# Compiler Construction

## Lecture 14: Semantic Analysis II (Definition and Circularity of 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/cc09/

Winter semester 2009/10

# Outline

# 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 leafs to the root)

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

# Example: Knuth's Binary Numbers I

## Example (only synthesized attributes)

Binary numbers (with fraction):

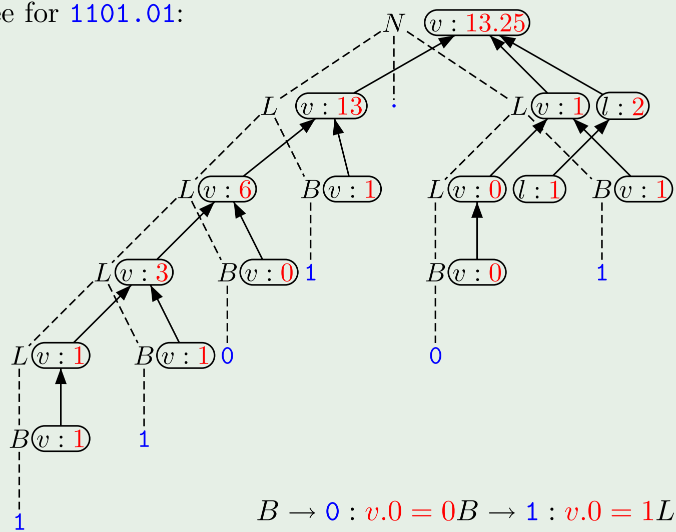
$$
\begin{array}{llrcllcl}
G_B: & \text{Numbers} & N & \to & L & v.0 & = & v.1 \\
 & & N & \to & L\,.\,L & v.0 & = & v.1 + v.3/2^{l.3} \\
 & \text{Lists} & L & \to & B & v.0 & = & v.1 \\
 & & & & & l.0 & = & 1 \\
 & & L & \to & LB & v.0 & = & 2 * v.1 + v.2 \\
 & & & & & l.0 & = & l.1 + 1 \\
 & \text{Bits} & B & \to & \texttt{0} & v.0 & = & 0 \\
 & \text{Bits} & B & \to & \texttt{1} & v.0 & = & 1
\end{array}
$$

Synthesized attributes of $N, L, B$: $v$ (value; domain: $V^v := \mathbb{Q}$)
of $L$: $l$ (length; domain: $V^l := \mathbb{N}$)

Semantic rules: equations with attribute variables
(index = position of symbol; 0 = left-hand side)

## Example (continued)

Syntax tree for `1101.01`:




$B \to 0 : v.0 = 0 B \to 1 : v.0 = 1 L \to B :$
$v.0 = v.1 L \to B : l.0 = 1 L \to LB : v.0 = 2 * v.1 + v.2 L \to LB : l.0 =$

# Outline

# Adding Inherited Attributes I

## Example 14.1 (synthesized + inherited attributes)

Binary numbers (with fraction):

$$
\begin{array}{llrcl}
G'_B: & \text{Numbers} & N \to L & & v.0 = v.1 \\
& & & & p.1 = 0 \\
& & N \to L\,.\,L & & v.0 = v.1 + v.3 \\
& & & & p.1 = 0 \\
& & & & p.3 = -l.3 \\
& \text{Lists} & L \to B & & v.0 = v.1 \\
& & & & l.0 = 1 \\
& & & & p.1 = p.0 \\
& & L \to LB & & v.0 = v.1 + v.2 \\
& & & & l.0 = l.1 + 1 \\
& & & & p.1 = p.0 + 1 \\
& & & & p.2 = p.0 \\
& \text{Bits} & B \to \mathtt{0} & & v.0 = 0 \\
& \text{Bits} & B \to \mathtt{1} & & v.0 = 2^{p.0}
\end{array}
$$

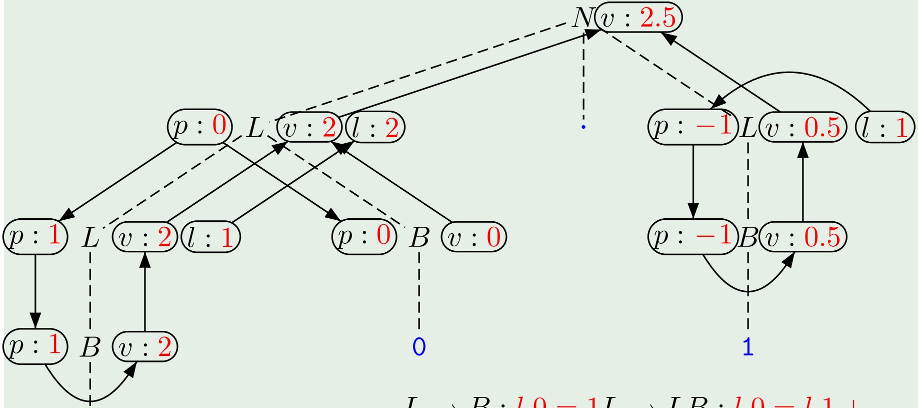
Synthesized attributes of $N, L, B$: $v$ (value; domain: $V^v := \mathbb{Q}$)

of $L$: $l$ (length; domain: $V^l := \mathbb{N}$)

Inherited attribute of $L, B$: $p$ (position; domain: $V^p := \mathbb{Z}$)

## Example 14.1 (continued)

Syntax tree for `10.1`:




$L \rightarrow B : l.0 = 1 L \rightarrow LB : l.0 = l.1 +$

$1 N \rightarrow L\,.\,L : p.1 = 0 N \rightarrow L\,.\,L : p.3 = -l.3 L \rightarrow LB : p.1 = p.0 + 1 L \rightarrow$

$LB : p.2 = p.0 L \rightarrow B : p.1 = p.0 B \rightarrow 0 : v.0 = 0 B \rightarrow 1 : v.0 = 2^{p.0} L \rightarrow$

# Outline

# Formal Definition of Attribute Grammars I

## Definition 14.2 (Attribute grammar)

Let $G = \langle N, \Sigma, P, S \rangle \in \mathit{CFG}_\Sigma$ with $X := N \uplus \Sigma$.

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

# Formal Definition of Attribute Grammars II

## Example 14.3 (cf. Example 14.1)

$\mathfrak{A}_B \in AG$ for binary numbers:

- Attributes: $Att = Syn \uplus Inh$ with $Syn = \{v, l\}$ and $Inh = \{p\}$
- Value sets: $V^v = \mathbb{Q}$, $V^l = \mathbb{N}$, $V^p = \mathbb{Z}$
- Attribute assignment:

| $Y \in X$ | $N$ | $L$ | $B$ | 0 | 1 |
|---|---|---|---|---|---|
| syn($Y$) | $\{v\}$ | $\{v, l\}$ | $\{v\}$ | $\emptyset$ | $\emptyset$ |
| inh($Y$) | $\emptyset$ | $\{p\}$ | $\{p\}$ | $\emptyset$ | $\emptyset$ |

- Attribute variables:

| $\pi \in P$ | $N \to L$ | $N \to L.L$ | $L \to B$ |
|---|---|---|---|
| $In_\pi$ | $\{v.0, p.1\}$ | $\{v.0, p.1, p.3\}$ | $\{v.0, l.0, p.1\}$ |
| $Out_\pi$ | $\{v.1, l.1\}$ | $\{v.1, l.1, v.3, l.3\}$ | $\{v.1, p.0\}$ |

| $\pi \in P$ | $L \to LB$ | $B \to 0$ | $B \to 1$ |
|---|---|---|---|
| $In_\pi$ | $\{v.0, l.0, p.1, p.2\}$ | $\{v.0\}$ | $\{v.0\}$ |
| $Out_\pi$ | $\{v.1, v.2, l.1, p.0\}$ | $\{p.0\}$ | $\{p.0\}$ |

- Semantic rules: see Example 14.1
  (e.g., $E_{N \to L} = \{v.0 = v.1, p.1 = 0\}$)

## Definition 14.4 (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 $\pi = Y_0 \to Y_1 \ldots Y_r \in P$ is applied, and let $k_1, \ldots, k_r \in K$ 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 $i \in \{0, \ldots, r\}$ 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

**Corollary 14.5**

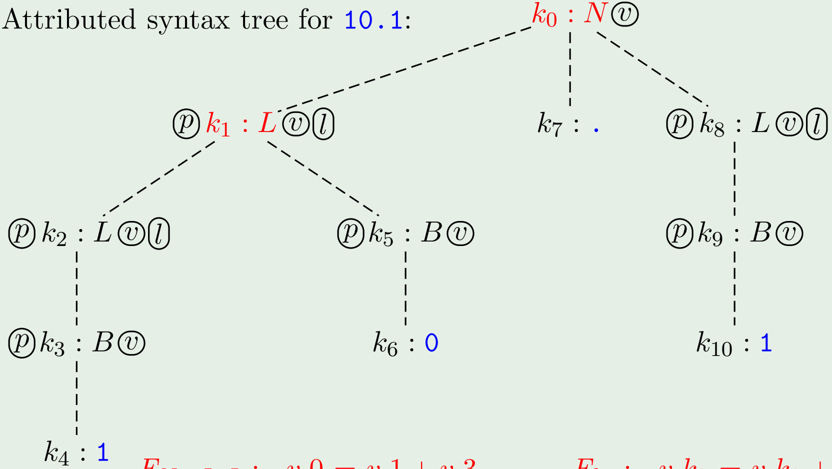
*For each $\alpha.k \in Var_t$ except the inherited attribute variables at the root and the synthesized attribute variables at the leafs of $t$, $E_t$ contains exactly one equation with left-hand side $\alpha.k$.*

**Assumptions:**

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

## Example 14.6 (cf. Example 14.1)

Attributed syntax tree for `10.1`:




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

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

# Outline

# Solvability of Attribute Equation System I

**Definition 14.7 (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 $\alpha.k = f(\alpha.k_1, \ldots, \alpha.k_n) \in E_t$,

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

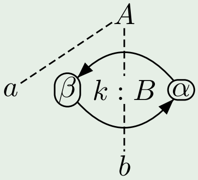
In general, the attribute equation system $E_t$ of a given syntax tree $t$ can have

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

## Example 14.8

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

$\Longrightarrow$ cyclic dependency:




$E_t: \quad \beta.k = f(\alpha.k)$
$\qquad \alpha.k = g(\beta.k)$

$\Longrightarrow$ 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 14.9 (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 (see Corollary 14.11).

# Outline

# Attribute Dependency Graphs I

**Goal:** graphic representation of attribute dependencies

**Definition 14.10 (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

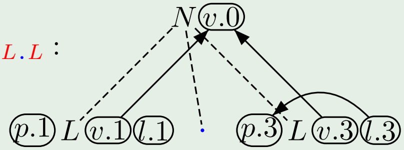
$$x \to_\pi y \quad \text{iff} \quad y = f(\ldots, x, \ldots) \in E_\pi.$$

**Corollary 14.11**

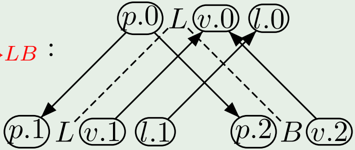
*The dependency graph of a production is acyclic (since $\to_\pi \subseteq Out_\pi \times In_\pi$).*

## Example 14.12 (cf. Example 14.1)

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




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



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

## Corollary 14.14

*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.*

## Example 14.15 (cf. Example 14.1)

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