Compiler Construction

Lecture 14: Semantic Analysis II (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/cc08/

Summer semester 2008

Outline

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 $\pi = Y_0 \to Y_1 \dots Y_r \in P$ 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 inner and outer variables:
$$In_\pi := \{\alpha.i \mid (i = 0, \alpha \in \mathrm{syn}(Y_i)) \text{ or } (i \in [r], \alpha \in \mathrm{inh}(Y_i))\}$$
$$Out_\pi := Var_\pi \setminus In_\pi$$
- A semantic rule of π 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 $f : V^{\alpha_1} \times \dots \times V^{\alpha_n} \to V^\alpha$.
- 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: $\mathfrak{A} \in AG$.

Example: Knuth's Binary Numbers I

Example (synthesized + inherited attributes)

Binary numbers (with fraction):

G'_B : Numbers $N \to L$

$N \to L \,.\, L$

Lists $L \to B$

$L \to LB$

Bits $B \to \texttt{0}$

Bits $B \to \texttt{1}$

Example: Knuth's Binary Numbers I

Example (synthesized + inherited attributes)

Binary numbers (with fraction):

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

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

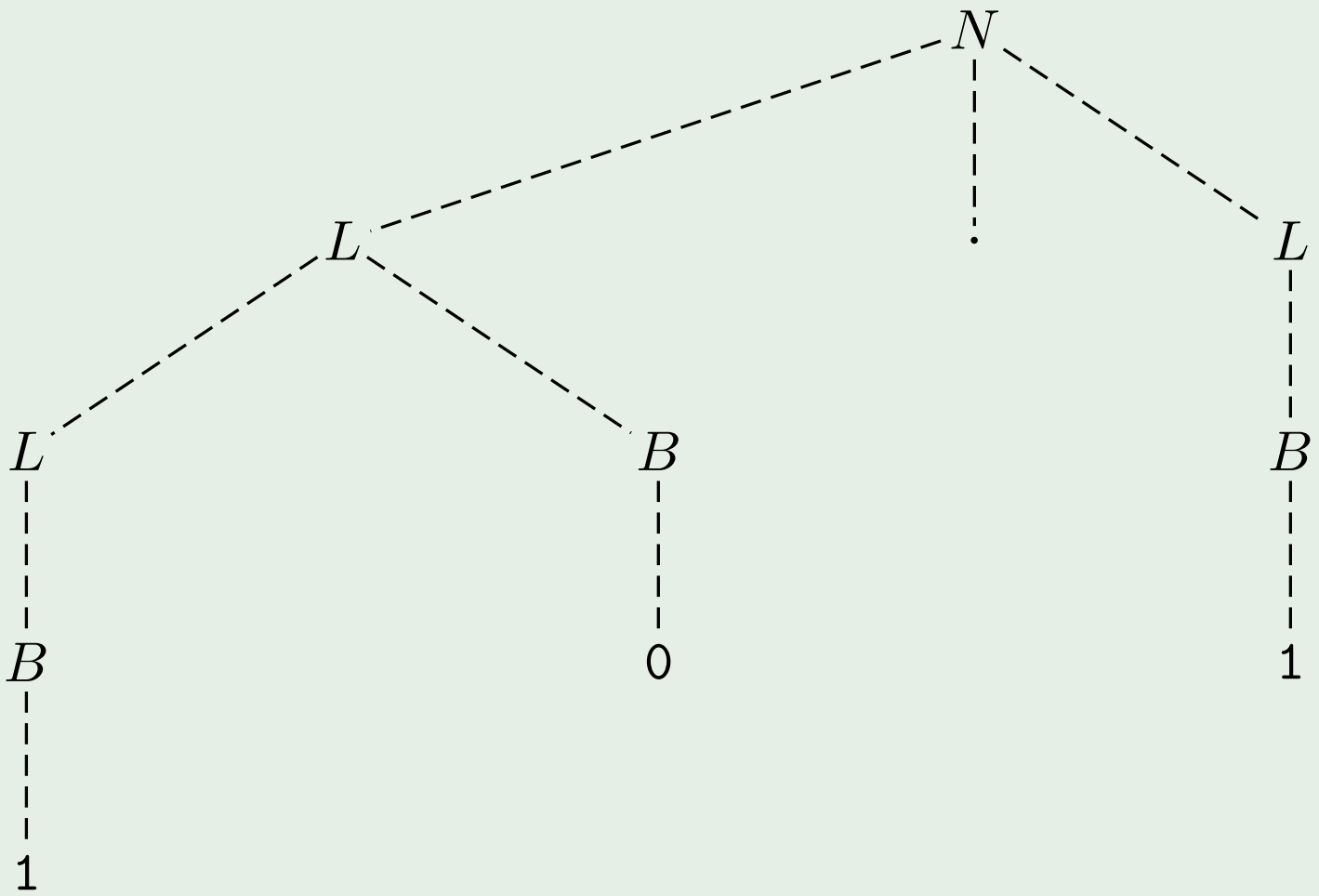
of L: l (length; domain: $V^l := \mathbb{N}$)

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

Example: Knuth's Binary Numbers II

Example (continued)

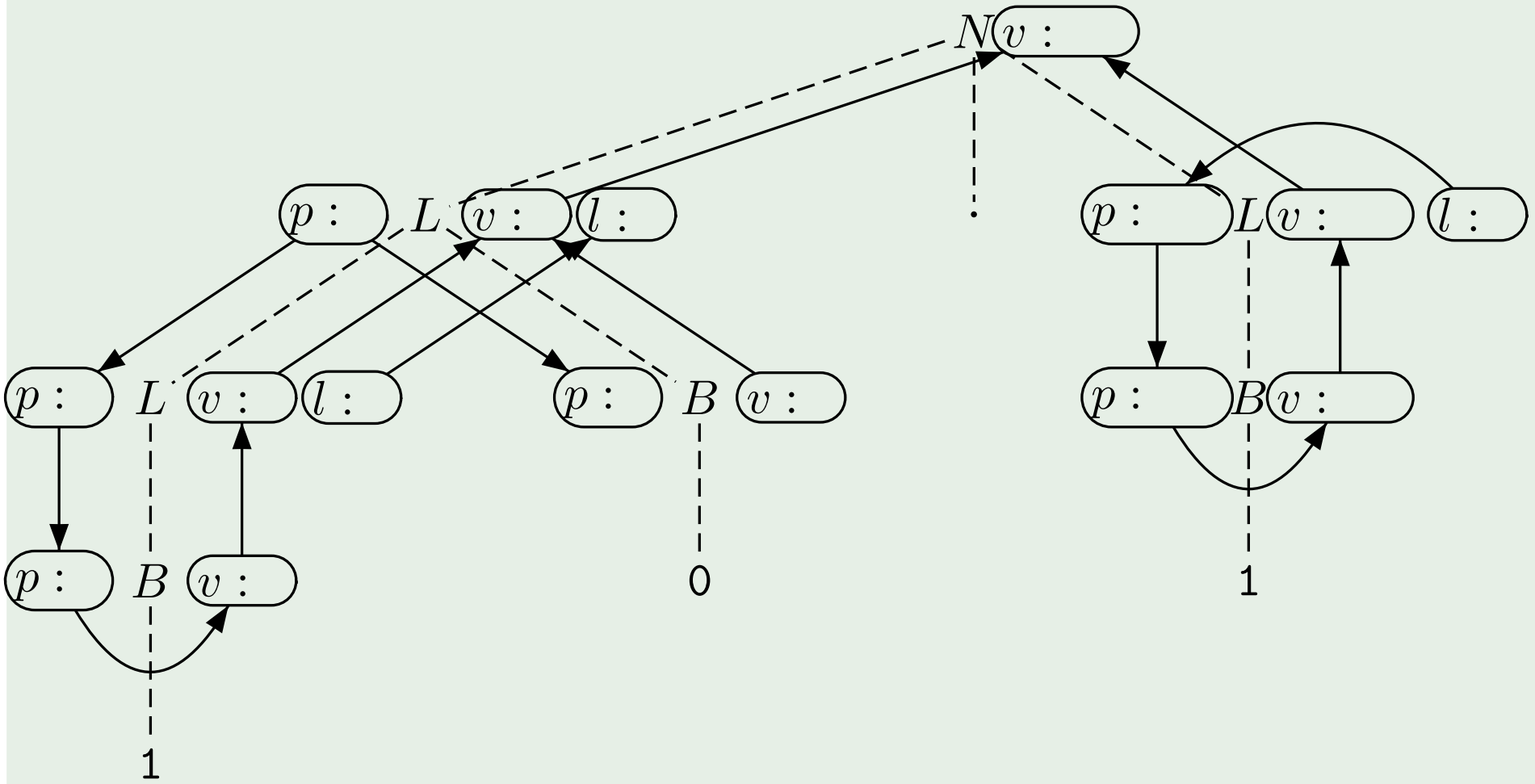
Syntax tree for `10.1`:



Example: Knuth's Binary Numbers II

Example (continued)

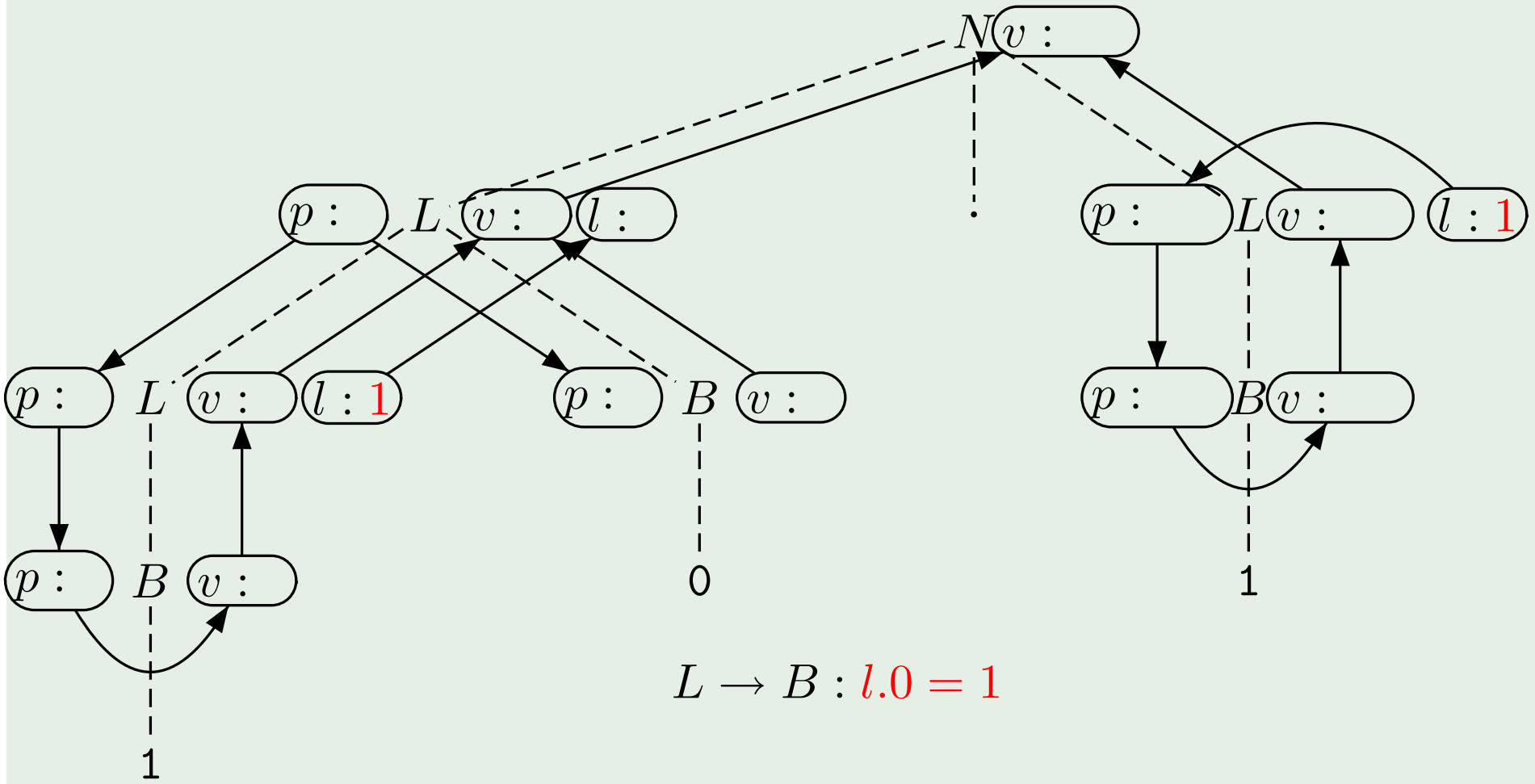
Attributed syntax tree for `10.1`:



Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

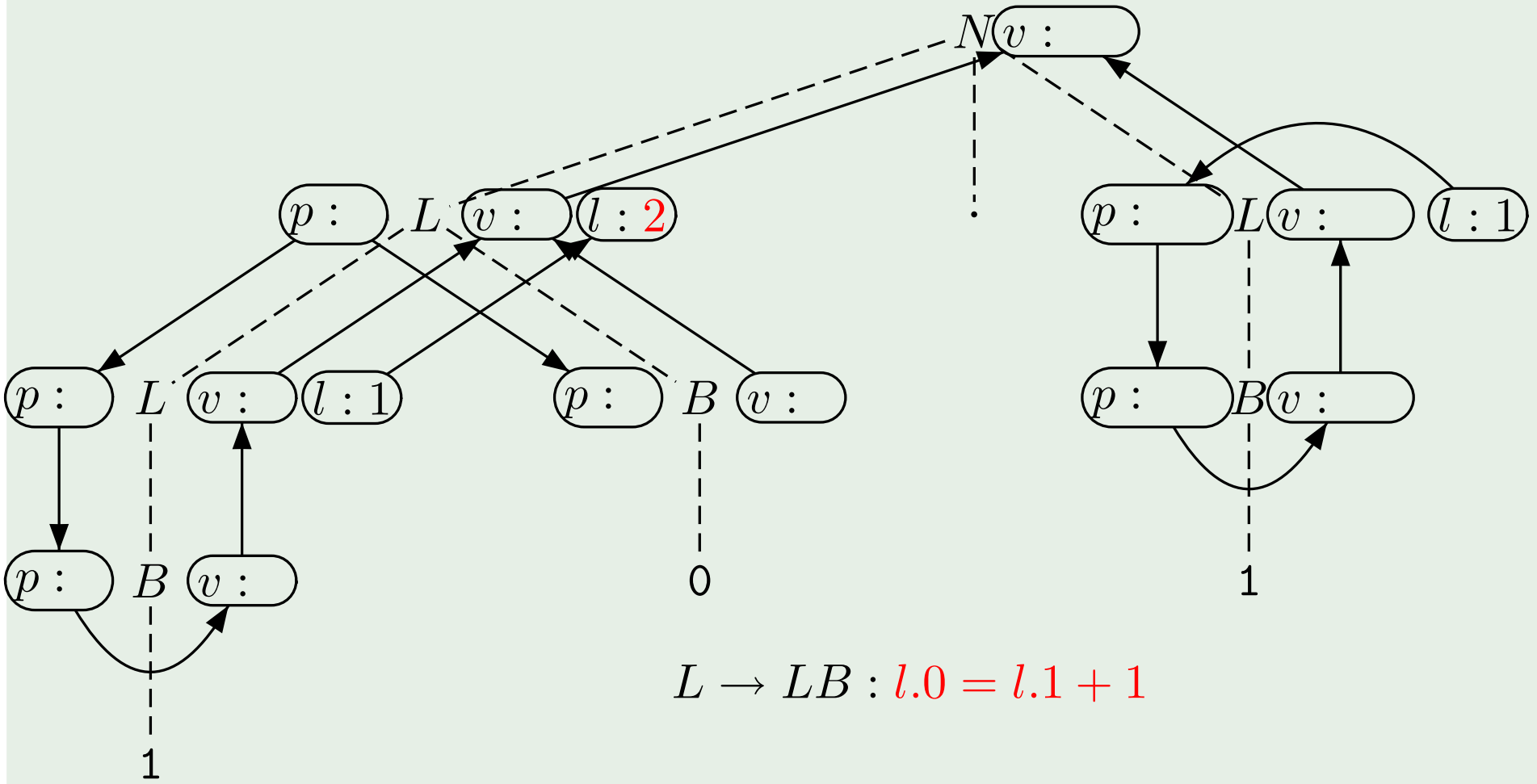


$$L \to B : l.0 = 1$$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

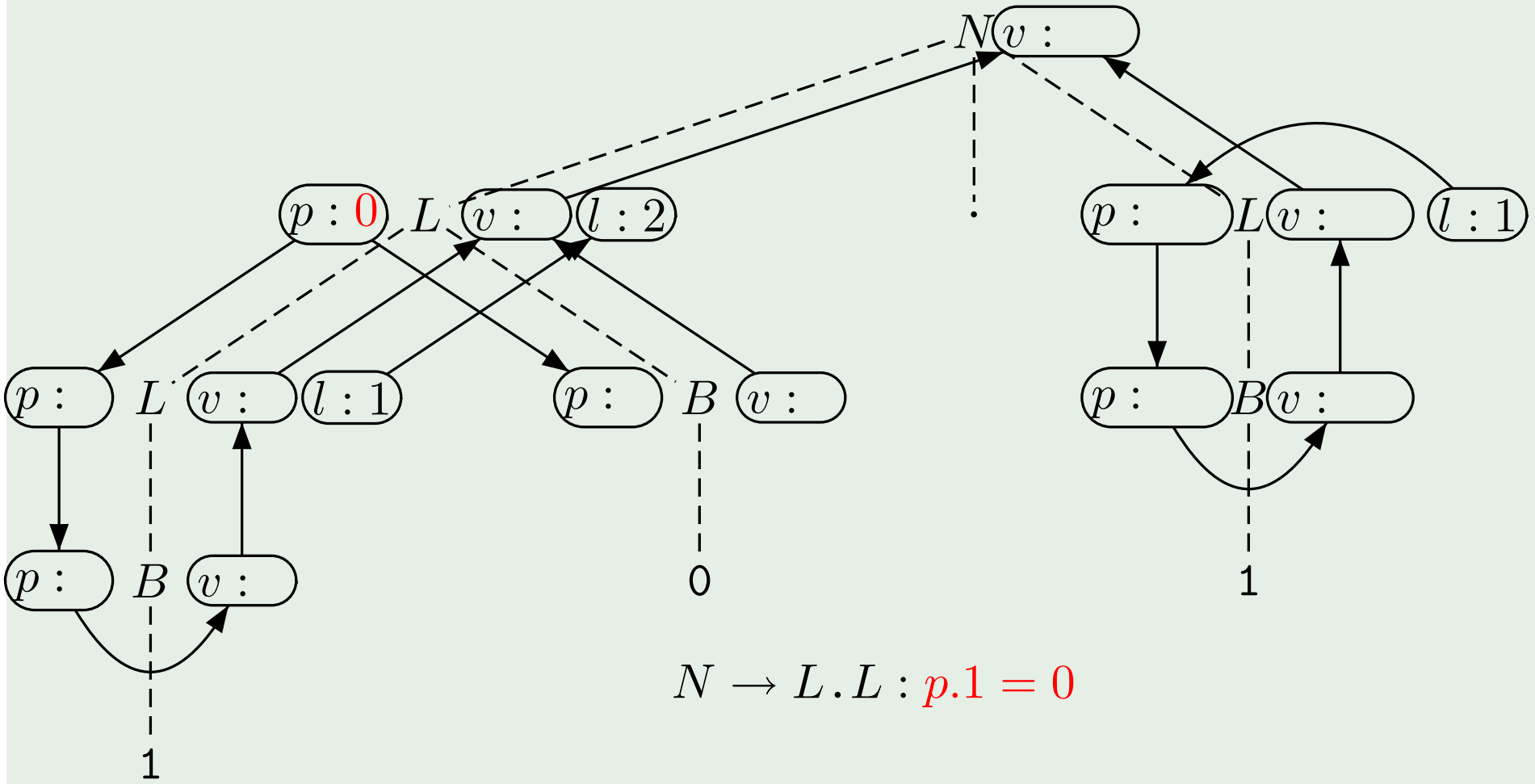


$$L \to LB : l.0 = l.1 + 1$$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

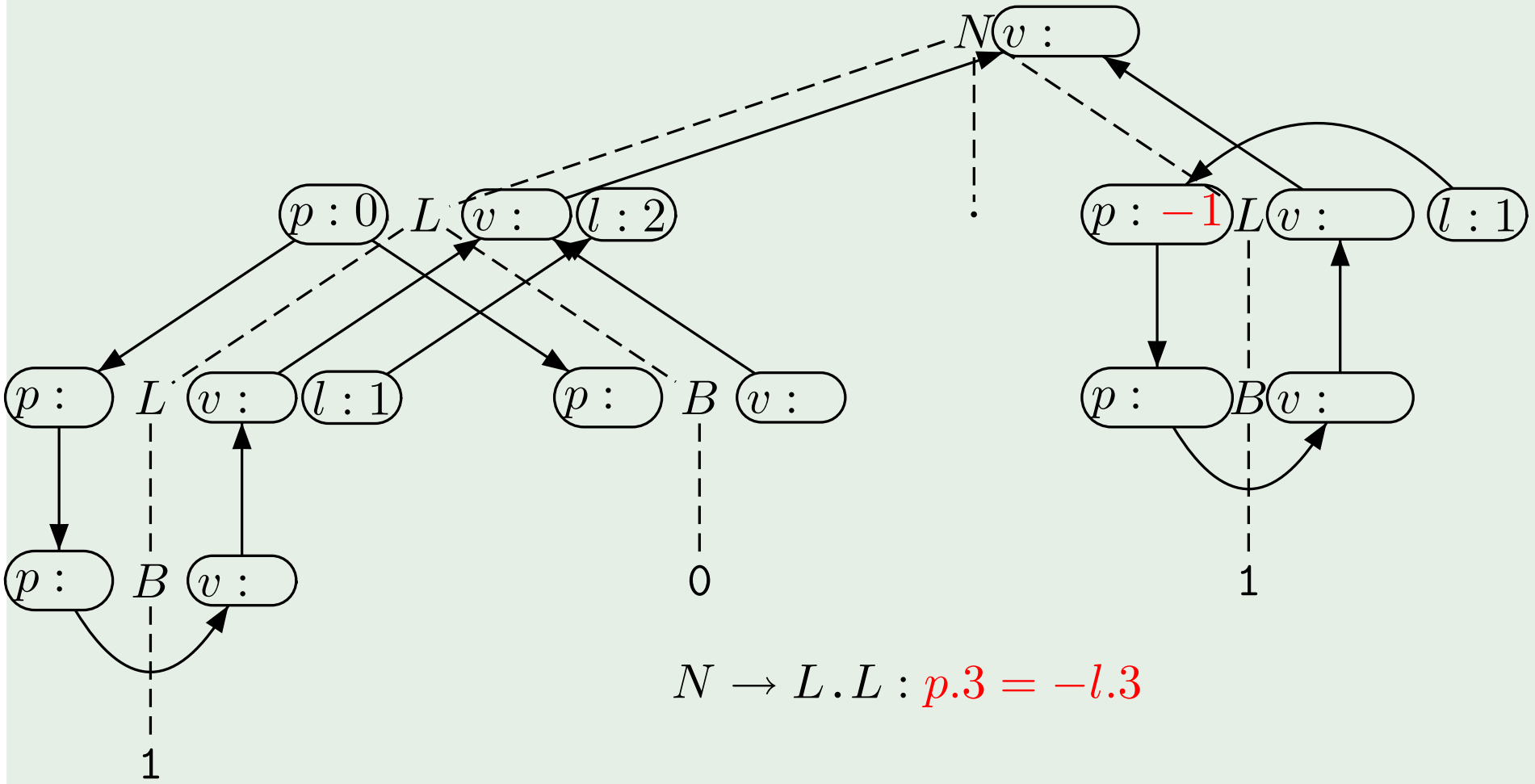


$N \to L\,.\,L : p.1 = 0$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

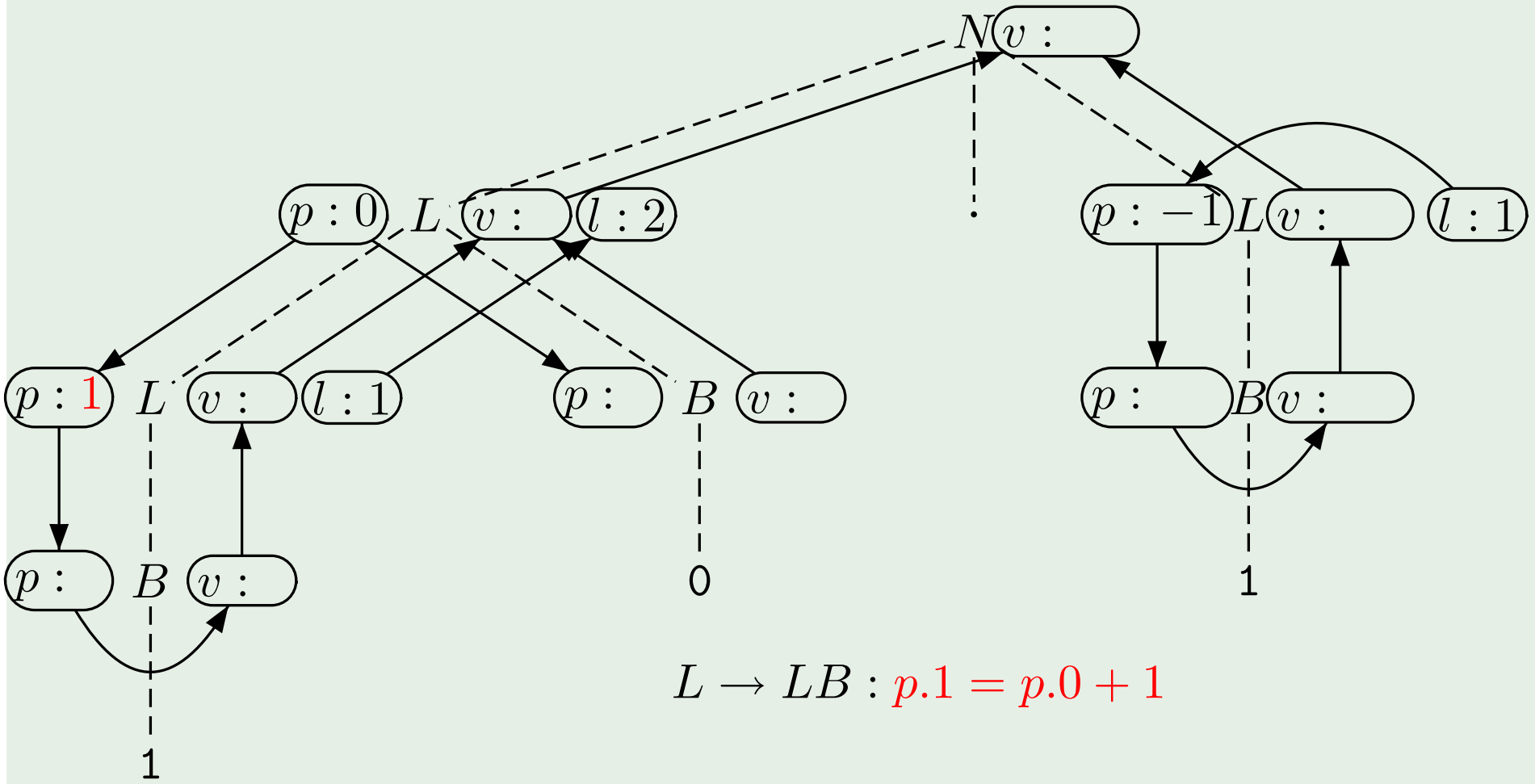


$N \to L\,.\,L : p.3 = -l.3$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

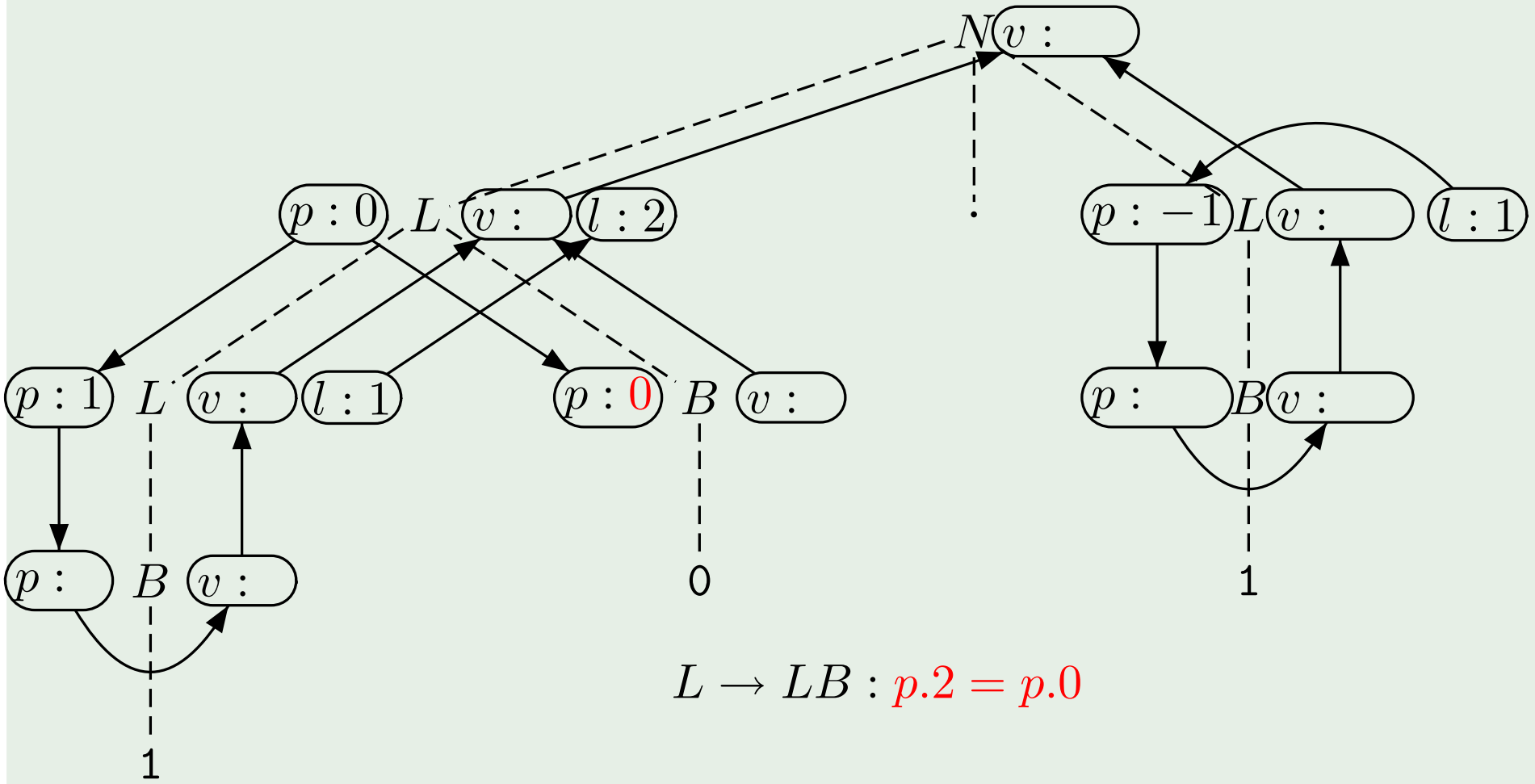


$$L \to LB : p.1 = p.0 + 1$$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

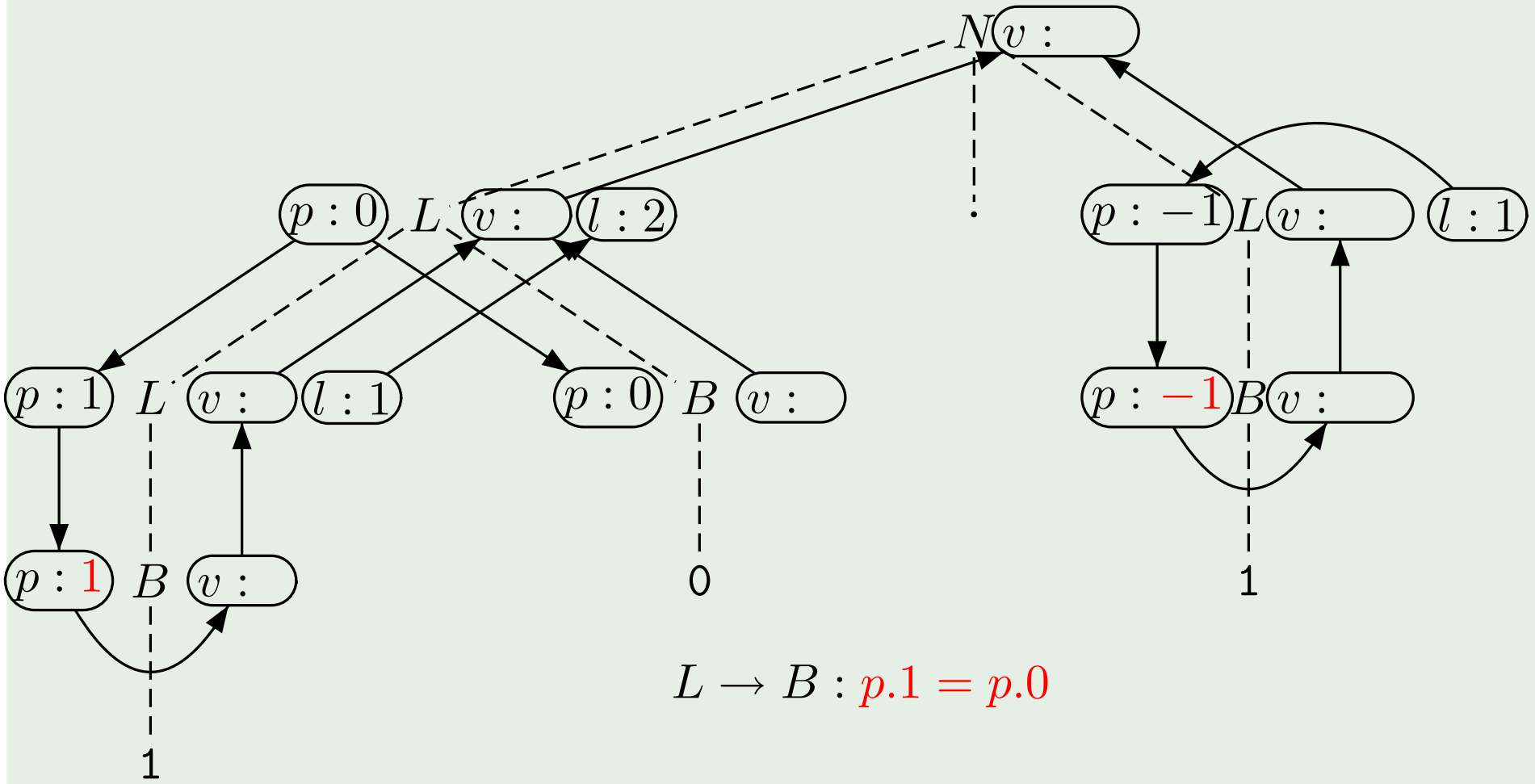


$L \to LB : p.2 = p.0$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

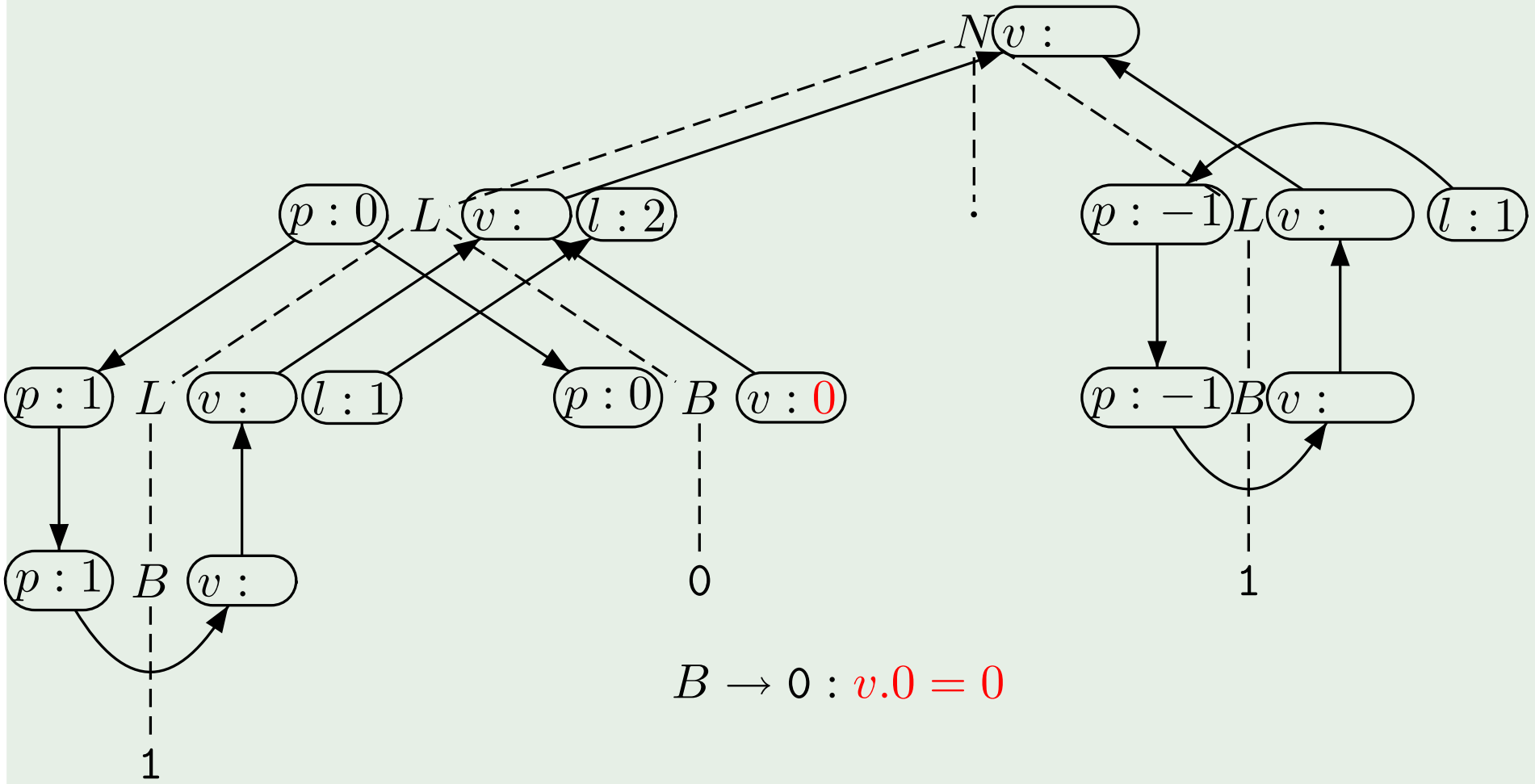


$L \to B : p.1 = p.0$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

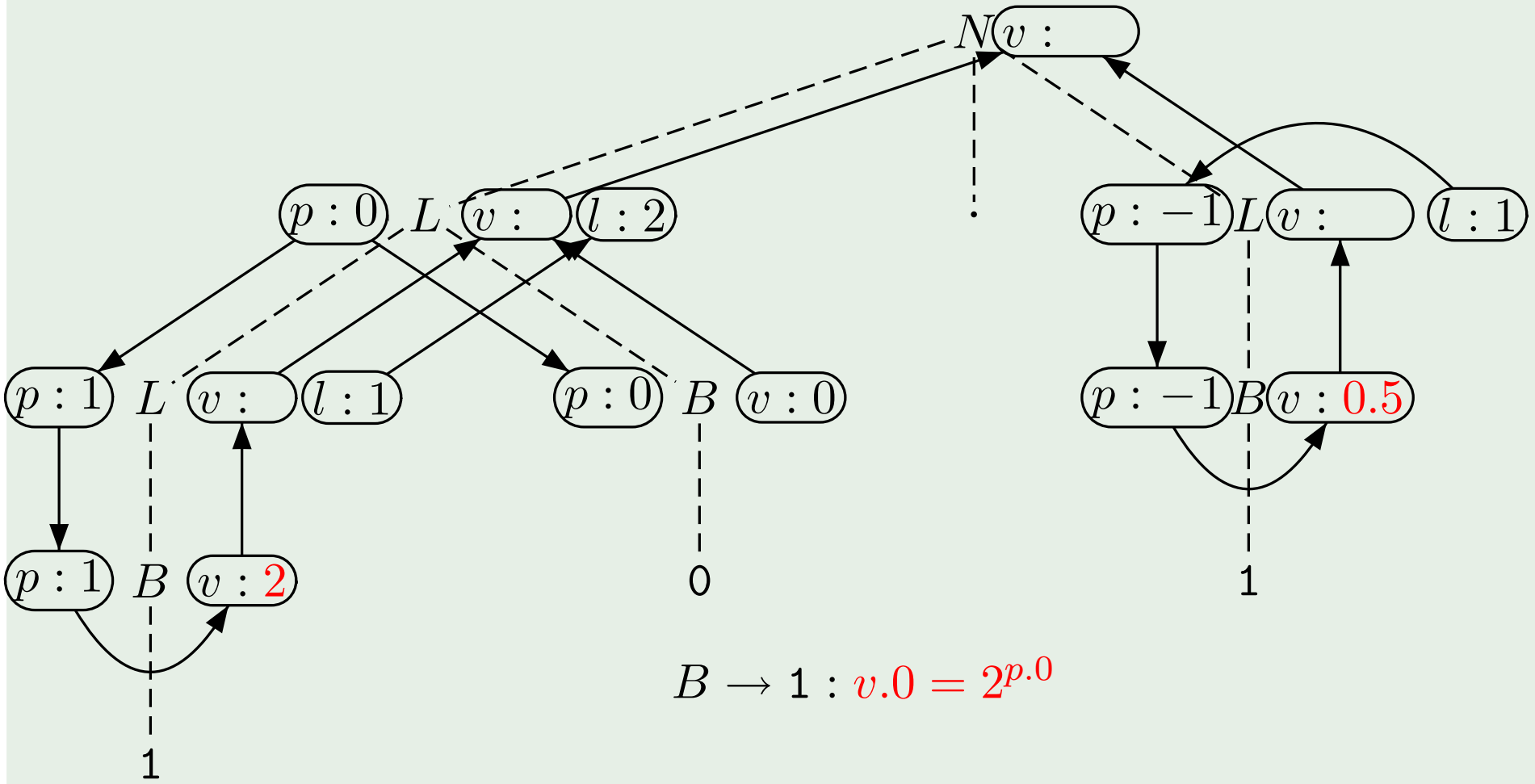


$B \to \mathtt{0} : v.0 = 0$

Example: Knuth's Binary Numbers II

Example (continued)

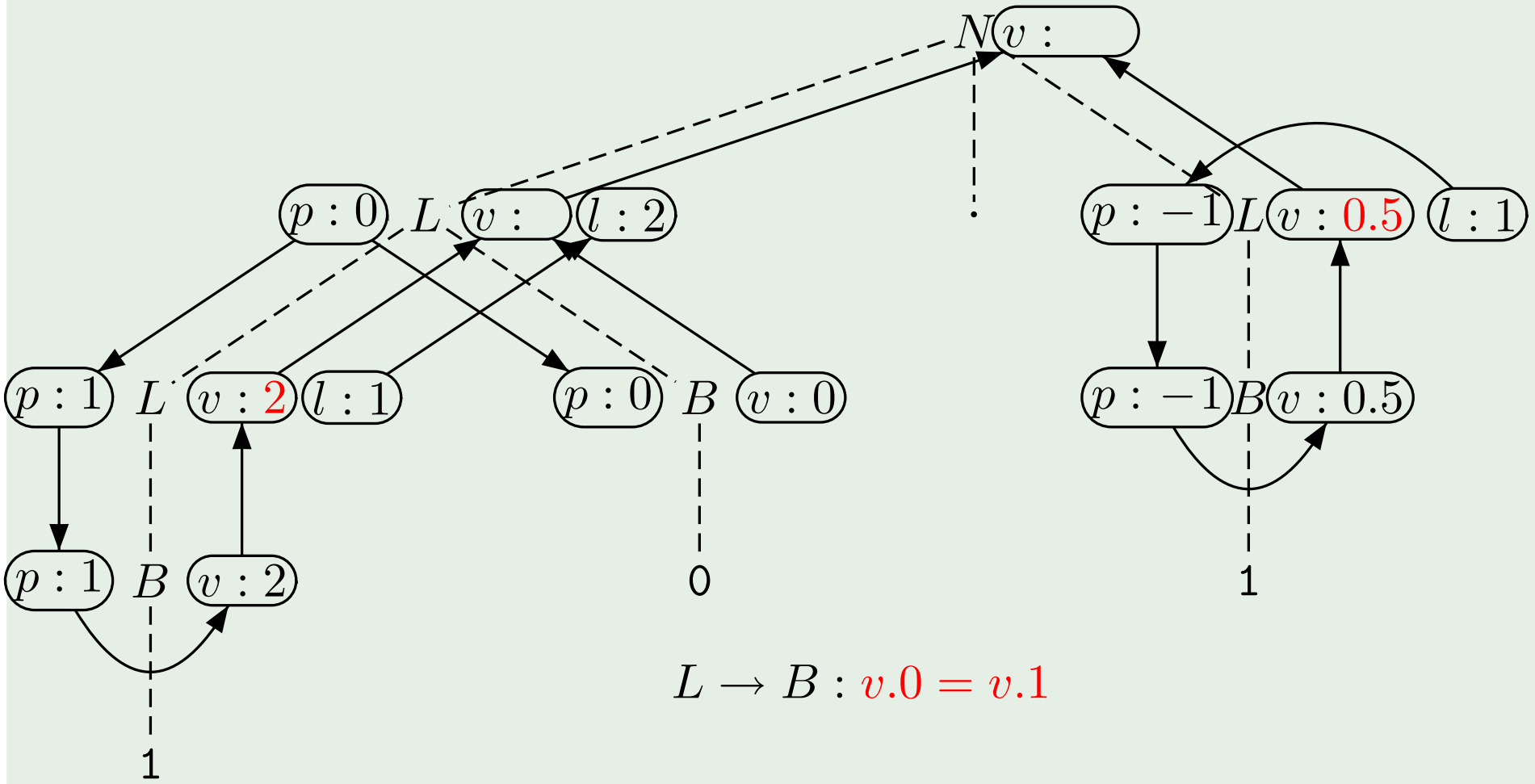
Attributed syntax tree for `10.1`:



Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

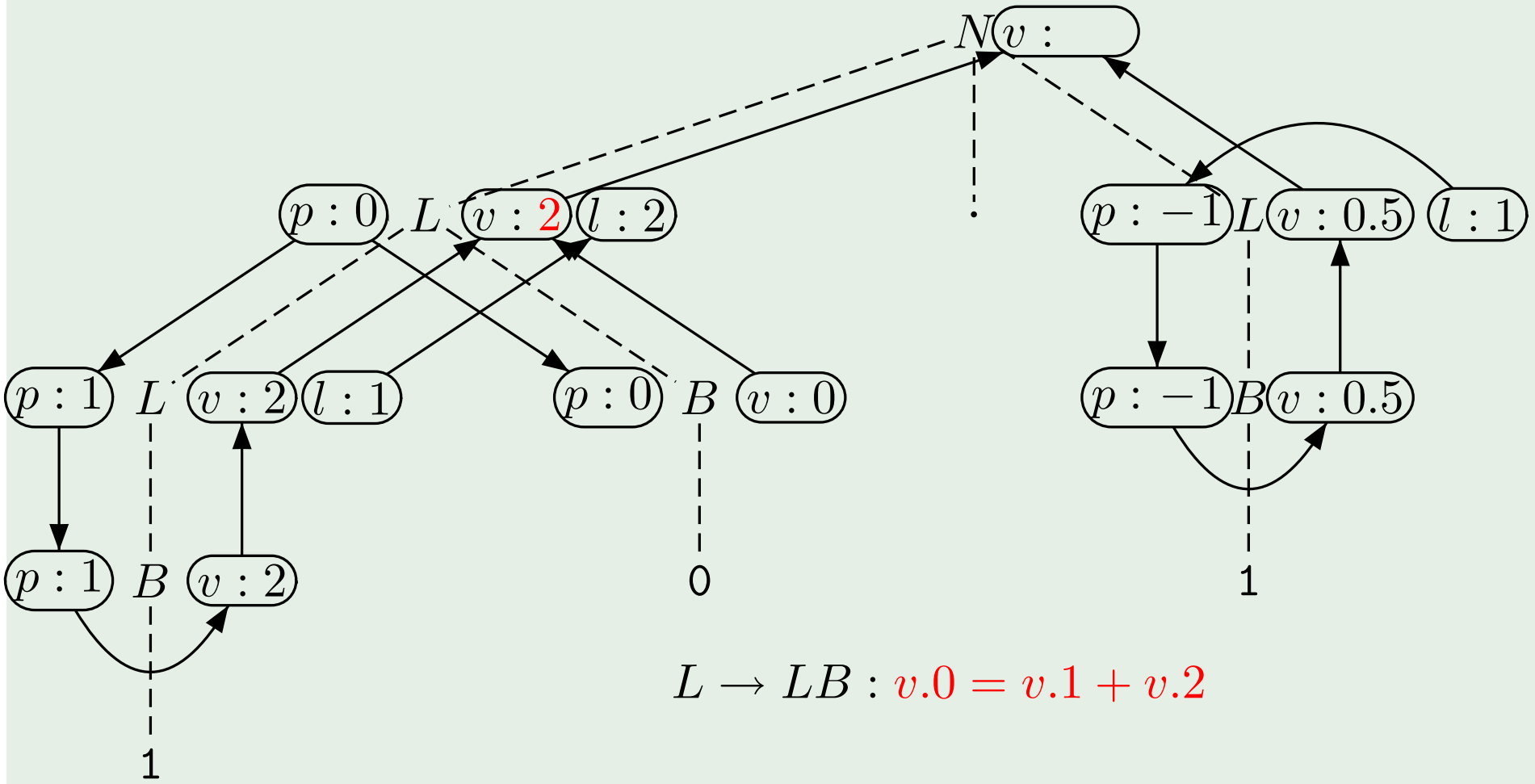


$L \to B : v.0 = v.1$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:

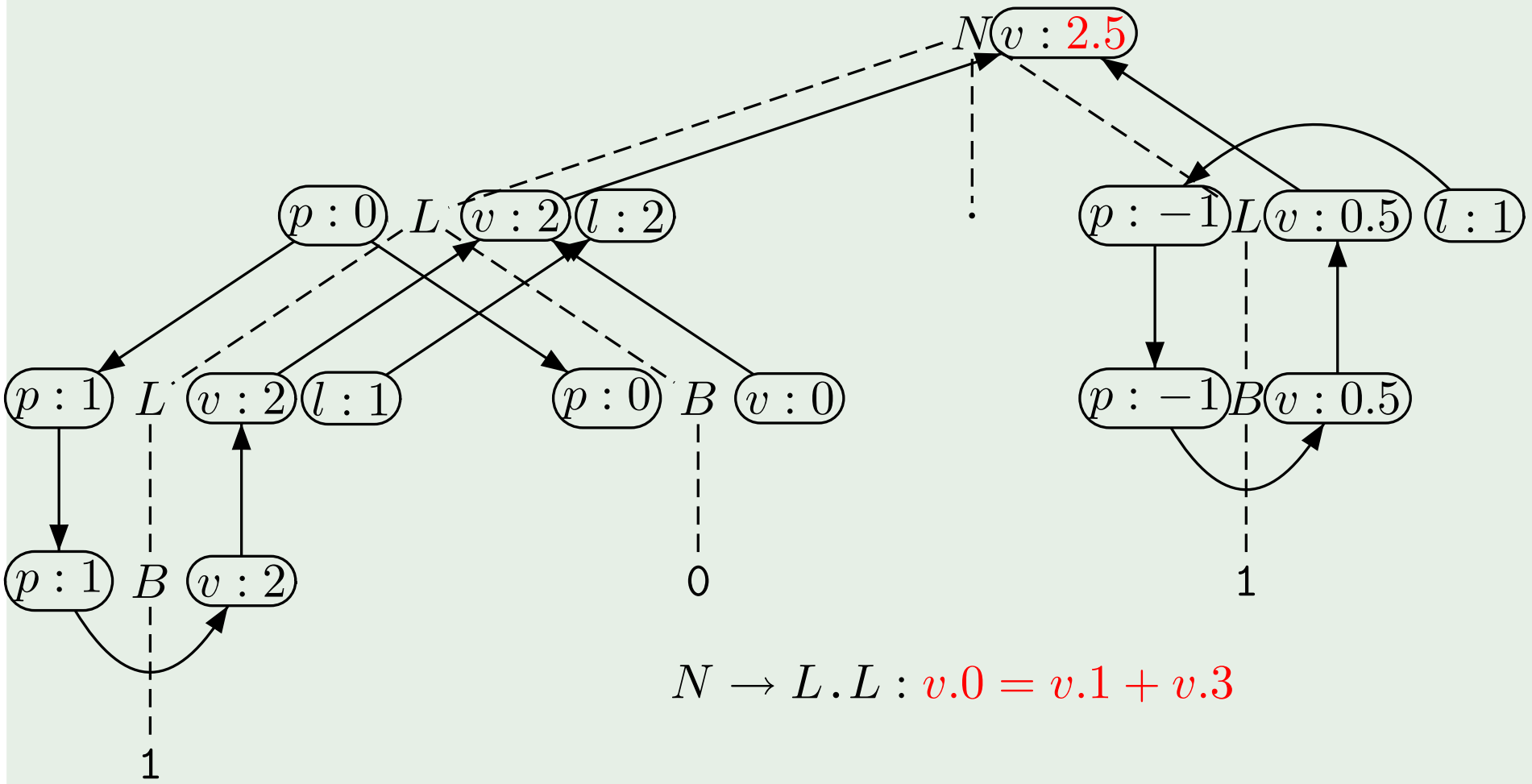


$$L \to LB : v.0 = v.1 + v.2$$

Example: Knuth's Binary Numbers II

Example (continued)

Attributed syntax tree for `10.1`:



$N \to L\,.\,L : v.0 = v.1 + v.3$

Outline

Solvability of the 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

$$v : Var_t \to V$$

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

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

Solvability of the 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

$$v : Var_t \to V$$

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.

Solvability of the Attribute Equation System II

Example 14.2

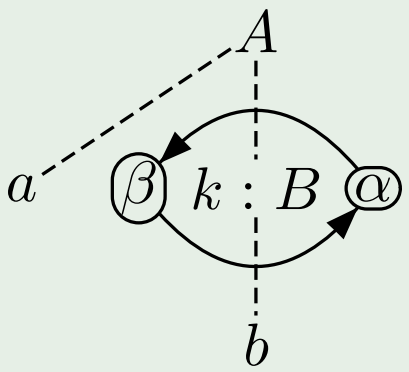
- $A \to aB, B \to b \in P$
- $\alpha \in \text{syn}(B)$, $\beta \in \text{inh}(B)$
- $\beta.2 = f(\alpha.2) \in E_{A \to aB}$
- $\alpha.0 = g(\beta.0) \in E_{B \to b}$

Solvability of the Attribute Equation System II

Example 14.2

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

$\Longrightarrow$ cyclic dependency:



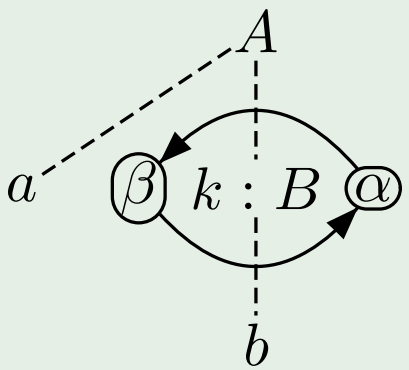
$$E_t: \quad \begin{aligned} \beta.k &= f(\alpha.k) \\ \alpha.k &= g(\beta.k) \end{aligned}$$

Solvability of the Attribute Equation System II

Example 14.2

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

$\Longrightarrow$ cyclic dependency:



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

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

Circularity of Attribute Grammars

Goal: unique solvability of equation system

$\Longrightarrow$ avoid cyclic dependencies

Circularity of Attribute Grammars

Goal: unique solvability of equation system
$\Longrightarrow$ avoid cyclic dependencies

Definition 14.3 (Circularity)

An attribute grammar $\mathfrak{AG} = \langle G, E, V \rangle \in AG$ 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.

Circularity of Attribute Grammars

Goal: unique solvability of equation system
$\Longrightarrow$ avoid cyclic dependencies

Definition 14.3 (Circularity)

An attribute grammar $\mathfrak{AG} = \langle G, E, V \rangle \in AG$ 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 (see Corollary 14.5).

Outline

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 $D_\pi := \langle Var_\pi, \rightarrow_\pi \rangle$ where the set of edges $\rightarrow_\pi \subseteq Var_\pi \times Var_\pi$ is given by

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

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 $D_\pi := \langle Var_\pi, \rightarrow_\pi \rangle$ where the set of edges $\rightarrow_\pi \subseteq Var_\pi \times Var_\pi$ is given by

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

Corollary 14.5

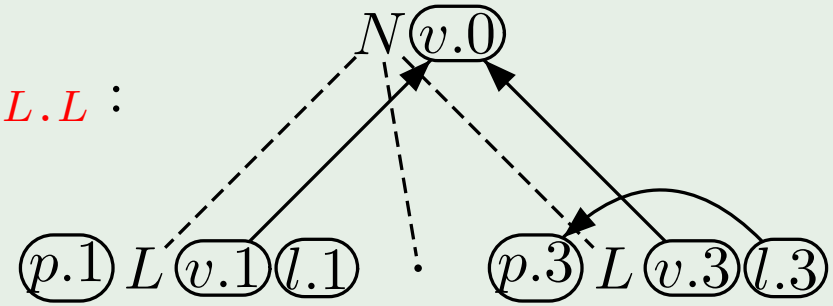
The dependency graph of a production is acyclic (since $\rightarrow_\pi \subseteq Out_\pi \times In_\pi$).

Attribute Dependency Graphs II

Example 14.6 (cf. Example 13.2)

1. $N \to L\,.\,L$: $\quad \Longrightarrow \quad D_{N \to L\,.\,L}$:

$$
\begin{aligned}
v.0 &= v.1 + v.3 \\
p.1 &= 0 \\
p.3 &= -l.3
\end{aligned}
$$

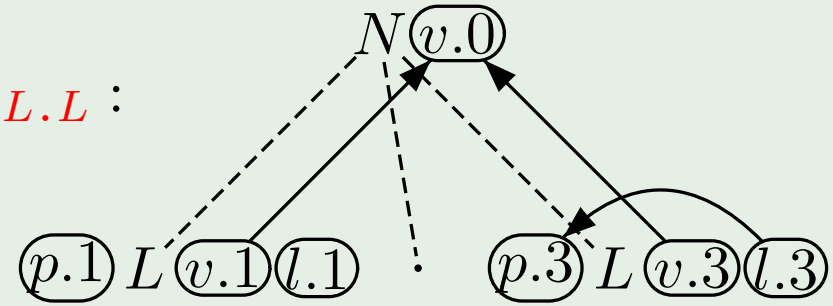


Attribute Dependency Graphs II

Example 14.6 (cf. Example 13.2)

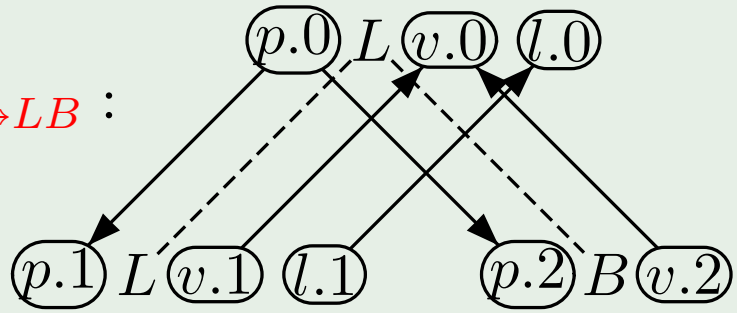
1. $N \to L\,.\,L$: $\Longrightarrow$ $D_{N \to L\,.\,L}$:

$$v.0 = v.1 + v.3$$
$$p.1 = 0$$
$$p.3 = -l.3$$



2. $L \to LB$: $\Longrightarrow$ $D_{N \to LB}$:

$$v.0 = v.1 + v.2$$
$$l.0 = l.1 + 1$$
$$p.1 = p.0 + 1$$
$$p.2 = p.0$$



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.

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 $D_t := \langle Var_t, \rightarrow_t \rangle$ 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 $x \rightarrow_t^+ x$.

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 $D_t := \langle Var_t, \rightarrow_t \rangle$ 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 $x \rightarrow_t^+ x$.

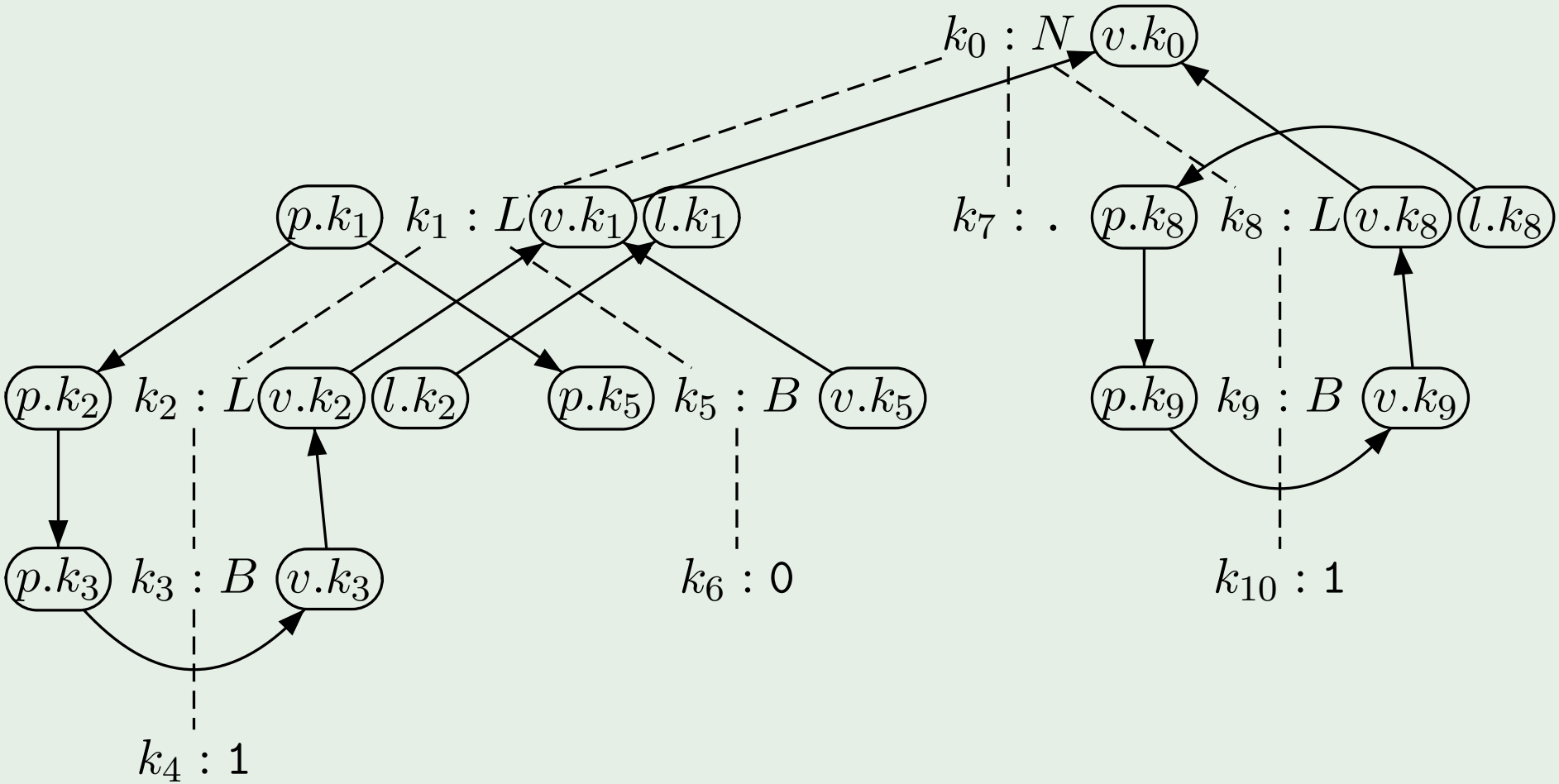
Corollary 14.8

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`:



Outline

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 $\mathrm{inh}(A_i)$.

Example 14.10

on the board

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 $\mathrm{inh}(A_i)$.

Example 14.10

on the board

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

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, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \rightarrow_t^+ \alpha.k$, then α is dependent on β below A in t (notation: $\beta \overset{A}{\hookrightarrow} \alpha$).

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, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \rightarrow_t^+ \alpha.k$, then α is dependent on β below A in t (notation: $\beta \overset{A}{\hookrightarrow} \alpha$).
- For every syntax tree t with root label $A \in N$,

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

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, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \rightarrow_t^+ \alpha.k$, 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 $A \in N$,
$$IS(A) := \{is(A,t) \mid t \text{ syntax tree with root label A}\} \subseteq 2^{Inh \times Syn}.$$

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, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \rightarrow_t^+ \alpha.k$, 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 (later: strong noncircularity).

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, $\alpha \in \mathrm{syn}(A)$, and $\beta \in \mathrm{inh}(A)$ such that $\beta.k \rightarrow_t^+ \alpha.k$, 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 (later: strong noncircularity).

Example 14.12

on the board

Outline

The Circularity Test I

In the circularity test, the dependency systems $IS(A)$ are iteratively computed. It employs the following notation:

Definition 14.13

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 every $i \in [r]$, let

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

be given 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 Test I

In the circularity test, the dependency systems $IS(A)$ are iteratively computed. It employs the following notation:

Definition 14.13

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 every $i \in [r]$, let

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

be given by

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

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

Example 14.14

on the board

The Circularity Test II

Algorithm 14.15 (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* $\pi = A \to w \in P$, *then* $is[\pi] \in IS(A)$
   2. *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:*

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

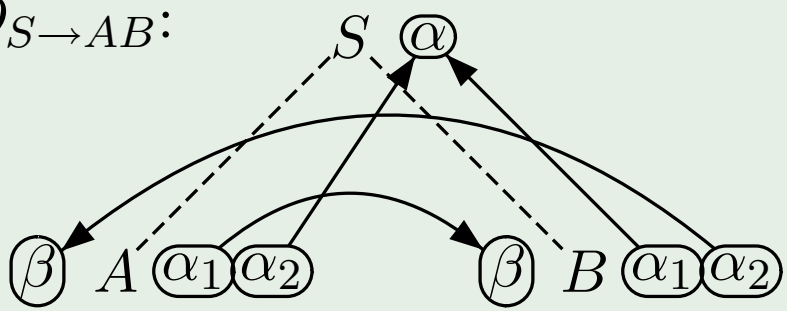
(where $p_i := \sum_{j=1}^{i} |w_{j-1}| + i$*)*

Output: *"yes" or "no"*

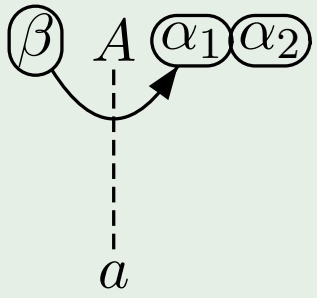
The Circularity Test III

Example 14.16

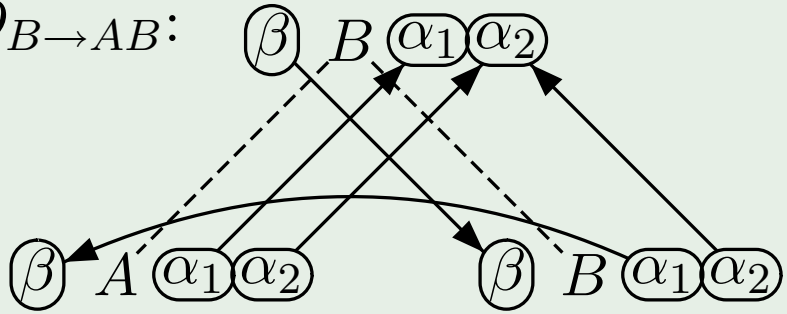
$D_{S \to AB}$:



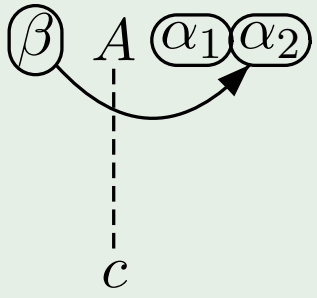
$D_{A \to a}$:



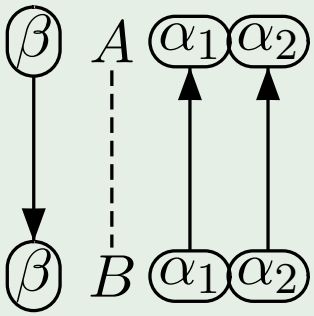
$D_{B \to AB}$:



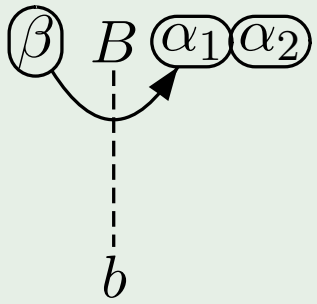
$D_{A \to c}$:



$D_{A \to B}$:



$D_{B \to b}$:



Application of Algorithm 14.15: on the board