Discrete Applied Mathematics 32 (1991) 103-129
North-Holland

Grammatical codes of trees

A. Ehrenfeucht

Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA

G. Rozenberg

Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, Netherlands; and Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA

Received 10 April 1988
Revised 2 October 1989

Abstract

Ehrenfeucht, A. and G. Rozenberg, Grammatical codes of trees, Discrete Applied Mathematics 32 (1991) 103-129.

The problem of coding (chain free) trees by words where the length of the word coding a tree t equals the number of leaves of t is investigated. The notion of an insertive strict code is introduced and investigated—these are codes of a grammatical nature. It is shown that there are exactly 120 insertive strict codes. A characterization of these codes (and their various subclasses) is given in grammatical terms.

Introduction

The notion of a *tree* plays an important role in (among others) linguistics, logic, mathematics, and computer science. The concept itself was used before it got its current name and graphical notation—see, e.g., the notation of a semantic tableaux as used in logic by Jaskowski [8] and Gentzen [6], or the notion of prime constituents as used in linguistics by Harris [7] and Fris [5].

The concept of a *deep structure* used by Chomsky [2] is different from the notion of surface structure (representing syntax) and the notion of semantical structure (representing the meaning). For Chomsky, deep structure is a blueprint for the construction of the surface structure. In the formalization of Chomsky's ideas by mathematicians (such as Bar-Hillel) the notion of a deep structure became the notion of a derivation tree—hence trees became algorithms for constructing strings.

In this paper we view a tree very much in the line of a deep structure by Chomsky—it is an object that can be "matched" in two directions—towards syntax and towards semantics. As a matter of fact this point of view corresponds quite closely to the concept of a tree as used in mathematics.

The notion of a *grammar* in formal language theory corresponds closely to an algorithm that either (nondeterministically) generates text or (nondeterministically) parses text. In general linguistics and in logic (see, e.g., [9]) a grammar provides relationships between concepts (objects) that are given by the lexicon of the language. Still another approach, represented by [1], is to view a grammar as a system which protects text from a "noise" (by inserting "check bits" into a text).

In this paper we view grammar as coding deep structures, where we assume that a deep structure is an ordered chain-free tree. Accordingly, the main question investigated in this paper is: "What are good grammatical linear codes for ordered chain-free trees?"

Preliminaries

We assume the reader to be familiar with basic notions of graph theory and in particular with the basic theory of trees, and with the basics of formal languages and automata theory.

We will recall now some notions and establish the notation to be used in this paper.

For a set Z, $\# Z$ denotes its cardinality; $\emptyset$ denotes the empty set. $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{N}_+ = \mathbb{N} - \{0\}$, and for each $k \geq 2$, $\mathbb{N}^k$ is the k-folded cartesian product of $\mathbb{N}$.

For a function $\varphi: X \to Y$, $\mathrm{Dom}(\varphi)$ denotes X, $\mathrm{Ran}(\varphi)$ denotes Y, and $\mathrm{Rran}(\varphi) = \{y \in Y\colon y = \varphi(x) \text{ for some } x \in X\}$. We consider only total functions.

For a sequence x, $|x|$ denotes its length, and $\underline{i}(x)$ for an $1 \leq i \leq |x|$ denotes the ith element of x (this notation carries over to words which are sequences of letters); also we use $\mathrm{first}(x)$ to denote the first element of x and $\mathrm{last}(x)$ to denote the last element of x.

For a word x, $\pi(x)$ denotes the Parikh vector of x, and $\mathrm{alph}(x)$ is the set of letters appearing in x. For words x, y we say that x is a *segment of* y iff $y = y_1 x y_2$ for some words y_1, y_2 and we say that x is a *subword of* y iff $y = y_0 a_1 y_1 a_2 \dots a_n y_n$ for some words $y_0, y_1, \dots, y_n$ and letters $a_1, \dots, a_n$, where $x = a_1 \dots a_n$.

We consider only trees *without chains*, i.e., each internal node has more than one direct descendant. Hence, by a *tree* we mean a nonempty rooted directed ordered tree without chains (where "ordered" means that for each node all its direct descendants are linearly ordered).

Let t be a tree.

$\mathrm{ND}(t)$ denotes the set of all nodes of t, $\mathrm{IN}(t)$ denotes the set of internal nodes of t, $\mathrm{LEAF}(t)$ denotes the set of leaves of t, and $\mathrm{root}(t)$ denotes the root of t.

For an internal node v of t, $\mathrm{DDES}_t(v)$ is the set of all direct descendants of v in t, and $\mathrm{ddes}_t(v)$ is the sequence of all direct descendants of v in t (i.e., the elements of $\mathrm{DDES}_t(v)$ ordered according to the order of t).

Frontier of t, denoted $\mathrm{front}(t)$, is the sequence of all leaves of t ordered according to the order of t.

For a $1 \le i \le |\mathrm{front}(t)|$, $\mathrm{leaf}_t(i)$ denotes the ith leaf of t, hence the ith element of $\mathrm{front}(t)$.

If $x \in \mathrm{LEAF}(t)$, then

- x is a *leftmost child* iff there exists $v \in \mathrm{IN}(t)$ such that $x = \mathrm{first}(\mathrm{ddes}_t(v))$,
- x is a *rightmost child* iff there exists $v \in \mathrm{IN}(t)$ such that $x = \mathrm{last}(\mathrm{ddes}_t(v))$,
- x is a *middle child* iff there exists $v \in \mathrm{IN}(t)$ such that $x \in \mathrm{DDES}_t(v)$ and x is neither a leftmost nor a rightmost child.

If w is a segment of $\mathrm{front}(t)$ (i.e., w is a sequence of consecutive elements of $\mathrm{front}(t)$), then

- w is a *sibling segment* (*of* $\mathrm{front}(t)$) iff $|w| = 2$ and there exists $v \in \mathrm{IN}(t)$ such that w is a segment of $\mathrm{ddes}_t(v)$, and
- w is a *complete segment* (*of* $\mathrm{front}(t)$) iff there exists $v \in \mathrm{IN}(t)$ such that $w = \mathrm{ddes}_t(v)$.

For a $1 \le i \le |\mathrm{front}(t)|$ and $n \ge 2$, $\mathrm{sub}_t(i, n)$ denotes the family of all trees resulting from t by adding n new nodes and making them the direct descendants of $\mathrm{leaf}_t(i)$ (which in the resulting tree becomes an internal node).

A *node-labeling of t* (by an alphabet Σ) is a function $\psi : \mathrm{ND}(t) \to \Sigma$.

If we don't want to distinguish between isomorphic trees, then we can consider a *selector set* (*of trees*) which is a set of trees T such that, for each tree t there exists $t' \in T$ isomorphic with t, and moreover, for all different $t_1, t_2 \in T$, t_1 is not isomorphic with t_2.

A 0*S system* (see, e.g., [4]) is like a context-free grammar except that one does not distinguish between terminal and nonterminal symbols. An *unlimited* 0*S system* is like a 0S system except that it has infinitely many productions. Hence an unlimited 0S system is a triple $G = (\Sigma, P, \sigma)$, where Σ is the (finite) alphabet of G, $P \subseteq \Sigma \times \Sigma^+$ is the infinite set of productions of G, and $\sigma \in \Sigma$ is the axiom of G; *we assume that P does not contain chain productions, hence $|x| \ge 2$ for each $a \to x$ in P.* We use the standard notation for grammars: $a \to_P x$ (or $a \to_G x$) denotes the fact that $a \to x$ is in P, $y \Rightarrow_G z$ means that y *directly derives z in G*, and $y \Rightarrow_G^* z$, $y \Rightarrow_G^+ z$ stands for y *derives z in G*, and y *derives z in G in at least one step*, respectively. If $y \Rightarrow_G^* uzv$, for some $u, v \in \Sigma^*$, then we say that z *is reachable from y in G*. If $a \to_P xay$ for some $x, y \in \Sigma^*$, then a is *directly recursive*. Also, we use L_G and R_G (or simply L and R whenever G is understood from the context of considerations) to denote

$$\{b \in \Sigma\colon\ b = \mathrm{first}(x) \text{ for some } a \to x \text{ in } P\},$$

$$\{b \in \Sigma\colon\ b = \mathrm{last}(x) \text{ for some } a \to x \text{ in } P\}.$$

1. Codes and strict codes

In this section we formulate the notion of a code and introduce the subclass of strict codes which are the subject of investigation of this paper.

Definition 1.1. Let T be a selector set of trees, let Σ be an alphabet, and let $\varphi: T \to \Sigma^*$.

(i) φ is *length-preserving* iff, for all $t \in T$, $|\varphi(t)| = |\text{front}(t)|$.

(ii) φ is *local* iff there exists a mapping $\psi: \Sigma \times (\mathbb{N}_+ - \{1\}) \to \Sigma^+$ such that, for all $t_1, t_2 \in T$, where $t_2 \in \text{sub}_{t_1}(i, n)$ for some $i \in \mathbb{N}_+$, $n \in \mathbb{N}_+ - \{1\}$, if $\varphi(t_1) = xay$ with $|x| = i-1$ and $a \in \Sigma$, then $\varphi(t_2) = x\psi(a, n)y$.

(iii) φ is a *code* (*of* T) iff φ is injective, length-preserving, and local.

Remark 1.2. (1) Note that we have not required that Σ is finite.

(2) For technical reasons we have defined φ on a selector set T rather than on the class of all trees. However, φ is easily extended to the class of all trees: for a tree t, $\varphi(t) = \varphi(t')$ where $t' \in T$ is isomorphic with t; hence we will freely write $\varphi(t)$ for an arbitrary tree t.

(3) Since a code φ is length-preserving, for a $t \in T$, the ith element of $\varphi(t)$ corresponds to the ith leaf of t, for all $1 \leq i \leq |\varphi(t)|$. In this sense one may consider $\varphi(t)$ to be a labeling of leaves of t: the ith leaf of t is labeled by $\underline{i}(\varphi(t))$.

(4) Clearly iff φ is a code and ψ as in the definition above, then φ is uniquely determined by the pair $(\text{one}_\varphi, \psi)$, where one_φ is the value of φ for the one node tree from T; this pair is referred to as *a defining pair of* φ.

Definition 1.3. Let $\varphi: T \to \Sigma^*$ be a code.

(i) φ is *sibling-consistent* iff for all $x \in \Sigma^+$ and all $y, z \in \text{Rran}(\varphi)$ such that $|x| = 2$, $y = y_1 x y_2$, and $z = z_1 x z_2$ for some $y_1, y_2, z_1, z_2 \in \Sigma^*$ with $|y_1| = i$ and $|z_1| = j$,

$$\text{leaf}_{\varphi^{-1}(y)}(i+1)\text{leaf}_{\varphi^{-1}(y)}(i+2) \text{ is a sibling segment of front}(\varphi^{-1}(y))$$
$$\text{iff} \quad \text{leaf}_{\varphi^{-1}(z)}(j+1)\text{leaf}_{\varphi^{-1}(z)}(j+2) \text{ is a sibling segment of front}(\varphi^{-1}(z)).$$

(ii) φ is *completeness-consistent* iff

(1) for all $x \in \Sigma^+$ and all $y, z \in \text{Rran}(\varphi)$ such that $|x| = n$, $y = y_1 x y_2$, $z = z_1 x z_2$, for some $n \geq 2$, $y_1, y_2, z_1, z_2 \in \Sigma^*$ with $|y_1| = i$ and $|z_1| = j$,

$$\text{leaf}_{\varphi^{-1}(y)}(i+1) \ldots \text{leaf}_{\varphi^{-1}(y)}(i+n) \text{ is a complete segment of front}(\varphi^{-1}(y))$$
$$\text{iff} \quad \text{leaf}_{\varphi^{-1}(z)}(j+1) \ldots \text{leaf}_{\varphi^{-1}(z)}(j+n) \text{ is a complete segment of front}(\varphi^{-1}(z)),$$

and

(2) there is a defining pair $(\text{one}_\varphi, \psi)$ of φ such that for all $a, b \in \Sigma$ and all $n \in \mathbb{N}_+ - \{1\}$, $\psi(a, n) = \psi(b, n)$ implies $a = b$.

(iii) φ is *rich* iff for each $x \in \Sigma^+$ there exist $y, z \in \Sigma^+$ such that $yxz \in \mathrm{Rran}(\varphi)$.
(iv) φ is *strict* iff φ is sibling-consistent, completeness-consistent, and rich.

Clearly we require sibling consistency and completeness consistency to guarantee the unique parsability of words coding trees, and the richness requirement guarantees that a strict code is as close as possible to an onto mapping.

Remark 1.4. Clearly, if φ is a rich code, then φ has exactly one defining pair—we refer to it as *the defining pair of* φ.

2. Basic properties of strict codes

We will prove now some basic properties of strict codes—they will be quite fundamental in the sequel of this paper.

The following classification of the letters from the range of a code is essential in our investigation of strict codes.

Definition 2.1. Let $\varphi: T \to \Sigma^*$ be a code and let $a \in \Sigma$.

(i) a is *left* (*w.r.t.* φ) iff, there exist $x \in \mathrm{Rran}(\varphi)$ and $1 \le i \le |x|$, such that $\underline{i}(x) = a$ and $\mathrm{leaf}_{\varphi^{-1}(x)}(i)$ is a leftmost child.

(ii) a is *right* (*w.r.t.* φ) iff, there exist $x \in \mathrm{Rran}(\varphi)$ and $1 \le i \le |x|$, such that $\underline{i}(x) = a$ and $\mathrm{leaf}_{\varphi^{-1}(x)}(i)$ is a rightmost child.

(iii) a is *middle* (*w.r.t.* φ) iff, there exist $x \in \mathrm{Rran}(\varphi)$ and $1 \le i \le |x|$, such that $\underline{i}(x) = a$ and $\mathrm{leaf}_{\varphi^{-1}(x)}(i)$ is a middle child.

We use L_φ, R_φ, M_φ to denote the sets of left, right, and middle letters (w.r.t. φ), respectively.

Remark 2.2. If a code $\varphi: T \to \Sigma^*$ is sibling-consistent, then there exists a relation $S_\varphi \subseteq \Sigma \times \Sigma$ such that, for each $x \in \mathrm{Rran}(\varphi)$ and for each $1 \le i < |x|$, $\mathrm{leaf}_{\varphi^{-1}(x)}(i)$ $\mathrm{leaf}_{\varphi^{-1}(x)}(i+1)$ is a sibling segment of $\varphi^{-1}(x)$ iff $(\underline{i}(x), \underline{i+1}(x)) \in S_\varphi$.

Lemma 2.3. *For each strict code* $\varphi: T \to \Sigma^*$, $\{L_\varphi, R_\varphi, M_\varphi\}$ *is a partition of* Σ.

Proof. (i) Obviously each of L_φ, R_φ, M_φ is nonempty.

(ii) Since φ is rich, $\Sigma = L_\varphi \cup R_\varphi \cup M_\varphi$.

(iii) Consider $a \in L_\varphi$.

Since $a \in L_\varphi$, $x = x_1 a z x_2 \in \mathrm{Rran}(\varphi)$ and $y = x_1 c x_2 \in \mathrm{Rran}(\varphi)$ for some $x_1, x_2, z \in \Sigma^+$ and $c \in \Sigma$, where $\underline{i+1}(\varphi^{-1}x))$ is a leftmost child for $i = |x_1|$.

If $a \in R_\varphi \cup M_\varphi$, then there exists $b \in \Sigma$ such that $(b, a) \in S_\varphi$. Since φ is rich, $u_1 bcu_2 \in \mathrm{Rran}(\varphi)$ for some $u_1, u_2 \in \Sigma^+$, and consequently $u = u_1 bazu_2 \in \mathrm{Rran}(\varphi)$. However $\underline{k+2}(\varphi^{-1}(u))$ is again a leftmost child for $k = |u_1|$ and so $\underline{k+1}(\varphi^{-1}(u))\underline{k+2}(\varphi^{-1}(u))$

is not a sibling segment of front$(\varphi^{-1}(u))$. Since φ is *sibling-consistent*, this contradicts the fact that $(b,a)\in S_\varphi$. Hence $a\notin R_\varphi\cup M_\varphi$. Consequently, $L_\varphi\cap(R_\varphi\cup M_\varphi)=\emptyset$.

Similarly one can prove that $R_\varphi\cap(L_\varphi\cup M_\varphi)=\emptyset$. Consequently the sets L_φ, R_φ, M_φ are mutually disjoint. The lemma follows from (i), (ii), and (iii). $\square$

Remark 2.4. In our proof of the above lemma we have used the fact that φ is sibling-consistent and rich but not that φ is completeness-consistent. Consequently, it is easily seen that Definition 1.3(ii.1) is redundant. We have included Definition 1.3(ii.1), because it seems to be more natural to define strict codes this way.

Remark 2.5. It is instructive to notice that if $\varphi:T\to\Sigma^*$ is a strict code, then

$$S_\varphi=(L_\varphi\times M_\varphi)\cup(L_\varphi\times R_\varphi)\cup(M_\varphi\times M_\varphi)\cup(M_\varphi\times R_\varphi).$$

This is seen as follows.

(i) By Lemma 2.3,

$$S_\varphi\subseteq L_\varphi\times M_\varphi\cup L_\varphi\times R_\varphi\cup M_\varphi\times M_\varphi\cup M_\varphi\times R_\varphi.$$

(ii) To prove the reverse inclusion we proceed as follows. Consider $L_\varphi\times M_\varphi$. Let $a\in L_\varphi$ and $b\in M_\varphi$. Since φ is rich there exist $y,z\in\Sigma^*$ such that $yabz\in\mathrm{Rran}(\varphi)$. Consequently, by Lemma 2.3, and because φ is sibling-consistent, $(a,b)\in S_\varphi$. Hence $L_\varphi\times M_\varphi\subseteq S_\varphi$. Reasoning analogously, we prove that $L_\varphi\times R_\varphi\subseteq S_\varphi$, $M_\varphi\times M_\varphi\subseteq S_\varphi$, and $M_\varphi\times R_\varphi\subseteq S_\varphi$. Consequently,

$$L_\varphi\times M_\varphi\cup L_\varphi\times R_\varphi\cup M_\varphi\times M_\varphi\cup M_\varphi\times R_\varphi\subseteq S_\varphi.$$

By (i) and (ii), $S_\varphi=L_\varphi\times M_\varphi\cup L_\varphi\times R_\varphi\cup M_\varphi\times M_\varphi\cup M_\varphi\times R_\varphi$.

Definition 2.6. Let $\varphi:T\to\Sigma^*$ be a code, and let $x\in\Sigma^+$, where $|x|=n$. x is *complete* (*w.r.t.* φ) iff, there exist $y\in\mathrm{Rran}(\varphi)$ and $i\in\mathbb{N}_+$, such that $\underline{i}(y)\underline{i+1}(y)\dots\underline{i+n}(y)=x$, and $\mathrm{leaf}_{\varphi^{-1}(y)}(i+1)\dots\mathrm{leaf}_{\varphi^{-1}(y)}(i+n)$ is a complete segment of front$(\varphi^{-1}(y))$.

We use C_φ to denote the set of all complete words (w.r.t. φ) of Σ^*.

Lemma 2.7. *For each strict code* φ, $C_\varphi=L_\varphi M_\varphi^* R_\varphi$.

Proof. (i) Let $x\in C_\varphi$. By Definition 2.1, $x=ayb$, where $a\in L_\varphi$, $y\in M_\varphi^*$, and $b\in R_\varphi$. Hence $C_\varphi\subseteq L_\varphi M_\varphi^* R_\varphi$.

(ii) Let $x\in L_\varphi M_\varphi^* R_\varphi$. Since φ is rich, there exist $y,z\in\mathrm{Ran}(\varphi)$ such that $u=yxz\in\mathrm{Rran}(\varphi)$. Thus, by Lemma 2.3,

$$\mathrm{leaf}_{\varphi^{-1}(u)}(i+1)\mathrm{leaf}_{\varphi^{-1}(u)}(i+2)\dots\mathrm{leaf}_{\varphi^{-1}(u)}(i+n)$$

is a complete segment of front$(\varphi^{-1}(u))$, where $i=|y|$ and $n=|x|$. Hence $x\in C_\varphi$.

The lemma follows from (i) and (ii). $\square$

3. The size of the alphabet of a strict code

It turns out (quite unexpectedly) that if $\varphi : T \to \Sigma^*$ is a strict code, where Σ is finite, then Σ contains exactly (!) 6 letters. This result is proved in this section.

First we need the following lemma.

Lemma 3.1. *Let $\varphi : T \to \Sigma^*$ be a strict code, and let* $(\mathrm{one}_\varphi, \psi)$ *be the defining pair of φ.*

(1) *For all $a \in \Sigma$, $n \in \mathbb{N}_+ - \{1\}$, $|\psi(a,n)| = n$.*

(2) *ψ is a bijection onto C_φ.*

Proof. (1) This follows from the fact that φ is length-preserving and rich.

(2) This follows from the fact that φ is injective, completeness-consistent, and rich. $\square$

Theorem 3.2. *For each strict code $\varphi : T \to \Sigma^*$ where Σ is finite, $\#\Sigma = 6$.*

Proof. Let $(\mathrm{one}_\varphi, \psi)$ be the defining pair of φ, and let ψ_2 and ψ_3 be restrictions of ψ to $\Sigma \times \{2\}$ and $\Sigma \times \{3\}$, respectively.

By Lemma 2.7 and Lemma 3.1, ψ_2 is a bijection onto $L_\varphi \times R_\varphi$ and ψ_3 is a bijection onto $L_\varphi \times M_\varphi \times R_\varphi$. Since $\#(\Sigma \times \{2\}) = \#(\Sigma \times \{3\}) = \#\Sigma$, this implies that

$$\#\Sigma = (\#L_\varphi)(\#R_\varphi) = (\#L_\varphi)(\#M_\varphi)(\#R_\varphi). \tag{1}$$

From (1) it follows that $M_\varphi = 1$. From (1) and Lemma 2.3 it follows that $\#L_\varphi + \#M_\varphi + \#R_\varphi = (\#L_\varphi)(\#R_\varphi)$ and so by the above

$$\#L_\varphi + \#R_\varphi + 1 = (\#L_\varphi)(\#R_\varphi). \tag{2}$$

Consequently,

$$\#L_\varphi > 1 \quad \text{and} \quad \#R_\varphi > 1. \tag{3}$$

By (2) we get $\#R_\varphi + 1 = \#L_\varphi(\#R_\varphi - 1)$, and so, by (3), $\#R_\varphi + 1 \geq 2(\#R_\varphi - 1)$. Hence $\#R_\varphi \leq 3$. Analogously, from (2) and (3) we get $\#L_\varphi \leq 3$. Thus $1 < \#L_\varphi, \#R_\varphi \leq 3$. There are obviously two solutions of (2):

$$\#L_\varphi = 2, \quad \#R_\varphi = 3, \qquad \#L_\varphi = 3, \quad \#R_\varphi = 2.$$

Thus, by (1), $\#\Sigma = 6$, and the theorem holds. $\square$

Remark 3.3. (1) *In the rest of this paper we will consider finite alphabets only.*

(2) According to the proof of the above theorem, for each strict code $\varphi : T \to \Sigma^*$, $\#M_\varphi = 1$ and either $\#L_\varphi = 3$ and $\#R_\varphi = 2$, or $\#L_\varphi = 2$ and $\#R_\varphi = 3$; if the former holds, then we say that φ is of the *type* $(3,2)$ and if the latter holds, then we say that φ is of the *type* $(2,3)$. We will reserve the symbol m_φ to denote the element of M_φ; we will write simply m whenever φ is understood from the context of considerations.

(3) *For technical convenience, in the rest of this paper we will assume that for each strict code φ, $\varphi(t)=m$ for each one-node tree t.*

4. Strict codes and unlimited 0S systems

In this section we will demonstrate that strict codes are of grammatical nature in the sense that they correspond (in a well-defined sense) to a subclass of the class of unlimited 0S systems.

The subclass of unlimited 0S systems corresponding to strict codes is defined as follows.

Definition 4.1. Let $G=(\Sigma, P, \sigma)$ be an unlimited 0S system.

(1) G is *semi-deterministic* iff, for each $a \in \Sigma$ and each $n \geq 2$, there exists precisely one production $a \to x$ in P such that $|x|=n$.

(2) G is *backwards-deterministic* iff for each $x \in \Sigma^+$ there exists at most one $a \in \Sigma$ such that $a \to_P x$.

(3) G is *strict* iff
   (i) G is semi-deterministic,
   (ii) G is backwards-deterministic, and
   (iii) there exists a partition of Σ into three sets L, M, R such that
   - $M=\{\sigma\}$,
   - either $\#L=3$ and $\#R=2$, or $\#L=2$ and $\#R=3$, and
   - for each production $a \to_P x$, $\mathrm{first}(x) \in L$, $\mathrm{last}(x) \in R$, and $\underline{i}(x) \in M$ for all $1<i<|x|$.

The following lemma follows directly from the definition of a strict unlimited 0S system.

Lemma 4.2. *Let $G=(\Sigma, P, \sigma)$ be a strict unlimited 0S system. For each $l \in L$, $r \in R$, $k \geq 0$ there exists $a \in \Sigma$ such that $a \to_P l\sigma^k r$.*

In order to establish the relationship between strict codes and strict unlimited 0S systems we need the following three definitions.

Definition 4.3. Let $\varphi: T \to \Sigma^*$ be a strict code, let $(\mathrm{one}_\varphi, \psi)$ be the defining pair of φ, and let $t \in T$. The *node-labeling of t induced by φ*, denoted $\mathrm{lab}_{t,\varphi}$, is defined as follows:

$$\mathrm{lab}_{t,\varphi}(\mathrm{root}(t)) = \mathrm{one}_\varphi.$$

If $v \in \mathrm{IN}(t)$ is such that $\mathrm{ddes}_t(v)=v_1 \dots v_k$ for $k \geq 2$, $v_i \in \mathrm{ND}(t)$ for all $1 \leq i \leq k$, and $\mathrm{lab}_{t,\varphi}(v)=a$, then $\mathrm{lab}_{t,\varphi}(v_i)=\underline{i}(\psi(a,k))$ for each $1 \leq i \leq k$.

Remark 4.4. The above definition of $\mathrm{lab}_{t,\varphi}$ is in a "top-down fashion". One can also define $\mathrm{lab}_{t,\varphi}$ in a 'bottom-up fashion" as follows.

(i) If $v = \mathrm{leaf}_t(i)$ for some $1 \le i \le |\mathrm{front}(t)|$, then $\mathrm{lab}_{t,\varphi}(v) = \underline{i}(\varphi(t))$.

(ii) If t_1 is a tree and $v \in \mathrm{IN}(t_1)$ is such that $\mathrm{DDES}_{t_1}(v) \subseteq \mathrm{LEAF}(t_1)$, then $\mathrm{lab}_{t,\varphi}(v) = \underline{i}(\varphi(t_2))$, where t_2 results from t_1 by removing $\mathrm{DDES}_{t_1}(v)$ for $1 \le i \le |\mathrm{front}(t_2)|$, and $v = \mathrm{leaf}_{t_2}(i)$.

Hence starting with t by successive removals of a complete subsequence of the frontier of a current tree, one gets a sequence of trees $t_1, \ldots, t_s$ where $t_1 = t$ and t_s is the one-node tree consisting of $\mathrm{root}(t)$. For each $1 \le i \le s$, leafs of t_i are labeled according to $\varphi(t_i)$. Since $\bigcup_{1 \le i \le s} \{\mathrm{LEAF}(t_i)\} = \mathrm{ND}(t)$, in this way we get a labeling of *all* nodes of t.

The bottom-up definition of $\mathrm{lab}_{t,\varphi}$ as above is quite natural, because, to start with (see Remark 1.2) $\varphi(t)$ can be considered to be a labeling of leaves of t.

Definition 4.5. Let $\varphi : T \to \Sigma^*$ be a strict code and let $(\mathrm{one}_\varphi, \psi)$ be the defining pair of φ.

(i) Let $t \in T$ and let $v \in \mathrm{IN}(t)$. The *φ-production of v in t*, denoted $\mathrm{prod}_\varphi(v, t)$, is the production $a \to x$, where $\mathrm{lab}_{t,\varphi}(v) = a$, $\#\mathrm{DDES}_t(v) = k$ and $\psi(a, k) = x$.

(ii) Let $t \in T$. The *set of φ-productions induced by t*, denoted $\mathrm{PROD}_\varphi(t)$, is the set $\{\mathrm{prod}_\varphi(v, t) \colon v \in \mathrm{IN}(t)\}$.

(iii) The *set of φ-productions*, denoted $\mathrm{PROD}(\varphi)$, is the set $\bigcup_{t \in T} \mathrm{PROD}_\varphi(t)$.

(iv) The *unlimited 0S system induced by φ*, denoted $0\mathrm{S}(\varphi)$, is the unlimited 0S system $(\Sigma, \mathrm{PROD}(\varphi), m)$.

Definition 4.6. Let $G = (\Sigma, P, \sigma)$ be a strict unlimited 0S system and let T be a selector (of trees). The *code induced by G (on T)*, denoted $\mathrm{COD}_{G,T}$, is the mapping of T into Σ^* defined as follows. Let $t \in T$.

(i) Let η be the following node-labeling of t:

$\eta(\mathrm{root}(t)) = \sigma$.

If $v \in \mathrm{IN}(t)$ is such that $\mathrm{ddes}_t(v) = v_1 \ldots v_k$ for $k \ge 2$, $v_i \in \mathrm{ND}(t)$ for all $1 \le i \le k$, and $\eta(v) = a$, then $\eta(v_i) = a_i$ for each $1 \le i \le k$, where $a \to a_1 \ldots a_k \in P$.

(ii) $\mathrm{COD}_{G,T}(t) = \eta(d_1) \ldots \eta(d_n)$, where $\mathrm{front}(t) = d_1 \ldots d_n$ with $d_i \in \mathrm{LEAF}(t)$ for each $1 \le i \le n$.

We refer to η as above as the *node-labeling of t induced by G*.

Remark 4.7. Since we have assumed (see Remark 3.3) that, for each strict code φ, $\varphi(t) = m$ for each one-node tree t, *we will assume in the sequel that $\sigma = m$ for each unlimited 0S system $G = (\Sigma, P, \sigma)$.*

Now the relationship between strict codes and strict unlimited 0S systems can be stated in the form of the following result.

Theorem 4.8. (i) *For each strict code $\varphi : T \to \Sigma^*$, $0\mathrm{S}(\varphi)$ is a strict unlimited 0S system, and moreover $\mathrm{COD}_{0\mathrm{S}(\varphi),T} = \varphi$.*

(ii) *For each strict unlimited* 0*S system* G *and each selector set of trees* T, $\mathrm{COD}_{G,T}$ *is a strict code, and moreover* $0\mathrm{S}(\mathrm{COD}_{G,T}) = G$.

Proof. (i) This follows directly from the construction of $0\mathrm{S}(\varphi)$ for a given strict code φ.

(ii) Let $G = (\Sigma, P, m)$ be a strict unlimited 0S system. It is easily seen that for each selector set of trees T, $\mathrm{COD}_{G,T}$ is a code which is sibling-consistent and completeness-consistent.

To prove that $\mathrm{COD}_{G,T}$ is rich is more involved. To this aim we will prove that each $w \in \Sigma^+$ is reachable from m (in G). First however we state an obvious combinatorial observation.

Claim 4.9. *Let* $Z \subseteq X \times Y$ *for some sets* X, Y. *If* $\# Z > \# X + \# Y$, *then* $\# Z \geq 6$.

Lemma 4.10. *Each* $w \in \Sigma^+$ *is reachable from* m.

Proof. We prove the lemma by induction on $|w|$.

Base. Assume that $|w| = 1$, hence $w \in \Sigma$. Let

$$L_w = \{a \in L_G\colon a \text{ is reachable from } w \text{ in } G\}$$

and

$$R_w = \{a \in R_G\colon a \text{ is reachable from } w \text{ in } G\}.$$

Consider now all productions of the type $a \to x$ where $a \in \Sigma$ and $|x| = 2$, hence $x \in L_G R_G$. Since G is semi-deterministic, there is exactly one such production for each letter of Σ; let for each $a \in \Sigma$, $l_a \in L_G$ and $r_a \in R_G$ be such that $a \to_P l_a r_a$.

Clearly if $a \in \Sigma$ is reachable from w then so are l_a and r_a; let

$$Z_m = \{(l_a, r_a)\colon a \in \Sigma \text{ is reachable from } m\}.$$

Since G is strict, if $a \neq b$ then $(l_a, r_a) \neq (l_b, r_b)$, and so $\# Z_m = \# L_m + \# R_m + 1$ (because m is obviously reachable from m). Hence $\# Z_m > \# L_m + \# R_m$, and consequently, by Claim 4.9, $\# Z_m \geq 6$ which implies that each letter of Σ is reachable from m.

Inductive step. Assume that for some $n > 1$, each $u \in \Sigma^+$ such that $|u| < n$ is reachable from m, and consider $aw \in \Sigma^+$ such that $|w| = n$.

We will consider five possible cases for w.

Case 1: $w \in M^+$. Then clearly w is reachable from m.

Case 2: $w = w_1 l m^k r w_2$ *for some* $w_1, w_2 \in \Sigma^*$, $k \geq 0$, $l \in L$ *and* $r \in R$. By Lemma 4.2 there exists $a \in \Sigma$ such that $a \to_P l m^k r$. Hence $w_1 a w_2 \Rightarrow_G w$. But $|w_1 a w_2| < |w|$, and so by the inductive assumption $w_1 a w_2$ is reachable from m which obviously implies that w is reachable from m.

Case 3: $w = m^k r w_2$ *for some* $w_2 \in \Sigma^*$, $k \geq 1$, *and* $r \in R$. Let $a \in \Sigma$ be such that $a \to_P l m^k r$ for some $l \in L$. Then $a w_2 \Rightarrow_G l w$. But $|a w_2| < |w|$, and so by the inductive assumption $a w_2$ is reachable from m. Consequently w is reachable from m.

Case 4: $w = w_1 l m^k$ *for some* $w_1 \in \Sigma^*$, $k \geq 1$, *and* $l \in L$. Analogously to Case 3 we prove that w is reachable from m.

Case 5: *None of the above four cases hold.* It is easy to see that Case 5 holds iff one of the following three cases holds.

(i) $w \in (L \cup M)^+$ and $\mathrm{last}(w) \in L$.

(ii) $w \in (R \cup M)^+$ and $\mathrm{first}(w) \in R$.

(iii) $\mathrm{first}(w) \in R$ and $\mathrm{last}(w) \in L$.

We will consider each of these cases separately but first we need the following claim.

Claim 4.11. (1) *For each $r \in R$, either there exist $y, z \in \Sigma$ such that $y \notin R$ and $y \to_P zr$ or there exist $y_1, z_1, y_2, z_2 \in \Sigma$ such that $y_1 \notin R$, $y_1 \to_P z_1 y_2$, and $y_2 \to_P z_2 r$.*

(2) *For each $l \in L$, either there exist $y, z \in \Sigma$ such that $y \notin L$ and $y \to_P lz$ or there exist $y_1, z_1, y_2, z_2 \in \Sigma$ such that $y_1 \notin L$, $y_1 \to_P y_2 z_2$, and $y_2 \to_P l z_1$.*

Proof. (1) Let $r \in R$. Let

$$A(r) = \{a \in \Sigma\colon\ a \to_P x,\ |x| = 2, \text{ and } \mathrm{last}(x) = r\};$$

by Lemma 4.2, either $\#A(r) = 3$ or $\#A(r) = 2$.

(1.1) If $A(r) - R \neq \emptyset$, then (the "either" case of) the claim holds.

(1.2) Assume that $A(r) \subseteq R$. Let $y_2 \in A(r)$ be such that $y_2 \neq r$, and consider

$$A(y_2) = \{a \in \Sigma\colon\ a \to_P x,\ |x| = 2, \text{ and } \mathrm{last}(x) = y_2\}.$$

If $A(y_2) \subseteq R$, then $\#R \geq 4$ (because $A(r) \subseteq R$); a contradiction. Hence $A(y_2) - R \neq \emptyset$. If we choose now $y_1 \in A(y_2) - R$, then (the "or" case of) the claim holds.

(2) This is proved analogously to (1). $\square$

Case 5(i). Let $w = ul$ for some $u \in (L \cup M)^+$ and $l \in L$. By Claim 4.11, either there exist $y, z \in \Sigma$ such that $y \notin L$ and $uy \Rightarrow_G ulz = wz$, or there exist $y, z_1, z_2 \in \Sigma$ such that $y \notin L$ and $uy \Rightarrow_G^+ ulz_1 z_2 = wz_1 z_2$.

It is easily seen that in both cases uy is in one of the considered Cases 1–4, and hence uy is reachable from m. But then w is reachable from m.

Case 5(ii). Let $w = ru$ for some $r \in R$ and $u \in (R \cup M)^+$. By Claim 4.11, either there exist $y, z \in \Sigma$ such that $y \notin R$ and $yu \Rightarrow_G zru = zw$, or there exist $y, z_1, z_2 \in \Sigma$ such that $y \notin R$ and $yu \Rightarrow_G^+ z_1 z_2 ru = z_1 z_2 w$.

It is easily seen that in both cases yu is one of the Cases 1–4, and hence yu is reachable from m. But then w is reachable from m.

Case 5(iii). Let $w = rul$ for some $u \in \Sigma^*$. By Claim 4.11, either there exist $y, y', z, z' \in \Sigma$ such that $y \notin R$, $y' \notin L$, and

$$yuy' \Rightarrow_G^+ zrulz' = zwz',$$

or there exist $y, z_1, z_2, y', z' \in \Sigma$ such that $y \notin R$, $y' \notin L$, and

$$yuy' \Rightarrow_G^+ z_1 z_2 rulz' = z_1 z_2 wz',$$

or there exist $y, z, y', z_1', z_2' \in \Sigma$ such that $y \notin R$, $y' \notin L$, and

$$yuy' \Rightarrow_G^+ zrulz_1'z_2' = zwz_1'z_2',$$

or there exist $y, z_1, z_2, y', z_1', z_2' \in \Sigma$ such that $y \notin R$, $y' \notin L$, and

$$yuy' \Rightarrow_G^+ z_1 z_2 rulz_1'z_2' = z_1 z_2 wz_1'z_2'.$$

It is easily seen that in each of the above cases yuy' is in one of the previously considered Cases 1-5(ii), and hence yuy' is reachable from m. But then w is reachable from m.

Altogether from Cases 1-5(iii) it follows that the inductive step holds. Consequently the lemma holds. □

Proof of Theorem 4.8 (continued). Since m is the axiom of G and $m \rightarrow_P lmr$ for some $l \in L$ and $r \in R$, the above lemma implies that $\mathrm{COD}_{G,T}$ is rich. Consequently $\mathrm{COD}_{G,T}$ is strict. Since it is easy to see that $0\mathrm{S}(\mathrm{COD}_{G,T}) = G$, (ii) holds.

This completes the proof of the theorem. □

According to the above theorem we may consider strict codes to be strict unlimited 0S systems, and we may consider strict unlimited 0S systems (together with a selector set) to be strict codes. This means in particular that we may *specify* strict codes in the form of strict unlimited 0S systems. Thus typically we will write "let $\varphi = (\Sigma, P, m)$ be a strict code", where (Σ, P, m) is a strict unlimited 0S system (and a selector set is clear from the context of considerations); then $L_\varphi = L$, $M_\varphi = M$, and $R_\varphi = R$, where $\{L, M, R\}$ is the partition of Σ required in the definition of a strict unlimited 0S system. We will assume then that the selector set which is the domain of φ is clear from the context of considerations, or otherwise one can consider an arbitrary but fixed selector set of trees. For this reason we will sometimes write COD_G rather than $\mathrm{COD}_{G,T}$ to denote the code induced by a strict unlimited 0S system.

Let $G = (\Sigma, P, m)$ be a strict unlimited 0S system, where $\{L, M, R\}$ is the partition of G (required by the definition of a strict unlimited 0S system). If $\# L = 3$ and $\# R = 2$, then we say that G is of the *type* (3, 2), and if $\# L = 2$ and $\# R = 3$, then we say that G is of the *type* (2, 3). We consider Σ, L, R to be ordered:

- $L = (l_1, l_2, l_3)$, $R = (r_1, r_2)$, and
- $\Sigma = (l_1, l_2, l_3, r_1, r_2, m)$ if G is of the type (3, 2),

and

- $L = (l_1, l_2)$, $R = (r_1, r_2, r_3)$, and
- $\Sigma = (l_1, l_2, r_1, r_2, r_3, m)$ if G is of the type (2, 3),

where $M = \{m\}$.

Example 4.12. Let $G = (\Sigma, P, m)$ be the unlimited 0S system such that $\Sigma = L \cup R \cup M$, where $L = \{l_1, l_2, l_3\}$, $R = \{r_1, r_2\}$, $M = \{m\}$, and P consists of the following productions:

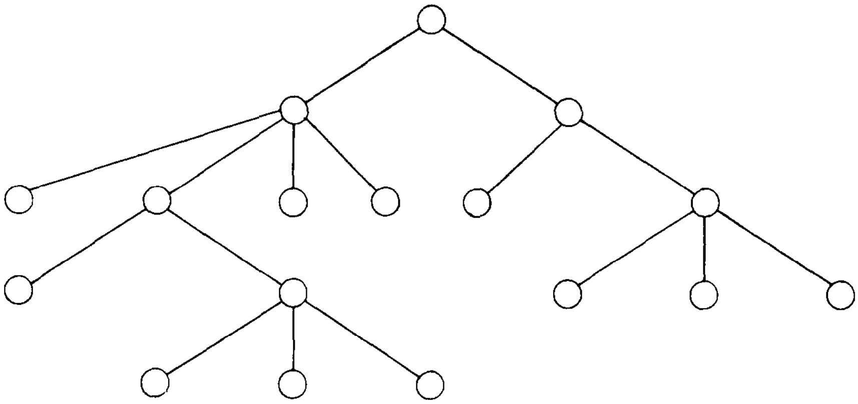

Fig. 1.

- $m \to l_1 m^k r_1$ for all $k \geq 0$,
- $l_1 \to l_1 m^k r_2$ for all $k \geq 0$,
- $l_2 \to l_2 m^k r_2$ for all $k \geq 0$,
- $l_3 \to l_3 m^k r_1$ for all $k \geq 0$,
- $r_1 \to l_2 m^k r_1$ for all $k \geq 0$,
- $r_2 \to l_3 m^k r_2$ for all $k \geq 0$.

Clearly G is a strict unlimited 0S system of the type $(3, 2)$.

Let t_1 be the tree in Fig. 1. Then the node-labeling of t_1 induced by G is shown in Fig. 2. Hence $\varphi(t) = l_1 l_1 l_2 m r_1 m r_2 l_2 l_2 m r_1$, where $\varphi = \mathrm{COD}_G$.

5. Insertive strict codes

In this section we will investigate insertive strict codes. In strict codes of this kind

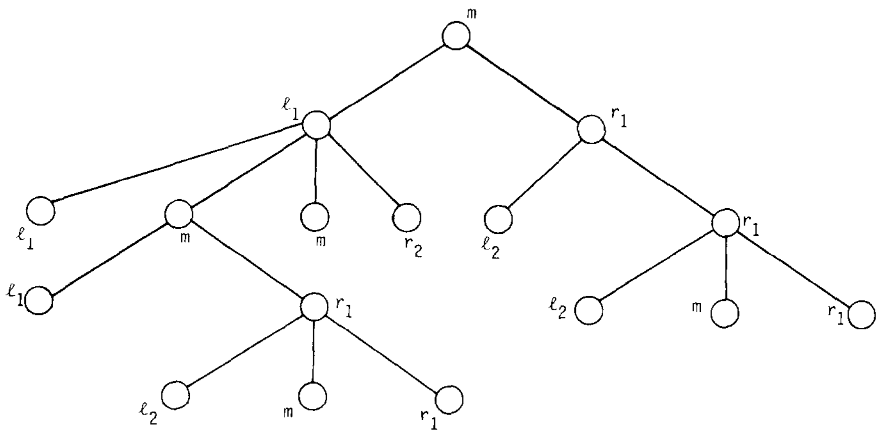

Fig. 2.

a production for a letter b with longer right-hand side β is obtained from a production for b with a shorter right-hand side α by inserting segments into α. The situation like this is quite typical in linguistics—take, e.g., a grammar for a fragment of English where productions for the noun phrase (NP) will be of the form: NP → the car, NP → the nice car, NP → the long nice car, NP → the red long nice car, ...

Formally insertive strict codes are defined as follows.

Definition 5.1. A strict code $\varphi=(\Sigma,P,m)$ is *insertive* iff for each $a\in\Sigma$ and all $\alpha,\beta\in\Sigma^+$ such that $a\rightarrow_P\alpha$ and $a\rightarrow_P\beta$, if $|\alpha|<|\beta|$, then α is a subword of β.

The following technical result follows directly from the definition of an insertive strict code.

Lemma 5.2. *Let $\varphi=(\Sigma,P,m)$ be a strict insertive code. For each $a\in\Sigma$ and for all $x,y\in\Sigma^+$ such that $a\rightarrow_P x$ and $a\rightarrow_P y$,* first$(x)=$first(y) *and* last$(x)=$last(y).

The way of specifying strict codes (through strict unlimited 0S systems) discussed at the end of the last section is especially attractive for strict insertive codes because there is a nice notation for unlimited 0S systems corresponding to strict insertive codes. This notation, that we are going to discuss now is based on Lemma 5.2.

If $G=(\Sigma,P,m)$ is of type (3, 2), then the *tableau of G* is the following tableau:

	r_1	r_2
l_1	a_{11}	a_{12}
l_2	a_{21}	a_{22}
l_3	a_{31}	a_{32}

,

where $\Sigma=\{a_{ij}\colon 1\le i\le 3 \text{ and } 1\le j\le 2\}$, and for all $1\le i\le 3$, $1\le j\le 2$, $a_{ij}\rightarrow_P l_i m^k r_j$ for all $k\ge 0$.

If $G=(\Sigma,P,m)$ is of type (2, 3), then the *tableau of G* is the following tableau:

	r_1	r_2	r_2
l_1	a_{11}	a_{12}	a_{13}
l_2	a_{21}	a_{22}	a_{23}

,

where $\Sigma = \{a_{ij}\colon 1 \leq i \leq 3 \text{ and } 1 \leq j \leq 2\}$, and for all $1 \leq i \leq 3$, $1 \leq j \leq 2$, $a_{ij} \to_P l_i m^k r_j$ for all $k \geq 0$.

We will use Q_G to denote the tableau of G and $Q_G(i,j)$ to denote the $a_{i,j}$ entry of Q_G.

Example 5.3. It is easily seen that the strict unlimited 0S system G from Example 4.12 is insertive. The tableau

$$Q_G = \begin{array}{|c||c|c|} \hline & r_1 & r_2 \\ \hline\hline l_1 & m & l_1 \\ \hline l_2 & r_1 & l_2 \\ \hline l_3 & l_3 & r_2 \\ \hline \end{array}\,.$$

Then $Q_G(1,1) = m$, $Q_G(1,2) = l_1$, $Q_G(2,1) = r_1$, $Q_G(2,2) = l_2$, $Q_3(3,1) = l_3$, and $Q_G(3,2) = r_2$.

The restriction of insertiveness applied to strict codes has quite a dramatic effect: there is only a finite number of strict insertive codes, where we do not distinguish between "isomorphic" codes, i.e., codes that result from each other just by "renaming" letters of the alphabets involved.

Definition 5.4. Codes $\varphi_1 : T \to \Sigma_1^*$ and $\varphi_2 : T \to \Sigma_2^*$ are *isomorphic* iff there exists a bijection $\eta : \Sigma_1 \to \Sigma_2$ such that for each $t \in T$, $\eta(\varphi_1(t)) = \varphi_2(t)$.

Theorem 5.5. *There are only finitely many nonisomorphic insertive strict codes.*

Proof. Assume that $\varphi = (\Sigma, P, m)$ is an insertive strict code.

Since φ is insertive, by Lemma 5.2, if $X \to_\varphi lm^k r$ and $X \to_\varphi l'm^s r'$, where $X \in \Sigma$, $k, s \geq 0$, $l, l' \in L_\varphi$, and $r, r' \in R_\varphi$, then $l = l'$ and $r = r'$. Thus, *all* productions for X in φ are uniquely determined by the pair (l, r) (because φ is semi-deterministic).

Consequently the number of insertive strict codes is not larger than the number of functions from Σ into $L_\varphi \times R_\varphi$.

Since $\#\Sigma = 6$, the theorem holds. $\square$

As a matter of fact we can compute the exact number of insertive strict codes—or more precisely, the exact number of nonisomorphic insertive strict codes. This "nonisomorphy" assumption will hold also in the sequel of this paper in the sense that, *whenever we have a result saying that there are exactly n codes of a given kind, we mean that there are exactly n mutually nonisomorphic codes of a given kind.*

Theorem 5.6. *There are exactly* 120 *insertive strict codes.*

Proof. Since the number of different tableaux of insertive strict codes obviously equals 6! and it can be easily checked that by permuting letters so that left remains left, right remains right, and m remains m, one obtaines 12 different isomorphic tableaux, the number of nonisomorphic codes of a given type ((2, 3) or (3, 2)) equals $6!/12 = 60$. Consequently, the number of insertive strict codes equals $2 \cdot 60 = 120$. □

Remark 5.7 In the sequel of this paper we will give the precise number of insertive strict codes satisfying various additional conditions. The proofs of the corresponding theorems are organized in such a way that:

(1) while counting the number of codes of a given sort, the proof of the corresponding theorem gives the precise form of all codes of this kind,

(2) when we prove that there are at most n codes of a given kind, the construction used in the proof is done in such a way that it is clear that all constructed codes are mutually nonisomorphic (the proof of the latter fact will be left to the reader), hence we will conclude that there are *precisely* n codes of a given kind.

6. Strongly recursive insertive strict codes

Often in considerations concerning grammars one wants to get a clear cut division between recursive and nonrecursive letters, and so one applies transformations leading to grammars with "as many as possible" recursive letters (because nonrecursive letters lead often to tedious "exceptions" in reasoning about grammars). A typical desired situation then is that a recursive letter must be directly recursive and a nonrecursive letter must lead directly (i.e. in one step) to a recursive letter.

Based on this motivation we will introduce now strongly recursive insertive strict codes.

Definition 6.1. An insertive strict code $\varphi = (\Sigma, P, m)$ is *strongly recursive* iff for each production $a \rightarrow_P lm^k r$, where $l \in L_\varphi$, $r \in R_\varphi$, and $k \geq 0$, if $a \neq l$ and $a \neq r$, then l and r are directly recursive.

Again, we can compute the exact number of strongly recursive strict codes.

Theorem 6.2. *There are exactly* 8 *strongly recursive insertive strict codes.*

Proof. Consider an arbitrary strongly recursive insertive strict code φ.

We will compute the number of different forms that Q_φ may have.

Case 1: *Assume that φ is of the type* (3, 2). Again, we may assume that $Q_\varphi(1, 1) = m$. Hence each production for m is of the form $m \Rightarrow_\varphi l_1 m^k r_1$, for some $k \geq 0$, and so, because φ is strongly recursive, both l_1 and r_1 must be directly recursive. Consequently $Q_\varphi(1,2)=l_1$ and $Q_\varphi(2,1)=r_1$.

We have three possibilities for r_2: either $Q_\varphi(2,2)=r_2$ or $Q_\varphi(3,1)=r_2$ or $Q_\varphi(3,2)=r_2$.

(i) $Q_\varphi(2,2)=r_2$. Then $\{Q_\varphi(3,1), Q_\varphi(3,2)\}=\{l_2,l_3\}$, and so we have two possibilities for Q_φ.

(ii) $Q_\varphi(3,1)=r_2$. Then each production for r_2 is of the form $r_2 \Rightarrow_\varphi l_3 m^k r_1$, for some $k\geq 0$, and so, because φ is strongly recursive, l_3 must be directly recursive. Consequently, $Q_\varphi(3,2)=l_3$ and consequently $Q_\varphi(2,2)=l_2$.

(iii) $Q_\varphi(3,2)=r_2$. If $Q_\varphi(2,2)=l_3$ and $Q_\varphi(3,1)=l_2$, then each production for l_3 is of the form $l_3 \Rightarrow_\varphi l_2 m^k r_2$, for some $k\geq 0$, and each production for l_2 is of the form $l_2 \Rightarrow_\varphi l_3 m^k r_1$, for some $k\geq 0$. This however contradicts the fact that φ is strongly recursive.

The remaining case is $Q_\varphi(2,2)=l_2$ and $Q_\varphi(3,1)=l_3$.

Thus we have two possibilities in Case 1(i), one possibility in Case 1(ii) and one possibility in Case 1(iii); altogether four possibilities.

Case 2: *Assuming that φ is of the type* (2,3), by analogous reasoning we arrive at four possibilities for Q_φ.

Thus altogether we have eight possibilities for Q_φ, and consequently there are exactly eight strongly recursive insertive strict codes.

Since it is easily seen that all eight codes we have constructed above are mutually nonisomorphic, the theorem holds. □

7. Dependent insertive strict codes

We will consider now the notion of a dependent (insertive) strict code. It formalizes the classical notion of dependency in parenthesis notation for derivation trees of context-free grammar. E.g., if we consider the context-free grammar G_0 with productions $S\to(S)$, $S\to SS$, $S\to\Lambda$ generating well-formed parenthesis expressions, then the number of right and left parentheses in each sentential form will be equal (i.e., will "cancel" each other). Hence there exists a fixed vector $\delta=(-1,+1,0)$ such that for all sentential forms x, y in G_0 such that x derives y, $\delta\pi(x)=\delta\pi(y)$, where $\pi(z)$ for a word z is the Parikh column vector of z, and the fixed order of the alphabet is $(,),S$. This idea is now carried over to the framework of dependent strict codes.

Recall that we assume the alphabet Σ of a strict code φ to be ordered, i.e. $\Sigma=(l_1,l_2,l_3,r_1,r_2,m)$ if φ is of the (3,2) type, and $\Sigma=(l_1,l_2,r_1,r_2,r_3,m)$ if φ is of the (2,3) type. Consequently, given a vector $\delta=(e_1,\dots,e_6)\in\mathbb{N}^6$ we consider $e_1,\dots,e_6$ to be the values of δ for $l_1,l_2,\dots,m$ respectively, i.e., $\delta(l_1)=e_1$, $\delta(l_2)=e_2$, $\delta(l_3)=e_3$, $\delta(r_1)=e_4$, $\delta(r_2)=e_5$, and $\delta(m)=e_6$, if φ is of the (3,2) type, and $\delta(l_1)=e_1$, $\delta(l_2)=e_2$, $\delta(r_1)=e_3$, $\delta(r_2)=e_4$, $\delta(r_3)=e_5$, and $\delta(m)=e_6$, if φ is of the (2,3) type.

Definition 7.1. A strict code $\varphi = (\Sigma, P, m)$ is *dependent* iff there exists a nonzero vector $\delta \in \mathbb{N}^6$ such that, for all $x, y \in \Sigma^+$, if $x \Rightarrow_\varphi y$, then $\delta\pi(x) = \delta\pi(y)$. Each nonzero vector δ satisfying the above is called a *dependency (vector) for* φ.

Let φ be a strict code of the (3,2) type with

$Q_\varphi =$

	r_1	r_2
l_1	a_{11}	a_{12}
l_2	a_{21}	a_{22}
l_3	a_{31}	a_{32}

.

For a vector $\delta \in \mathbb{N}^6$, the *φ-tableau of δ* is the following tableau:

	$\delta(r_1)$	$\delta(r_2)$
$\delta(l_1)$	$\delta(a_{11})$	$\delta(a_{12})$
$\delta(l_2)$	$\delta(a_{21})$	$\delta(a_{22})$
$\delta(l_3)$	$\delta(a_{31})$	$\delta(a_{32})$

.

Similarly, if φ is a strict code of the (2,3) type with

$Q_\varphi =$

	r_1	r_2	r_3
l_1	a_{11}	a_{12}	a_{13}
l_2	a_{21}	a_{22}	a_{23}

and $\delta \in \mathbb{N}^6$, then the *φ-tableau of δ* is the following tableau:

	$\delta(r_1)$	$\delta(r_2)$	$\delta(r_3)$
$\delta(l_1)$	$\delta(a_{11})$	$\delta(a_{12})$	$\delta(a_{13})$
$\delta(l_2)$	$\delta(a_{21})$	$\delta(a_{22})$	$\delta(a_{23})$

We will use $Q_{\varphi,\delta}$ to denote the φ-tableau of δ.

If, for a strict code φ and a vector $\delta \in \mathbb{N}^6$, it holds that $\delta(a_{ij}) = \delta(l_i) + \delta(r_j)$, hence $Q_{\varphi,\delta}(a_{ij}) = \delta(l_i) + \delta(r_j)$, then we say that $Q_{\varphi,\delta}$ is *additive*.

Lemma 7.2. *Let $\varphi = (\Sigma, P, m)$ be an insertive strict code.A nonzero vector $\delta \in \mathbb{N}^6$ is a dependency for φ iff $\delta(m) = 0$ and $Q_{\varphi,\delta}$ is additive.*

Proof. ($\Rightarrow$) Assume that a nonzero vector δ is a dependency for φ. Since φ is strict insertive there exist $l \in L_\varphi$, and $r \in R_\varphi$ such that $m \to_\varphi lm^k r$ for each $k \geq 0$. Since δ is a dependency vector for φ, we get then $\delta(m) = \delta(l) + k\delta(m) + \delta(r)$ for each $k \geq 0$. Consequently $\delta(m) = 0$.

Consider an arbitrary entry a_{ij} of Q_φ. Then $a_{ij} \to_\varphi l_i m^k r_j$ for each $k \geq 0$, and consequently $\delta(a_{ij}) = \delta(l_i) + k\delta(m) + \delta(r_j)$ for each $k \geq 0$. Since $\delta(m) = 0$, $\delta(a_{ij}) = \delta(l_i) + \delta(r_j)$. Consequently, $Q_{\varphi,\delta}$ is additive.

($\Leftarrow$) Assume that $\delta \in \mathbb{N}^6$ is a nonzero vector such that $\delta(m) = 0$ and $Q_{\varphi,\delta}$ is additive.

Consider an arbitrary entry a_{ij} of Q_φ. Then $a_{ij} \to_\varphi l_i m^k r_j$ for each $k \geq 0$, and so for arbitrary $x, y \in \Sigma^*$, and arbitrary $k \geq 0$, $x a_{ij} y \Rightarrow_\varphi x l_i m^k r_j y$. Hence $\zeta = \pi(x l_i m^k r_j y) - \pi(x a_{ij} y)$ is such that one of the following possibilities holds:

(1) $\zeta(a_{ij}) = -1$, $\zeta(l_i) = +1$, $\zeta(r_j) = +1$, $\zeta(m) = k$, and $\zeta(u) = 0$ otherwise, or
(2) $\zeta(a_{ij}) = 0$, $\zeta(l_i) = 0$, $\zeta(r_j) = +1$, $\zeta(m) = k$, and $\zeta(u) = 0$ otherwise, or
(3) $\zeta(a_{ij}) = 0$, $\zeta(l_i) = +1$, $\zeta(r_j) = 0$, $\zeta(m) = k$, and $\zeta(u) = 0$ otherwise, or
(4) $\zeta(a_{ij}) = k - 1$, $\zeta(l_i) = 0$, $\zeta(r_j) = +1$, $\zeta(m) = k$, and $\zeta(u) = 0$ otherwise.

Consequently,

$$\delta\zeta = \delta(a_{ij})\,\zeta(a_{ij}) + \delta(l_i)\,\zeta(l_i) + \delta(r_j)\,\zeta(r_j) + \delta(m)\,\zeta(m)$$

and so:

- if (1), then $\delta\zeta = 0$, because $\delta\zeta = -\delta(a_{ij}) + \delta(l_i) + \delta(r_j)$ and $Q_{\varphi,\delta}$ is additive,
- if (2), then $\delta\zeta = \delta(r_j) = 0$, because $\delta(l_i) + \delta(r_j) = \delta(a_{ij}) = \delta(l_i)$,
- if (3), then $\delta\zeta = \delta(l_i) = 0$, because $\delta(l_i) + \delta(r_j) = \delta(a_{ij}) = \delta(r_j)$,
- if (4), then $\delta\zeta = (k-1)\delta(a_{ij}) + \delta(l_i) + \delta(r_j) = 0$, because $\delta(a_{ij}) = \delta(m) = 0$.

Thus δ is a dependency for φ. $\square$

The above result (and its proof) yields the following corollary that will be useful in the sequel.

Corollary 7.3. *Let $\varphi = (\Sigma, P, m)$ be an insertive strict code and let δ be a dependency for φ.*

(i) *For all $l \in L_\varphi$, if $l \to_\varphi lm^k r$ for some $k \geq 0$, $r \in R_\varphi$, then $\delta(r) = 0$.*
(ii) *For all $r \in R_\varphi$, if $r \to_\varphi lm^k r$ for some $k \geq 0$, $l \in L_\varphi$, then $\delta(l) = 0$.*

We will demonstrate now that there are exactly 12 dependent insertive strict codes.

Let $SI_{(3,2)}$ be the set of those insertive strict codes that are isomorphic with either one of the following insertive strict codes:

	r_1	r_2
l_1	m	x_1
l_2	r_1	u_1
l_3	u_2	x_2

or with one of the following insertive strict codes:

	r_1	r_2
l_1	r_2	x_1
l_2	r_1	m
l_3	l_2	x_2

,

where $\{x_1, x_2\} = \{l_1, l_3\}$ and $\{u_1, u_2\} = \{r_2, l_2\}$.

Let $SI_{(2,3)}$ be the set of those insertive strict codes that are isomorphic with either one of the following insertive strict codes:

	r_1	r_2	r_3
l_1	m	l_1	u_2
l_2	x_1	u_1	x_2

or with one of the following insertive strict codes:

	r_1	r_2	r_3
l_1	l_2	l_1	r_2
l_2	x_1	m	x_2

,

where $\{x_1, x_2\} = \{r_1, r_3\}$ and $\{u_1, u_2\} = \{r_2, l_2\}$.
Let $\mathrm{SI} = \mathrm{SI}_{(3,2)} \cup \mathrm{SI}_{(2,3)}$.

Lemma 7.4. *An insertive strict code φ is dependent iff $\varphi \in SI$.*

Proof. ($\Rightarrow$) Assume that φ is dependent. We consider two cases:

Case 1. Assume that φ is of the type (3,2). Let

$$Q_0 = \begin{array}{c||c|c} & -1 & 0 \\ \hline\hline 1 & 0 & 1 \\ \hline 0 & -1 & 0 \\ \hline 1 & 0 & 1 \end{array}.$$

Claim 7.5. *If δ is a dependency for φ, then (modulo a multiplicative constant and a permutation of rows and columns) $Q_{\varphi,\delta}$ equals Q_0.*

Proof. Let δ be a dependency vector of φ and let

$$Q_{\varphi,\delta} = \begin{array}{c||c|c} & y_1 & y_2 \\ \hline\hline x_1 & b_{11} & b_{12} \\ \hline x_2 & b_{21} & b_{22} \\ \hline x_3 & b_{31} & b_{32} \end{array}.$$

We split the proof in five parts (I)–(V).

(I) For no $1 \le i \le 3$, $1 \le j \le 2$, $x_i y_j > 0$.

This is seen as follows. Assume that for some $1 \le i_0 \le 3$, $1 \le j_0 \le 2$, $x_{i_0} > 0$ and $y_{j_0} > 0$. Let $b_{i_1 j_1} > 0$ be the maximal entry of $Q_{\varphi,\delta}$ in the sense that for all $1 \le i \le 3$ and $1 \le j \le 2$, $b_{ij} \le b_{i_1 j_1}$.

By Lemma 7.2, $Q_{\varphi,\delta}$ is additive and so, if $b_{i_1 j_1} = l_{i_2}$ for some $1 \le i_2 \le 3$ then $b_{i_2 j_0} = x_{i_2} + y_{j_0} > b_{i_1 j_1}$; a contradiction, and if $b_{i_1 j_1} = r_{j_2}$ for some $1 \le j_2 \le 2$, then $b_{i_0 j_2} = x_{i_0} + y_{j_2} > b_{i_1 j_1}$; a contradiction. Consequently there does not exist $1 \le i_0 \le 3$ and $1 \le j_0 \le 2$, such that $x_{i_0} > 0$ and $y_{j_0} > 0$.

Similarly we prove that there does not exist $1 \le i_0 \le 3$ and $1 \le j_0 \le 2$, such that $x_{i_0} < 0$ and $y_{j_0} < 0$.

(II) There exist $1 \le i_0 \le 3$ and $1 \le j_0 \le 2$, such that $x_{i_0} \ne 0$ and $y_{j_0} \ne 0$.

This is seen as follows. Assume to the contrary that, for all $1 \le i \le 3$, $x_i = 0$. Since $\delta(m) = 0$ and $Q_{\varphi,\delta}$ is additive, this implies that also $y_1 = 0$ or $y_2 = 0$. Hence from the values $\delta(l_1)$, $\delta(l_2)$, $\delta(l_3)$, $\delta(r_1)$, $\delta(r_2)$ at most one differs from zero, which implies that they are all equal to zero, contradicting the fact that δ is a dependency vector. Consequently there exists $1 \le i_0 \le 3$ such that $x_{i_0} \ne 0$.

Similarly we prove that there exists $1 \le j_0 \le 2$, such that $y_{j_0} \ne 0$.

(III) For all $1 \le i \le 3$, $1 \le j \le 2$, either $x_i \ge 0$ and $y_j \le 0$ or $x_j \le 0$ and $y_j \ge 0$.

This follows directly from (I) and (II).

(IV) There exist $1 \le i_0 \le 3$ and $1 \le j_0 \le 2$, such that $x_{i_0} = 0$ and $y_{j_0} = 0$.

This is seen as follows. Consider a $1 \le j_1 \le 2$ such that $r_{j_1} \Rightarrow_\varphi^+ l_{k_1} l_{k_2} r_{j_1}$ for some $1 \le k_1, k_2 \le 2$; clearly such a j_1 exists. Since δ is a dependency for φ, $\delta(r_{j_1}) = \delta(l_{k_1}) + \delta(l_{k_2}) + \delta(r_{j_1})$, and so from (III) it follows that $x_{k_1} = x_{k_2} = 0$.

Similarly we prove that there exists $1 \le j_0 \le 2$ such that $y_{j_0} = 0$.

(V) $Q_{\varphi,\delta}$ is as follows:

	-1	0
x_1	b_{11}	x_1
0	-1	0
x_3	b_{31}	x_3

for some x_1, x_3, b_{11}, b_{31}.

This is seen as follows. By (IV) at least one $x \in \{x_1, x_2, x_3\}$ and exactly one $y \in \{y_1, y_2\}$ equal 0. Since we consider the form of $Q_{\varphi,\delta}$ modulo permutation of rows and columns we set $x_2 = 0$ and $y_2 = 0$. Since by Lemma 7.2, $Q_{\varphi,\delta}$ is additive, $b_{12} = x_1$, $b_{22} = 0$, and $b_{32} = x_3$. Since we consider the form of $Q_{\varphi,\delta}$ modulo a multiplicative constant, we set $y_1 = -1$, and consequently (because $Q_{\varphi,\delta}$ is additive) $b_{21} = -1$.

Now we conclude the proof of the claim as follows. $Q_{\varphi,\delta}$ must have the form stated in (V). Since

$$\{x_1, x_2, x_3, y_1, y_2, m\} = \{b_{11}, b_{12}, b_{21}, b_{22}, b_{31}, b_{32}\} \quad \text{and} \quad \delta(m) = 0,$$

$\{b_{11}, b_{31}\} = \{x_2, y_2\} = \{0, 0\}$, and so $b_{11} = b_{31} = 0$. Since, by Lemma 7.2, $Q_{\varphi,\delta}$ is additive, $x_1 = 1$ and $x_3 = 1$. Thus $Q_{\varphi,\delta}$ equals Q_0. Hence the claim holds. □

Proof of Lemma 7.4 (continued). Consider now the form of Q_φ. We have:

$$m \in \{Q_\varphi(1,1), Q_\varphi(2,2), Q_\varphi(3,1)\}.$$

This follows directly from Claim 7.5 and Lemma 7.2 (because $\delta(m)=0$).

So we will consider separately two cases.

Case 1.1. $m \in \{Q_\varphi(1,1), Q_\varphi(3,1)\}$. Since $\delta(r_1)=-1$, $Q_\varphi(2,1)=r_1$ and so

$$\{x_1, x_3\} = \{Q_\varphi(1,2), Q_\varphi(3,2)\}$$

and $\{x_2, y_2\} = \{Q_\varphi(2,2), Q_\varphi(3,1)\}$ (if $m = Q_\varphi(1,1)$), or $\{Q_\varphi(1,1), Q_\varphi(2,2)\}$ (if $m = Q_\varphi(3,1)$).

Hence in this case $Q_\varphi \in \mathrm{SI}_{(3,2)}$.

Case 1.2. $m = Q_\varphi(2,2)$. Since $\delta(r_1)=-1$, $Q_\varphi(2,1)=r_1$, and reasoning similarly as above we conclude that also in this case $Q_\varphi \in \mathrm{SI}_{(3,2)}$. Consequently $Q_\varphi \in \mathrm{SI}$.

Case 2. Similarly, assuming that φ is of the type $(2,3)$ we prove that $Q_\varphi \in \mathrm{SI}$. Consequently $\varphi \in \mathrm{SI}$.

($\Leftarrow$) We assume that $\varphi \in \mathrm{SI}$ and again we consider two cases:

Case 2.1. Assume that $\varphi \in \mathrm{SI}_{(3,2)}$. We notice that then the vector δ such that $\delta(l_1)=1$, $\delta(l_2)=0$, $\delta(l_3)=1$, $\delta(r_1)=-1$, $\delta(r_2)=\delta(m)=0$ is such that $\delta(m)=0$ and $Q_{\varphi,\delta}$ is additive. Consequently, by Lemma 7.2, δ is a dependency for φ and so φ is dependent.

Case 2.2. Similarly we prove that if $\varphi \in \mathrm{SI}_{(2,3)}$ then φ is dependent. Consequently φ is dependent.

Now Lemma 7.4 is proven. □

Theorem 7.6. *There are exactly* 12 *dependent insertive strict codes.*

Proof. Follows directly from Lemma 7.2 and from an easy observation that no two codes in SI are isomorphic. □

8. Dependent strong recursive insertive strict codes

We will demonstrate now that requiring strong recursivity decreases the number of dependent insertive strict codes to 4.

Theorem 8.1. *There are exactly* 4 *dependent strongly recursive insertive strict codes.*

Proof. Let φ be a strongly recursive insertive strict code.

Case 1. Assume that φ is of the $(3,2)$ type. From the proof of Theorem 6.2 (see also Remark 5.7) we know that $\varphi \in \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ where

$Q_{\varphi_1} =$

	r_1	r_2
l_1	m	l_1
l_2	r_1	r_2
l_3	l_2	l_3

,

$Q_{\varphi_2} =$

	r_1	r_2
l_1	m	l_1
l_2	r_1	r_2
l_3	l_3	l_2

,

$Q_{\varphi_3} =$

	r_1	r_2
l_1	m	l_1
l_2	r_1	l_2
l_3	l_3	r_2

,

$Q_{\varphi_4} =$

	r_1	r_2
l_1	m	l_1
l_2	r_1	l_2
l_3	r_2	l_3

.

(i) Now consider the vector $\delta = (1, 0, 1, -1, 0, 0)$. Hence

$$Q_{\varphi_1,\delta} = \begin{array}{|c||c|c|} \hline & -1 & 0 \\ \hline\hline 1 & 0 & 1 \\ \hline 0 & -1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array},$$

and

$$Q_{\varphi_4,\delta} = \begin{array}{|c||c|c|} \hline & -1 & 0 \\ \hline\hline 1 & 0 & 1 \\ \hline 0 & -1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}.$$

Thus, by Lemma 7.2, δ is a dependency for both φ_1 and φ_4.

(ii) Consider now φ_2 and assume that δ_2 is a dependency for φ_2. By Lemma 7.2, $\delta_2(m)=0$. Corollary 7.3 (together with the form of φ_2) implies that $\delta_2(r_2)=0$, $\delta_2(l_2)=0$, and $\delta_2(r_1)=0$. Since $\delta_2(r_1)=0$, and $\delta_2(m)=0$, by Lemma 7.2, $\delta_2(l_1)=0$. Since $\delta_2(r_2)=0$, and $\delta_2(l_2)=0$, by Lemma 7.2, $\delta_2(l_3)=0$. Consequently δ_2 is the zero vector in $\mathbb{N}^6$; a contradiction.

Thus φ_2 is not dependent.

(iii) Consider now φ_3 and assume that δ_3 is a dependency vector for φ_3. By Lemma 7.2, $\delta_3(m)=0$. Corollary 7.3 (together with the form of φ_3) implies that $\delta_3(r_2)=0$, $\delta_3(r_1)=0$, $\delta_3(l_2)=0$, and $\delta_3(l_3)=0$. Since $\delta_3(m)=0$ and $\delta_3(r_1)=0$, by Lemma 7.2, $\delta_3(l_1)=0$. Consequently δ_3 is the zero vector in $\mathbb{N}^6$, a contradiction.

Thus φ_3 is not dependent.

From (i)–(iii) it follows that only two strongly recursive insertive strict codes of the type $(3,2)$ are dependent.

Case 2. Analogously we prove that only two strict insertive strongly recursive codes of the type $(2,3)$ are dependent.

Altogether there are exactly four dependent strongly recursive insertive strict codes. □

9. Discussion

As we have indicated in the introduction, the role of a grammar is to code derivation trees of texts. We have introduced the notion of a strict code and shown that they correspond to grammars (strict unlimited 0S systems) where coding of derivation trees is done using six syntactic categories.

In general, if we want to code m objects by strings of length n, then the cardinality t of the alphabet Σ used (to form strings) must satisfy the inequality:

$$m \leq t^n.$$

In our paper we are interested in the problem of coding (directed ordered chain-free) trees by length-preserving codes, hence m in the above corresponds to the number of ordered chain-free trees with n leaves.

It can be proved (e.g., using estimations from [3]) that the number of trees with n leaves is bigger than 5^n and hence the alphabet Σ used must have at least $t=6$ letters. In this sense our Theorem 3.2 says that strict codes are "informationally optimal"—they use precisely 6 letters.

In this paper we have introduced a classification of strict codes. Further insight into this classification as well as applications of strict codes to parsing and to two-dimensional text representation will be considered in the sequel of this paper.

We would like to conclude this paper with the following remark.

Remark 9.1. Clearly there are other ways of setting up the notion of a strict code. E.g., one could start with the notion of a *marked code* which would be a mapping $\varphi: T \to \Sigma^*$ satisfying the following conditions

(1) *length-preserving* (Definition 1.1(i)),
(2) *local* (Definition 1.1(ii)),
(3) "injectiveness" of ψ (Definition 1.3(ii.2)), and
(4) "left and right consistency" (in the sense that left and right letters are consistent in all words of Rran(φ)—hence $\{L_\varphi, M_\varphi, R_\varphi\}$ is a partition).

Then one could prove that for a marked code $\varphi: T \to \Sigma^*$ (with finite Σ) it must be that $\#\Sigma \geq 6$. Then one can show the existence of "minimal marked codes"—i.e., codes for which $\#\Sigma = 6$. A way to ensure minimality is to impose the richness condition (Definition 1.3(iii)). In this way a *strict code* is introduced as a *rich marked code*.

Both ways of introducing the notion of a strict code formalize our (somewhat different) intuitions of what a "good way" of coding trees is.

Acknowledgement

The authors are indebted to J. Engelfriet, T. Harju and K. Salomaa for very useful comments on the previous version of this paper, and to Ms. M. van der Nat for the expert typing of this paper.

References

[1] P. Beckmann, The Structure of Language—A New Approach (Golem Press, Boulder, NC, 1972).
[2] N. Chomsky, Syntactic Structures (Mouton, The Hague, 1957).
[3] A. Ehrenfeucht, J. Haemer and D. Haussler, Quasi-monotonic sequences: algorithms and applications, SIAM J. Algorithms Discrete Math. 8 (1987) 410–429.
[4] A. Ehrenfeucht and G. Rozenberg, The sequence equivalence problem is decidable for 0S systems, J. ACM 27 (1980) 656–663.
[5] C.C. Fries, The Structure of English: An Introduction to the Construction of English Sentences (Harcourt Brase, London, 1957).
[6] G. Gentzen, Untersuchungen über das Logische Schliessen, Math. Z. 39 (1934) 176–210, 405–431.
[7] Z. Harris, Methods in Structural Linguistics (Univ. of Chicago Press, Chicago, IL, 1951).
[8] S. Jaskowski, On the rules of suppositions in formal logic, Studia Logica 1 (1934) 27–30.
[9] W. Kintsch, The Representation of Meaning in Memory (Erlbaum, Hillsdale, NJ, 1974).