Compiler Construction

Lecture 13: Semantic Analysis II (Circularity Check)

Winter Semester 2018/19

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ws-1819/cc/

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Outline of Lecture 13

Recap: Attribute Grammars

Checking Attribute Grammars for Circularity

The Circularity Check

Correctness and Complexity of the Circularity Check

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 (synthesised or inherited) attributes, and let $V = \bigcup_{\alpha \in Att} V^\alpha$ be a collection 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 $\pi = Y_0 \to Y_1 \dots Y_r \in P$ determines the set
$$Var_\pi := \{\alpha.i \mid \alpha \in \mathrm{att}(Y_i), i \in \{0, \dots, r\}\}$$
of attribute variables of π with the subsets of internal and external variables:
$$Int_\pi := \{\alpha.i \mid (i = 0, \alpha \in \mathrm{syn}(Y_i)) \text{ or } (i \in [r], \alpha \in \mathrm{inh}(Y_i))\} \quad Ext_\pi := Var_\pi \setminus Int_\pi$$
- A semantic rule of π is an equation of the form
$$\alpha_0.i_0 = f(\alpha_1.i_1, \dots, \alpha_n.i_n)$$
where $n \in \mathbb{N}$, $\alpha_0.i_0 \in Int_\pi$, $\alpha_j.i_j \in Ext_\pi$, and $f : V^{\alpha_1} \times \dots \times V^{\alpha_n} \to V^{\alpha_0}$.
- For each $\pi \in P$, let E_π be a set with exactly one semantic rule for every internal 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

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 $\pi = Y_0 \to Y_1 \ldots Y_r \in P$ 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_π by substituting every attribute index $i \in \{0, \ldots, r\}$ by k_i.
- The attribute equation system of t is given by
$$E_t := \bigcup \{E_k \mid k \text{ inner node of } t\}.$$

Solvability of Attribute Equation System

Definition (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 $\alpha_0.k_0 \in Var_t$ and $\alpha_0.k_0 = f(\alpha_1.k_1, \ldots, \alpha_n.k_n) \in E_t$,

$$v(\alpha_0.k_0) = f(v(\alpha_1.k_1), \ldots, v(\alpha_n.k_n)).$$

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

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

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 Int_π and Ext_π, cyclic dependencies cannot occur at production level.

Attribute Dependency Graphs I

Goal: graphical representation of attribute dependencies

Definition (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

The dependency graph of a production is acyclic (since $\to_\pi \subseteq Ext_\pi \times Int_\pi$ and $Ext_\pi \cap Int_\pi = \emptyset$).

Attribute Dependency Graphs II

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 $x \to_t^+ x$.

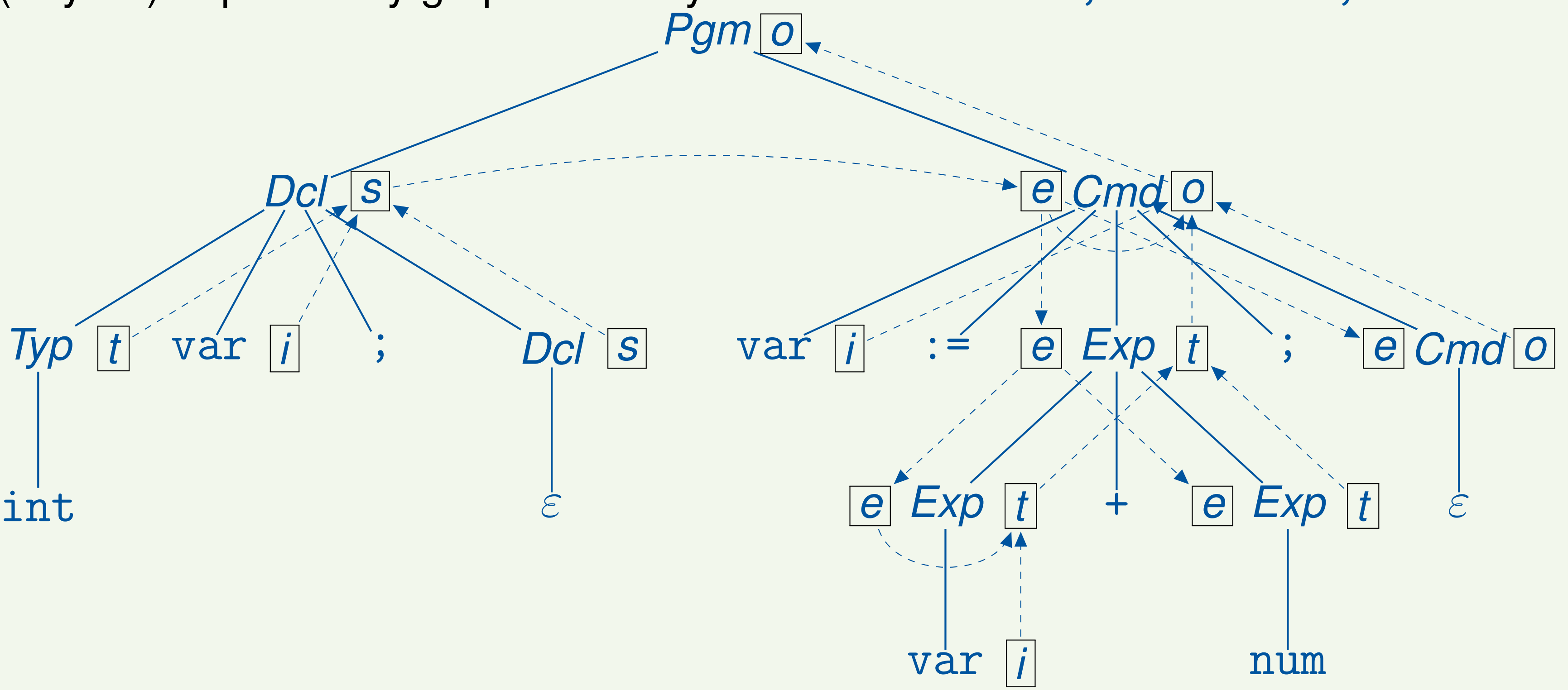
Corollary

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 III

Example (cf. Example 12.1)

(Acyclic) dependency graph of the syntax tree for `int x; x := x+1;`:



Outline of Lecture 13

Recap: Attribute Grammars

Checking Attribute Grammars for Circularity

The Circularity Check

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

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 $\pi = A_0 \to w_0 A_1 w_1 \dots A_r w_r \in P$ in a node k_0 of t such that

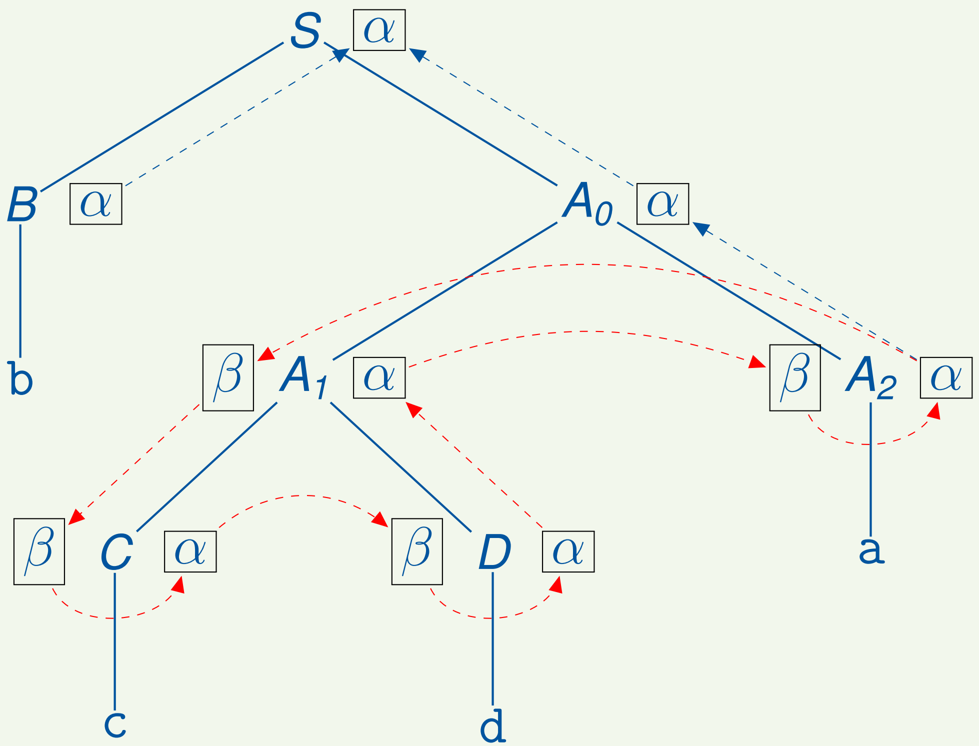
- the dependencies in E_{k_0} yield the "upper end" of the cycle and
- for at least one $i \in [r]$, some attributes in $\mathrm{syn}(A_i)$ depend on attributes in $\mathrm{inh}(A_i)$.

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

Example 13.1

Typical situation (with “cover” production $\pi = A_0 \rightarrow A_1 A_2 \in P$):



Attribute Dependency Graphs and Circularity III

Definition 13.2 (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, $\alpha \in \mathrm{syn}(A)$, 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: $\beta \overset{A}{\hookrightarrow} \alpha$).

Attribute Dependency Graphs and Circularity III

Definition 13.2 (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, $\alpha \in \mathrm{syn}(A)$, 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 $A \in N$,

$$is(A, t) := \{(\beta, \alpha) \in \mathrm{inh}(A) \times \mathrm{syn}(A) \mid \beta \overset{A}{\hookrightarrow} \alpha \text{ in } t\}.$$

Attribute Dependency Graphs and Circularity III

Definition 13.2 (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, $\alpha \in \mathrm{syn}(A)$, 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 $A \in N$,
$$IS(A) := \{is(A, t) \mid t \text{ syntax tree with root label } A\} \subseteq 2^{Inh \times Syn}.$$

Attribute Dependency Graphs and Circularity III

Definition 13.2 (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, $\alpha \in \mathrm{syn}(A)$, 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 later).

Attribute Dependency Graphs and Circularity III

Definition 13.2 (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, $\alpha \in \mathrm{syn}(A)$, 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 later).

Example 13.3

on the board

Outline of Lecture 13

The Circularity Check I

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

Definition 13.4

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

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

is defined by

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

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

The Circularity Check I

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

Definition 13.4

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 each $i \in [r]$,

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

is defined 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 13.5

on the board

The Circularity Check II

Algorithm 13.6 (Circularity check for attribute grammars)

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

The Circularity Check II

Algorithm 13.6 (Circularity check 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:*

i. *if* $\pi = A \to w \in P$, *then*

$$is[\pi] \in IS(A)$$

ii. *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*

$$is[\pi; is_1, \ldots, is_r] \in IS(A)$$

The Circularity Check II

Algorithm 13.6 (Circularity check 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:

i. if $\pi = A \to w \in P$, then
$$is[\pi] \in IS(A)$$

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

The Circularity Check II

Algorithm 13.6 (Circularity check 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:*

i. *if* $\pi = A \to w \in P$, *then*

$$is[\pi] \in IS(A)$$

ii. *if* $\pi = A \to w_0 A_1 w_1 \dots A_r w_r \in P$ *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* $\pi = A \to w_0 A_1 w_1 \dots A_r w_r \in P$ *and* $is_i \in IS(A_i)$ *for every* $i \in [r]$ *such that the following relation is cyclic:*

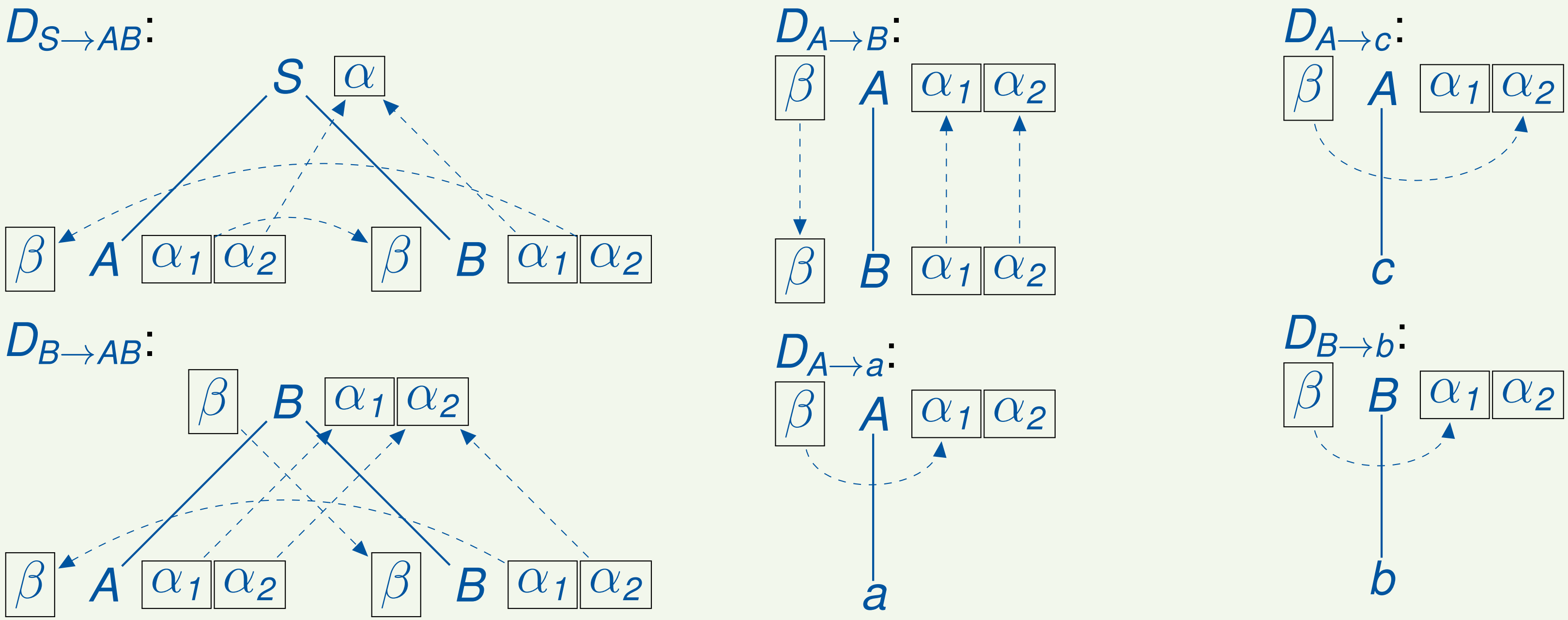
$$\to_\pi \cup \bigcup_{i=1}^{r} \{(\beta.p_i, \alpha.p_i) \mid (\beta, \alpha) \in is_i\}$$

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

Output: *"yes" or "no"*

The Circularity Check III

Example 13.7



Application of Algorithm 13.6: on the board

Outline of Lecture 13

Recap: Attribute Grammars

Checking Attribute Grammars for Circularity

The Circularity Check

Correctness and Complexity of the Circularity Check

Correctness and Complexity of Circularity Check

Theorem 13.8 (Correctness of circularity check)

An attribute grammar is circular iff Algorithm 13.6 yields the answer "yes"

Correctness and Complexity of Circularity Check

Theorem 13.8 (Correctness of circularity check)

An attribute grammar is circular iff Algorithm 13.6 yields the answer "yes"

Proof.

by induction on the syntax tree t with cyclic D_t □

Correctness and Complexity of Circularity Check

Theorem 13.8 (Correctness of circularity check)

An attribute grammar is circular iff Algorithm 13.6 yields the answer "yes"

Proof.

by induction on the syntax tree t with cyclic D_t □

Lemma 13.9

The time complexity of the circularity check is exponential in the size of the attribute grammar (= maximal length of right-hand sides of productions).

Correctness and Complexity of Circularity Check

Theorem 13.8 (Correctness of circularity check)

An attribute grammar is circular iff Algorithm 13.6 yields the answer "yes"

Proof.

by induction on the syntax tree t with cyclic D_t □

Lemma 13.9

The time complexity of the circularity check 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. ACM 28(4), 1981, pp. 715–720) □