

Compiler Construction

Lecture 15: Semantic Analysis III (Attribute Evaluation)

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/cc08/`

Summer semester 2008

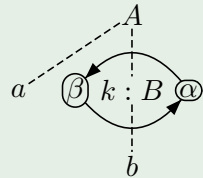
- 1 Repetition: Circularity of Attribute Grammars
- 2 Correctness and Complexity of the Circularity Test
- 3 Strongly Noncircular Attribute Grammars
- 4 Attribute Evaluation
- 5 Attribute Evaluation by Topological Sorting
- 6 Attribute Evaluation by Recursive Functions

Solvability of the Attribute Equation System

Example

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

$\Rightarrow$ cyclic dependency:



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

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

Goal: **unique solvability** of equation system
 $\implies$ 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.

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

on the board

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

Definition (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: $\beta \xrightarrow{A} \alpha$).
- For every syntax tree t with root label $A \in N$,
$$\text{is}(A, t) := \{(\beta, \alpha) \in \text{inh}(A) \times \text{syn}(A) \mid \beta \xrightarrow{A} \alpha \text{ in } t\}.$$
- For every $A \in N$,
$$\text{IS}(A) := \{\text{is}(A, t) \mid t \text{ syntax tree with root label } A\} \\ \subseteq 2^{\text{Inh} \times \text{Syn}}.$$

Remark: it is important that $\text{IS}(A)$ is a **system** of attribute dependence sets, not a **union** (later: **strong noncircularity**).

Example

on the board

The Circularity Test

Algorithm (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 $\pi = A \rightarrow w \in P$, then $is[\pi] \in IS(A)$*
 - ② *if $\pi = A \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$ and $is_i \in IS(A_i)$ for every $i \in [r]$, then $is[\pi; is_1, \dots, is_r] \in IS(A)$*
- ② *test whether $\mathfrak{A}$ is circular by checking if there exist $\pi = A \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$ and $is_i \in IS(A_i)$ for every $i \in [r]$ such that the following relation is cyclic:*
$$\rightarrow_\pi \cup \bigcup_{i=1}^r \{(\beta.p_i, \alpha.p_i) \mid (\beta, \alpha) \in is_i\}$$
(where $p_i := \sum_{j=1}^i |w_{j-1}| + i$)

Output: “yes” or “no”

- 1 Repetition: Circularity of Attribute Grammars
- 2 Correctness and Complexity of the Circularity Test
- 3 Strongly Noncircular Attribute Grammars
- 4 Attribute Evaluation
- 5 Attribute Evaluation by Topological Sorting
- 6 Attribute Evaluation by Recursive Functions

Theorem 15.1 (Correctness of the circularity test)

An attribute grammar is circular iff Algorithm 14.15 yields the answer “yes”.

Correctness and Complexity of Circularity Test

Theorem 15.1 (Correctness of the circularity test)

An attribute grammar is circular iff Algorithm 14.15 yields the answer “yes”.

Proof.

by induction on the syntax tree t with cyclic D_t



Correctness and Complexity of Circularity Test

Theorem 15.1 (Correctness of the circularity test)

An attribute grammar is circular iff Algorithm 14.15 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).*

Correctness and Complexity of Circularity Test

Theorem 15.1 (Correctness of the circularity test)

An attribute grammar is circular iff Algorithm 14.15 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) □

- 1 Repetition: Circularity of Attribute Grammars
- 2 Correctness and Complexity of the Circularity Test
- 3 Strongly Noncircular Attribute Grammars**
- 4 Attribute Evaluation
- 5 Attribute Evaluation by Topological Sorting
- 6 Attribute Evaluation by Recursive Functions

Simplifying the Circularity Test

Idea: to simplify the circularity test, do not distinguish between attribute dependences which are caused by different syntax trees

Simplifying the Circularity Test

Idea: to simplify the circularity test, do not distinguish between attribute dependences which are caused by different syntax trees

Definition 15.3 (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 , $\alpha \in \text{syn}(A)$, 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: $\beta \xrightarrow{A} \alpha$).

Simplifying the Circularity Test

Idea: to simplify the circularity test, do not distinguish between attribute dependences which are caused by different syntax trees

Definition 15.3 (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 , $\alpha \in \text{syn}(A)$, 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: $\beta \xrightarrow{A} \alpha$).
- For every $A \in N$,

$$\begin{aligned} IS'(A) &:= \{(\beta, \alpha) \mid \beta \xrightarrow{A} \alpha \text{ in some syntax tree with root label } A\} \\ &\subseteq \text{Inh} \times \text{Syn} \end{aligned}$$

The Strong Circularity Test

Algorithm 15.4 (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 $\pi = A \rightarrow w \in P$, then $is[\pi] \subseteq IS'(A)$
- ② if $\pi = A \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$, then $is[\pi; IS'(A_1), \dots, IS'(A_r)] \subseteq IS'(A)$

② test whether there exists

$\pi = A \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$ such that the following relation is cyclic:

$$\rightarrow_\pi \cup \bigcup_{i=1}^r \{(\beta.p_i, \alpha.p_i) \mid (\beta, \alpha) \in IS'(A_i)\}$$

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

Output: “yes” or “no”

The Strong Circularity Test

Algorithm 15.4 (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 $\pi = A \rightarrow w \in P$, then $is[\pi] \subseteq IS'(A)$*
- ② *if $\pi = A \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$, then $is[\pi; IS'(A_1), \dots, IS'(A_r)] \subseteq IS'(A)$*

② *test whether there exists*

$\pi = A \rightarrow w_0 A_1 w_1 \dots A_r w_r \in P$ such that the following relation is cyclic:

$$\rightarrow_{\pi} \cup \bigcup_{i=1}^r \{(\beta.p_i, \alpha.p_i) \mid (\beta, \alpha) \in IS'(A_i)\}$$

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

Output: “yes” or “no”

Example 15.5

on the board

Strongly Noncircular Attribute Grammars I

Definition 15.6 (Strong noncircularity)

An attribute grammar is called **strongly noncircular** if Algorithm 15.4 yields the answer “no”.

Strongly Noncircular Attribute Grammars I

Definition 15.6 (Strong noncircularity)

An attribute grammar is called **strongly noncircular** if Algorithm 15.4 yields the answer “no”.

Lemma 15.7

*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



Lemma 15.8

- ① *Every strongly noncircular attribute grammar is noncircular.*
- ② *There are noncircular attribute grammars which are not strongly noncircular.*

Lemma 15.8

- 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 15.5 is noncircular but not strongly noncircular (on the board).



- 1 Repetition: Circularity of Attribute Grammars
- 2 Correctness and Complexity of the Circularity Test
- 3 Strongly Noncircular Attribute Grammars
- 4 Attribute Evaluation
- 5 Attribute Evaluation by Topological Sorting
- 6 Attribute Evaluation by Recursive Functions

Attribute Evaluation Methods

- Given:
- (strongly) noncircular attribute grammar
 $\mathfrak{A} = \langle G, E, V \rangle \in AG$
 - syntax tree t of G
 - valuation $v : Syn_{\Sigma} \rightarrow V$ where
 $Syn_{\Sigma} := \{\alpha.k \mid k \text{ labelled by } a \in \Sigma, \alpha \in \text{syn}(a)\} \subseteq Var_t$

Attribute Evaluation Methods

- Given:
- (strongly) noncircular attribute grammar
 $\mathfrak{A} = \langle G, E, V \rangle \in AG$
 - syntax tree t of G
 - valuation $v : Syn_{\Sigma} \rightarrow V$ 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** $v : Var_t \rightarrow V$

Attribute Evaluation Methods

- Given:
- (strongly) noncircular attribute grammar
 $\mathcal{A} = \langle G, E, V \rangle \in AG$
 - syntax tree t of G
 - valuation $v : Syn_{\Sigma} \rightarrow V$ 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** $v : Var_t \rightarrow V$

- Methods:
- ① **Topological sorting** of D_t :
 - ① start with attribute variables which depend at most on synthesized attributes of terminals
 - ② proceed by successive substitution

Attribute Evaluation Methods

- Given:
- (strongly) noncircular attribute grammar
 $\mathcal{A} = \langle G, E, V \rangle \in AG$
 - syntax tree t of G
 - valuation $v : Syn_{\Sigma} \rightarrow V$ 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** $v : Var_t \rightarrow V$

- Methods:
- ① **Topological sorting** of D_t :
 - ① start with attribute variables which depend at most on synthesized attributes of terminals
 - 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

Attribute Evaluation Methods

- Given:
- (strongly) noncircular attribute grammar
 $\mathcal{A} = \langle G, E, V \rangle \in AG$
 - syntax tree t of G
 - valuation $v : Syn_{\Sigma} \rightarrow V$ 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** $v : Var_t \rightarrow V$

- Methods:
- ➊ **Topological sorting** of D_t :
 - ➊ start with attribute variables which depend at most on synthesized attributes of terminals
 - ➋ proceed by successive substitution
 - ➋ **Recursive functions** (for strongly noncircular AGs):
 - ➊ 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
 - ➋ for every $\alpha \in \text{syn}(S)$, evaluate $g_{S,\alpha}(k_0)$ where k_0 denotes the root of t
 - ➌ Special cases: **S-attributed grammars** (yacc), **L-attributed grammars**

- 1 Repetition: Circularity of Attribute Grammars
- 2 Correctness and Complexity of the Circularity Test
- 3 Strongly Noncircular Attribute Grammars
- 4 Attribute Evaluation
- 5 Attribute Evaluation by Topological Sorting
- 6 Attribute Evaluation by Recursive Functions

Attribute Evaluation by Topological Sorting

Algorithm 15.9 (Evaluation by topological sorting)

Input: *noncircular* $\mathfrak{A} = \langle G, E, V \rangle \in AG$, *syntax tree* t of G ,
valuation $v : \text{Syn}_\Sigma \rightarrow V$

Procedure:

- ① *let* $\text{Var} := \text{Var}_t \setminus \text{Syn}_\Sigma$ (** attributes to be evaluated **)
- ② *while* $\text{Var} \neq \emptyset$ *do*
 - ① *let* $x \in \text{Var}$ *such that* $\{y \in \text{Var} \mid y \rightarrow_t x\} = \emptyset$
 - ② *let* $x = f(x_1, \dots, x_n) \in E_t$
 - ③ *let* $v(x) := f(v(x_1), \dots, v(x_n))$
 - ④ *let* $\text{Var} := \text{Var} \setminus \{x\}$

Output: *solution* $v : \text{Var}_t \rightarrow V$

Attribute Evaluation by Topological Sorting

Algorithm 15.9 (Evaluation by topological sorting)

Input: *noncircular* $\mathfrak{A} = \langle G, E, V \rangle \in AG$, *syntax tree* t of G ,
valuation $v : \text{Syn}_\Sigma \rightarrow V$

Procedure:

- ① *let* $\text{Var} := \text{Var}_t \setminus \text{Syn}_\Sigma$ (** attributes to be evaluated **)
- ② *while* $\text{Var} \neq \emptyset$ *do*
 - ① *let* $x \in \text{Var}$ *such that* $\{y \in \text{Var} \mid y \rightarrow_t x\} = \emptyset$
 - ② *let* $x = f(x_1, \dots, x_n) \in E_t$
 - ③ *let* $v(x) := f(v(x_1), \dots, v(x_n))$
 - ④ *let* $\text{Var} := \text{Var} \setminus \{x\}$

Output: *solution* $v : \text{Var}_t \rightarrow V$

Remark: noncircularity guarantees that in step 2.1 at least one such x is available

Attribute Evaluation by Topological Sorting

Algorithm 15.9 (Evaluation by topological sorting)

Input: *noncircular* $\mathfrak{A} = \langle G, E, V \rangle \in AG$, *syntax tree* t of G ,
valuation $v : \text{Syn}_\Sigma \rightarrow V$

Procedure:

- ❶ *let* $\text{Var} := \text{Var}_t \setminus \text{Syn}_\Sigma$ (** attributes to be evaluated **)
- ❷ *while* $\text{Var} \neq \emptyset$ *do*
 - ❶ *let* $x \in \text{Var}$ *such that* $\{y \in \text{Var} \mid y \rightarrow_t x\} = \emptyset$
 - ❷ *let* $x = f(x_1, \dots, x_n) \in E_t$
 - ❸ *let* $v(x) := f(v(x_1), \dots, v(x_n))$
 - ❹ *let* $\text{Var} := \text{Var} \setminus \{x\}$

Output: *solution* $v : \text{Var}_t \rightarrow V$

Remark: noncircularity guarantees that in step 2.1 at least one such x is available

Example 15.10

see Examples 13.1 and 13.2 (Knuth's binary numbers)

- 1 Repetition: Circularity of Attribute Grammars
- 2 Correctness and Complexity of the Circularity Test
- 3 Strongly Noncircular Attribute Grammars
- 4 Attribute Evaluation
- 5 Attribute Evaluation by Topological Sorting
- 6 Attribute Evaluation by Recursive Functions

Attribute Evaluation by Recursive Functions

Restriction: only for **strongly noncircular** attribute 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 $\alpha \in \text{syn}(S)$

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 15.4)
- $\text{inh}(A, \alpha) := \{\beta \in \text{inh}(A) \mid (\beta, \alpha) \in IS'(A)\}$
- $A \rightarrow \gamma_1 \mid \dots \mid \gamma_m$ all A -productions in P

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 15.4)
- $\text{inh}(A, \alpha) := \{\beta \in \text{inh}(A) \mid (\beta, \alpha) \in IS'(A)\}$
- $A \rightarrow \gamma_1 \mid \dots \mid \gamma_m$ all A -productions in P

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

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

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 15.4)
- $\text{inh}(A, \alpha) := \{\beta \in \text{inh}(A) \mid (\beta, \alpha) \in IS'(A)\}$
- $A \rightarrow \gamma_1 \mid \dots \mid \gamma_m$ all A -productions in P

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

$g_{A,\alpha}(k_0, \text{inh}(A, \alpha)) :=$ **case** production applied at k_0 **of**

⋮
 $A \rightarrow \gamma_j : \text{eval}(\alpha.0)$
⋮
end

with

$$\text{eval}(\alpha.i) := \begin{cases} \alpha & \text{if } \alpha \in \text{inh}(A), i = 0 \\ f(\text{eval}(\alpha_1.i_1), \dots, \text{eval}(\alpha_n.i_n)) & \text{if } \alpha.i \in \text{In}_{A \rightarrow \gamma_j}, \alpha.i = f(\alpha_1.i_1, \dots, \alpha_n.i_n) \in E_{A \rightarrow \gamma_j} \\ g_{Y_i, \alpha}(k_i, \text{eval}(\beta_1.i), \dots, \text{eval}(\beta_l.i)) & \text{if } \alpha \in \text{Syn}, i > 0, Y_i \in N, \\ & \text{inh}(Y_i, \alpha) = \{\beta_1, \dots, \beta_l\} \\ v(\alpha.i) & \text{if } \alpha \in \text{Syn}, i > 0, Y_i \in \Sigma \end{cases}$$

where $\gamma_j = Y_1 \dots Y_r$, and where k_i denotes the i th successor of k_0

Definition of Evaluation Functions II

Example 15.11 (cf. Example 13.2)

G'_B :

$S \rightarrow L \quad v.0 = v.1$

$p.1 = 0$

$S \rightarrow L.L \quad v.0 = v.1 + v.3$

$p.1 = 0$

$p.3 = -l.3$

$L \rightarrow B \quad v.0 = v.1$

$l.0 = 1$

$p.1 = p.0$

$L \rightarrow LB \quad v.0 = v.1 + v.2$

$l.0 = l.1 + 1$

$p.1 = p.0 + 1$

$p.2 = p.0$

$B \rightarrow 0 \quad v.0 = 0$

$B \rightarrow 1 \quad v.0 = 2^{p.0}$

Definition of Evaluation Functions II

Example 15.11 (cf. Example 13.2)

G'_B :

$S \rightarrow L \quad v.0 = v.1$

$p.1 = 0$

$S \rightarrow L.L \quad v.0 = v.1 + v.3$

$p.1 = 0$

$p.3 = -l.3$

$L \rightarrow B \quad v.0 = v.1$

$l.0 = 1$

$p.1 = p.0$

$L \rightarrow LB \quad v.0 = v.1 + v.2$

$l.0 = l.1 + 1$

$p.1 = p.0 + 1$

$p.2 = p.0$

$B \rightarrow 0 \quad v.0 = 0$

$B \rightarrow 1 \quad v.0 = 2^{p.0}$

$A \in N$	S	L	B
$IS'(A)$	$\emptyset$	$\{(p, v)\}$	$\{(p, v)\}$

Definition of Evaluation Functions II

Example 15.11 (cf. Example 13.2)

$G'_B:$		$g_{S,v}(k_0) = \text{case production}(k_0) \text{ of}$
$S \rightarrow L$	$v.0 = v.1$	$S \rightarrow L : g_{L,v}(k_1, 0)$
	$p.1 = 0$	$S \rightarrow L.L : g_{L,v}(k_1, 0) +$
$S \rightarrow L.L$	$v.0 = v.1 + v.3$	$g_{L,v}(k_3, -g_{L,l}(k_3))$
	$p.1 = 0$	end
	$p.3 = -l.3$	$g_{L,v}(k_0, p) = \text{case production}(k_0) \text{ of}$
$L \rightarrow B$	$v.0 = v.1$	$L \rightarrow B : g_{B,v}(k_1, p)$
	$l.0 = 1$	$L \rightarrow LB : g_{L,v}(k_1, p + 1)$
	$p.1 = p.0$	$+ g_{B,v}(k_2, p)$
$L \rightarrow LB$	$v.0 = v.1 + v.2$	end
	$l.0 = l.1 + 1$	$g_{L,l}(k_0) = \text{case production}(k_0) \text{ of}$
	$p.1 = p.0 + 1$	$L \rightarrow B : 1$
	$p.2 = p.0$	$L \rightarrow LB : g_{L,l}(k_1) + 1$
$B \rightarrow 0$	$v.0 = 0$	end
$B \rightarrow 1$	$v.0 = 2^{p.0}$	$g_{B,v}(k_0, p) = \text{case production}(k_0) \text{ of}$
		$B \rightarrow 0 : 0$
		$B \rightarrow 1 : 2^p$
		end

$A \in N$	S	L	B
$IS'(A)$	$\emptyset$	$\{(p, v)\}$	$\{(p, v)\}$

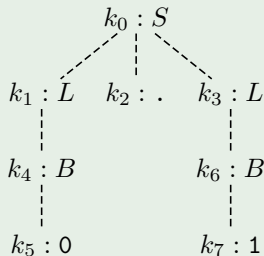
Example 15.11 (continued)

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

Example 15.11 (continued)

```
 $g_{S,v}(k_0) = \text{case production}(k_0) \text{ of}$   
     $S \rightarrow L : g_{L,v}(k_1, 0)$   
     $S \rightarrow L.L : g_{L,v}(k_1, 0) +$   
         $g_{L,v}(k_3, -g_{L,l}(k_3))$   
     $\text{end}$   
 $g_{L,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : g_{B,v}(k_1, p)$   
     $L \rightarrow LB : g_{L,v}(k_1, p + 1)$   
         $+ g_{B,v}(k_2, p)$   
     $\text{end}$   
 $g_{L,l}(k_0) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : 1$   
     $L \rightarrow LB : g_{L,l}(k_1) + 1$   
     $\text{end}$   
 $g_{B,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $B \rightarrow 0 : 0$   
     $B \rightarrow 1 : 2^p$   
     $\text{end}$ 
```

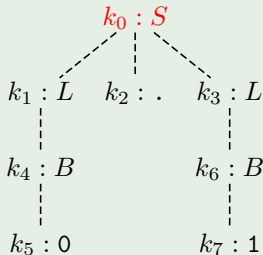
Syntax tree t :



Example 15.11 (continued)

```
 $g_{S,v}(k_0) = \text{case production}(k_0) \text{ of}$   
     $S \rightarrow L : g_{L,v}(k_1, 0)$   
     $S \rightarrow L.L : g_{L,v}(k_1, 0) +$   
         $g_{L,v}(k_3, -g_{L,l}(k_3))$   
     $\text{end}$   
 $g_{L,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : g_{B,v}(k_1, p)$   
     $L \rightarrow LB : g_{L,v}(k_1, p + 1)$   
         $+ g_{B,v}(k_2, p)$   
     $\text{end}$   
 $g_{L,l}(k_0) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : 1$   
     $L \rightarrow LB : g_{L,l}(k_1) + 1$   
     $\text{end}$   
 $g_{B,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $B \rightarrow 0 : 0$   
     $B \rightarrow 1 : 2^p$   
     $\text{end}$ 
```

Syntax tree t :

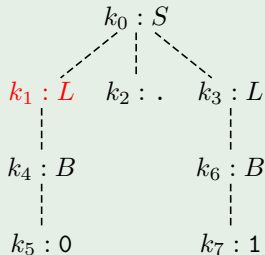


$g_{S,v}(k_0)$

Example 15.11 (continued)

```
 $g_{S,v}(k_0) = \text{case production}(k_0) \text{ of}$   
     $S \rightarrow L : g_{L,v}(k_1, 0)$   
     $S \rightarrow L.L : g_{L,v}(k_1, 0) +$   
         $g_{L,v}(k_3, -g_{L,l}(k_3))$   
     $\text{end}$   
 $g_{L,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : g_{B,v}(k_1, p)$   
     $L \rightarrow LB : g_{L,v}(k_1, p + 1)$   
         $+ g_{B,v}(k_2, p)$   
     $\text{end}$   
 $g_{L,l}(k_0) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : 1$   
     $L \rightarrow LB : g_{L,l}(k_1) + 1$   
     $\text{end}$   
 $g_{B,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $B \rightarrow 0 : 0$   
     $B \rightarrow 1 : 2^p$   
     $\text{end}$ 
```

Syntax tree t :

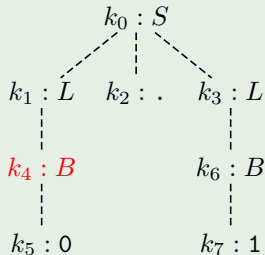


$$g_{S,v}(k_0) = g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3))$$

Example 15.11 (continued)

```
 $g_{S,v}(k_0) = \text{case production}(k_0) \text{ of}$   
     $S \rightarrow L : g_{L,v}(k_1, 0)$   
     $S \rightarrow L.L : g_{L,v}(k_1, 0) +$   
         $g_{L,v}(k_3, -g_{L,l}(k_3))$   
     $\text{end}$   
 $g_{L,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : g_{B,v}(k_1, p)$   
     $L \rightarrow LB : g_{L,v}(k_1, p + 1)$   
         $+ g_{B,v}(k_2, p)$   
     $\text{end}$   
 $g_{L,l}(k_0) = \text{case production}(k_0) \text{ of}$   
     $L \rightarrow B : 1$   
     $L \rightarrow LB : g_{L,l}(k_1) + 1$   
     $\text{end}$   
 $g_{B,v}(k_0, p) = \text{case production}(k_0) \text{ of}$   
     $B \rightarrow 0 : 0$   
     $B \rightarrow 1 : 2^p$   
     $\text{end}$ 
```

Syntax tree t :

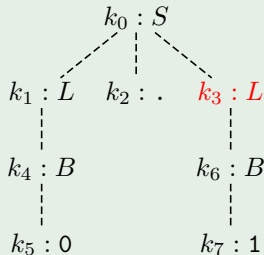


$$\begin{aligned} g_{S,v}(k_0) &= g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\ &= g_{B,v}(k_4, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \end{aligned}$$

Example 15.11 (continued)

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

Syntax tree t :

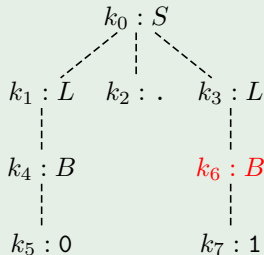


$$\begin{aligned}
 g_{S,v}(k_0) &= g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= g_{B,v}(k_4, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{L,v}(k_3, -g_{L,l}(k_3))
 \end{aligned}$$

Example 15.11 (continued)

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

Syntax tree t :

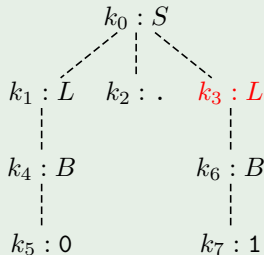


$$\begin{aligned}
 g_{S,v}(k_0) &= g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= g_{B,v}(k_4, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{B,v}(k_6, -g_{L,l}(k_3))
 \end{aligned}$$

Example 15.11 (continued)

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

Syntax tree t :

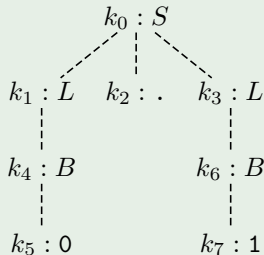


$$\begin{aligned}
 g_{S,v}(k_0) &= g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= g_{B,v}(k_4, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{B,v}(k_6, -g_{L,l}(k_3)) \\
 &= 0 + 2^{-g_{L,l}(k_3)}
 \end{aligned}$$

Example 15.11 (continued)

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

Syntax tree t :

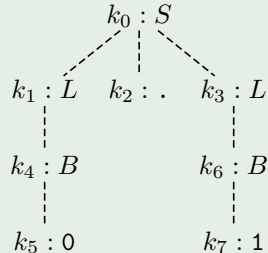


$$\begin{aligned}
 g_{S,v}(k_0) &= g_{L,v}(k_1, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= g_{B,v}(k_4, 0) + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{L,v}(k_3, -g_{L,l}(k_3)) \\
 &= 0 + g_{B,v}(k_6, -g_{L,l}(k_3)) \\
 &= 0 + 2^{-g_{L,l}(k_3)} \\
 &= 0 + 2^{-1}
 \end{aligned}$$

Example 15.11 (continued)

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

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 15.12 (cf. Example 15.5)

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$$S \rightarrow A \quad \alpha.0 = \alpha_2.1$$

$$\beta_1.1 = \alpha_1.1$$

$$\beta_2.1 = \alpha_2.1$$

$$A \rightarrow a \quad \alpha_1.0 = \beta_2.0$$

$$\alpha_2.0 = 2$$

$$A \rightarrow b \quad \alpha_1.0 = 1$$

$$\alpha_2.0 = \beta_1.0$$

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$S \rightarrow A$ $\alpha.0 = \alpha_2.1$

$\beta_1.1 = \alpha_1.1$

$\beta_2.1 = \alpha_2.1$

$A \rightarrow \mathbf{a}$ $\alpha_1.0 = \beta_2.0$

$\alpha_2.0 = 2$

$A \rightarrow \mathbf{b}$ $\alpha_1.0 = 1$

$\alpha_2.0 = \beta_1.0$

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

$g_{S,\alpha}(k_0)$

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$$S \rightarrow A \quad \alpha.0 = \alpha_2.1$$

$$\beta_1.1 = \alpha_1.1$$

$$\beta_2.1 = \alpha_2.1$$

$$A \rightarrow \mathbf{a} \quad \alpha_1.0 = \beta_2.0$$

$$\alpha_2.0 = 2$$

$$A \rightarrow \mathbf{b} \quad \alpha_1.0 = 1$$

$$\alpha_2.0 = \beta_1.0$$

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

$$g_{S,\alpha}(k_0)$$

$$= \text{eval}(\alpha.0)$$

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$$S \rightarrow A \quad \alpha.0 = \alpha_2.1$$

$$\beta_1.1 = \alpha_1.1$$

$$\beta_2.1 = \alpha_2.1$$

$$A \rightarrow \mathbf{a} \quad \alpha_1.0 = \beta_2.0$$

$$\alpha_2.0 = 2$$

$$A \rightarrow \mathbf{b} \quad \alpha_1.0 = 1$$

$$\alpha_2.0 = \beta_1.0$$

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

$$g_{S,\alpha}(k_0)$$

$$= \text{eval}(\alpha.0)$$

$$= \text{eval}(\alpha_2.1)$$

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

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

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

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

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$$\begin{array}{ll} S \rightarrow A & \alpha.0 = \alpha_2.1 \\ & \beta_1.1 = \alpha_1.1 \\ & \beta_2.1 = \alpha_2.1 \\ A \rightarrow \mathbf{a} & \alpha_1.0 = \beta_2.0 \\ & \alpha_2.0 = 2 \\ A \rightarrow \mathbf{b} & \alpha_1.0 = 1 \\ & \alpha_2.0 = \beta_1.0 \end{array}$$

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

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

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$S \rightarrow A \quad \alpha.0 = \alpha_2.1$

$\beta_1.1 = \alpha_1.1$

$\beta_2.1 = \alpha_2.1$

$A \rightarrow a \quad \alpha_1.0 = \beta_2.0$

$\alpha_2.0 = 2$

$A \rightarrow b \quad \alpha_1.0 = 1$

$\alpha_2.0 = \beta_1.0$

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

$g_{S,\alpha}(k_0)$

$= \text{eval}(\alpha.0)$

$= \text{eval}(\alpha_2.1)$

$= g_{A,\alpha_2}(k_1, \text{eval}(\beta_1.1))$

$= g_{A,\alpha_2}(k_1, \text{eval}(\alpha_1.1))$

$= g_{A,\alpha_2}(k_1, g_{A,\alpha_1}(k_1, \text{eval}(\beta_2.1)))$

In Example 15.5:

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

Why Strong Noncircularity?

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

Example 15.12 (cf. Example 15.5)

$S \rightarrow A \quad \alpha.0 = \alpha_2.1$

$\beta_1.1 = \alpha_1.1$

$\beta_2.1 = \alpha_2.1$

$A \rightarrow a \quad \alpha_1.0 = \beta_2.0$

$\alpha_2.0 = 2$

$A \rightarrow b \quad \alpha_1.0 = 1$

$\alpha_2.0 = \beta_1.0$

In Example 15.5:

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

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

$g_{S,\alpha}(k_0)$

$= \text{eval}(\alpha.0)$

$= \text{eval}(\alpha_2.1)$

$= g_{A,\alpha_2}(k_1, \text{eval}(\beta_1.1))$

$= g_{A,\alpha_2}(k_1, \text{eval}(\alpha_1.1))$

$= g_{A,\alpha_2}(k_1, g_{A,\alpha_1}(k_1, \text{eval}(\beta_2.1)))$

$= g_{A,\alpha_2}(k_1, g_{A,\alpha_1}(k_1, \text{eval}(\alpha_2.1)))$

$\Rightarrow$ does not terminate!