Compiler Construction

Lecture 15: Semantic Analysis III (Attribute Evaluation)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/cc08/

Summer semester 2008

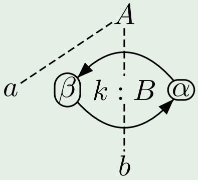
Outline

Solvability of the Attribute Equation System

Example

- $A \to aB, B \to b \in P$
- $\alpha \in \text{syn}(B)$, $\beta \in \text{inh}(B)$
- $\beta.2 = f(\alpha.2) \in E_{A \to aB}$
- $\alpha.0 = g(\beta.0) \in E_{B \to b}$

$\Longrightarrow$ cyclic dependency:



$$E_t : \quad \begin{aligned} \beta.k &= f(\alpha.k) \\ \alpha.k &= g(\beta.k) \end{aligned}$$

$\Longrightarrow$ for $V^\alpha := V^\beta := \mathbb{N}$, $g(x) := x$, and

- $f(x) := x + 1$: no solution
- $f(x) := 2x$: exactly one solution $(v(\alpha.k) = v(\beta.k) = 0)$
- $f(x) := x$: infinitely many solutions $(v(\alpha.k) = v(\beta.k) = y$ for any $y \in \mathbb{N})$

Circularity of Attribute Grammars

Goal: unique solvability of equation system

$\Longrightarrow$ avoid cyclic dependencies

Definition (Circularity)

An attribute grammar $\mathfrak{A} = \langle G, E, V \rangle \in AG$ is called circular if there exists a syntax tree t such that the attribute equation system E_t is recursive (i.e., some attribute variable of t depends on itself). Otherwise it is called noncircular.

Remark: because of the division of Var_π into In_π and Out_π, cyclic dependencies cannot occur at production level.

Attribute Dependency Graphs and Circularity I

Observation: a cycle in the dependency graph D_t of a given syntax tree t is caused by the occurrence of a "cover" production $\pi = A_0 \rightarrow w_0 A_1 w_1 \ldots A_r w_r \in P$ in a node k_0 of t such that

- the dependencies in E_{k_0} yield the "upper end" of the cycle and
- for at least one $i \in [r]$, some attributes in $\text{syn}(A_i)$ depend on attributes in $\text{inh}(A_i)$.

Example

on the board

To identify such "critical" situations we need to determine the possible ways in which attributes in $\text{syn}(A_i)$ can depend on attributes in $\text{inh}(A_i)$.

Definition (Attribute dependence)

Let $\mathfrak{A} = \langle G, E, V \rangle \in AG$ with $G = \langle N, \Sigma, P, S \rangle$.

- If t is a syntax tree with root label $A \in N$ and root node k, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \to_t^+ \alpha.k$, then α is dependent on β below A in t (notation: $\beta \stackrel{A}{\hookrightarrow} \alpha$).
- For every syntax tree t with root label $A \in N$,
$$is(A, t) := \{(\beta, \alpha) \in \mathrm{inh}(A) \times \mathrm{syn}(A) \mid \beta \stackrel{A}{\hookrightarrow} \alpha \text{ in } t\}.$$
- For every $A \in N$,
$$IS(A) := \{is(A, t) \mid t \text{ syntax tree with root label A}\} \subseteq 2^{Inh \times Syn}.$$

Remark: it is important that $IS(A)$ is a system of attribute dependence sets, not a union (later: strong noncircularity).

Example

on the board

The Circularity Test

Algorithm (Circularity test for attribute grammars)

Input: $\mathfrak{A} = \langle G, E, V \rangle \in AG$ *with* $G = \langle N, \Sigma, P, S \rangle$

Procedure:

1. *for every* $A \in N$, *iteratively construct* $IS(A)$ *as follows:*
   1. *if* $\pi = A \to w \in P$, *then* $is[\pi] \in IS(A)$
   2. *if* $\pi = A \to w_0 A_1 w_1 \ldots A_r w_r \in P$ *and* $is_i \in IS(A_i)$ *for every* $i \in [r]$, *then* $is[\pi; is_1, \ldots, is_r] \in IS(A)$
2. *test whether* $\mathfrak{A}$ *is circular by checking if there exist* $\pi = A \to w_0 A_1 w_1 \ldots A_r w_r \in P$ *and* $is_i \in IS(A_i)$ *for every* $i \in [r]$ *such that the following relation is cyclic:*
   $$\to_\pi \cup \bigcup_{i=1}^{r} \{(\beta.p_i, \alpha.p_i) \mid (\beta, \alpha) \in is_i\}$$
   (where $p_i := \sum_{j=1}^{i} |w_{j-1}| + i$*)*

Output: *"yes" or "no"*

Outline

Correctness and Complexity of Circularity Test

Theorem 15.1 (Correctness of the circularity test)

An attribute grammar is circular iff Algorithm 14.15 yields the answer "yes".

Proof.

by induction on the syntax tree t with cyclic D_t □

Lemma 15.2

The time complexity of the circularity test is exponential in the size of the attribute grammar (= maximal length of right-hand sides of productions).

Proof.

by reduction of the word problem of alternating Turing machines (see M. Jazayeri: *A Simpler Construction for Showing the Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars*, Comm. of the ACM 28(4), 1981, pp. 715–720) □

Outline

Simplifying the Circularity Test

Idea: to simplify the circularity test, do not distinguish between attribute dependences which are caused by different syntax trees

Definition 15.3 (Attribute dependence (modified))

Let $\mathfrak{A} = \langle G, E, V \rangle \in AG$ with $G = \langle N, \Sigma, P, S \rangle$.

- Reminder: if t is a syntax tree with root label $A \in N$ and root node k, $\alpha \in \text{syn}(A)$, and $\beta \in \text{inh}(A)$ such that $\beta.k \to_t^+ \alpha.k$, then α is dependent on β below A in t (notation: $\beta \overset{A}{\hookrightarrow} \alpha$).
- For every $A \in N$,

$$\begin{aligned} IS'(A) &:= \{(\beta, \alpha) \mid \beta \overset{A}{\hookrightarrow} \alpha \text{ in some syntax tree with root label A}\} \\ &\subseteq \mathit{Inh} \times \mathit{Syn} \end{aligned}$$

The Strong Circularity Test

Algorithm 15.4 (Strong circularity test for attribute grammars)

Input: $\mathfrak{A} = \langle G, E, V \rangle \in AG$ *with* $G = \langle N, \Sigma, P, S \rangle$

Procedure:

1. *for every* $A \in N$, *iteratively construct* $IS'(A)$ *as follows:*
   1. *if* $\pi = A \to w \in P$, *then* $is[\pi] \subseteq IS'(A)$
   2. *if* $\pi = A \to w_0 A_1 w_1 \ldots A_r w_r \in P$, *then* $is[\pi; IS'(A_1), \ldots, IS'(A_r)] \subseteq IS'(A)$
2. *test whether there exists* $\pi = A \to w_0 A_1 w_1 \ldots A_r w_r \in P$ *such that the following relation is cyclic:*

   $$\to_\pi \cup \bigcup_{i=1}^{r} \{(\beta.p_i, \alpha.p_i) \mid (\beta, \alpha) \in IS'(A_i)\}$$

   (where $p_i := \sum_{j=1}^{i} |w_{j-1}| + i$*)*

Output: *"yes" or "no"*

Example 15.5

on the board

Strongly Noncircular Attribute Grammars I

Definition 15.6 (Strong noncircularity)

An attribute grammar is called strongly noncircular if Algorithm 15.4 yields the answer “no”.

Lemma 15.7

The time complexity of the strong circularity test is polynomial in the size of the attribute grammar (= maximal length of right-hand sides of productions).

Proof.

omitted □

Strongly Noncircular Attribute Grammars II

Lemma 15.8

1. *Every strongly noncircular attribute grammar is noncircular.*
2. *There are noncircular attribute grammars which are not strongly noncircular.*

Proof.

1. Clear since $is \subseteq IS'(A)$ for every $A \in N$ and $is \in IS(A)$
2. The attribute grammar in Example 15.5 is noncircular but not strongly noncircular (on the board).

□

Outline

Attribute Evaluation Methods

Given:
- (strongly) noncircular attribute grammar $\mathfrak{A} = \langle G, E, V \rangle \in AG$
- syntax tree t of G
- valuation $v : Syn_\Sigma \to V$ where $Syn_\Sigma := \{\alpha.k \mid k \text{ labelled by } a \in \Sigma, \alpha \in \text{syn}(a)\} \subseteq Var_t$

Goal: extend v to (partial) solution $v : Var_t \to V$

Methods:
1. Topological sorting of D_t:
   1. start with attribute variables which depend at most on synthesized attributes of terminals
   2. proceed by successive substitution
2. Recursive functions (for strongly noncircular AGs):
   1. for every $A \in N$ and $\alpha \in \text{syn}(A)$, define evaluation function $g_{A,\alpha}$ with the following parameters:
      - the node of t where α has to be evaluated and
      - all inherited attributes of A on which α (potentially) depends
   2. for every $\alpha \in \text{syn}(S)$, evaluate $g_{S,\alpha}(k_0)$ where k_0 denotes the root of t
3. Special cases: S-attributed grammars (`yacc`), L-attributed grammars

Outline

Attribute Evaluation by Topological Sorting

Algorithm 15.9 (Evaluation by topological sorting)

Input: *noncircular* $\mathfrak{A} = \langle G, E, V \rangle \in AG$, *syntax tree* t *of* G, *valuation* $v : Syn_\Sigma \to V$

Procedure:

1. *let* $Var := Var_t \setminus Syn_\Sigma$ (* *attributes to be evaluated* *)
2. *while* $Var \neq \emptyset$ *do*
   1. *let* $x \in Var$ *such that* $\{y \in Var \mid y \to_t x\} = \emptyset$
   2. *let* $x = f(x_1, \ldots, x_n) \in E_t$
   3. *let* $v(x) := f(v(x_1), \ldots, v(x_n))$
   4. *let* $Var := Var \setminus \{x\}$

Output: *solution* $v : Var_t \to V$

Remark: noncircularity guarantees that in step 2.1 at least one such x is available

Example 15.10

see Examples 13.1 and 13.2 (Knuth's binary numbers)

Outline

Attribute Evaluation by Recursive Functions

Restriction: only for strongly noncircular attribute grammars

Principle:

1. for every $A \in N$ and $\alpha \in \text{syn}(A)$, define evaluation function $g_{A,\alpha}$ with the following parameters:
   - the node of t where α has to be evaluated (which is labelled by A) and
   - all inherited attributes of A on which α (potentially) depends (that is, $\{\beta \in \text{inh}(A) \mid (\beta, \alpha) \in IS'(A)\}$)
2. given a syntax tree t with root k_0, evaluate $g_{S,\alpha}(k_0)$ for every $\alpha \in \text{syn}(S)$

Result: evaluates synthesized attribute variables at root of t and all attribute variables on which they actually depend (according to E_t)

Definition of Evaluation Functions I

For every $A \in N$ and $\alpha \in \text{syn}(A)$, let

- $IS'(A) \subseteq \text{inh}(A) \times \text{syn}(A)$ as computed by strong circularity test (Algorithm 15.4)
- $\text{inh}(A, \alpha) := \{\beta \in \text{inh}(A) \mid (\beta, \alpha) \in IS'(A)\}$
- $A \to \gamma_1 \mid \ldots \mid \gamma_m$ all A-productions in P

Then $g_{A,\alpha}$ is given by

$$g_{A,\alpha}(k_0, \text{inh}(A,\alpha)) := \textbf{case } \text{production applied at } k_0 \textbf{ of} \\ \vdots \\ A \to \gamma_j : \text{eval}(\alpha.0) \\ \vdots \\ \textbf{end}$$

with

$$\text{eval}(\alpha.i) := \begin{cases} \alpha & \text{if } \alpha \in \text{inh}(A), i = 0 \\ f(\text{eval}(\alpha_1.i_1), \ldots, \text{eval}(\alpha_n.i_n)) & \text{if } \alpha.i \in In_{A\to\gamma_j}, \alpha.i = \\ & \quad f(\alpha_1.i_1, \ldots, \alpha_n.i_n) \in E_{A\to\gamma_j} \\ g_{Y_i,\alpha}(k_i, \text{eval}(\beta_1.i), \ldots, \text{eval}(\beta_l.i)) & \text{if } \alpha \in Syn, i > 0, Y_i \in N, \\ & \quad \text{inh}(Y_i, \alpha) = \{\beta_1, \ldots, \beta_l\} \\ v(\alpha.i) & \text{if } \alpha \in Syn, i > 0, Y_i \in \Sigma \end{cases}$$

where $\gamma_j = Y_1 \ldots Y_r$, and where k_i denotes the ith successor of k_0

Definition of Evaluation Functions II

Example 15.11 (cf. Example 13.2)

G'_B:

$S \to L$	$v.0 = v.1$ $p.1 = 0$
$S \to L\,.\,L$	$v.0 = v.1 + v.3$ $p.1 = 0$ $p.3 = -l.3$
$L \to B$	$v.0 = v.1$ $l.0 = 1$ $p.1 = p.0$
$L \to LB$	$v.0 = v.1 + v.2$ $l.0 = l.1 + 1$ $p.1 = p.0 + 1$ $p.2 = p.0$
$B \to \mathtt{0}$	$v.0 = 0$
$B \to \mathtt{1}$	$v.0 = 2^{p.0}$

$A \in N$	S	L	B
$IS'(A)$	$\emptyset$	$\{(p,v)\}$	$\{(p,v)\}$

$$
\begin{aligned}
g_{S,v}(k_0) = \ & \textbf{case}\ \text{production}(k_0)\ \textbf{of} \\
& \quad S \to L : g_{L,v}(k_1, 0) \\
& \quad S \to L\,.\,L : g_{L,v}(k_1, 0) + \\
& \qquad g_{L,v}(k_3, -g_{L,l}(k_3)) \\
& \textbf{end}
\end{aligned}
$$

$$
\begin{aligned}
g_{L,v}(k_0, p) = \ & \textbf{case}\ \text{production}(k_0)\ \textbf{of} \\
& \quad L \to B : g_{B,v}(k_1, p) \\
& \quad L \to LB : g_{L,v}(k_1, p+1) \\
& \qquad + g_{B,v}(k_2, p) \\
& \textbf{end}
\end{aligned}
$$

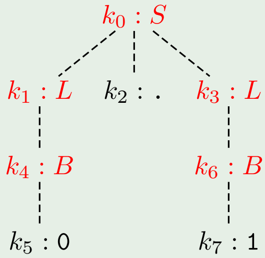
$$
\begin{aligned}
g_{L,l}(k_0) = \ & \textbf{case}\ \text{production}(k_0)\ \textbf{of} \\
& \quad L \to B : 1 \\
& \quad L \to LB : g_{L,l}(k_1) + 1 \\
& \textbf{end}
\end{aligned}
$$

$$
\begin{aligned}
g_{B,v}(k_0, p) = \ & \textbf{case}\ \text{production}(k_0)\ \textbf{of} \\
& \quad B \to \mathtt{0} : 0 \\
& \quad B \to \mathtt{1} : 2^p \\
& \textbf{end}
\end{aligned}
$$

Example 15.11 (continued)

$$
\begin{aligned}
g_{S,v}(k_0) = \;& \textbf{case } \text{production}(k_0) \textbf{ of} \\
& \quad S \to L : g_{L,v}(k_1, 0) \\
& \quad S \to L\,.\,L : g_{L,v}(k_1, 0) + \\
& \qquad\quad g_{L,v}(k_3, -g_{L,l}(k_3)) \\
& \textbf{end} \\
g_{L,v}(k_0, p) = \;& \textbf{case } \text{production}(k_0) \textbf{ of} \\
& \quad L \to B : g_{B,v}(k_1, p) \\
& \quad L \to LB : g_{L,v}(k_1, p+1) \\
& \qquad\quad + g_{B,v}(k_2, p) \\
& \textbf{end} \\
g_{L,l}(k_0) = \;& \textbf{case } \text{production}(k_0) \textbf{ of} \\
& \quad L \to B : 1 \\
& \quad L \to LB : g_{L,l}(k_1) + 1 \\
& \textbf{end} \\
g_{B,v}(k_0, p) = \;& \textbf{case } \text{production}(k_0) \textbf{ of} \\
& \quad B \to \mathtt{0} : 0 \\
& \quad B \to \mathtt{1} : 2^p \\
& \textbf{end}
\end{aligned}
$$

Syntax tree t:



$$
\begin{aligned}
& g_{S,v}(k_0) \\
& \quad = g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
& \quad = g_{B,v}(k_4, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
& \quad = 0 + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
& \quad = 0 + g_{B,v}(k_6, -g_{L,l}(k_3)) \\
& \quad = 0 + 2^{-g_{L,l}(k_3)} \\
& \quad = 0 + 2^{-1} \\
& \quad = 0.5
\end{aligned}
$$

Why Strong Noncircularity?

If the attribute grammar is not strongly noncircular, then the construction of the evaluation functions fails.

Example 15.12 (cf. Example 15.5)

$S \rightarrow A$ $\quad \alpha.0 = \alpha_2.1$

$\quad \beta_1.1 = \alpha_1.1$

$\quad \beta_2.1 = \alpha_2.1$

$A \rightarrow \texttt{a}$ $\quad \alpha_1.0 = \beta_2.0$

$\quad \alpha_2.0 = 2$

$A \rightarrow \texttt{b}$ $\quad \alpha_1.0 = 1$

$\quad \alpha_2.0 = \beta_1.0$

In Example 15.5:

$IS'(A) = \{(\beta_2, \alpha_1), (\beta_1, \alpha_2)\}$

Definition of $g_{S,\alpha}$:

$$\begin{aligned} g_{S,\alpha}(k_0) &= \text{eval}(\alpha.0) \\ &= \text{eval}(\alpha_2.1) \\ &= g_{A,\alpha_2}(k_1, \text{eval}(\beta_1.1)) \\ &= g_{A,\alpha_2}(k_1, \text{eval}(\alpha_1.1)) \\ &= g_{A,\alpha_2}(k_1, g_{A,\alpha_1}(k_1, \text{eval}(\beta_2.1))) \\ &= g_{A,\alpha_2}(k_1, g_{A,\alpha_1}(k_1, \text{eval}(\alpha_2.1)) \end{aligned}$$

$\Longrightarrow$ does not terminate!