Semantic Analysis with Attribute Grammars Part 2

Y.N. Srikant

Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012

NPTEL Course on Principles of Compiler Design

Outline of the Lecture

- Introduction (covered in lecture 1)
- Attribute grammars
- Attributed translation grammars
- Semantic analysis with attributed translation grammars

Attribute Grammars

- Let $G = (N, T, P, S)$ be a CFG and let $V = N \cup T$.
- Every symbol X of V has associated with it a set of *attributes*
- Two types of attributes: *inherited* and *synthesized*
- Each attribute takes values from a specified domain
- A production $p \in P$ has a set of attribute computation rules for
  - synthesized attributes of the LHS non-terminal of p
  - inherited attributes of the RHS non-terminals of p
- Rules are strictly local to the production p (no side effects)

Synthesized and Inherited Attributes

- An attribute cannot be both synthesized and inherited, but a symbol can have both types of attributes
- Attributes of symbols are evaluated over a parse tree by making passes over the parse tree
- Synthesized attributes are computed in a bottom-up fashion from the leaves upwards
  - Always synthesized from the attribute values of the children of the node
  - Leaf nodes (terminals) have synthesized attributes (only) initialized by the lexical analyzer and cannot be modified
- Inherited attributes flow down from the parent or siblings to the node in question

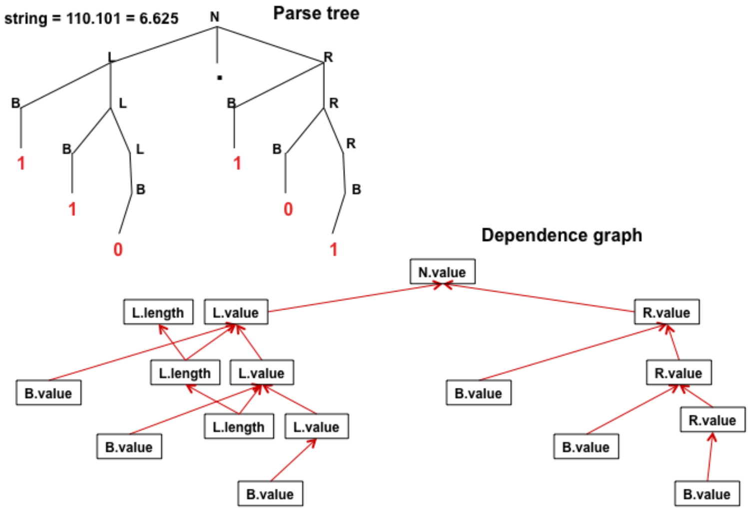
Attribute Evaluation Strategy

- Construct the parse tree
- Construct the dependence graph
- Perform topological sort on the dependence graph and obtain an evaluation order
- Evaluate attributes according to this order using the corresponding attribute evaluation rules attached to the respective productions

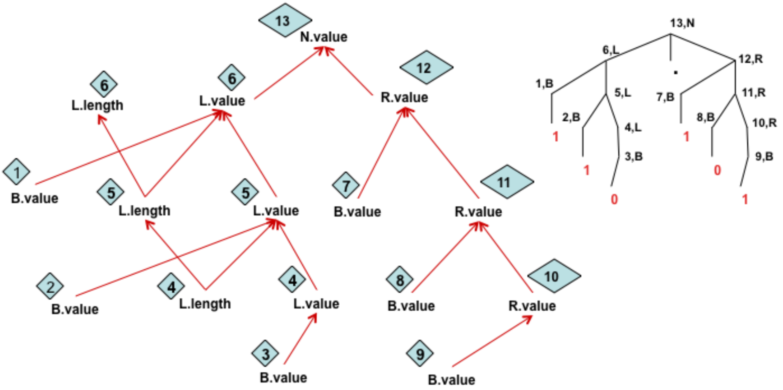
Attribute Grammar - Example 2

- AG for the evaluation of a real number from its bit-string representation
  Example: 110.101 = 6.625
- $N \rightarrow L.R,\ L \rightarrow BL \mid B,\ R \rightarrow BR \mid B,\ B \rightarrow 0 \mid 1$
- $AS(N) = AS(R) = AS(B) = \{value\uparrow : real\}$,
  $AS(L) = \{length\uparrow : integer,\ value\uparrow : real\}$
  1. $N \rightarrow L.R$ $\{N.value\uparrow := L.value\uparrow + R.value\uparrow\}$
  2. $L \rightarrow B$ $\{L.value\uparrow := B.value\uparrow;\ L.length\uparrow := 1\}$
  3. $L_1 \rightarrow BL_2$ $\{L_1.length\uparrow := L_2.length\uparrow + 1;$
     $L_1.value\uparrow := B.value\uparrow * 2^{L_2.length\uparrow} + L_2.value\uparrow\}$
  4. $R \rightarrow B$ $\{R.value\uparrow := B.value\uparrow /2\}$
  5. $R_1 \rightarrow BR_2$ $\{R_1.value\uparrow := (B.value\uparrow + R_2.value\uparrow)/2\}$
  6. $B \rightarrow 0$ $\{B.value\uparrow := 0\}$
  7. $B \rightarrow 1$ $\{B.value\uparrow := 1\}$

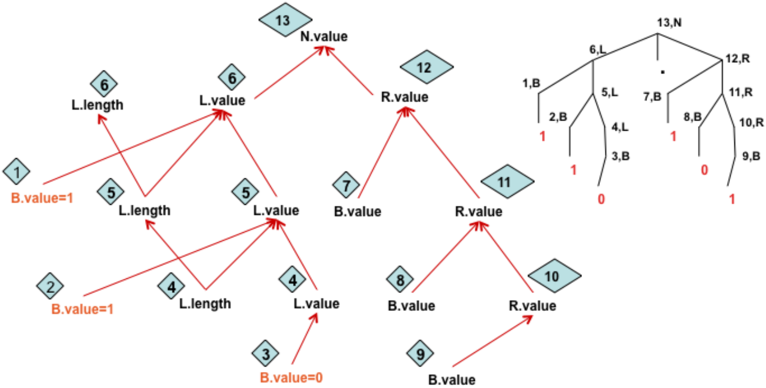
Dependence Graph for Example 2



Attribute Evaluation for Example 2 - 1

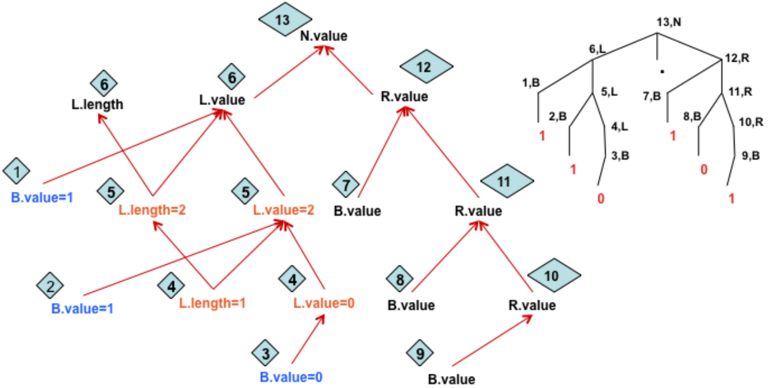


Attribute Evaluation for Example 2 - 2



Nodes 1,2: $B \rightarrow 1$ $\{B.value \uparrow := 1\}$
Node 3: $B \rightarrow 0$ $\{B.value \uparrow := 0\}$

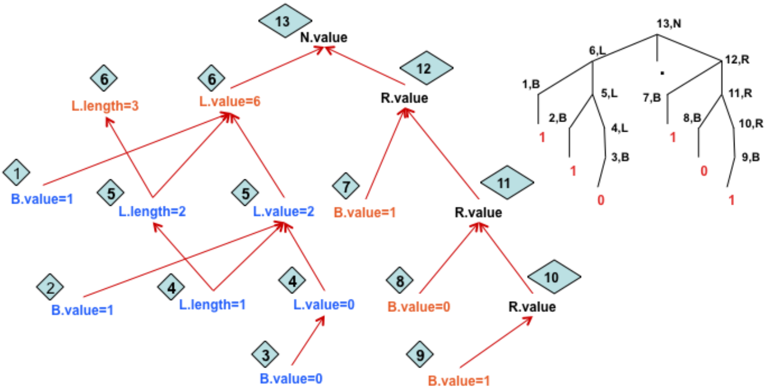
Attribute Evaluation for Example 2 - 3



Node 4: $L \to B$ $\{L.value \uparrow := B.value \uparrow;\ \ L.length \uparrow := 1\}$

Node 5: $L_1 \to BL_2$ $\{L_1.length \uparrow := L_2.length \uparrow +1;$

$L_1.value \uparrow := B.value \uparrow * 2^{L_2.length \uparrow} + L_2.value \uparrow\}$



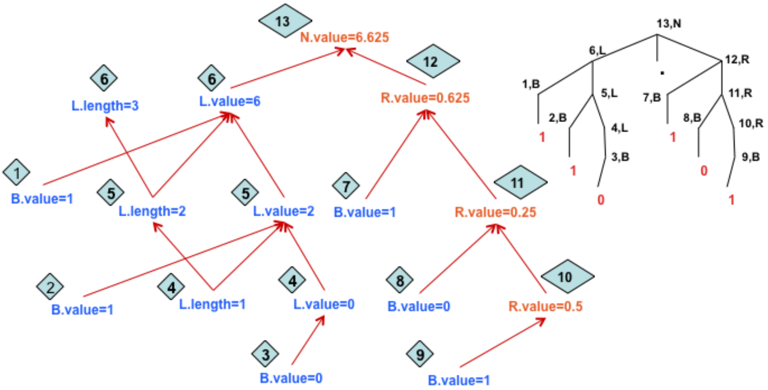
Node 6: $L_1 \rightarrow BL_2$ $\{L_1.length \uparrow := L_2.length \uparrow +1;$

$L_1.value \uparrow := B.value \uparrow * 2^{L_2.length\uparrow} + L_2.value \uparrow\}$

Nodes 7,9: $B \rightarrow 1$ $\{B.value \uparrow := 1\}$

Node 8: $B \rightarrow 0$ $\{B.value \uparrow := 0\}$

Attribute Evaluation for Example 2 - 5



Node 10: $R \rightarrow B$ $\{R.value\uparrow := B.value\uparrow /2\}$

Nodes 11,12:

$R_1 \rightarrow BR_2$ $\{R_1.value\uparrow := (B.value\uparrow + R_2.value\uparrow)/2\}$

Node 13: $N \rightarrow L.R$ $\{N.value\uparrow := L.value\uparrow + R.value\uparrow\}$

Attribute Grammar - Example 3

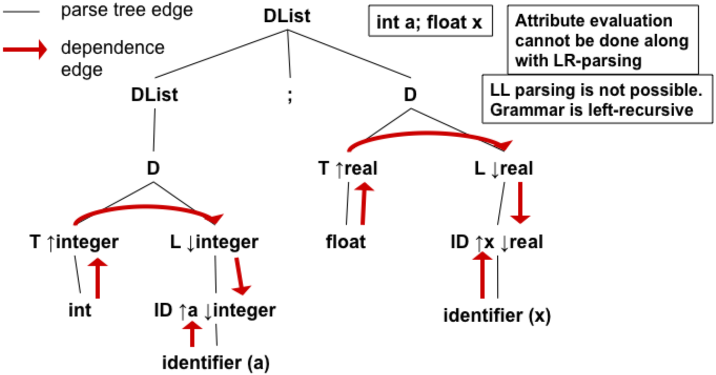
- A simple AG for the evaluation of a real number from its bit-string representation
  Example: $110.1010 = 6 + 10/2^4 = 6 + 10/16 = 6 + 0.625 = 6.625$
- $N \rightarrow X.X,\ X \rightarrow BX \mid B,\ B \rightarrow 0 \mid 1$
- $AS(N) = AS(B) = \{value \uparrow : real\}$,
  $AS(X) = \{length \uparrow : integer,\ value \uparrow : real\}$
  1. $N \rightarrow X_1.X_2$ $\{N.value \uparrow := X_1.value \uparrow + X_2.value \uparrow / 2^{X_2.length}\}$
  2. $X \rightarrow B$ $\{X.value \uparrow := B.value \uparrow ;\ X.length \uparrow := 1\}$
  3. $X_1 \rightarrow BX_2$ $\{X_1.length \uparrow := X_2.length \uparrow + 1;$
     $X_1.value \uparrow := B.value \uparrow * 2^{X_2.length \uparrow} + X_2.value \uparrow\}$
  4. $B \rightarrow 0$ $\{B.value \uparrow := 0\}$
  5. $B \rightarrow 1$ $\{B.value \uparrow := 1\}$

Attribute Grammar - Example 4

- An AG for associating *type* information with names in variable declarations
- $AI(L) = AI(ID) = \{type \downarrow : \{integer, real\}\}$
  $AS(T) = \{type \uparrow : \{integer, real\}\}$
  $AS(ID) = AS(identifier) = \{name \uparrow : string\}$
  1. $DList \rightarrow D \mid DList \; ; \; D$
  2. $D \rightarrow T \; L \; \{L.type \downarrow := T.type \uparrow\}$
  3. $T \rightarrow int \; \{T.type \uparrow := integer\}$
  4. $T \rightarrow float \; \{T.type \uparrow := real\}$
  5. $L \rightarrow ID \; \{ID.type \downarrow := L.type \downarrow\}$
  6. $L_1 \rightarrow L_2 \; , \; ID \; \{L_2.type \downarrow := L_1.type \downarrow; \; ID.type \downarrow := L_1.type \downarrow\}$
  7. $ID \rightarrow identifier \; \{ID.name \uparrow := identifier.name \uparrow\}$

Example: *int* a,b,c; *float* x,y
a,b, and c are tagged with type *integer*
x,y, and z are tagged with type *real*

Attribute Evaluation for Example 4



1. $DList \rightarrow D \mid DList\ ;\ D$ **2.** $D \rightarrow T\ L$ $\{L.type\downarrow := T.type\uparrow\}$

3. $T \rightarrow int$ $\{T.type\uparrow := integer\}$ **4.** $T \rightarrow float$ $\{T.type\uparrow := real\}$

5. $L \rightarrow ID$ $\{ID.type\downarrow := L.type\downarrow\}$

6. $L_1 \rightarrow L_2\ ,\ ID$ $\{L_2.type\downarrow := L_1.type\downarrow; ID.type\downarrow := L_1.type\downarrow\}$

7. $ID \rightarrow identifier$ $\{ID.name\uparrow := identifier.name\uparrow\}$

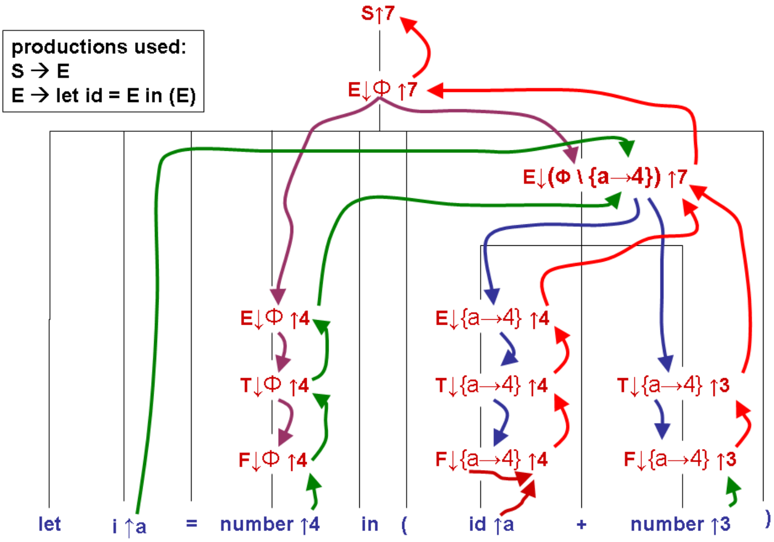
Attribute Grammar - Example 5

- Let us first consider the CFG for a simple language
  1. $S \longrightarrow E$
  2. $E \longrightarrow E + T \mid T \mid let\ id = E\ in\ (E)$
  3. $T \longrightarrow T * F \mid F$
  4. $F \longrightarrow (E) \mid number \mid id$
- This language permits expressions to be nested inside expressions and have scopes for the names
  - *let A = 5 in ((let A = 6 in (A*7)) - A)* evaluates correctly to 37, with the scopes of the two instances of *A* being different
- It requires a scoped symbol table for implementation
- An abstract attribute grammar for the above language uses both inherited and synthesized attributes
- Both inherited and synthesized attributes can be evaluated in one pass (from left to right) over the parse tree
- Inherited attributes cannot be evaluated during LR parsing

Attribute Grammar - Example 5

1. $S \longrightarrow E$ $\{E.symtab \downarrow := \phi; S.val \uparrow := E.val \uparrow\}$
2. $E_1 \longrightarrow E_2 + T$ $\{E_2.symtab \downarrow := E_1.symtab \downarrow;$ $E_1.val \uparrow := E_2.val \uparrow + T.val \uparrow; T.symtab \downarrow := E_1.symtab \downarrow\}$
3. $E \longrightarrow T$ $\{T.symtab \downarrow := E.symtab \downarrow; E.val \uparrow := T.val \uparrow\}$
4. $E_1 \longrightarrow let\ id = E_2\ in\ (E_3)$ $\{E_1.val \uparrow := E_3.val \uparrow; E_2.symtab \downarrow := E_1.symtab \downarrow;$ $E_3.symtab \downarrow := E_1.symtab \downarrow \backslash \{id.name \uparrow \rightarrow E_2.val \uparrow\}\}$
5. $T_1 \longrightarrow T_2 * F$ $\{T_1.val \uparrow := T_2.val \uparrow * F.val \uparrow;$ $T_2.symtab \downarrow := T1.symtab \downarrow; F.symtab \downarrow := T_1.symtab \downarrow\}$
6. $T \longrightarrow F$ $\{T.val \uparrow := F.val \uparrow; F.symtab \downarrow := T.symtab \downarrow\}$
7. $F \longrightarrow (E)$ $\{F.val \uparrow := E.val \uparrow; E.symtab \downarrow := F.symtab \downarrow\}$
8. $F \longrightarrow number$ $\{F.val \uparrow := number.val \uparrow\}$
9. $F \longrightarrow id$ $\{F.val \uparrow := F.symtab \downarrow [id.name \uparrow]\}$

Attribute Flow and Evaluation - Example 5



L-Attributed and S-Attributed Grammars

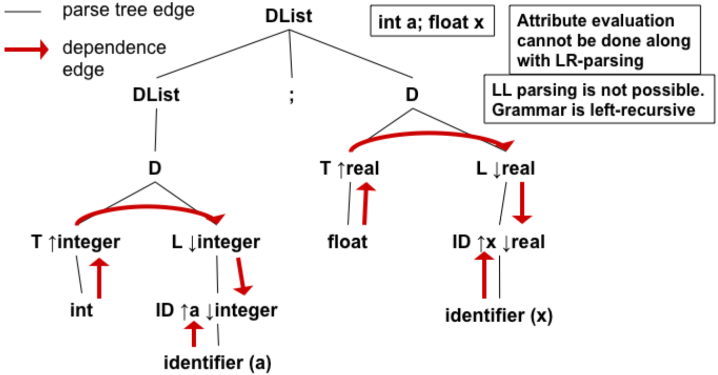
- An AG with only synthesized attributes is an S-attributed grammar
  - Attributes of SAGs can be evaluated in any bottom-up order over a parse tree (single pass)
  - Attribute evaluation can be combined with LR-parsing (YACC)
- In L-attributed grammars, attribute dependencies always go from *left to right*
- More precisely, each attribute must be
  - Synthesized, or
  - Inherited, but with the following limitations:
    consider a production $p: \ A \rightarrow X_1 X_2 ... X_n$. Let $X_i.a \in AI(X_i)$.
    $X_i.a$ may use only
    - elements of $AI(A)$
    - elements of $AI(X_k)$ or $AS(X_k)$, $k = 1, ..., i-1$
      (*i.e.,* attibutes of $X_1, ..., X_{i-1}$)
- We concentrate on SAGs, and 1-pass LAGs, in which attribute evaluation can be combined with LR, LL or RD parsing

Attribute Evaluation Algorithm for LAGs

Input: A parse tree *T* with unevaluated attribute instances
Output: *T* with consistent attribute values

```
void dfvisit(n: node)
{  for each child m of n, from left to right do
     {  evaluate inherited attributes of m;
        dfvisit(m)
     };
   evaluate synthesized attributes of n
}
```

Example of LAG - 1



1. $DList \rightarrow D \mid DList\ ;\ D$ **2.** $D \rightarrow T\ L$ $\{L.type\downarrow := T.type\uparrow\}$

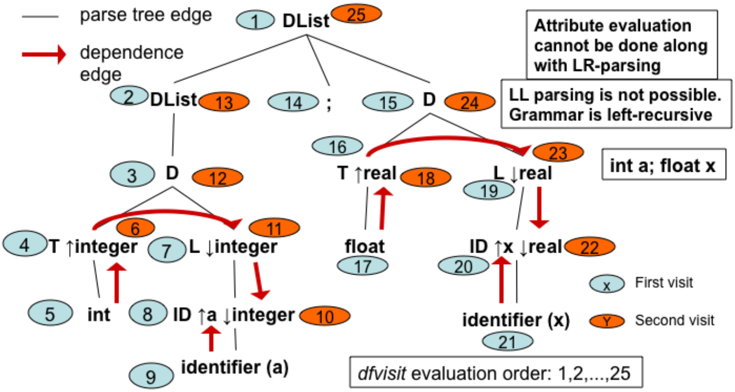
3. $T \rightarrow int$ $\{T.type\uparrow := integer\}$ **4.** $T \rightarrow float$ $\{T.type\uparrow := real\}$

5. $L \rightarrow ID$ $\{ID.type\downarrow := L.type\downarrow\}$

6. $L_1 \rightarrow L_2\ ,\ ID$ $\{L_2.type\downarrow := L_1.type\downarrow;\ ID.type\downarrow := L_1.type\downarrow\}$

7. $ID \rightarrow identifier$ $\{ID.name\uparrow := identifier.name\uparrow\}$

Example of LAG - 1, Evaluation Order



1. $DList \rightarrow D \mid DList\ ;\ D$ **2.** $D \rightarrow T\ L\ \{L.type \downarrow := T.type \uparrow\}$

3. $T \rightarrow int\ \{T.type \uparrow := integer\}$ **4.** $T \rightarrow float\ \{T.type \uparrow := real\}$

5. $L \rightarrow ID\ \{ID.type \downarrow := L.type \downarrow\}$

6. $L_1 \rightarrow L_2\ ,\ ID\ \{L_2.type \downarrow := L_1.type \downarrow; ID.type \downarrow := L_1.type \downarrow\}$

7. $ID \rightarrow identifier\ \{ID.name \uparrow := identifier.name \uparrow\}$