Compiler Construction

Lecture 15: Semantic Analysis III (Circularity Test)

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/cc09/

Winter semester 2009/10

Outline

1. Repetition: Definition and Circularity of Attribute Grammars
2. Testing Attribute Grammars for Circularity
3. The Circularity Test
4. Correctness and Complexity of the Circularity Test

Formal Definition of Attribute Grammars

Definition (Attribute grammar)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ with $X := N \uplus \Sigma$.

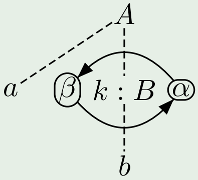
- Let $Att = Syn \uplus Inh$ be a set of (synthesized or inherited) attributes, and let $V = \bigcup_{\alpha \in Att} V^\alpha$ be a union of value sets.
- Let $\text{att} : X \to 2^{Att}$ be an attribute assignment, and let $\text{syn}(Y) := \text{att}(Y) \cap Syn$ and $\text{inh}(Y) := \text{att}(Y) \cap Inh$ for every $Y \in X$.
- Every production $\pi = Y_0 \to Y_1 \ldots Y_r \in P$ determines the set
  $$Var_\pi := \{\alpha.i \mid \alpha \in \text{att}(Y_i), i \in \{0, \ldots, r\}\}$$
  of attribute variables of π with the subsets of inner and outer variables:
  $$\begin{aligned} In_\pi &:= \{\alpha.i \mid (i = 0, \alpha \in \text{syn}(Y_i)) \text{ or } (i \in [r], \alpha \in \text{inh}(Y_i))\} \\ Out_\pi &:= Var_\pi \setminus In_\pi \end{aligned}$$
- A semantic rule of π is an equation of the form
  $$\alpha.i = f(\alpha_1.i_1, \ldots, \alpha_n.i_n)$$
  where $n \in \mathbb{N}$, $\alpha.i \in In_\pi$, $\alpha_j.i_j \in Out_\pi$, and $f : V^{\alpha_1} \times \ldots \times V^{\alpha_n} \to V^\alpha$.
- For each $\pi \in P$, let E_π be a set with exactly one semantic rule for every inner variable of π, and let $E := (E_\pi \mid \pi \in P)$.

Then $\mathfrak{A} := \langle G, E, V \rangle$ is called an attribute grammar: $\mathfrak{A} \in AG$.

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 (see Corollary 14.11).

Attribute Dependency Graphs I

Just as the attribute equation system E_t of a syntax tree t is obtained from the semantic rules of the contributing productions, the dependency graph of t is obtained by "glueing together" the dependency graphs of the productions.

Definition (Tree dependency graph)

Let $\mathfrak{A} = \langle G, E, V \rangle \in AG$, and let t be a syntax tree of G.

- The dependency graph of t is defined by $D_t := \langle Var_t, \to_t \rangle$ where the set of edges $\to_t \subseteq Var_t \times Var_t$ is given by
$$x \to_t y \quad \text{iff} \quad y = f(\ldots, x, \ldots) \in E_t.$$
- D_t is called cyclic if there exists $x \in Var_t$ such that $x \to_t^+ x$.

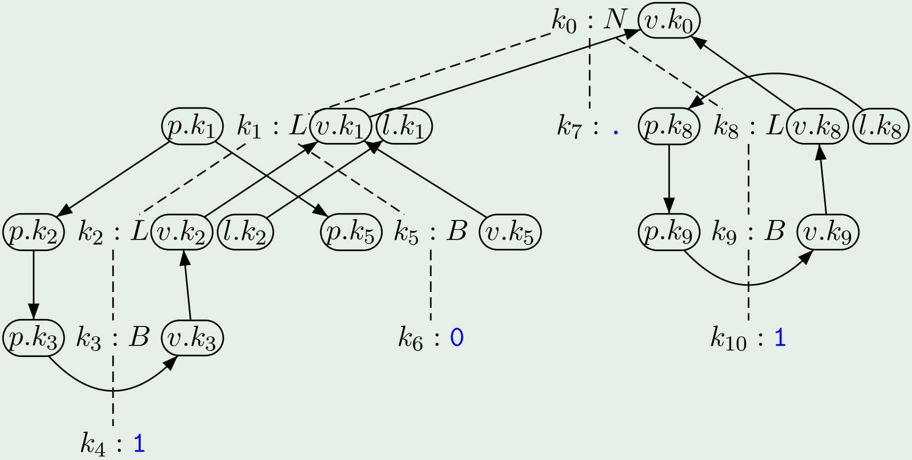
Corollary

An attribute grammar $\mathfrak{A} = \langle G, E, V \rangle \in AG$ is circular iff there exists a syntax tree t of G such that D_t is cyclic.

Attribute Dependency Graphs II

Example (cf. Example 14.1)

(Acyclic) dependency graph of the syntax tree for `10.1`:



Outline

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 15.1

on the board

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

Attribute Dependency Graphs and Circularity II

Definition 15.2 (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, $\alpha \in \text{syn}(A)$, and $\beta \in \text{inh}(A)$ 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 syntax tree t with root label $A \in N$,
  $$is(A, t) := \{(\beta, \alpha) \in \text{inh}(A) \times \text{syn}(A) \mid \beta \overset{A}{\hookrightarrow} \alpha \text{ in } t\}.$$
- 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 15.3

on the board

Outline

1. Repetition: Definition and Circularity of Attribute Grammars
2. Testing Attribute Grammars for Circularity
3. The Circularity Test
4. Correctness and Complexity of the Circularity Test

The Circularity Test I

In the circularity test, the dependency systems $IS(A)$ are iteratively computed. It employs the following notation:

Definition 15.4

Given $\pi = A \to w_0 A_1 w_1 \ldots A_r w_r \in P$ and $is_i \subseteq \text{inh}(A_i) \times \text{syn}(A_i)$ for every $i \in [r]$, let

$$is[\pi; is_1, \ldots, is_r] \subseteq \text{inh}(A) \times \text{syn}(A)$$

be given by

$$is[\pi; is_1, \ldots, is_r] :=$$
$$\left\{(\beta, \alpha) \mid (\beta.0, \alpha.0) \in \left(\to_\pi \cup \bigcup_{i=1}^{r} \{(\beta'.p_i, \alpha'.p_i) \mid (\beta', \alpha') \in is_i\}\right)^+\right\}$$

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

Example 15.5

on the board

The Circularity Test II

Algorithm 15.6 (Circularity test for attribute grammars)

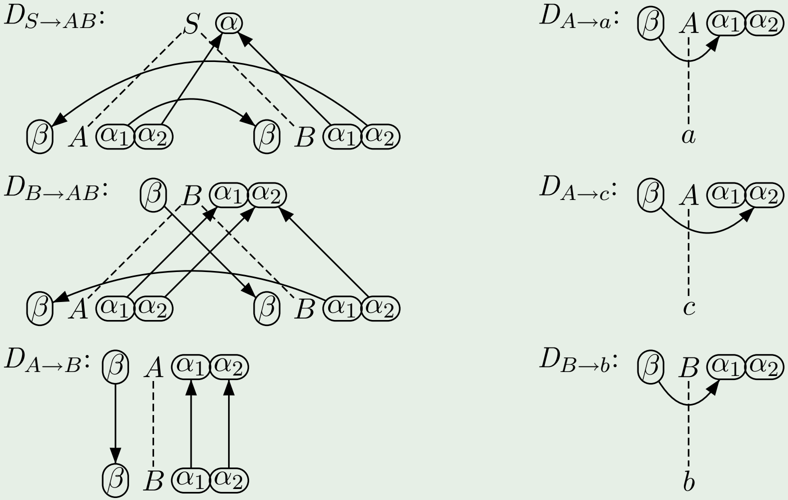
Input: $\mathfrak{A} = \langle G, E, V \rangle \in AG$ *with* $G = \langle N, \Sigma, P, S \rangle$

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"*

Example 15.7



Application of Algorithm 15.6: on the board

Outline

Correctness and Complexity of Circularity Test

Theorem 15.1 (Correctness of circularity test)

An attribute grammar is circular iff Algorithm 15.6 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) □