

Syntax-Directed Definitions

Compiler Construction, Lecture 12

K.R. Chowdhary, Professor of CS

August 19, 2019 | Fall 2019

Inherited and Synthesized Attributes

A **syntax-directed definition (SDD)** is a context-free grammar plus attributes (on symbols) and rules (on productions). $X.a$ is the value of attribute a at a parse-tree node labelled X ; attributes can be numbers, types, table references, strings, even code.

Synthesized attribute of A at node N

Defined by a rule of the production **with** A as **head**, only from attributes of N 's **children** and of N itself.

Inherited attribute of B at node N

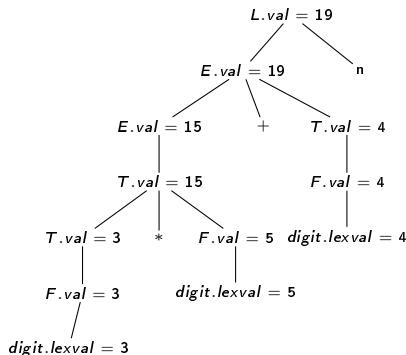
Defined by a rule of the production at N 's **parent**, with B in the body; only from the parent, N itself and N 's **siblings**.

- An inherited attribute may *not* depend on the node's children; a synthesized one *may* use inherited attributes at the same node
- **Terminals** have synthesized attributes (e.g. `lexval`, supplied by the lexical analyser) but no inherited ones, and no rules compute them
- Inherited attributes help when the parse-tree structure does not match the abstract syntax, e.g. in grammars designed for top-down parsing

Example: Desk Calculator (S-Attributed) and Annotated Tree

	Production	Semantic rule
1	$L \rightarrow E n$	$L.val = E.val$
2	$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3	$E \rightarrow T$	$E.val = T.val$
4	$T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
5	$T \rightarrow F$	$T.val = F.val$
6	$F \rightarrow (E)$	$F.val = E.val$
7	$F \rightarrow digit$	$F.val = digit.lexval$

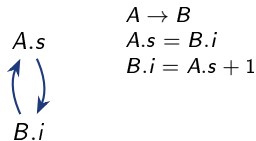
Every attribute is synthesized, so the SDD is **S-attributed**: each rule computes the head's attribute from the body.



Annotated parse tree for $3 * 5 + 4n$, computed bottom-up.

Evaluation Order and Dependency Graphs

With only synthesized attributes, any bottom-up order (e.g. postorder) works. With inherited ones there may be **no valid order**:



These rules are circular: neither value can be computed first. Detecting cycles for all parse trees is computationally hard, so useful SDD subclasses that guarantee an order are used.

Dependency graph

A node for each attribute of each parse-tree node; an edge $M \rightarrow N$ means M 's value is needed to compute N 's.

- **Synthesized** $A.b$ defined from child $X.c$: edge $X.c \rightarrow A.b$
- **Inherited** $B.i$ defined from $X.a$ (parent or sibling): edge $X.a \rightarrow B.i$

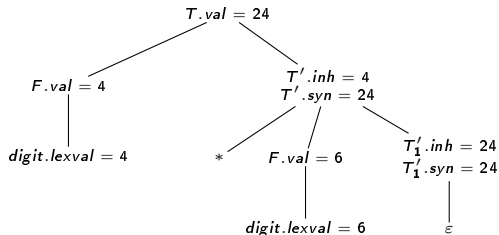
Topological sort

An allowed evaluation order lists the nodes so every edge goes from earlier to later. With a cycle, none exists; without one, one always does: remove a node with no incoming edge, repeat.

Example: Inherited Attributes for Top-Down Parsing

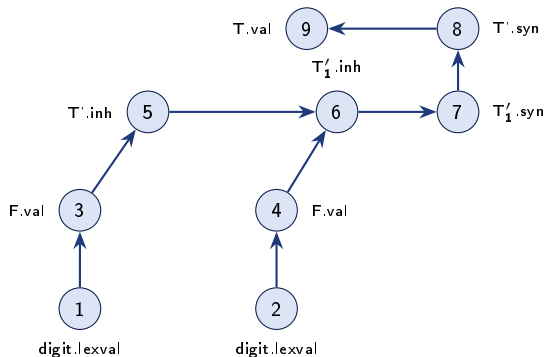
	Production	Semantic rules
1	$T \rightarrow F T'$	$T'.inh = F.val$ $T.val = T'.syn$
2	$T' \rightarrow * F T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3	$T' \rightarrow \epsilon$	$T'.syn = T'.inh$
4	$F \rightarrow digit$	$F.val = digit.lexval$

The left operand is passed to the operator by $T'.inh$ (left to right), and the final product returns up through $T'.syn$.



Annotated parse tree for $4 * 6$: $T'.inh = 4$, then
 $T'_1.inh = 4 \times 6 = 24$.

Dependency Graph for $4 * 6$ and Topological Sorts



- $F.val$ depends on $digit.lexval$ (edges $1 \rightarrow 3$, $2 \rightarrow 4$)
- $T'.inh = F.val$ gives $3 \rightarrow 5$
- $T'_1.inh = T'.inh \times F.val$ gives $5 \rightarrow 6$ and $4 \rightarrow 6$; the multiplication happens at node 6
- $T'_1.syn = T'_1.inh$, then $T'.syn$, then $T.val$: $6 \rightarrow 7 \rightarrow 8 \rightarrow 9$

Topological orders

No cycles, so valid orders exist:
1, 2, 3, 4, 5, 6, 7, 8, 9 and also
1, 3, 5, 2, 4, 6, 7, 8, 9.

S-Attributed vs. L-Attributed SDDs

S-attributed

Every attribute is synthesized. Evaluate by *postorder*: for each child left to right, recurse, then evaluate the node's attributes. This is the order in which an LR parser reduces, so it fits bottom-up parsing.

L-attributed

Each attribute is synthesized, or inherited under limits. For $A \rightarrow X_1 \cdots X_n$, rules for $X_i.a$ may use only:

1. inherited attributes of the head A
2. attributes of $X_1, \dots, X_{i-1}$ (to the left)
3. attributes of X_i itself, with no cycle

Information flows “from above or from the left”.

Table 12.2 is L-attributed: $T'.inh = F.val$ uses F , left of T' ; $T'_1.inh$ uses the head's $T'.inh$ and $F.val$, left of T'_1 .

Not L-attributed: $A \rightarrow BC$ with $A.s = B.b$, $B.i = f(C.c, A.s)$, since C is to the right of B .

Exercises

1. Annotated trees for $(3 + 4) * (5 + 6)$, $3 + 4 * (5 + 6)$, $3 + 4 * 5 + 6$
2. All topological orders of the $4 * 6$ graph
3. Why can't $a * b * c$ use only synthesized attributes in the top-down grammar?
4. Extend Table 12.2 to $+$ and parentheses
5. Circular rules: examples and challenges