Compiler Construction

Lecture 22:
Evaluation of Strongly Noncircular Attribute Grammars

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/cc10/

Winter semester 2010/11

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 (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, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \rightarrow_t^+ \alpha.k$, then α is dependent on β below A in t (notation: $\beta \overset{A}{\hookrightarrow} \alpha$).
- For every $A \in N$,

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

The Strong Circularity Test

Algorithm (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

on the board

Strongly Noncircular Attribute Grammars

Definition (Strong noncircularity)

An attribute grammar is called strongly noncircular if Algorithm 21.3 yields the answer "no".

Lemma

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 □

Outline

(Strongly) Noncircular Attribute Grammars

Lemma 22.1

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 21.4 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 (Syn_Σ)
   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

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 21.3)
- $\text{inh}(A, \alpha) := \{\beta \in \text{inh}(A) \mid (\beta, \alpha) \in IS'(A)\}$
- $A \to \delta_1 \mid \ldots \mid \delta_m$ all A-productions in P

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

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

with

$$\text{eval}(\gamma.i) := \begin{cases} \gamma & \text{if } \gamma \in Inh, i = 0 \\ f(\text{eval}(\gamma_1.i_1), \ldots, \text{eval}(\gamma_n.i_n)) & \text{if } \gamma.i \in In_{A \to \delta_j}, \gamma.i = f(\gamma_1.i_1, \ldots, \gamma_n.i_n) \in E_{A \to \delta_j} \\ g_{Y_i,\gamma}(k_i, \text{eval}(\beta_1.i), \ldots, \text{eval}(\beta_l.i)) & \text{if } \gamma \in Syn, i > 0, Y_i \in N, \text{inh}(Y_i, \gamma) = \{\beta_1, \ldots, \beta_l\} \\ v(\gamma.i) & \text{if } \gamma \in Syn, i > 0, Y_i \in \Sigma \end{cases}$$

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

Definition of Evaluation Functions II

Example 22.2 (cf. Example 15.2)

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
$IS'(A)$	$\emptyset$	$\{(p, v)\}$	$\{(p, v)\}$

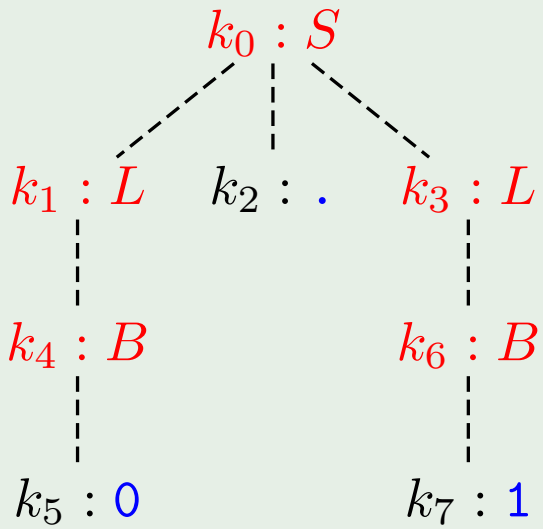
$$
\begin{aligned}
g_{S,v}(k_0) = \ & \textbf{case}\ \text{production}(k_0)\ \textbf{of} \\
& \quad S \to L : g_{L,v}(k_1, 0) \\
& \quad S \to L\,.\,L : g_{L,v}(k_1, 0) + \\
& \qquad g_{L,v}(k_3, -g_{L,l}(k_3)) \\
& \textbf{end} \\
g_{L,v}(k_0, p) = \ & \textbf{case}\ \text{production}(k_0)\ \textbf{of}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

Example Evaluation

Example 22.2 (continued)

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

Syntax tree t:



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

Why Strong Noncircularity?

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

Example 22.3 (cf. Example 21.4)

$$
\begin{array}{lrcl}
S \to A & \alpha.0 & = & \alpha_2.1 \\
 & \beta_1.1 & = & \alpha_1.1 \\
 & \beta_2.1 & = & \alpha_2.1 \\
A \to \texttt{a} & \alpha_1.0 & = & \beta_2.0 \\
 & \alpha_2.0 & = & 2 \\
A \to \texttt{b} & \alpha_1.0 & = & 1 \\
 & \alpha_2.0 & = & \beta_1.0
\end{array}
$$

From Example 21.4:

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

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

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

$\Longrightarrow$ does not terminate!