

Attribute Grammars

- An extension of CFGs to define “semantics” of sentences in language
- Knuth, 1968

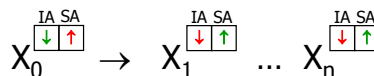
Attributes

- Let G be a context-free grammar $\langle V, \Sigma, S, \rightarrow \rangle$
- Associate with every $X \in (V \cup \Sigma)$ a set of attributes $A(X)$
- Notation. If $a \in A(X)$, we denote it $X.a$
- Let $A(X)$ be partitioned into disjoint sets
 - synthesized attributes, $SA(X)$
 - inherited attributes, $IA(X)$



Occurrences

- Let p be a production $X_0 \rightarrow X_1 \dots X_n$ of G
- Each X_i is a **symbol occurrence** of p
- **Input**(p) = $IA(X_0) \oplus SA(X_1) \oplus \dots \oplus SA(X_n)$
- **Output**(p) = $SA(X_0) \oplus IA(X_1) \oplus \dots \oplus IA(X_n)$
- Each attribute in **Input**(p) or **Output**(p) is an **attribute occurrence** of p



Equations

- Let p be a production $X_0 \rightarrow X_1 \dots X_n$ of G
- An **attribute equation** of p defines $a \in \text{Output}(p)$ in terms of attributes in $\text{Input}(p) \oplus \text{Output}(p)$
- An attribute grammar is **well formed** if
 - $IA(S) = \emptyset$
 - $SA(a) = \emptyset$, for all $a \in \Sigma$
 - Every output attribute of every production has precisely 1 defining equation
- An attribute grammar is in **normal form** if only input attributes occur on RHS of equations

Example

- **Productions**
 - $S \rightarrow E$
 - $E \rightarrow E + E$
 - $E \rightarrow \text{NUM}$
 - $E \rightarrow \text{ID}$
 - $E \rightarrow \text{let ID} = E \text{ in } E$
- **Sample sentence**
 - let** $x = 1$ **in let** $y = x+1$ **in** $x+y$
- **Attributes**
 - Inherited: $E.\text{env}$
 - Synthesized: $S.\text{value}, E.\text{value}, \text{NUM}.\text{value}, \text{ID}.\text{name}$

Example, cont.

$S \rightarrow E$

$E.\text{env} = \text{EmptyEnvironment}()$
 $S.\text{value} = E.\text{value}$

$E_0 \rightarrow E_1 + E_2$

$E_1.\text{env} = E_0.\text{env}$
 $E_2.\text{env} = E_0.\text{env}$
 $E_0.\text{value} = E_1.\text{value} + E_2.\text{value}$

$E \rightarrow \text{NUM}$

$E.\text{value} = \text{NUM}.\text{value}$

$E \rightarrow \text{ID}$

$E.\text{value} = \text{Lookup}(\text{ID}.\text{name}, E.\text{env})$

$E_0 \rightarrow \text{let ID} = E_1 \text{ in } E_2$

$E_1.\text{env} = E_0.\text{env}$
 $E_2.\text{env} = \text{Insert}(\text{ID}.\text{name}, E_1.\text{value}, E_0.\text{env})$
 $E_0.\text{value} = E_2.\text{value}$

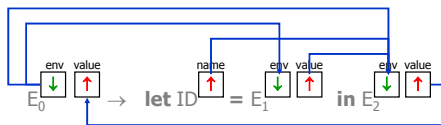
Direct Dependency Graph

- Let p be a production $X_0 \rightarrow X_1 \dots X_n$ of G
- D_p , the **direct dependency graph** of p , is the directed graph $\langle A(p), E(p) \rangle$, where
 - Nodes: $A(p) = \text{Input}(p) \oplus \text{Output}(p)$
 - Edges: $E(p) = \{ \langle a_1, a_2 \rangle \mid a_2 \text{ depends on } a_1 \}$
- An attribute grammar is **locally acyclic** if for every production p , D_p is acyclic

Example, cont.

$E_0 \rightarrow \text{let ID} = E_1 \text{ in } E_2$

$E_1.\text{env} = E_0.\text{env}$
 $E_2.\text{env} = \text{Insert}(\text{ID}.\text{name}, E_1.\text{value}, E_0.\text{env})$
 $E_0.\text{value} = E_2.\text{value}$

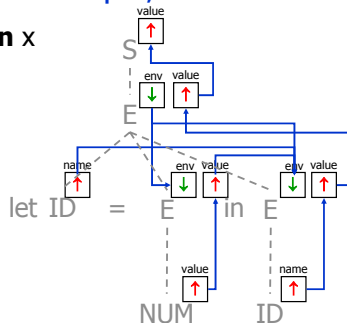


Dependency Graph

- Let T be a derivation tree for some $x \in L(G)$
 - Each subtree corresponding to production p is a **production instance** in T
 - Each symbol occurrence in p is a **symbol instance** in T
 - Each attribute occurrence in p is an **attribute instance** in T
 - Each edge in D_p is a dependence instance in T
- $D(T)$, the **dependency graph** for T , has
 - Nodes: the attribute instances of T
 - Edges: the dependence instances of T

Example, cont.

$\text{let } x = 1 \text{ in } x$



Noncircularity

- An attribute grammar is **noncircular** if for every derivation tree, $D(T)$ is acyclic
- We are only interested in noncircular grammars

Evaluation

- Given a derivation tree T , evaluate the attribute instances of T in **topological order** w.r.t. $D(T)$
- Dynamic evaluation: Obtain the topological order using either
 - topological sort, or
 - depth first search backwards from nodes of out-degree 0
- Static evaluation: Analyze the grammar in advance and determine tree traversal schemes with interleaved evaluations such that for any possible derivation tree T , evaluations will be in topological order

Topological Sort

```
W := ∅;
for each node n with indegree(n)=0 do
  W := W ∪ {n};
while S ≠ ∅ do
  select n from W;
  remove n from W;
  for each successor n' of n do
    remove edge <n,n'>;
    if indegree(n')=0 then W := W ∪ {n'}
```

S-attributed

- An attribute grammar is S-attributed iff it only has synthesized attributes.
- Evaluation: Use end-order traversal of derivation tree (e.g., during a bottom-up parse) to obtain topological evaluation order
- Yacc, Bison, & Cup only support S-attributed grammars

L-attributed

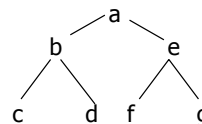
- Defined so that can be evaluated in one left-to-right pass, (e.g., during a top-down parse)
- Every RHS inherited attribute depends only on
 - LHS inherited
 - any RHS attribute to the left
- Every LHS synthesized attribute depends only on
 - LHS inherited
 - any RHS

Alternating Pass Evaluation

- Alternate between L-attributed and R-attributed passes.
- In pass i , all attributes evaluated in previous passes are known values.
- An attribute grammar is **alternating pass** if there exists k alternating passes sufficient to evaluate any derivation tree T

Efficient Use of Sequential Storage

- Reverse of left-to-right endorder is right-to-left preorder (and vice-versa) so can make efficient use of sequential storage medium



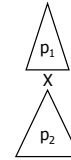
Endorder: c d b f g e a

Right-to-left preorder: a e g f b d c

Partitioned Evaluation

- Each production $X_0 \rightarrow X_1 \dots X_n$ will have an associated **plan**.
- A plan is a linear sequence of **instructions**.
- An instruction is one of
 - **EVAL** $X_i.a$ evaluate attribute a of symbol X_i
 - **VISIT**(r,i) visit neighbor i for the r -th time [child 0 = parent]

Partitioned Evaluation, cont.



- The attributes of X constitute the **interface** between plans for p_1 and p_2 . The plan for p_1 evaluates inherited attributes of X , and the plan for p_2 evaluates synthesized attributes of X

Partitioned Evaluation, cont.

- The plan for p_1 must be consistent with the plan for every production
 $X \rightarrow \alpha$
- The plan for p_2 must be consistent with the plans for every production
 $A \rightarrow \alpha X \beta$