

# Compiler Construction

## Lecture 14: Semantic Analysis II

### (Circularity Check)

Thomas Noll

Lehrstuhl für Informatik 2  
(Software Modeling and Verification)

RWTH Aachen University

[noll@cs.rwth-aachen.de](mailto:noll@cs.rwth-aachen.de)

<http://www-i2.informatik.rwth-aachen.de/i2/cc12/>

Summer Semester 2012

Studieren Ohne Grenzen e.V. präsentiert die

# Nacht der Professoren

15.06.

Apollo

22:00

Ab 23:00 legen eure Professoren von der RWTH für den guten Zweck auf:

**Prof. Reicher-Marek** | Philosophie

**Prof. Reicherter** | Neotektonik

**Prof. Bintinesi** | Informatik

**Prof. Panstruga** | Biologie

**Prof. Blank** | Biologie

**Dr. Pratzer** | Physik



4€, ab 23h 5€

Mitmachen? Besuch uns auf [facebook](#) oder unter [www.studieren-ohne-grenzen.org](http://www.studieren-ohne-grenzen.org)



- 1 Repetition: Attribute Grammars
- 2 Circularity of Attribute Grammars
- 3 Attribute Dependency Graphs
- 4 Testing Attribute Grammars for Circularity

**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**

⇒ **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.

## 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  $att : X \rightarrow 2^{Att}$  be an attribute assignment, and let  $syn(Y) := att(Y) \cap Syn$  and  $inh(Y) := att(Y) \cap Inh$  for every  $Y \in X$ .
- Every production  $\pi = Y_0 \rightarrow Y_1 \dots Y_r \in P$  determines the set

$$Var_{\pi} := \{\alpha.i \mid \alpha \in att(Y_i), i \in \{0, \dots, r\}\}$$

of attribute variables of  $\pi$  with the subsets of inner and outer variables:

$$In_{\pi} := \{\alpha.i \mid (i = 0, \alpha \in syn(Y_i)) \text{ or } (i \in [r], \alpha \in inh(Y_i))\}$$
$$Out_{\pi} := Var_{\pi} \setminus In_{\pi}$$

- A semantic rule of  $\pi$  is an equation of the form

$$\alpha.i = f(\alpha_1.i_1, \dots, \alpha_n.i_n)$$

where  $n \in \mathbb{N}$ ,  $\alpha.i \in In_{\pi}$ ,  $\alpha_j.i_j \in Out_{\pi}$ , and  $f : V^{\alpha_1} \times \dots \times V^{\alpha_n} \rightarrow V^{\alpha}$ .

- For each  $\pi \in P$ , let  $E_{\pi}$  be a set with exactly one semantic rule for every inner variable of  $\pi$ , and let  $E := (E_{\pi} \mid \pi \in P)$ .

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

## 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  $t$** :

$$Var_t := \{\alpha.k \mid k \in K \text{ labelled with } Y \in X, \alpha \in \text{att}(Y)\}.$$

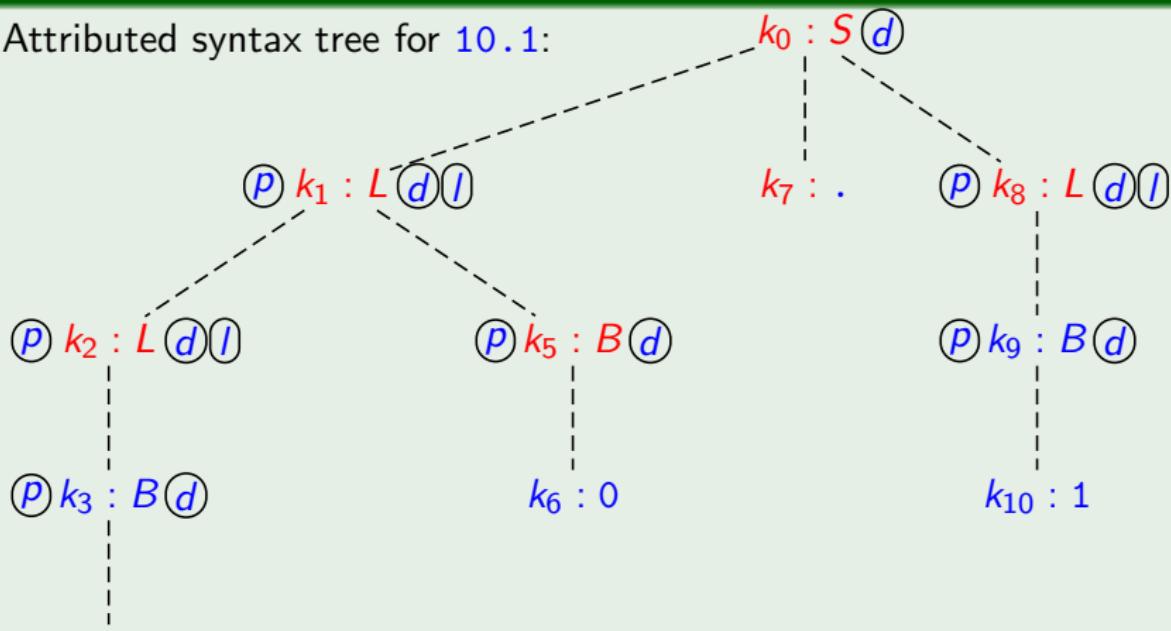
- Let  $k_0 \in K$  be an (inner) node where production  $\pi = Y_0 \rightarrow Y_1 \dots Y_r \in P$  is applied, and let  $k_1, \dots, k_r \in K$  be the corresponding successor nodes. The **attribute equation system  $E_{k_0}$**  of  $k_0$  is obtained from  $E_\pi$  by substituting every attribute index  $i \in \{0, \dots, r\}$  by  $k_i$ .
- The **attribute equation system** of  $t$  is given by

$$E_t := \bigcup \{E_k \mid k \text{ inner node of } t\}.$$

# Attribution of Syntax Trees II

Example (cf. Example 13.2)

Attributed syntax tree for 10.1:



$k_4 : 1$

$$\begin{aligned} E_{S \rightarrow L.L} : \quad & d.0 = d.1 + d.3 \\ & p.1 = 0 \\ & p.3 = -l.3 \end{aligned}$$

subst

$$\begin{aligned} E_{k_0} : \quad & d.k_0 = d.k_1 + d.k_8 \\ p.k_1 = 0 \\ p.k_8 = -l.k_8 \end{aligned}$$

$$E_{I \rightarrow I.R} : \quad d.0 = d.1 + d.2$$

$$E_{k_1} : \quad d.k_1 = d.k_2 + d.k_5$$

## Corollary

For each  $\alpha.k \in \text{Var}_t$  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:  $\text{inh}(S) = \emptyset$ .
- Synthesized attributes of terminal symbols are provided by the scanner.

- 1 Repetition: Attribute Grammars
- 2 Circularity of Attribute Grammars
- 3 Attribute Dependency Graphs
- 4 Testing Attribute Grammars for Circularity

# Solvability of Attribute Equation System I

Definition 14.1 (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

$$v : \text{Var}_t \rightarrow V$$

such that, for every  $\alpha.k \in \text{Var}_t$  and  $\alpha.k = f(\alpha.k_1, \dots, \alpha.k_n) \in E_t$ ,

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

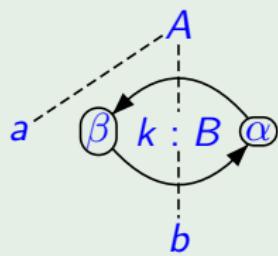
In general, the attribute equation system  $E_t$  of a given syntax tree  $t$  can have

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

## Example 14.2

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

$\implies$  cyclic dependency:



$\implies$  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  $y \in \mathbb{N}$ )

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

**Goal:** unique solvability of equation system  
⇒ avoid cyclic dependencies

## Definition 14.3 (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  $Var_\pi$  into  $In_\pi$  and  $Out_\pi$ , cyclic dependencies cannot occur at production level.

- 1 Repetition: Attribute Grammars
- 2 Circularity of Attribute Grammars
- 3 Attribute Dependency Graphs
- 4 Testing Attribute Grammars for Circularity

# Attribute Dependency Graphs I

**Goal:** graphic representation of attribute dependencies

## Definition 14.4 (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**  $D_\pi := \langle \text{Var}_\pi, \rightarrow_\pi \rangle$  where the set of edges  $\rightarrow_\pi \subseteq \text{Var}_\pi \times \text{Var}_\pi$  is given by

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

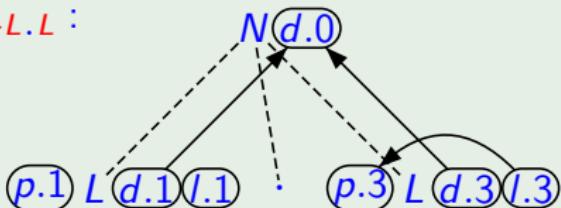
## Corollary 14.5

*The dependency graph of a production is acyclic  
(since  $\rightarrow_\pi \subseteq \text{Out}_\pi \times \text{In}_\pi$ ).*

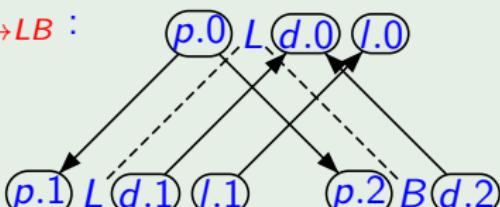
# Attribute Dependency Graphs II

## Example 14.6 (cf. Example 13.2)

1  $N \rightarrow L.L :$   $\Rightarrow D_{N \rightarrow L.L} :$   
 $d.0 = d.1 + d.3$   
 $p.1 = 0$   
 $p.3 = -l.3$



2  $L \rightarrow LB :$   $\Rightarrow D_{N \rightarrow LB} :$   
 $d.0 = d.1 + d.2$   
 $l.0 = l.1 + 1$   
 $p.1 = p.0 + 1$   
 $p.2 = p.0$



# 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 14.7 (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  $D_t := \langle \text{Var}_t, \rightarrow_t \rangle$  where the set of edges,  $\rightarrow_t \subseteq \text{Var}_t \times \text{Var}_t$ , is given by

$$x \rightarrow_t y \quad \text{iff} \quad y = f(\dots, x, \dots) \in E_t.$$

- $D_t$  is called **cyclic** if there exists  $x \in \text{Var}_t$  such that  $x \rightarrow_t^+ x$ .

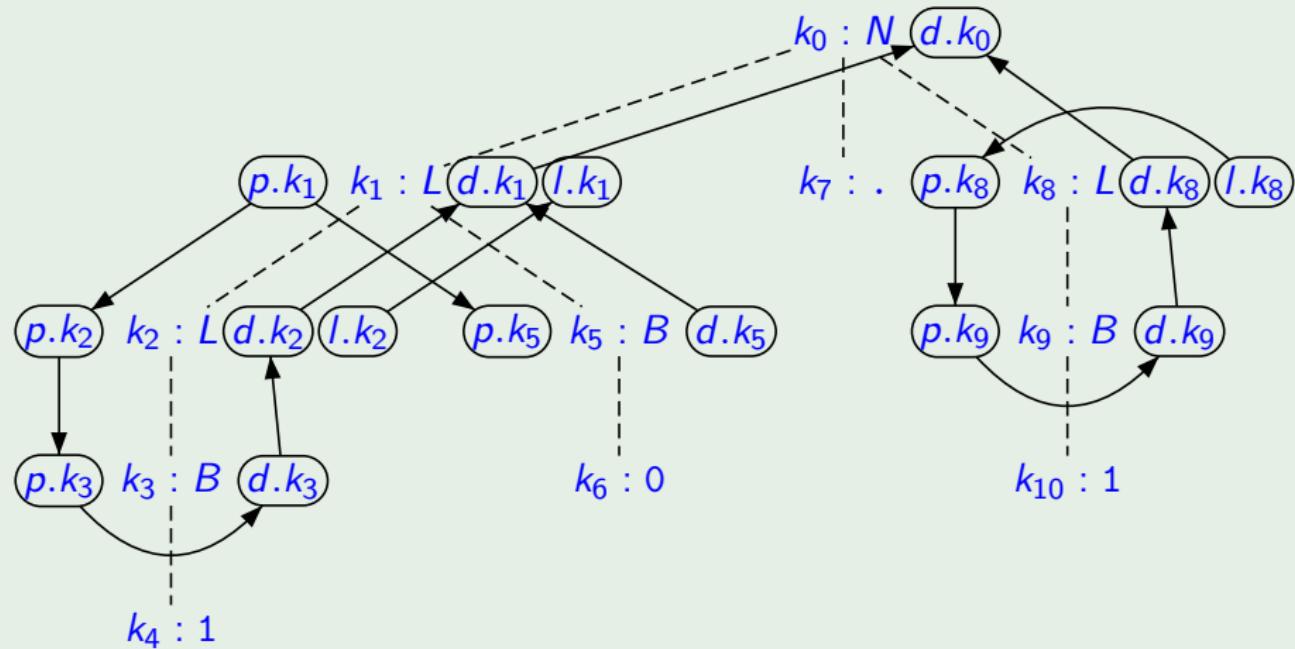
## Corollary 14.8

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 14.9 (cf. Example 13.2)

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



- 1 Repetition: Attribute Grammars
- 2 Circularity of Attribute Grammars
- 3 Attribute Dependency Graphs
- 4 Testing Attribute Grammars for Circularity

**Observation:** a cycle in the dependency graph  $D_t$  of a given syntax tree  $t$  is caused by the occurrence of a “cover” production

$\pi = A_0 \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$  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 14.10

on the board

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

## Definition 14.11 (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  $k$ ,  $\alpha \in \text{syn}(A)$ , and  $\beta \in \text{inh}(A)$  such that  $\beta.k \rightarrow_t^+ \alpha.k$ , then  $\alpha$  is dependent on  $\beta$  below  $A$  in  $t$  (notation:  $\beta \xrightarrow{A} \alpha$ ).
- For every syntax tree  $t$  with root label  $A \in N$ ,  
$$is(A, t) := \{(\beta, \alpha) \in \text{inh}(A) \times \text{syn}(A) \mid \beta \xrightarrow{A} \alpha \text{ in } t\}.$$
- 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 14.12

on the board