SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

19CS502- Automata Theory and Compiler Design

III YEAR / V SEMESTER

UNIT – 3 : Semantics and Context Sensitive Features

Topic : **Syntax Directed Translation**

Syntax Directed Translation

- Associate information with the programming language constructs by attaching attributes to grammar symbols.

- Values of these attributes are evaluated by the semantic rules associated with the production rules.

- An attribute may hold almost any thing.
    - ➢ a string, a number, a memory location, a complex record.

Grammar + Semantic Rules = SDT

Syntax Directed Translation

There are two ways to represent the semantic rules associated with grammar symbol

- ❑ Syntax Directed Definitions (SDD)
- ❑ Syntax Directed Translation Schemes(SDTs)

Syntax-Directed Definitions:

- ✓ Give high-level specifications for translations
- ✓ Hide many implementation details such as order of evaluation of semantic actions.
- ✓ We associate a production rule with a set of semantic actions, and we do not say when they will be evaluated.

Syntax Directed Translation Schemes:

- ✓ Indicate the order of evaluation of semantic actions associated with a production rule.
- ✓ In other words, translation schemes give a little bit information about implementation details.

Eg : SDD vs SDT scheme

SDT Scheme

E → E +T	{print ‘+’}
E → E –T	{print ‘–’}
E → T	
T → 0	{print ‘0’ }
T → 1	{print ‘1’ }
…	
T → 9	{print ‘9’ }

SDD

E → E +T	E.code = E.code||T.code||’+’
E → E –T	E.code = E.code||T.code||’–’
E → T	E.code = T.code
T → 0	T.code =‘0’
T → 1	T.code =‘1’
…	
T → 9	T.code =‘9’

Syntax Directed Translation

Conceptually with both the syntax directed translation and translation scheme we

- Parse the input token stream
- Build the parse tree
- Traverse the tree to evaluate the semantic rules at the parse tree nodes.

Input string → parse tree → dependency graph → evaluation order for semantic rules

Conceptual view of syntax directed translation

Syntax-Directed Definitions

- A syntax-directed definition is a generalization of a context-free grammar in which:
  - ➢ Each grammar symbol is associated with a set of attributes.
  - ➢ This set of attributes for a grammar symbol is partitioned into two subsets called
    - ❑ **Synthesized** and
    - ❑ **Inherited** attributes of that grammar symbol.
  - ➢ Each production rule is associated with a set of semantic rules.

- The value of an attribute at a parse tree node is defined by the semantic rule associated with a production at that node.
- The value of a **Synthesized attribute** at a node is computed from the values of attributes at the children in that node of the parse tree
- The value of an **Inherited attribute** at a node is computed from the values of attributes at the siblings and parent of that node of the parse tree

Syntax-Directed Definitions

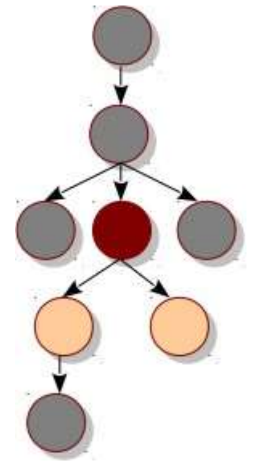
Examples:

- Synthesized attribute : E→E1+E2 {E.val =E1.val + E2.val}
- Inherited attribute :A→XYZ {Y.val = 2 * A.val}

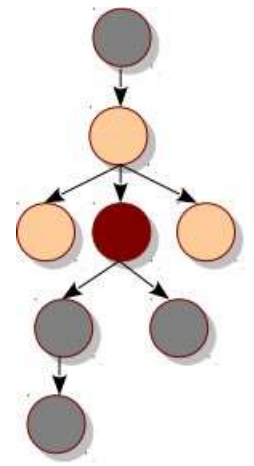
- Semantic rules set up dependencies between attributes which can be represented by a dependency graph.

- This dependency graph determines the evaluation order of these semantic rules.

- Evaluation of a semantic rule defines the value of an attribute. But a semantic rule may also have some side effects such as printing a value.



Synthesized attribute



Inherited attribute

Syntax-Directed Definitions Example

Production	Semantic Rules
L → E $	print(E.val)
E → 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

Symbols E, T, and F are associated with a synthesized attribute val.

The token digit has a synthesized attribute lexval (it is assumed that it is evaluated by the lexical analyzer).

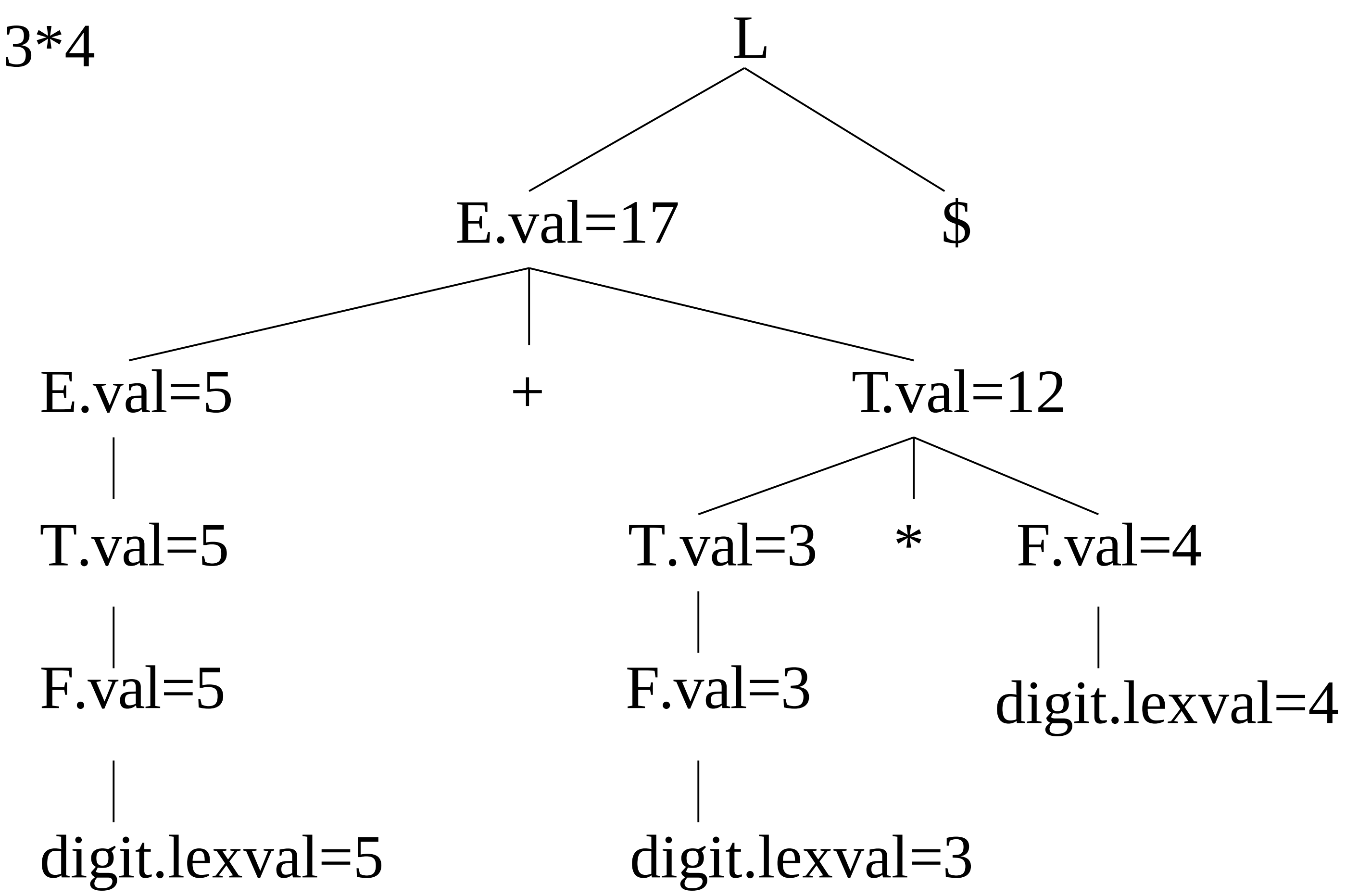
Terminals are assumed to have synthesized attributes only. Values for attributes of terminals are usually supplied by the lexical analyzer.

The start symbol does not have any inherited attribute unless otherwise stated.

Annotated Parse Tree

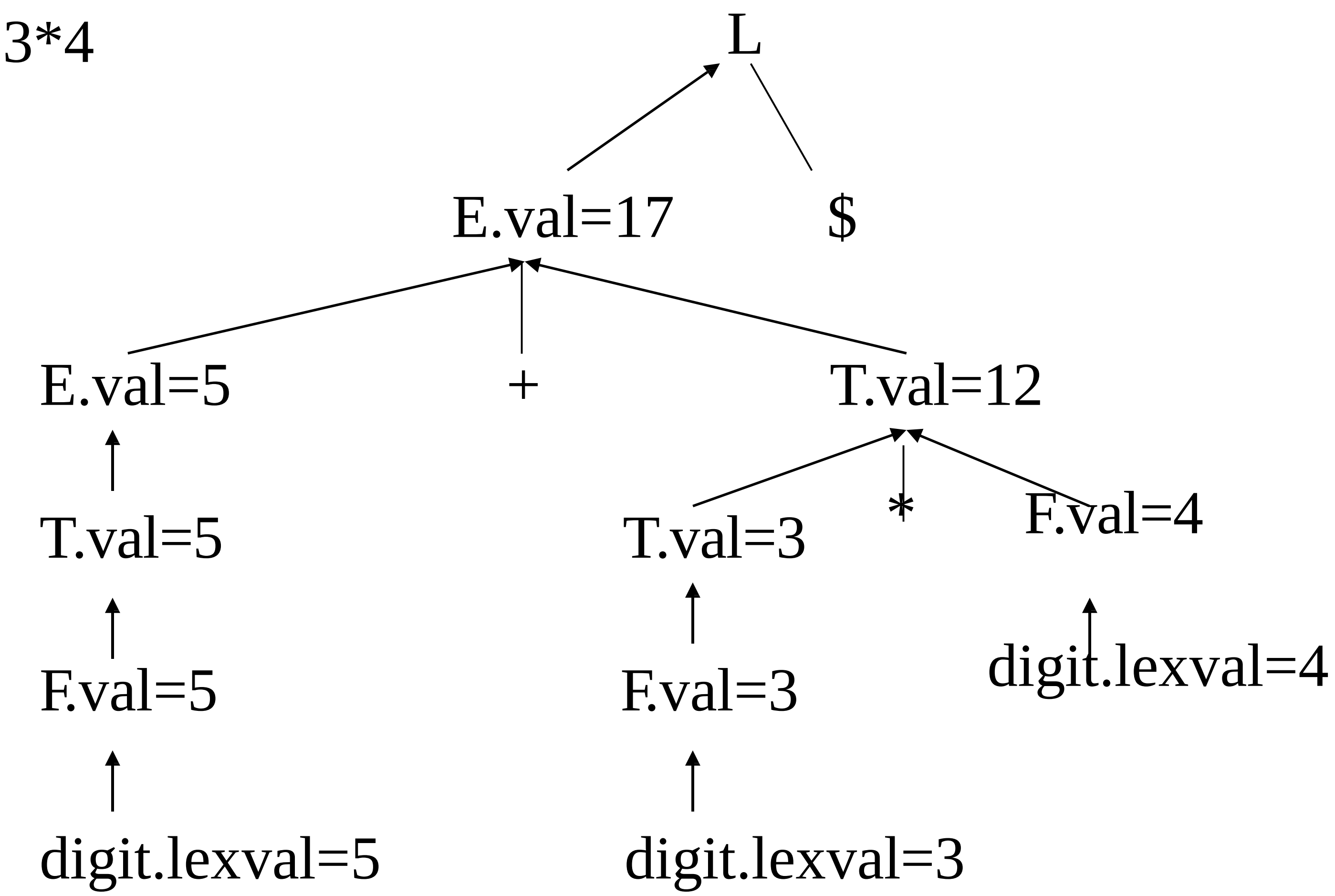
Production	Semantic Rules
L → E $	print(E.val)
E → E1 + T	E.val = E1.val + T.val
E → T	E.val = T.val
T → T1 * F	T.val = T1.val * F.val
T → F	T.val = F.val
F → (E)	F.val = E.val
F → digit	F.val = digit.lexval

Input: 5+3*4


L
E.val=17
$
E.val=5
+
T.val=12
T.val=5
T.val=3
*
F.val=4
F.val=5
F.val=3
digit.lexval=4
digit.lexval=5
digit.lexval=3

Dependency Graph

Input: 5+3*4



SDD – Inherited Attributes

Production	**Semantic Rules**
D → T L	L.in = T.type
T → **int**	T.type = integer
T → **real**	T.type = real
L → L_1 **id**	L_1.in = L.in, addtype(**id**.entry,L.in)
L → **id**	addtype(**id**.entry,L.in)

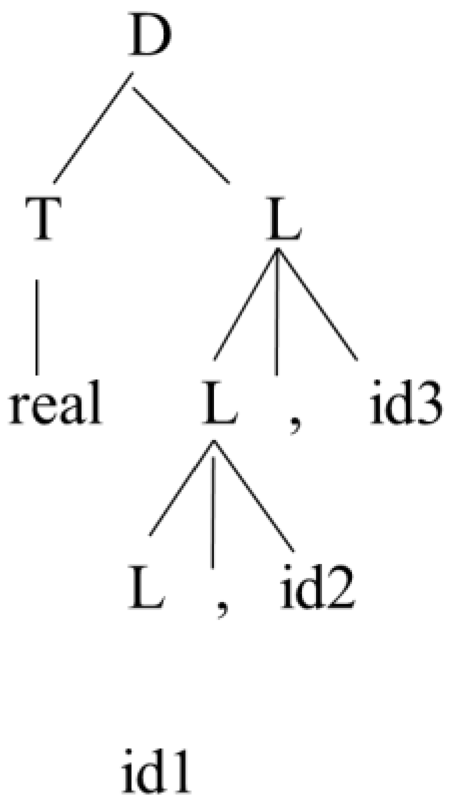
1. Symbol T is associated with a Synthesized attribute *type*.

2. Symbol L is associated with an inherited attribute *in*.

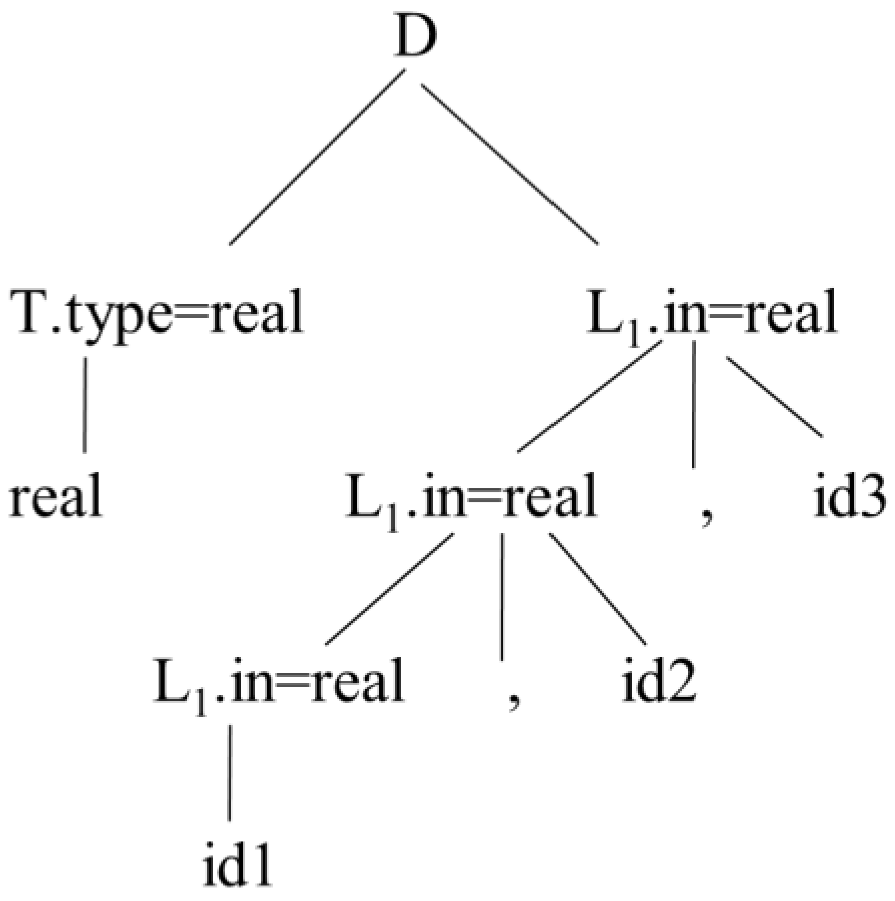
Annotated parse tree

Input: `real p,q,r`

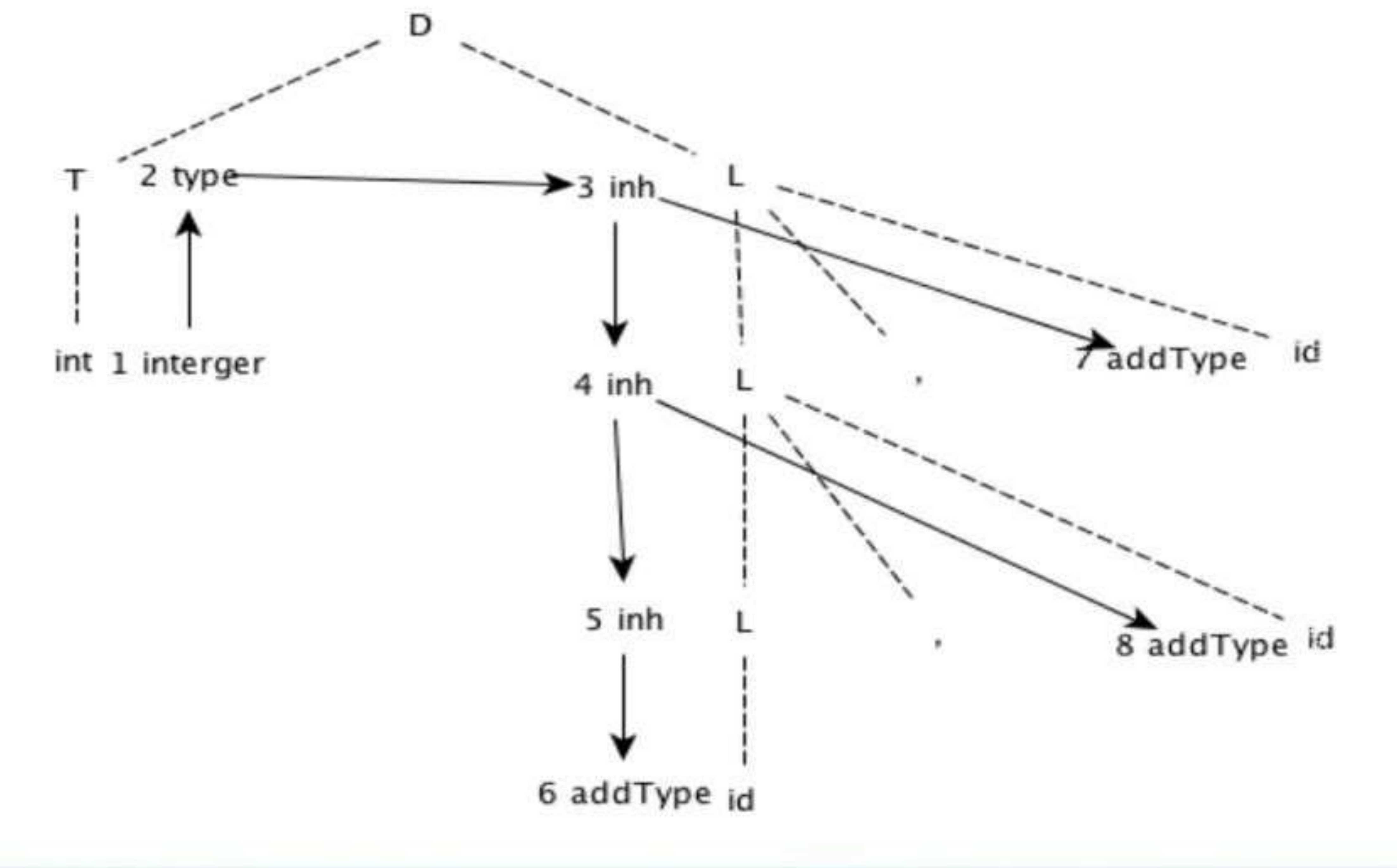
parse tree



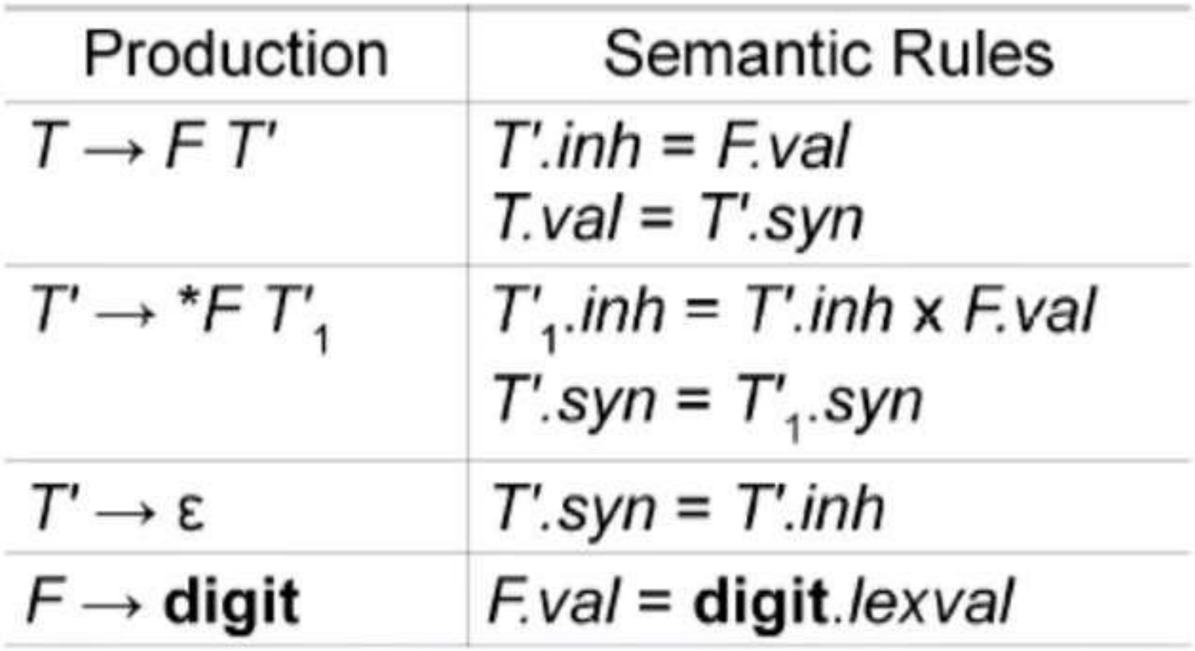
annotated parse tree



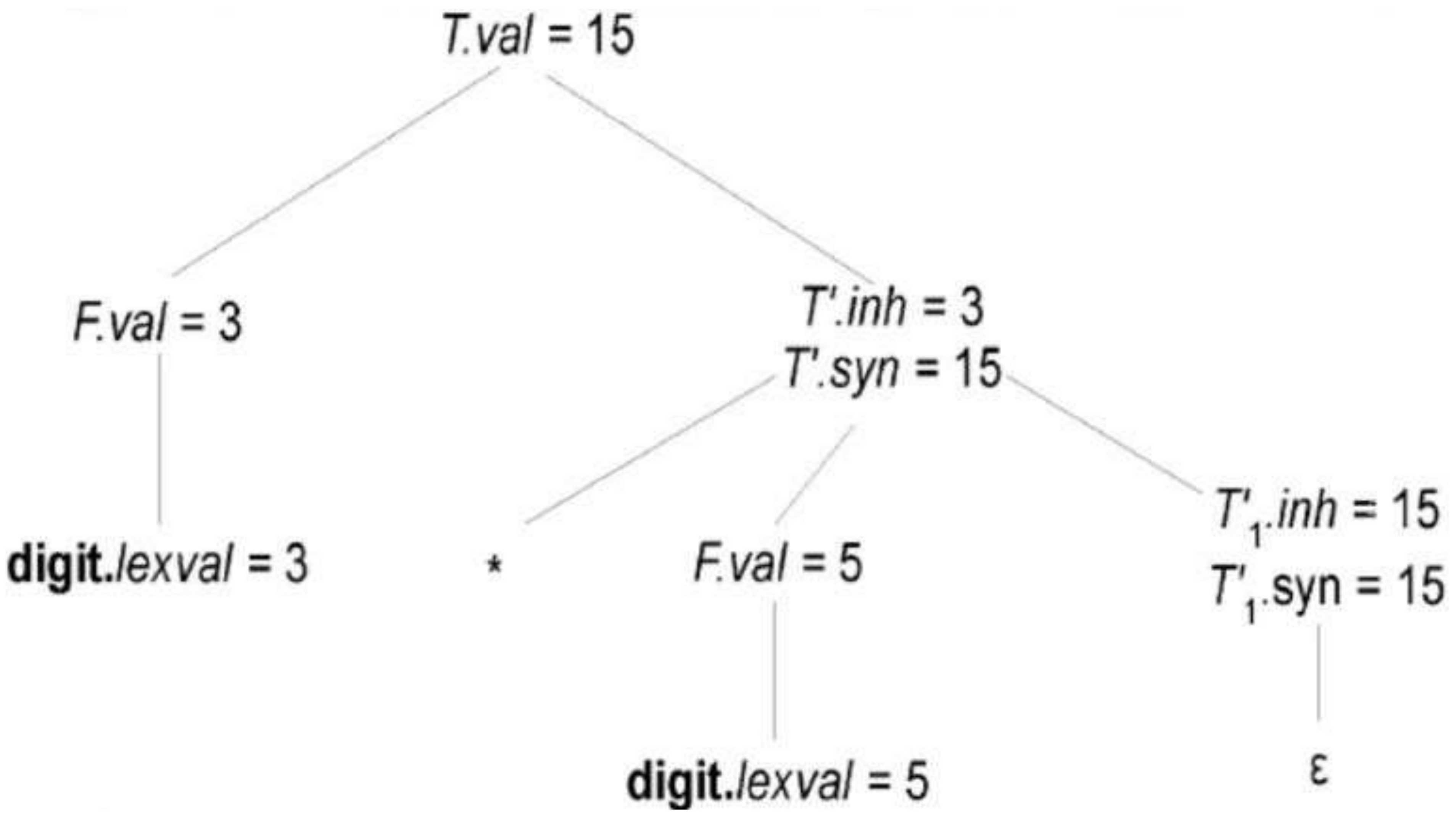
Dependency Graph



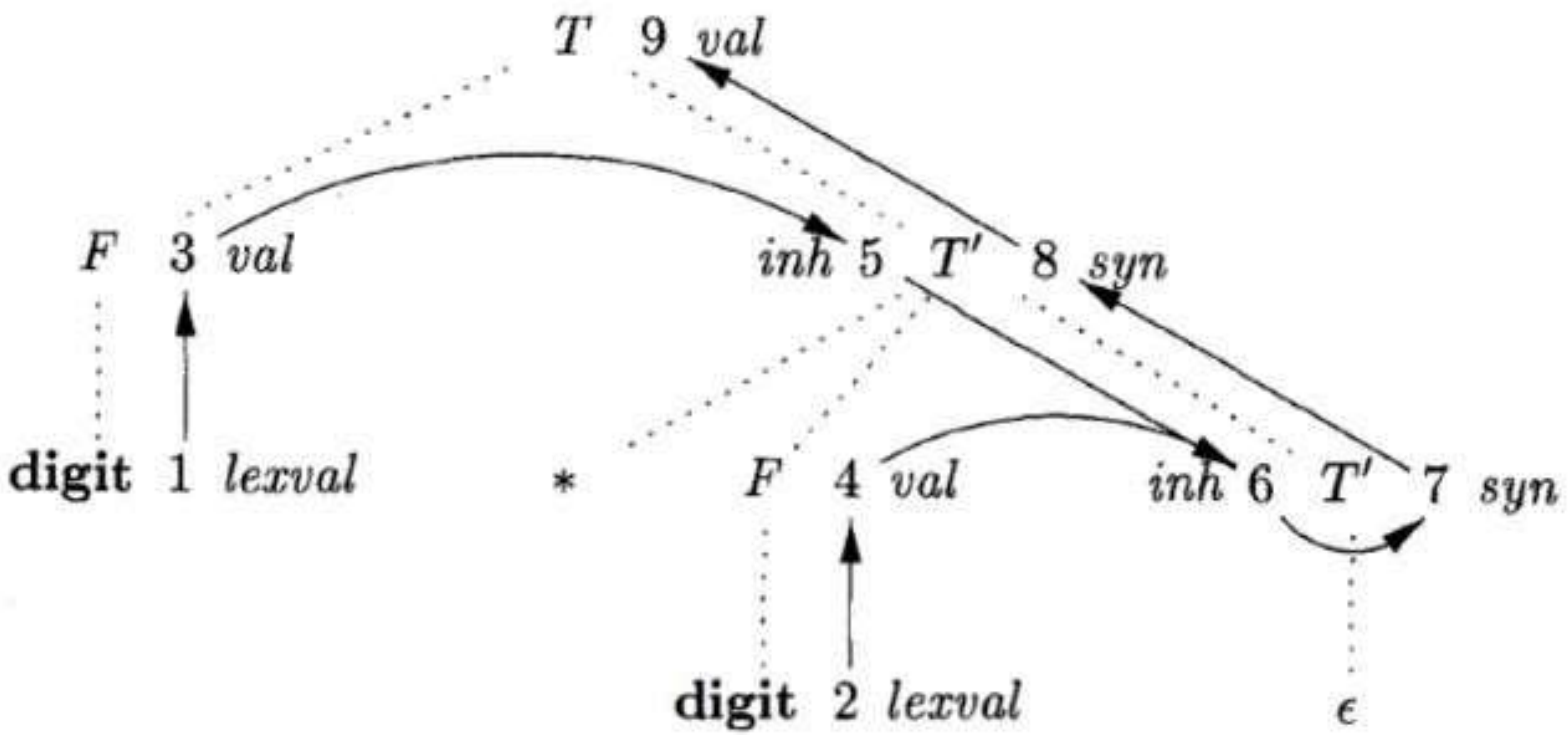
SDD of Inherited attribute



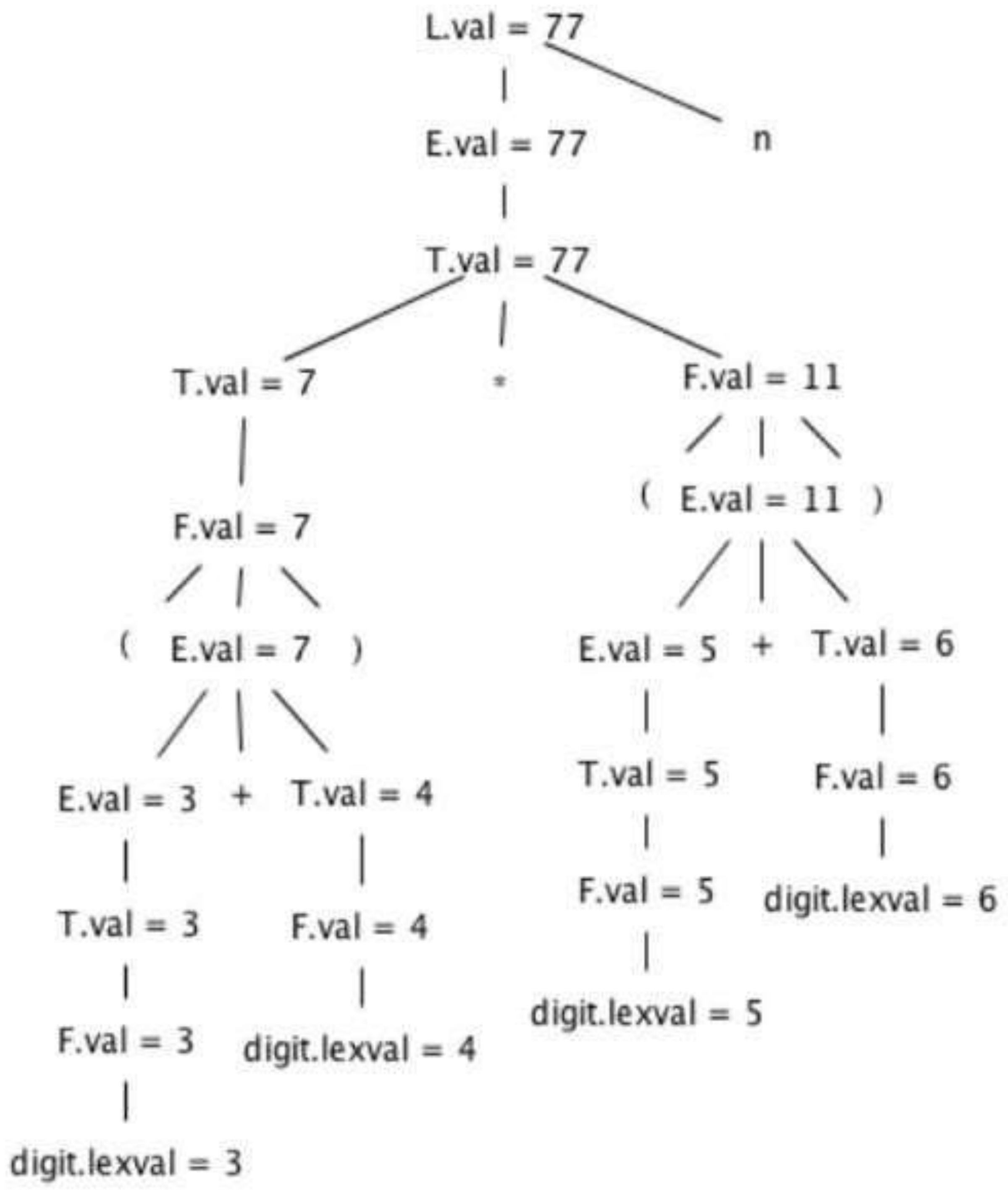
Production	Semantic Rules
$T \rightarrow F\ T'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow *F\ T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
$T' \rightarrow \varepsilon$	$T'.syn = T'.inh$
$F \rightarrow \textbf{digit}$	$F.val = \textbf{digit}.lexval$



Dependency Graph



Annotated Parse tree of (3+4)*(5+6)$



THANK
YOU