

## Semantic Actions

---

- This is what we will use to construct ASTs
- Each grammar symbol may have attributes
  - An attribute is a property of a programming language construct
  - For terminal symbols (lexical tokens) attributes can be calculated by the lexer
- Each production may have an action
  - Written as:  $X \rightarrow Y_1 \dots Y_n \quad \{ \text{action} \}$
  - That can refer to or compute symbol attributes

7

## Semantic Actions: An Example

---

- Consider the grammar
$$E \rightarrow \text{int} \mid E + E \mid ( E )$$
- For each symbol  $X$  define an attribute  $X.\text{val}$ 
  - For terminals,  $\text{val}$  is the associated lexeme
  - For non-terminals,  $\text{val}$  is the expression's value (which is computed from values of subexpressions)
- We annotate the grammar with actions:
$$\begin{array}{l} E \rightarrow \text{int} \quad \{ E.\text{val} = \text{int}.\text{val} \} \\ \quad \mid E_1 + E_2 \quad \{ E.\text{val} = E_1.\text{val} + E_2.\text{val} \} \\ \quad \mid ( E_1 ) \quad \{ E.\text{val} = E_1.\text{val} \} \end{array}$$

8

## Semantic Actions: An Example (Cont.)

---

- String:  $5 + (2 + 3)$
- Tokens:  $\text{int}_5 '+' '(' \text{int}_2 '+' \text{int}_3 ')'$

### Productions

$$E \rightarrow E_1 + E_2$$

$$E_1 \rightarrow \text{int}_5$$

$$E_2 \rightarrow (E_3)$$

$$E_3 \rightarrow E_4 + E_5$$

$$E_4 \rightarrow \text{int}_2$$

$$E_5 \rightarrow \text{int}_3$$

### Equations

$$E.\text{val} = E_1.\text{val} + E_2.\text{val}$$

$$E_1.\text{val} = \text{int}_5.\text{val} = 5$$

$$E_2.\text{val} = E_3.\text{val}$$

$$E_3.\text{val} = E_4.\text{val} + E_5.\text{val}$$

$$E_4.\text{val} = \text{int}_2.\text{val} = 2$$

$$E_5.\text{val} = \text{int}_3.\text{val} = 3$$

9

## Semantic Actions: Dependencies

---

- Semantic actions specify a system of equations
- Order of executing the actions is not specified

- Example:

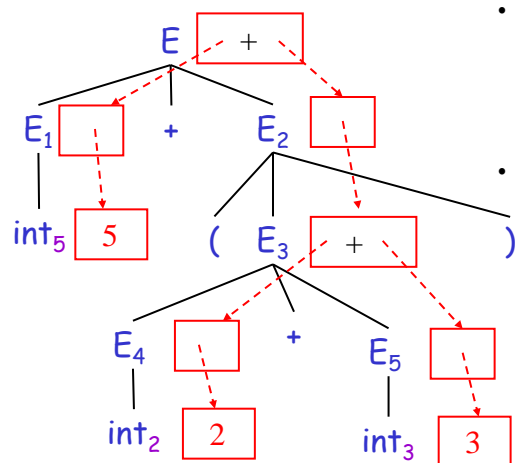
$$E_3.\text{val} = E_4.\text{val} + E_5.\text{val}$$

- Must compute  $E_4.\text{val}$  and  $E_5.\text{val}$  before  $E_3.\text{val}$
- We say that  $E_3.\text{val}$  *depends on*  $E_4.\text{val}$  and  $E_5.\text{val}$

- The parser must find the order of evaluation

10

## Dependency Graph



- Each node labeled with a non-terminal  $E$  has one slot for its  $val$  attribute
- Note the dependencies

11

## Evaluating Attributes

- An attribute must be computed after all its successors in the dependency graph have been computed
  - In the previous example attributes can be computed bottom-up
- Such an order exists when there are no cycles
  - Cyclically defined attributes are not legal

12

## Semantic Actions: Notes (Cont.)

- Synthesized attributes
  - Calculated from attributes of descendents in the parse tree
  - $E.val$  is a synthesized attribute
  - Can always be calculated in a bottom-up order
- Grammars with only synthesized attributes are called S-attributed grammars
  - Most frequent kinds of grammars

13

## Inherited Attributes

- Another kind of attributes
- Calculated from attributes of the parent node(s) and/or siblings in the parse tree
- Example: a line calculator

14

## A Line Calculator

- Each line contains an expression  
 $E \rightarrow \text{int} \mid E + E$
- Each line is terminated with the = sign  
 $L \rightarrow E = \mid + E =$
- In the second form, the value of evaluation of the previous line is used as starting value
- A program is a sequence of lines  
 $P \rightarrow \varepsilon \mid PL$

15

## Attributes for the Line Calculator

- Each  $E$  has a synthesized attribute **val**
  - Calculated as before
- Each  $L$  has a synthesized attribute **val**
  - $L \rightarrow E = \quad \{ L.\text{val} = E.\text{val} \}$
  - $\mid + E = \quad \{ L.\text{val} = E.\text{val} + L.\text{prev} \}$
- We need the value of the previous line
- We use an inherited attribute **L.prev**

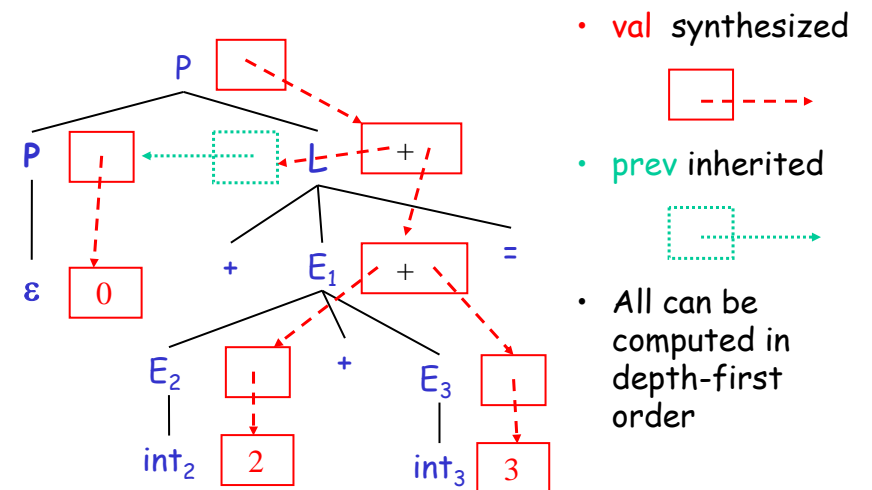
16

## Attributes for the Line Calculator (Cont.)

- Each  $P$  has a synthesized attribute **val**
  - The value of its last line
- $P \rightarrow \varepsilon \quad \{ P.\text{val} = 0 \}$   
 $\mid P_1 L \quad \{ P.\text{val} = L.\text{val};$   
 $\quad \quad \quad L.\text{prev} = P_1.\text{val} \}$
- Each  $L$  has an inherited attribute **prev**
  - $L.\text{prev}$  is inherited from sibling  $P_1.\text{val}$
- Example ...

17

## Example of Inherited Attributes



18

## Semantic Actions: Notes (Cont.)

---

- Semantic actions can be used to build ASTs
- And many other things as well
  - Also used for type checking, code generation, ...
- Process is called syntax-directed translation
  - Substantial generalization over CFGs