ΕΠΛ323 - Θεωρία και Πρακτική Μεταγλωττιστών

Lecture 8a

Syntax-directed Translation

Elias Athanasopoulos
eliasathan@cs.ucy.ac.cy

Syntax-directed Translation (SDT)

Μετάφραση Κατευθυνόμενη από τη Σύνταξη

- We associate information with a programming language construct by attaching attributes to the grammar symbols representing the construct.
- Values of attributes are computed based on semantic rules.
- We define Syntax-directed Definitions and Syntax-director Translations.
- We parse the input token stream, build the parse tree, and then traverse the tree as needed to evaluate the semantic rules at the parse-tree nodes.

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

Syntax-directed Definitions (SDTs)

Ορισμοί κατευθυνόμενοι από τη σύνταξη

- A syntax-directed definition is a generalization of a context-free grammar in which each grammar symbol has an associated set of attributes
  - Synthesized attributes (*παραγόμενα*),
  - Inherited attributes (*κληρονομούμενα*).
- Attributes can represent anything
  - Strings, numbers, types, memory locations, etc.
- A parse tree showing the values of attributes at each node is called an *annotated parse tree.*

Attributes

In a syntax-directed definition, each grammar production $A \rightarrow a$ has associated with it a set of semantic rules of the form $b := f(c_1, c_1, \ldots, c_k)$ where a f is a function.

- Synthesized
  - b is a synthesized attribute of *A,*
  - **Example***: $A \rightarrow BC$, $A.val = f(B.val, C.val)$, i.e.,* the attribute **val** of *A* is computed by attributes of its children (*B* and *C*)*.*
- Inherited
  - b is an inherited attribute of one of the grammar symbols on the right side of the production.
  - Terminals have **only** inherited attributes.
  - **Example***: $A \rightarrow BCD$, $C.val = f(A.val, B.val, D.val)$, i.e.,* the attribute **val** of *C* (on the right side of the production) is computed by attributes of its parent (*A*) and its siblings (*B* and *D*).

In either case, we say that attribute b depends on attributes $c_1, c_1, \ldots, c_k$.

Example

Synthesized Attributes

PRODUCTION	SEMANTIC RULES
$L \rightarrow E$ **n**	$print(E.val)$
$E \rightarrow E_1 + T$	$E.val := E_1.val + T.val$
$E \rightarrow T$	$E.val := T.val$
$T \rightarrow T_1 * T$	$T.val := T_1.val * T.val$
$T \rightarrow F$	$T.val := F.val$
$F \rightarrow (E)$	$F.val := E.val$
$F \rightarrow$ **digit**	$F.val :=$ **digit**.$lexval$

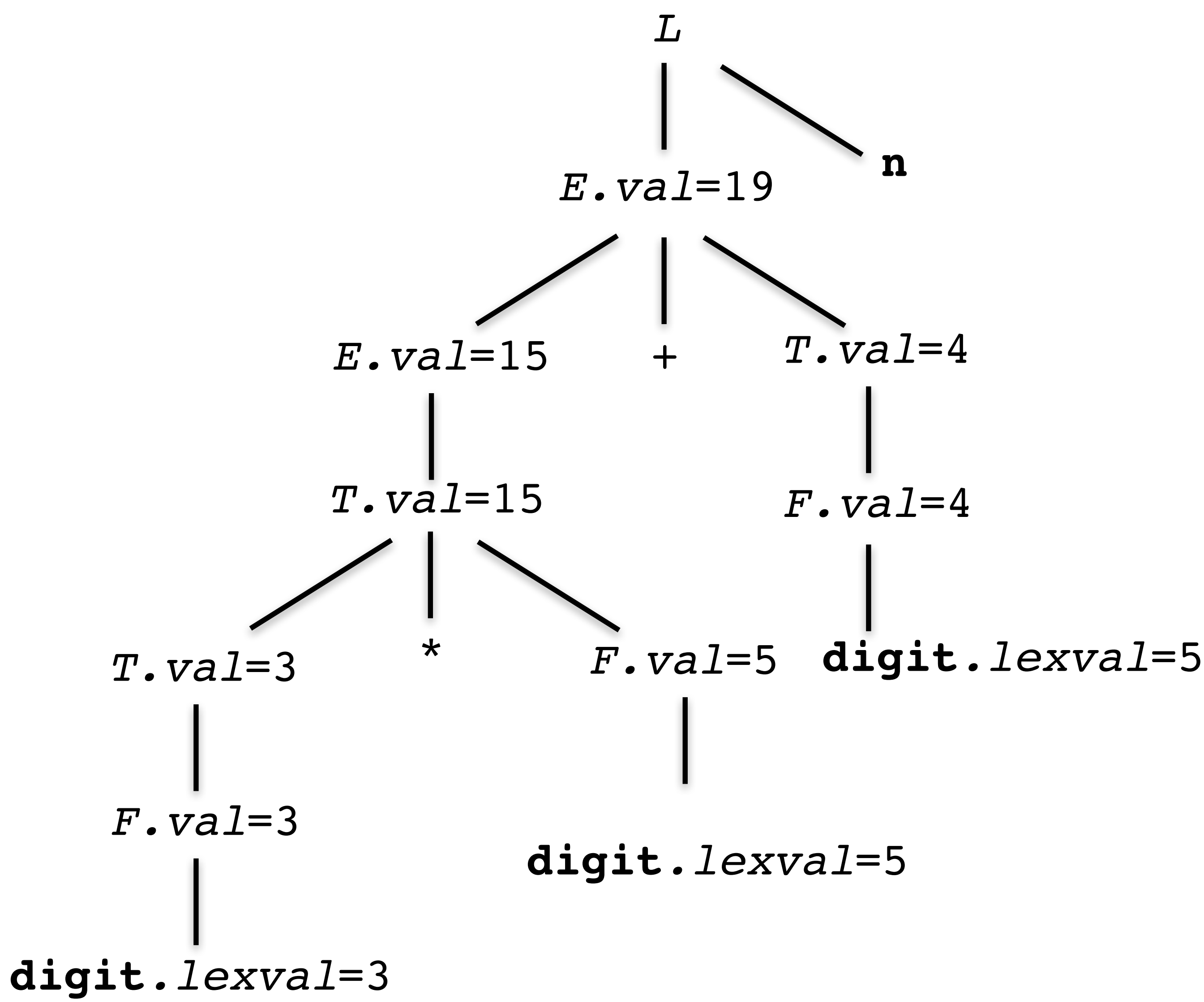
All semantic rules are written at the right end of the productions.

Attribute **val** of *E* is a function of the attributes of its children, namely E_1 and *T*.

Syntax-directed definition (called also *S-attributed definition*), which contains *only* synthesized attributes.

Annotated Parse Tree

Input: 3*5+4



Semantic Analysis

- **Syntax-directed Definition (SDD)**
  A context-free grammar with **semantic rules** for calculating the attributes

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

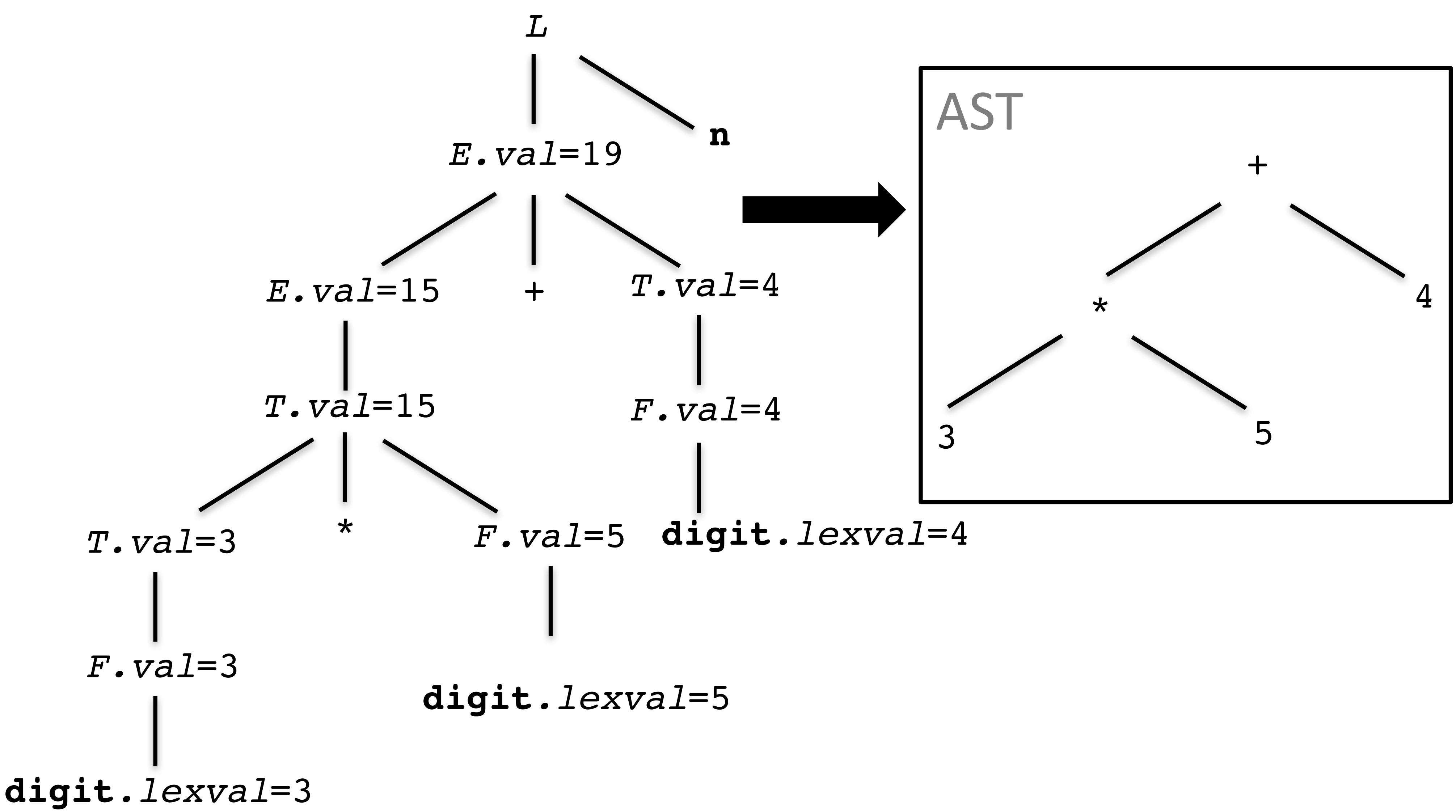
- **Syntax-directed Translations (SDTs)**
  A context-free grammar with **semantic actions** and their exact order (actions can appear anywhere in the right side of a production)

$T \rightarrow \mathbf{num}\ \{\ print(\mathbf{num}.val)\ \}$

Syntax Trees

- An (abstract) syntax tree (AST) is a condensed form of a parse tree, useful for representing language constructs.
- An AST is much more simplified compared to the parse tree
- Much more easier to be handled from following phases of the compiler

Example



L
n
E.val=19
E.val=15
+
T.val=4
T.val=15
F.val=4
T.val=3
*
F.val=5
digit.lexval=4
F.val=3
digit.lexval=5
digit.lexval=3
AST
+
*
4
3
5

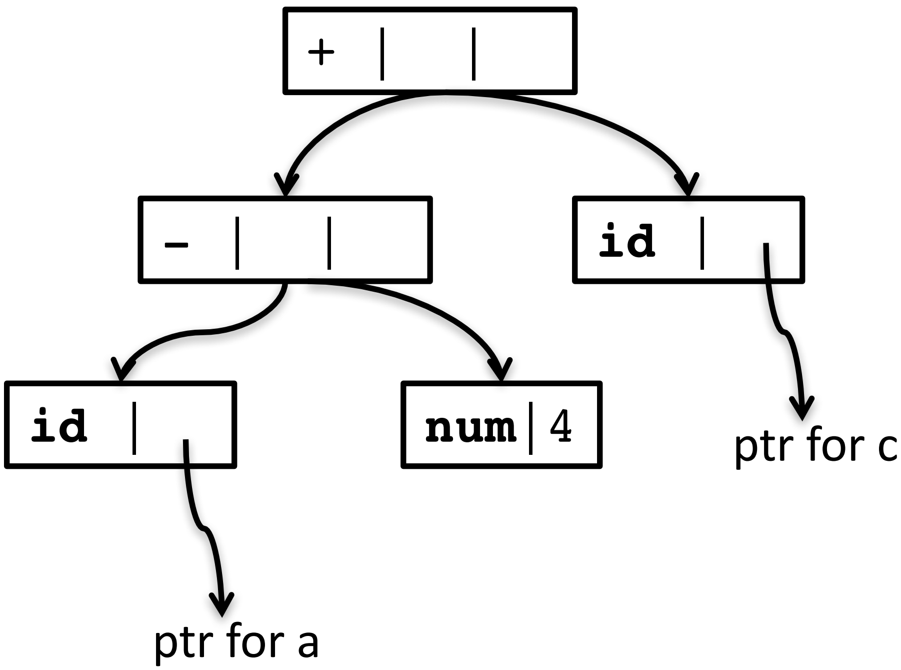
AST Construction

- We use the following functions to create an AST.
  - `mknode(op, left, right)` creates an operator node with label *op* and two fields containing pointers to *left* and *right*.
  - `mkleaf(`**`id`**`, entry)` creates an identifier node with label **id** and a field containing *entry*, a pointer to the symbol-table entry for the identifier.
  - `mkleaf(`**`num`**`, val)` creates a number node with label **num** and a field containing *val*, the value of the number.

Syntax-directed Definition for AST construction

PRODUCTION	SEMANTIC RULES
$E \to E_1 + T$	$E.nptr := mknode('+', E_1.nptr, T.nptr)$
$E \to E_1 - T$	$E.nptr := mknode('-', E_1.nptr, T.nptr)$
$E \to T$	$E.nptr := T.nptr$
$T \to (E)$	$T.nptr := E.nptr$
$T \to$ **id**	$T.nptr := mkleaf($**id**, **id**$.entry)$
$T \to$ **num**	$T.nptr :=$ mkleaf(**num**, **num**.val)

a-4+c




```
(1) p1 := mkleaf(id, entrya)
(2) p2 := mkleaf(num, 4)
(3) p3 := mknode('-', p1, p2)
(4) p4 := mkleaf(id, entryc)
(5) p5 := mknode('+', p3, p4)
```