CS 4300: Compiler Theory

Chapter 5
Syntax-Directed Translation

Dr. Xuejun Liang

Outlines (Sections)

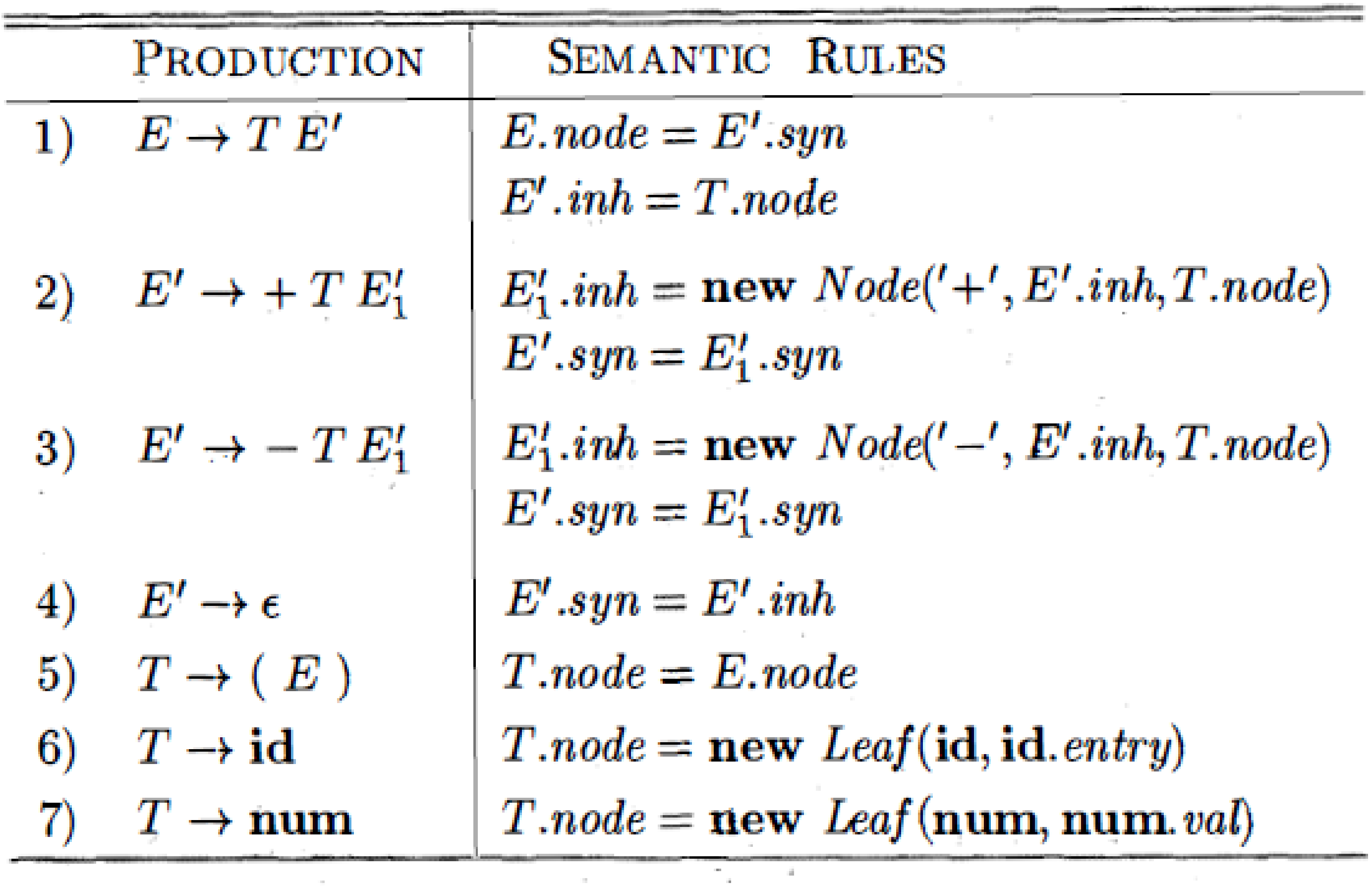
1. Syntax-Directed Definitions
2. Evaluation Orders for SDD's
3. Applications of Syntax-Directed Definition
4. Syntax-Directed Translation Schemes
5. Implementing L-Attributed SDD's

Quick Review of Last Lecture

- Syntax-Directed Definitions
  - Syntax-directed definition (SDD): Semantic rules
  - Syntax-directed translation scheme: Semantic actions
  - Attributes: Synthetic or Inherited
    - S-attributed
  - Annotated Parse Tree and Its Traversal
- Evaluation Orders for SDDs
  - Dependency graph and topological order
  - L-attributed SDD
    - Can apply depth-first and left to right
- Applications of SDD
  - Construction of Syntax Trees

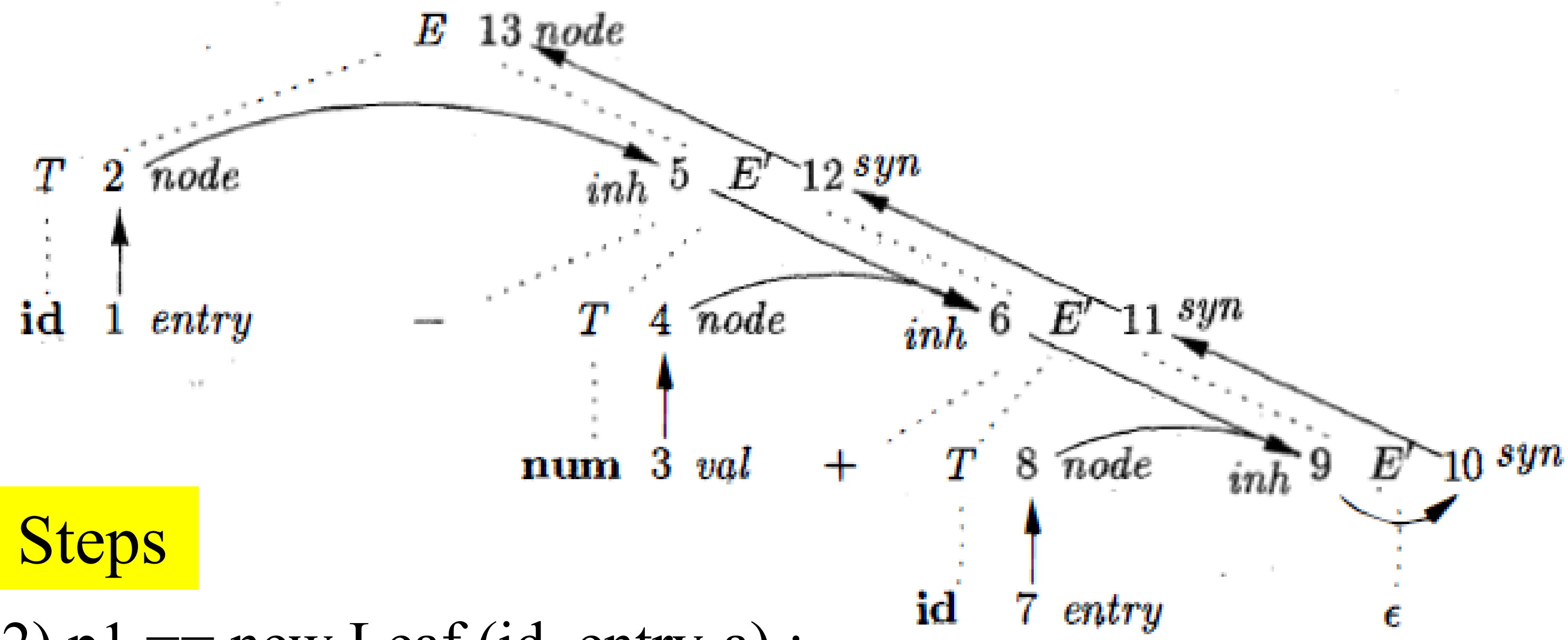
Constructing Syntax Tree During Top-Down Parsing

L-attributed Definition for Simple Expression



	PRODUCTION	SEMANTIC RULES
1)	$E \to T\ E'$	$E.node = E'.syn$ $E'.inh = T.node$
2)	$E' \to +\ T\ E'_1$	$E'_1.inh = \textbf{new}\ Node('+', E'.inh, T.node)$ $E'.syn = E'_1.syn$
3)	$E' \to -\ T\ E'_1$	$E'_1.inh = \textbf{new}\ Node('-', E'.inh, T.node)$ $E'.syn = E'_1.syn$
4)	$E' \to \epsilon$	$E'.syn = E'.inh$
5)	$T \to (\ E\ )$	$T.node = E.node$
6)	$T \to \textbf{id}$	$T.node = \textbf{new}\ Leaf(\textbf{id}, \textbf{id}.entry)$
7)	$T \to \textbf{num}$	$T.node = \textbf{new}\ Leaf(\textbf{num}, \textbf{num}.val)$

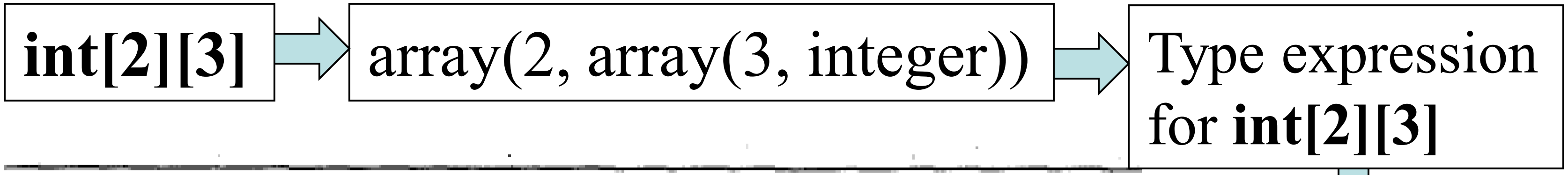
Example: Dependency Graph for a-4+c



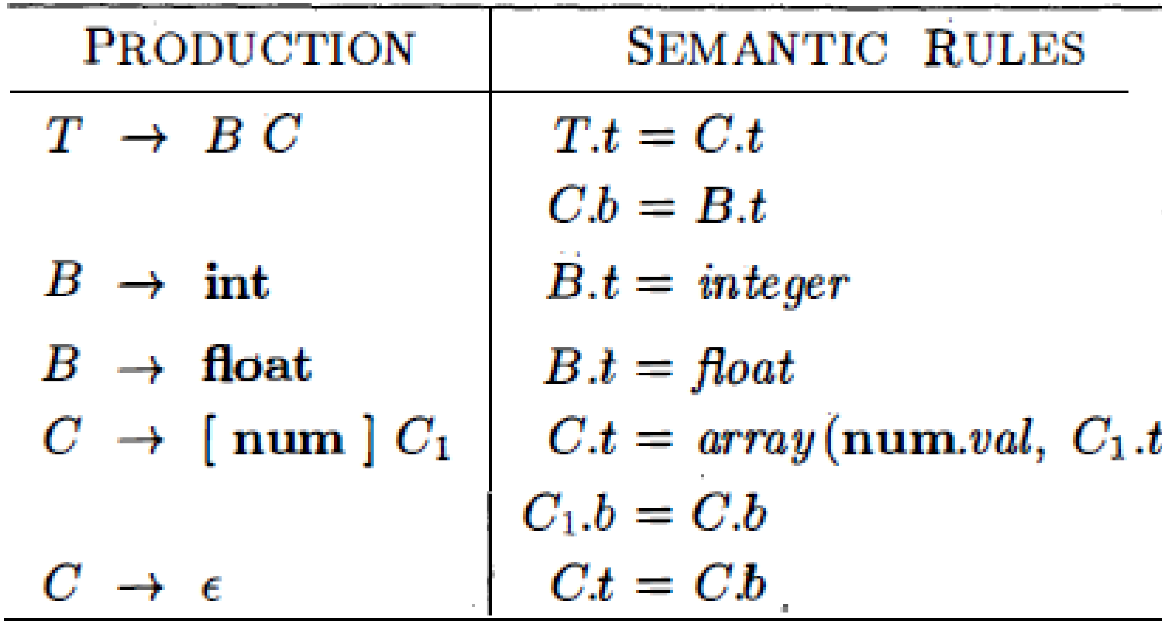
Steps

2) p1 == new Leaf (id, entry-a) ;
4) p2 == new Leaf (num, 4) ;
6) p3 == new Node('-', pl, p2) ;
8) p4 == new Leaf (id, entry-c) ;
9) p5 == new Node('+', p3, p4) ;

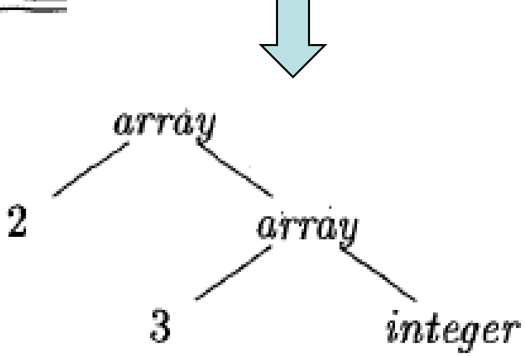
The Structure of a Type



int[2][3] → array(2, array(3, integer)) → Type expression for **int[2][3]**

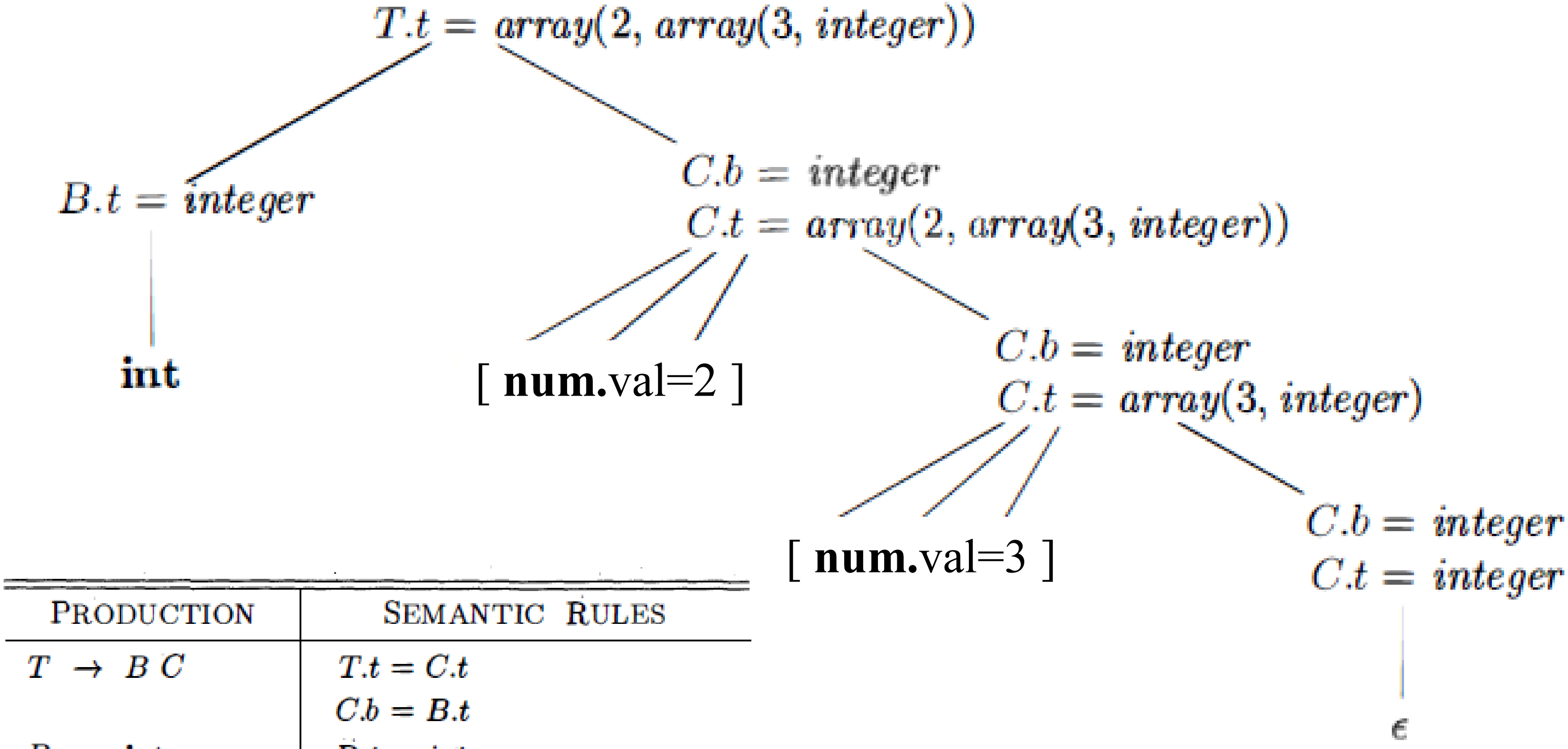


PRODUCTION	SEMANTIC RULES
$T \rightarrow B\ C$	$T.t = C.t$ $C.b = B.t$
$B \rightarrow$ **int**	$B.t = integer$
$B \rightarrow$ **float**	$B.t = float$
$C \rightarrow [\ \mathbf{num}\ ]\ C_1$	$C.t = array(\mathbf{num}.val,\ C_1.t)$ $C_1.b = C.b$
$C \rightarrow \epsilon$	$C.t = C.b$



T generates either a basic type or an array type

Annotated Parse Tree for int[2][3]



Production	Semantic Rules
$T \rightarrow B\ C$	$T.t = C.t$ $C.b = B.t$
$B \rightarrow$ **int**	$B.t = integer$
$B \rightarrow$ **float**	$B.t = float$
$C \rightarrow$ [**num**] C_1	$C.t = array($**num**$.val,\ C_1.t)$ $C_1.b = C.b$
$C \rightarrow \epsilon$	$C.t = C.b$