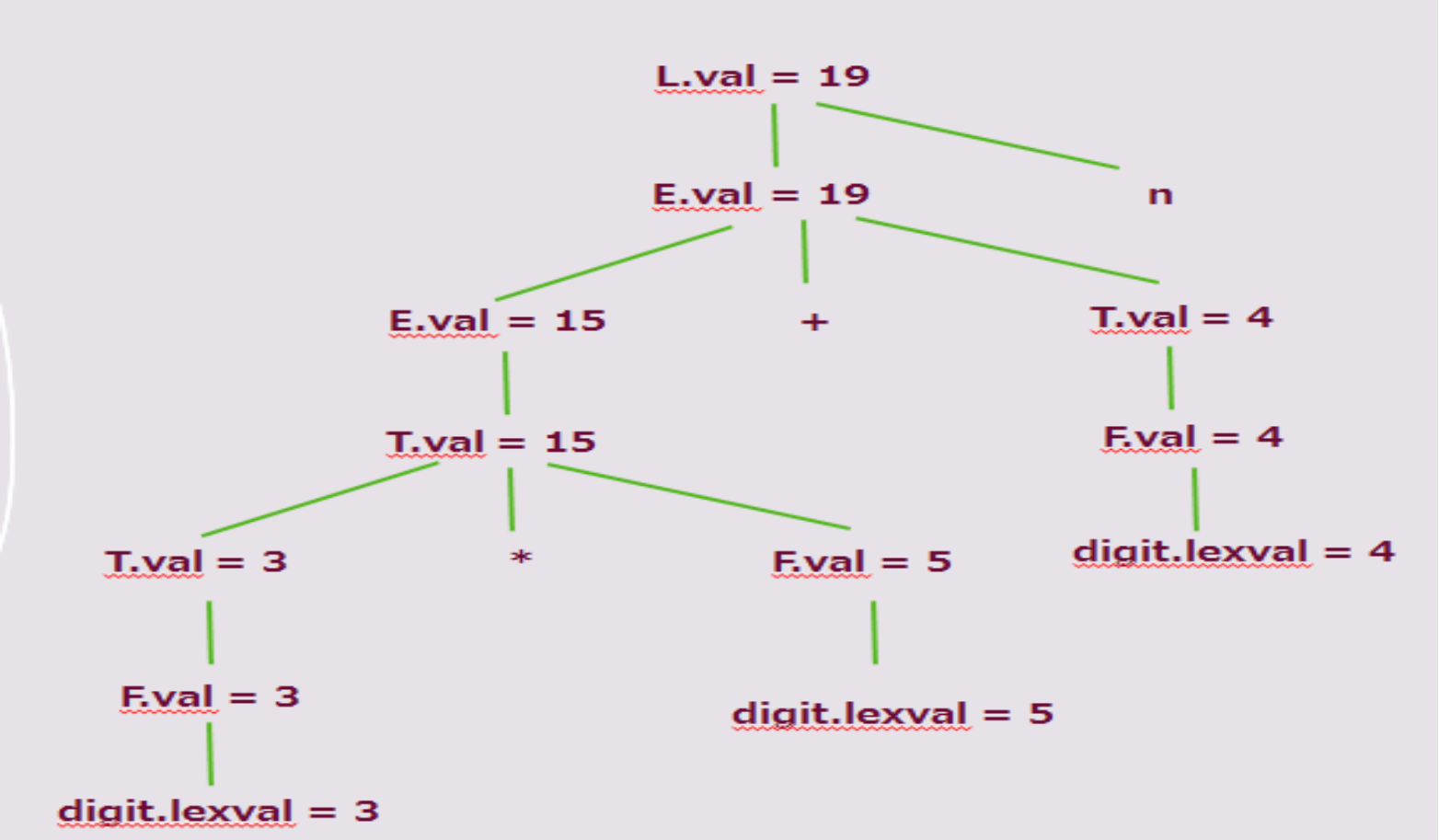
- An syntax directed definition that uses Synthesized attributes exclusively is said to be an S-attributed definition.

Synthesized Attributes

- A parse tree for an S-attributed definition can be annotated (interpreted) by evaluating semantic rules for attributes at each node

- To implement S-attributed definition LR parser can be used i.e. bottom up parsing is used

Synthesized Attributes



Annotated Parse Tree for 3*5+4n

Inherited Attributes

- An inherited attribute for a non terminal B at a parse-tree node N is defined by a semantic rule associated with the production at the parent of N.

- 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.

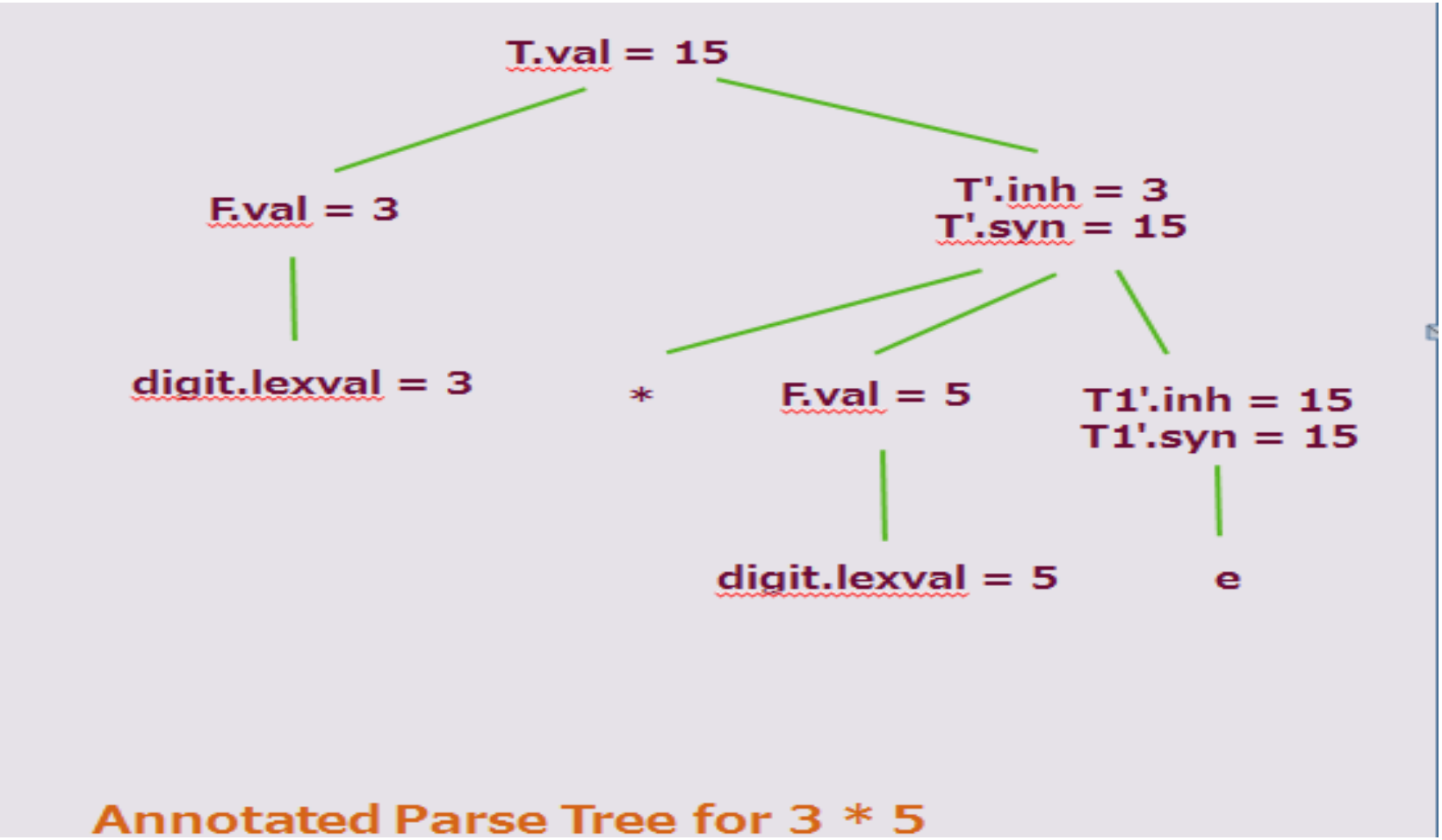
- Inherited attributes are convenient for expressing dependence of a program language construct on the context in which appears

Inherited Attributes

Production	Semantic Rules
T → F T'	T'.inh = F.val T.val = T'.syn
T' → * F T1'	T1'.inh = T'.inh * F.val T'.syn = T1'.syn
T' → e	T'.syn = T'.inh
F → digit	F.val = digit.lexval

SDD with inherited attribute L.in

Inherited Attributes



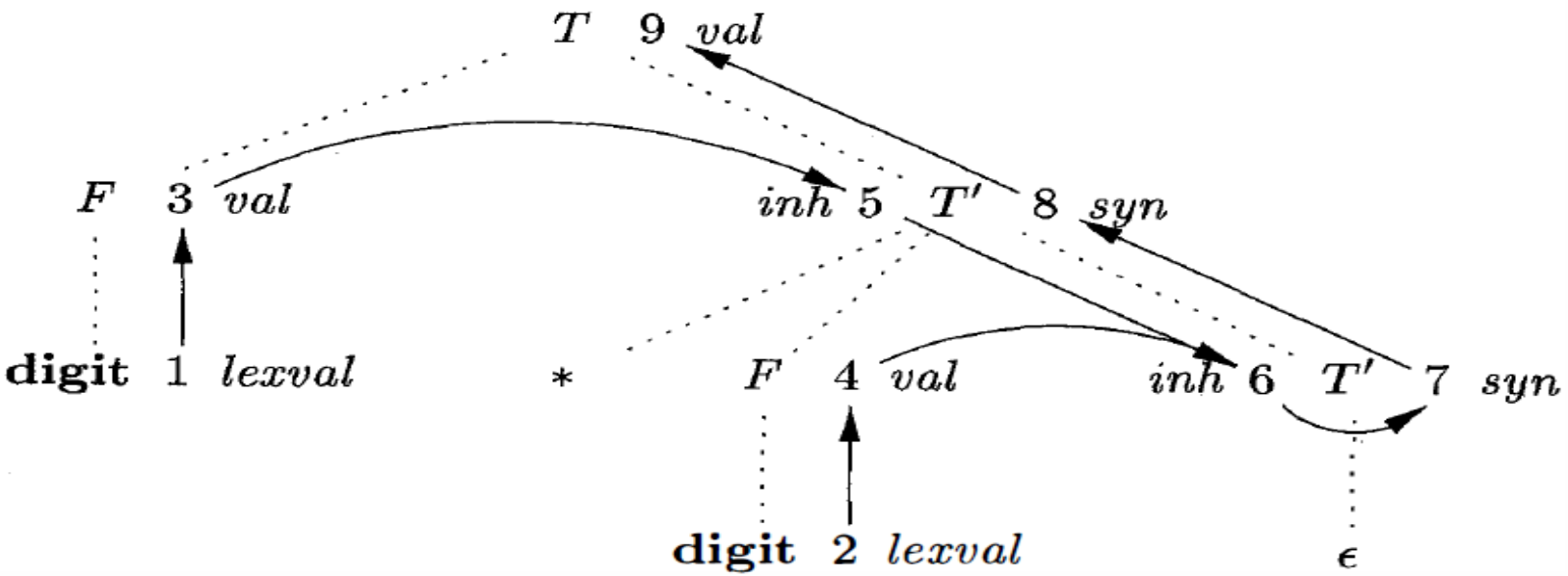
Annotated Parse Tree for 3 * 5

Evaluation Order for SDDs

- **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.

Dependency Graph



- A dependency graph shows 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.

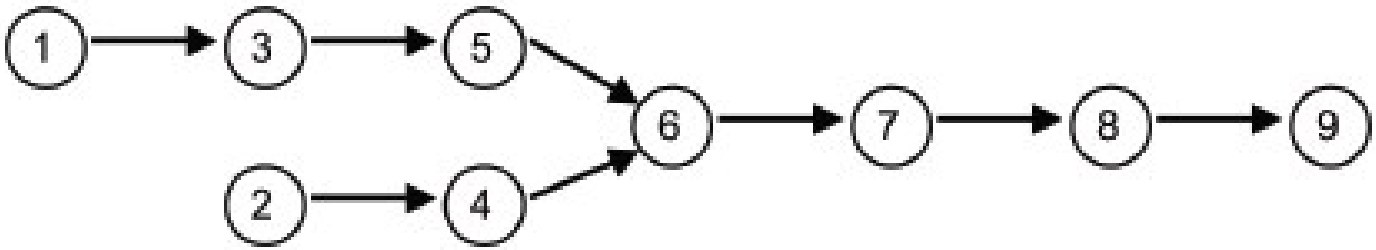
Topological Sort of Dependency Graph

- A dependency graph characterizes the possible order in which we can evaluate the attributes at various nodes of a parse tree.
- If there is an edge from node M to N, then attribute corresponding to M is first to be evaluated before evaluating N.
- Thus the only allowable orders of evaluation are N_1, N_2, ..., N_k such that if there is an edge from N_i to N_j then $i < j$.

Topological Sort of Dependency Graph

- Such an ordering embeds a directed graph into a linear order, and is called a topological sort of the graph.
- If there is any cycle in the graph, then there are no topological sorts; that is, there is no way to evaluate the SDD on this parse tree.
- If there are no cycles, however, then there is always at least one topological sort.

Topological Sort of Dependency Graph



Base sequences are {1 3 5} and {2 4} with the suffix {6 7 8 9} being constant as the last nodes of the topological sorting need to remain fixed.

1 3 5 2 4 6 7 8 9, 1 3 2 5 4 6 7 8 9, 1 2 3 5 4 6 7 8 9,

1 3 2 4 5 6 7 8 9, 1 2 3 4 5 6 7 8 9, 1 2 4 3 5 6 7 8 9,

2 1 3 5 4 6 7 8 9, 2 1 3 4 5 6 7 8 9, 2 1 4 3 5 6 7 8 9,

2 4 1 3 5 6 7 8 9

S-attributed Definition

- Uses only synthesized attributes
- Semantic actions are placed at right end of the production
- Also called as postfix SDT
- Attributes are evaluated during bottom up parsing

L-attributed Definition

- Uses both inherited and synthesized attributes.
- Each inherited attribute is restricted to inherit either from parent or left sibling
- Semantic actions are placed anywhere on the RHS
- Attributes are evaluated by traversing parse tree depth first and left to right