Compiler Construction

Lecture 13: Semantic Analysis II (Circularity Check)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

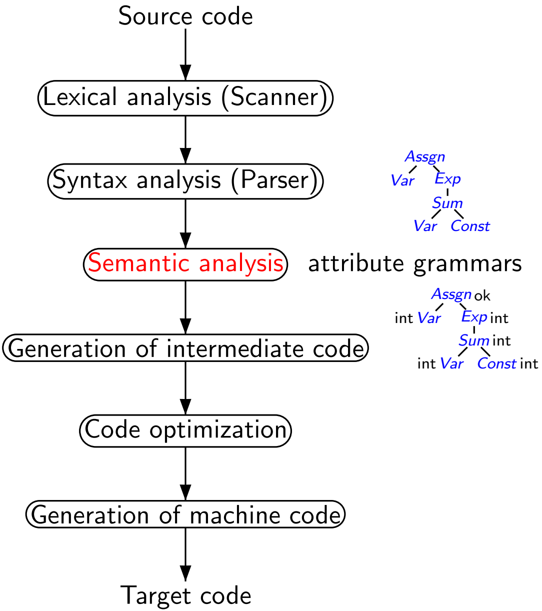
http://moves.rwth-aachen.de/teaching/ss-14/cc14/

Summer Semester 2014

Outline

1. Recap: Attribute Grammars
2. Circularity of Attribute Grammars
3. Attribute Dependency Graphs
4. Testing Attribute Grammars for Circularity
5. The Circularity Check

Conceptual Structure of a Compiler



Attribute Grammars

Goal: compute context-dependent but runtime-independent properties of a given program

Idea: enrich context-free grammar by semantic rules which annotate syntax tree with attribute values

$\Longrightarrow$ Semantic analysis = attribute evaluation

Result: attributed syntax tree

In greater detail:

- With every nonterminal a set of attributes is associated.
- Two types of attributes are distinguished:

  Synthesized: bottom-up computation (from the leaves to the root)

  Inherited: top-down computation (from the root to the leaves)
- With every production a set of semantic rules is associated.

Formal Definition of Attribute Grammars I

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 $\mathrm{att} : X \to 2^{Att}$ be an attribute assignment, and let $\mathrm{syn}(Y) := \mathrm{att}(Y) \cap Syn$ and $\mathrm{inh}(Y) := \mathrm{att}(Y) \cap Inh$ for every $Y \in X$.
- Every production $\pi = Y_0 \to Y_1 \ldots Y_r \in P$ determines the set
  $$Var_\pi := \{\alpha.i \mid \alpha \in \mathrm{att}(Y_i), i \in \{0, \ldots, r\}\}$$
  of attribute variables of π with the subsets of inner and outer variables:
  $$\begin{aligned} In_\pi &:= \{\alpha.i \mid (i = 0, \alpha \in \mathrm{syn}(Y_i)) \text{ or } (i \in [r], \alpha \in \mathrm{inh}(Y_i))\} \\ Out_\pi &:= Var_\pi \setminus In_\pi \end{aligned}$$
- 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: $\mathfrak{A} \in AG$.

Attribution of Syntax Trees I

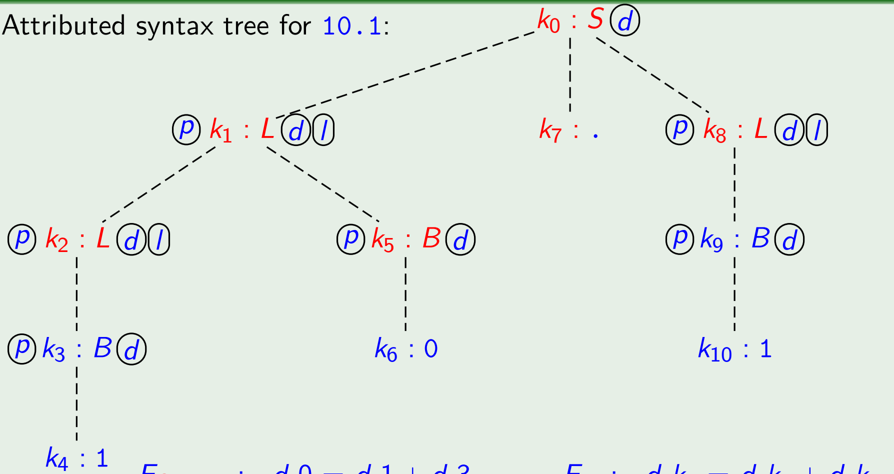
Definition (Attribution of syntax trees)

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

- K determines the set of attribute variables of t:
$$Var_t := \{\alpha.k \mid k \in K \text{ labelled with } Y \in X, \alpha \in \mathrm{att}(Y)\}.$$
- Let $k_0 \in K$ be an (inner) node where production $\pi = Y_0 \to Y_1 \dots Y_r \in P$ is applied, and let $k_1, \dots, k_r \in K$ be the corresponding successor nodes. The attribute equation system E_{k_0} of k_0 is obtained from E_π by substituting every attribute index $i \in \{0, \dots, r\}$ by k_i.
- The attribute equation system of t is given by
$$E_t := \bigcup \{E_k \mid k \text{ inner node of } t\}.$$

Example (cf. Example 12.2)

Attributed syntax tree for `10.1`:



$$E_{S \to L.L}: \quad \begin{aligned} d.0 &= d.1 + d.3 \\ p.1 &= 0 \\ p.3 &= -l.3 \end{aligned} \quad \xrightarrow{\text{subst}} \quad E_{k_0}: \quad \begin{aligned} d.k_0 &= d.k_1 + d.k_8 \\ p.k_1 &= 0 \\ p.k_8 &= -l.k_8 \end{aligned}$$

$$E_{L \to LB}: \quad d.0 = d.1 + d.2 \qquad E_{k_1}: \quad d.k_1 = d.k_2 + d.k_5$$

Attribution of Syntax Trees III

Corollary

For each $\alpha.k \in Var_t$ *except the inherited attribute variables at the root and the synthesized attribute variables at the leaves of* t, E_t *contains exactly one equation with left-hand side* $\alpha.k$.

Assumptions:

- The start symbol does not have inherited attributes: $\mathrm{inh}(S) = \emptyset$.
- Synthesized attributes of terminal symbols are provided by the scanner.

Outline

1. Recap: Attribute Grammars
2. Circularity of Attribute Grammars
3. Attribute Dependency Graphs
4. Testing Attribute Grammars for Circularity
5. The Circularity Check

Solvability of Attribute Equation System I

Definition 13.1 (Solution of attribute equation system)

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

$$v : Var_t \to V$$

such that, for every $\alpha.k \in Var_t$ and $\alpha.k = f(\alpha.k_1, \ldots, \alpha.k_n) \in E_t$,

$$v(\alpha.k) = f(v(\alpha.k_1), \ldots, v(\alpha.k_n)).$$

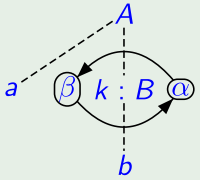
In general, the attribute equation system E_t of a given syntax tree t can have

- no solution,
- exactly one solution, or
- several solutions.

Example 13.2

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

$\Longrightarrow$ cyclic dependency:



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

$\Longrightarrow$ for $V^\alpha := V^\beta := \mathbb{N}$ 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 13.3 (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.

Outline

1. Recap: Attribute Grammars
2. Circularity of Attribute Grammars
3. Attribute Dependency Graphs
4. Testing Attribute Grammars for Circularity
5. The Circularity Check

Attribute Dependency Graphs I

Goal: graphic representation of attribute dependencies

Definition 13.4 (Production dependency graph)

Let $\mathfrak{A} = \langle G, E, V \rangle \in AG$ with $G = \langle N, \Sigma, P, S \rangle$. Every production $\pi \in P$ determines the dependency graph $D_\pi := \langle Var_\pi, \to_\pi \rangle$ where the set of edges $\to_\pi \subseteq Var_\pi \times Var_\pi$ is given by

$$x \to_\pi y \quad \text{iff} \quad y = f(\ldots, x, \ldots) \in E_\pi.$$

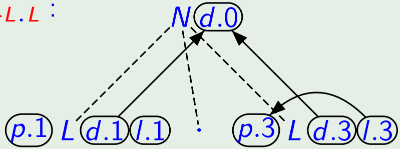
Corollary 13.5

The dependency graph of a production is acyclic (since $\to_\pi \subseteq Out_\pi \times In_\pi$).

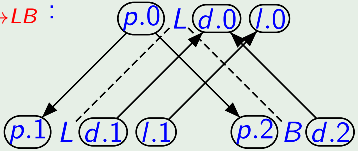
Attribute Dependency Graphs II

Example 13.6 (cf. Example 12.2)

1. $N \to L\,.\,L$:
$$\begin{aligned} d.0 &= d.1 + d.3 \\ p.1 &= 0 \\ p.3 &= -l.3 \end{aligned}$$
$\Longrightarrow$ $D_{N \to L.L}$:



2. $L \to LB$:
$$\begin{aligned} d.0 &= d.1 + d.2 \\ l.0 &= l.1 + 1 \\ p.1 &= p.0 + 1 \\ p.2 &= p.0 \end{aligned}$$
$\Longrightarrow$ $D_{N \to LB}$:



Attribute Dependency Graphs III

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 13.7 (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 $x \to_t^+ x$.

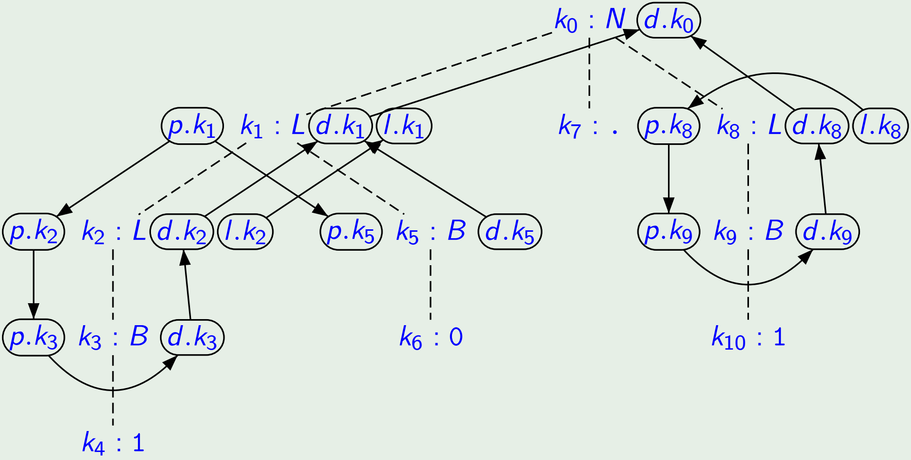
Corollary 13.8

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 IV

Example 13.9 (cf. Example 12.2)

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



Outline

1. Recap: Attribute Grammars
2. Circularity of Attribute Grammars
3. Attribute Dependency Graphs
4. Testing Attribute Grammars for Circularity
5. The Circularity Check

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 $\pi = A_0 \to w_0 A_1 w_1 \ldots A_r w_r \in P$ 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 $\mathrm{syn}(A_i)$ depend on attributes in $\mathrm{inh}(A_i)$.

Example 13.10

on the board

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

Attribute Dependency Graphs and Circularity II

Definition 13.11 (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: $\beta \overset{A}{\hookrightarrow} \alpha$).
- For every syntax tree t with root label $A \in N$,
$$is(A, t) := \{(\beta, \alpha) \in \mathrm{inh}(A) \times \mathrm{syn}(A) \mid \beta \overset{A}{\hookrightarrow} \alpha \text{ in } t\}.$$
- 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 (otherwise: strong noncircularity—see exercises).

Example 13.12

on the board

Outline

1. Recap: Attribute Grammars
2. Circularity of Attribute Grammars
3. Attribute Dependency Graphs
4. Testing Attribute Grammars for Circularity
5. The Circularity Check

The Circularity Check I

In the circularity check, the dependency systems $IS(A)$ are iteratively computed. The following notation is employed:

Definition 13.13

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

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

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 13.14

on the board