Grammars Working on Layered Strings

Paolo Bottoni, [*] Giancarlo Mauri, [†] Piero Mussio, [‡]
Gheorghe Păun [§]

Abstract

We consider first an operation with strings and languages suggested by superposed windows on the computer screen (as well as by cryptographic systems of Richelieu type): we assume that the strings contain usual symbols as well as a transparent symbol. Superposing two strings (justified to left), we produce a new string consisting of the symbols observable from above. This operation is investigated as an abstract operation on strings, then it is used in building a variant of grammar systems with the component grammars working on the layers of an array of strings. Each grammar can rewrite only symbols in its layer which are observable from above. The language generated in this way consists of strings of the observable symbols, produced at the end of a derivation. The power of several variants of these generative mechanisms is investigated for the case of two layered strings. When a matrix-like control on the work of the component grammars is considered, then a characterization of recursively enumerable languages is obtained.

1 Introduction

Recent work in the study of algebraic features of pictorial languages has shown how a natural operation between pictures is that of superposition. This operation is informally understood as the placing of one image (which can contain some *transparent* symbols) above another, so that only pixels in the second image which appear immediately below transparent pixels in the first image are *observable* and can contribute to the resulting image [2]. Actually, superposition of structures on the screen is continuously used in visual interactive systems. Consider for example windowing systems in which windows are allowed to overlap, as in the MacOs^{TM} and Windows^{TM} operating systems. In these cases what is actually observable by a

[*] Department of Computer Science, University of Rome "La Sapienza", Via Salaria 113, 00198 Roma, Italy, e-mail:bottoni@dsi.uniroma1.

[†] Department of Computer Science, University of Milano, Via Comelico 39, 20135 Milano, Italy, e-mail:mauri@dsi.unimi.it.

[‡] Department of Electronics for Automation, University of Brescia, Via Branze 38, 25123 Brescia, Italy, e-mail:mussio@bsing.ing.unibs.it.

[§] Institute of Mathematics of the Romanian Academy, PO Box 1 – 764, 70700 Bucureşti, Romania, e-mail:gpaun@imar.ro.

user results from the spatial relations among the windows currently on the screen. If one considers that each window contains a sentence in a language, the screen defines a sentence which results from the superposition of several such partial sentences. The study of the characteristics of such languages, and hence of the properties of the superposition operation becomes interesting if one wants to avoid situations which may generate disorientation in the user [3].

The idea of employing transparency as a language-defining tool predates the appearance of computers of a long time. It can be traced back to the Richelieu code, an elementary form of cryptography in which a message is embedded in a random text, and can be recovered by superposing the whole text by an opaque sheet with holes in it. The letters reading through the holes form the original message [1]. On the other hand, non-transparent pixels are those on which the user can interact, hence are those from which the transformations of the current sentence may originate. This suggests a notion of control in which the activation of symbols in a rewriting process is possible only if such symbols are *visible*, i.e., observable and non-transparent. Such a form of control appears to be very frequent in natural and artificial systems, in all cases in which layers may be defined and the development of a phenomenon depends on the characteristics both of the layer at which it occurs and of the above layers (e.g., rain permeability of a terrain, the growth of films in VLSI chips, competition for light in chlorophylliac plants, radiology methods based on differences in tissue absorbing properties, etc.). The notion of layer is also useful in defining systems in which different agents may cooperate to define the evolution of a substrate. Different systems may have different roles and be allowed to operate only on parts left undefined by prominent agents. A similar model was also at the basis of the proposal of Parallel Communicating Grammar Systems, where query symbols are used by a master agent to mark the places where other, specified, agents may contribute strings of unknown length [9]. In the notion of layered grammar proposed in this paper, instead, agents can operate autonomously and independently, but their contribution to the final result is limited to fill transparent "holes" left by an agent in an upper position. Which agent will contribute in which zones of the substrate to the final result cannot be established a priori, depending on the ability of an agent to synchronize its activity with that of agents at an upper layer.

Hence, two problems appear worth studying related to the notion of layer. First, what are the properties of the superposition operation and which properties and families of languages it preserves. Second, which is the expressive power of layers as a modelling tool. We start this study by considering closure properties of the operation and by exploring the generative power of context-free grammars with only two layers. It results that by such simple tools, more precisely by using layered right-linear grammars with a specific form of synchronization, we can provide a characterization of recursively enumerable languages. Further study can be performed on the characteristics of systems with more than two layers and on an extension of the notion of transparency (for instance by considering symbols with different levels of opacity).

2 Formal Language Theory Prerequisites

In this section we recall only a few notions, notations, and results needed below. Further details can be found in [11] and in references therein. For an alphabet V, we denote by V^* the free monoid generated by V; λ is the empty string, $|x|$ is the length of $x \in V^*$. The language of the non-empty strings over V, that is $V^* - \{\lambda\}$, is denoted by V^+. A morphism $h : V^* \longrightarrow U^*$ such that $h(a) \in U \cup \{\lambda\}$ for all $a \in V$ is called a *weak coding*; it is called a *coding* when $h(a) \in U$ for all $a \in V$ and a *projection* when $h(a) \in \{a, \lambda\}$ for all $a \in V$. A string $x \in V^*$ can be seen as a mapping $x : \{1, 2, \ldots, \infty\} \longrightarrow V \cup \{\#\}$, where $\#$ is the *blank* symbol, with the following properties: there is $i \geq 1$ such that $x(i) = \#$; moreover, if $x(j) = \#$, then $x(j+1) = \#, j \geq 1$ (this means that $x(j) \in V$ for $1 \leq j \leq |x|$ and $x(j) = \#$ otherwise). Sometimes, we shall use below such an interpretation of a string. A Chomsky grammar is a construct $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and P is a finite subset of $(N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$. The elements of N are called nonterminal symbols, those of T are called terminal symbols, S is the axiom, and the elements $(u, v) \in P$, written in the form $u \to v$, are called rewriting rules (for short, rules). The language generated by G is denoted $L(G)$. When all rules in P are of the form $A \to x, A \in N, x \in (N \cup T)^*$, then the grammar is said to be *context-free*; it is *linear* if x above contains at most one nonterminal symbol. A context-free grammar whose rules are of the forms $A \to xB, A \to x$, for $A, B \in N, x \in T^*$, is said to be *right-linear*; when x above is a single symbol in T, then the grammar is said to be *regular*. We denote by $FIN, REG, LIN, CF, CS, RE$ the families of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages, respectively.

For $x, y \in V^*$, we define the *shuffle* of x, y by

$$\begin{aligned} x \sqcup\!\sqcup\, y &= \{x_1 y_1 x_2 y_2 \ldots x_n y_n \mid n \geq 1, x = x_1 x_2 \ldots x_n, \\ &\quad\; y = y_1 y_2 \ldots y_n, x_i, y_i \in V^*, 1 \leq i \leq n\}. \end{aligned}$$

The concatenation has priority over shuffle: $L_1 L_2 \sqcup\!\sqcup\, L_3$ should be read as $(L_1 L_2) \sqcup\!\sqcup\, L_3$.

Convention. Two language generating mechanisms are considered equivalent if they generate languages which differ at most by the empty string.

3 The Operation of Superposition

In what follows, t is always a special symbol, which is considered *transparent*. Let V be an alphabet. Two strings $x, y \in (V \cup \{t\})^*$ can be *superposed*, producing a third string z which is constructed as follows:

1. the shortest of x, y is completed to the right with occurrences of t such that two strings of the same length are obtained; let us denote by x', y' the strings obtained in this way (at least one of them is not modified);

2. then, for $i \geq 1$, we set

$$z(i) = \begin{cases} x'(i), & \text{if } x'(i) \neq t, \\ y'(i), & \text{otherwise.} \end{cases}$$

(Clearly, $|z| = \max\{|x|, |y|\}$.) We denote the string z by $x \diamond y$ and we say that it is obtained by the *superposition* of x and y. We can imagine that $x \diamond y$ is obtained by writing the strings x, y one over the other, with x above, aligned to left, and looking from above to the "layered string" obtained in this way. Through transparent symbols we observe the corresponding symbols of y (maybe also the transparent symbol), otherwise we observe the symbols of x.

For **example**, for

$$x = abttabt, \quad y = tabttbbabt,$$

we obtain

$$x \diamond y = ab\underline{bt}ab\underline{babt},$$

where the underlined symbols are taken from y.

Obviously, the operation $\diamond$ is associative, but not commutative.

For $L_1, L_2 \subseteq (V \cup \{t\})^*$ we put

$$L_1 \diamond L_2 = \{x \diamond y \mid x \in L_1, y \in L_2\}.$$

Note that *every* string from L_1 is superposed with *every* string in L_2, irrespective of their length, because of the completion with occurrences of t of the shortest string in each pair.

We now investigate the operation of superposition as an abstract operation on languages, relating it to other operations on languages. In this way, the closure properties of families in the Chomsky hierarchy under this new operation will be settled.

Since our goal is the study of the closure properties of language families in the Chomsky hierarchy, all families we consider below are supposed to contain at least all regular languages (but this is not stated again and again).

Lemma 1. *If F is a family of languages which is closed under superposition, shuffle with regular languages, weak codings, and intersection with regular languages, then it is closed under intersection.*

Proof. Consider two languages $L_1, L_2 \subseteq V^*$. Denote $V' = \{a' \mid a \in V\}$, where a' is a new symbol associated with $a \in V$, and define the weak coding $h_1 : (V \cup V')^* \longrightarrow V^*$ by $h_1(a') = \lambda, a' \in V'$, and $h_1(a) = a, a \in V$, and the coding $h_2 : (V \cup \{t\})^* \longrightarrow (V' \cup \{t\})^*$ by $h_2(a) = a', a \in V$, and $h_2(t) = t$. Consider also the regular languages

$$R_1 = \{at \mid a \in V\}^*,$$
$$R_2 = \{ta \mid a \in V\}^*,$$
$$R_3 = \{aa' \mid a \in V\}^*.$$

Then the following equality holds

$$L_1 \cap L_2 = h_1((L_1' \diamond L_2') \cap R_3),$$

where

$$L_1' = (L_1 ⧢ t^*) \cap R_1,$$
$$L_2' = h_2((L_2 ⧢ t^*) \cap R_2).$$

(The intersection with R_1, R_2 forces the shuffling with t^* to pair symbols of strings in L_1, L_2 with symbols t, then the intersection with R_3 selects only those strings coresponding to equal strings from L_1, L_2. Finally, the weak coding h_1 discards symbols in V'.) In view of the closure properties of the family F, it follows that F is closed under intersection. □

Corollary 1. *The families LIN, CF are not closed under superposition.*

Lemma 2. *If F is a family of languages which is closed under intersection with regular languages, shuffle, codings, and inverse morphisms, then F is closed under superposition.*

Proof. Consider two languages $L_1, L_2 \subseteq (V \cup \{t\})^*$, denote as above $V' = \{a' \mid a \in V\}$, consider the new symbols c, c', t' and the new alphabet

$$\begin{aligned} U &= \{[ab'], [at'], [ac'], [ca'], [ta'] \mid a, b \in V\} \\ &\cup \{[tt'], [ct'], [tc']\}, \end{aligned}$$

and define the following morphisms:

$h_1 : U^* \longrightarrow V^*$,
  by $h_1([ab']) = h_1([at']) = h_1([ta']) = h_1([ca']) = h_1([ac']) = a$, for $a, b \in V$,
  and $h_1([tt']) = h_1([ct']) = h_1([tc']) = t$,
$h_2 : U^* \longrightarrow (V \cup V' \cup \{t, t', c, c'\})^*$,
  by $h_2([\alpha\beta]) = \alpha\beta$, for $[\alpha\beta] \in U$,
$h_3 : (V \cup \{t\})^* \longrightarrow (V' \cup \{t'\})^*$,
  by $h_3(a) = a'$, $a \in V$, and $h_3(t) = t'$.

With the regular language

$$R = [(V \cup \{t\})(V' \cup \{t'\})]^*[((V \cup \{t\})\{c'\})^* \cup (\{c\}(V' \cup \{t'\}))^*]$$

we obtain

$$L_1 \diamond L_2 = h_1(h_2^{-1}((L_1\{c\}^* ⧢ h_3(L_2)\{c'\}^*) \cap R)).$$

The intersection with R selects, from the strings produced by shuffling, those strings which are obtained by interleaving the symbols of strings in L_1, L_2, maybe prolonged with occurrences of c, c', but not containing superfluous occurrences of c, c';

then, h_2^{-1} replaces blocks $\alpha\beta$ by symbols $[\alpha\beta]$ which are "interpreted" by h_1 in the same way as when constructing the superposition of the two strings in L_1, L_2. The use of symbols c, c' prevents the addition of superfluous symbols t in the right end of strings. From the closure properties of F, we get $L_1 \diamond L_2 \in F$.

Note that, by our convention above, $\{c\}^* \in REG \subseteq F$ and $L\{c\}^* = (L ш \{c\}^*) \cap V^*\{c\}^*$, that is, F is closed under concatenation with $\{c\}^*$. (In fact, by the closure properties of F, we can get the closure of F under concatenation with any regular language, but we do not need here this general property.) □

Corollary 2. *The families REG, CS, RE are closed under the superposition.*

Corollary 3. *If F is a family of languages which is closed under intersection with regular languages, shuffle with regular languages, codings, and inverse morphisms, then F is closed under superposition with regular languages, in the following sense: if $L_1 \in F, L_2 \in REG$, then $L_1 \diamond L_2 \in F$ and $L_2 \diamond L_1 \in F$.*

Proof. If one of the languages L_1, L_2 is regular, then in the proof above we use a shuffle with regular languages. □

The families LIN, CF are closed under shuffle with regular languages, hence they are closed under superposition with regular languages.

4 Grammars Working on Layered Strings

For two strings $x, y \in (V \cup \{t\})^*$ we denote by $[x, y]$ the two-level sequence obtained by placing x over y, justified to left and completing the shortest string with occurrences of t; $[x, y]$ is called a *layered string*. Given a layered string $[x, y]$, any symbol $x(i) \in V$ is said to be *observable*. A symbol $y(i) \in V \cup \{t\}$ is *observable* if and only if $x(i) = t$. If $y(i) \in V$, then $y(i)$ is also *visible*. Therefore, the observable symbols in $[x, y]$ correspond to the symbols appearing in the string $x \diamond y$ defined as in the previous section. In the sequel, for simplicity, we will only use the term observable. We can now define the main notion investigated in this paper.

A *layered grammar* is a construct

$$\gamma = (N, T, t, (S_1, P_1), (S_2, P_2)),$$

where N, T are disjoint alphabets, t is a special symbol not in $N \cup T$, $S_1, S_2 \in N$, and P_1, P_2 are finite sets of context-free rules over $N \cup T \cup \{t\}$ (t is considered a terminal symbol); N is the *nonterminal* alphabet, T is the *terminal* alphabet, t is the transparent symbol, (S_i, P_i) are the *components* of the grammar; S_i is the *axiom* and P_i is the set of rules of component $i, i = 1, 2$. We also say that (S_1, P_1) is the upper component and (S_2, P_2) is the lower component of γ.

For $x_1, x_2, y_1, y_2 \in (N \cup T \cup \{t\})^*$ we write $[x_1, x_2] \Longrightarrow_s [y_1, y_2]$ if and only if both the following conditions hold:

(1) $x_1 = x_1' A x_1'', y_1 = x_1' u_1 x_1'', A \to u_1 \in P_1,$ or

$x_1 = y_1 \in (T \cup \{t\})^*$,

(2) $x_2 = x_2' A x_2'', y_2 = x_2' u_2 x_2'', A \to u_2 \in P_2, A$ is observable, or

$x_2 = y_2$ and no nonterminal symbol is observable in x_2.

The relation $\Longrightarrow_s$ is called a *synchronized derivation* in γ: each component has to use a rule, except the case when the corresponding string is terminal, or, in the case of the lower component, no nonterminal is observable. Therefore, only the observable symbols of the lower level are active and can be rewriten. A variant of the relation $\Longrightarrow_s$ is $\Longrightarrow_{ns}$, the *non-synchronized* derivation step: for $x_1, x_2, y_1, y_2 \in (N \cup T \cup \{t\})^*$ we write $[x_1, x_2] \Longrightarrow_{ns} [y_1, y_2]$ if and only if one of the following cases holds:

(1) $x_1 = x_1' A x_1'', y_1 = x_1' u_1 x_1'', A \to u_1 \in P_1$, and

$x_2 = y_2$,

(2) $x_1 = y_1$ and

$x_2 = x_2' A x_2'', y_2 = x_2' u_2 x_2'', A \to u_2 \in P_2, A$ is observable.

Only one of the two components works, rewriting any symbol in the upper level and an observable symbol in the lower level. By rewriting first in the lower level and then in the upper level we can simulate in this way a synchronized derivation. Thus, $\Longrightarrow_s$ is indeed a restricted version of $\Longrightarrow_{ns}$. For any $\Longrightarrow_\alpha, \alpha \in \{s, ns\}$ we denote by $\Longrightarrow_\alpha^*$ its reflexive and transitive closure. For each relation $\Longrightarrow_\alpha$ we can consider two languages associated to γ, by considering two stop conditions for a derivation: when no nonterminal is allowed in the last layered string, we get

$$L_{t,\alpha}(\gamma) = \{z_1 \diamond z_2 \in (T \cup \{t\})^* \mid [S_1, S_2] \Longrightarrow_\alpha^* [z_1, z_2],\ z_1, z_2 \in (T \cup \{t\})^*\}.$$

When we allow finishing the derivation with non-observable nonterminal symbols in the lower level, then we get

$$\begin{aligned} L_{nt,\alpha}(\gamma) &= \{z_1 \diamond z_2 \in (T \cup \{t\})^* \mid [S_1, S_2] \Longrightarrow_\alpha^* [z_1, z_2],\ z_1 \in (T \cup \{t\})^*, \\ &\quad z_2 \in (N \cup T \cup \{t\})^*, \text{ but no nonterminal in } z_2 \text{ is observable}\}. \end{aligned}$$

In both cases, $\alpha \in \{s, ns\}$. In this way, we associate with γ four languages, $L_{t,s}(\gamma)$, $L_{t,ns}(\gamma)$, $L_{nt,s}(\gamma)$, $L_{nt,ns}(\gamma)$. We denote by $TSL(X), TNSL(X), NTSL(X), NTNSL(X)$ the families of languages of these types generated by layered grammars with rules of type X; for X we consider here REG, RL, LIN, CF (regular, right-linear, linear, context-free, respectively); we do not distinguish between grammars allowed to contain λ-rules and λ-free grammars (that is, we allow erasing rules). In the proofs in the following section we shall present several specific layered grammars, hence we do not give here examples.

5 Preliminary Results

Directly from the definitions, we obtain

Lemma 3. *$YL(REG) \subseteq YL(RL) \subseteq YL(LIN) \subseteq YL(CF)$, for all $Y \in \{TS, TNS, NTS, NTNS\}$.*

By adding to each set P_1, P_2 rules $A \to A$ for each $A \in N$, we can simulate a non-synchronized derivation by a synchronized one, hence we get

Lemma 4. *$TNSL(X) \subseteq TSL(X), NTNSL(X) \subseteq NTSL(X)$, for each $X \in \{RL, LIN, CF\}$.*

The use of chain rules is important here. We shall see below that layered grammars with regular rules generate only regular languages, both in the synchronized and the non-synchronized modes, hence the result above holds in this indirect way also for the regular case.

Lemma 5. *$X \subseteq YL(X)$, for all $X \in \{REG, RL, LIN, CF\}$, $Y \in \{TS, TNS, NTS, NTNS\}$.*

Proof. For a usual grammar $G = (N, T, S, P)$ we construct the layered grammar

$$\gamma = (N \cup \{S_1\}, T, t, (S_1, \{S_1 \to t\}), (S, P)).$$

Because the upper component generates only one transparent symbol and then stops, we obviously obtain $L_{t,\alpha}(\gamma) = L_{nt,\alpha}(\gamma) = L(G), \alpha \in \{s, ns\}$. □

There is a close relation between the work of layered grammars and the superposition operation (between the superposition of context-free languages and languages generated by a layered grammar). The next result illustrates this.

Theorem 1. *For all context-free languages $L_1, L_2 \subseteq (T \cup \{t\})^*$ there are a weak coding h, a regular language R, and a layered context-free grammar γ such that*

$$L_1 \diamond L_2 = h(L_{t,ns}(\gamma) \cap R).$$

Proof. Let $G_i = (N_i, T \cup \{t\}, S_i, P_i)$ be two context-free grammars with $N_1 \cap N_2 = \emptyset$ such that $L_i = L(G_i), i = 1, 2$. Let S_1', S_2' and A be new nonterminals and let c_1, c_2 be new terminals. We construct the layered grammar

$$\gamma = (N', T', t, (S_1', P_1'), (S_2', P_2')),$$

with

$$\begin{aligned} N' &= N_1 \cup N_2 \cup \{S_1', S_2', A\}, \\ T' &= T \cup \{c_1, c_2, t'\}, \\ P_1' &= \{S_1' \to tS_1',\ S_1' \to c_1 S_1\} \cup P_1, \\ P_2' &= \{S_2' \to AS_2,\ A \to t'A,\ A \to c_2 t\} \cup P_2. \end{aligned}$$

Consider also the weak coding $h : T'^* \longrightarrow (T \cup \{t\})^*$ defined by $h(a) = a, a \in T$, $h(t') = h(c_1) = h(c_2) = \lambda$, as well as the regular language

$$R = \{t'\}^* \{c_2 c_1\} (T \cup \{t\})^*.$$

We obtain the equality $L_1 \diamond L_2 = h(L_{t,ns}(\gamma) \cap R)$. Indeed, the derivations in γ leading to strings in R are equivalent (modulo the order of some steps) to a derivation of the following form:

$$\begin{aligned}
[S_1', S_2'] &\Longrightarrow_{ns}^* [t^n S_1', S_2'] \Longrightarrow_{ns} [t^n c_1 S_1, S_2'], \text{ for } n \geq 1,\\
&\Longrightarrow_{ns}^* [t^n c_1 w_1, S_2'], \text{ for } w_1 \in L_1,\\
&\Longrightarrow_{ns} [t^n c_1 w_1, AS_2] \Longrightarrow_{ns}^* [t^n c_1 w_1, Aw_2], \text{ for } w_2 \in L_2, |w_2| \leq n-1,\\
&\Longrightarrow_{ns}^* [t^n c_1 w_1, t'^{n-1} c_2 t w_2].
\end{aligned}$$

Clearly, $(t^n c_1 w_1) \diamond (t'^{n-1} c_2 w_2) = t'^{n-1} c_2 c_1 (w_1 \diamond w_2)$. With the weak coding h, we obtain the string $w_1 \diamond w_2$. □

Corollary 4. *$TNSL(CF) - CF \neq \emptyset$, $TSL(CF) - CF \neq \emptyset$.*

Proof. The family CF is not closed under the operation $\diamond$, but it is closed under intersection with regular languages and arbitrary morphisms. Moreover, $TNSL(CF) \subseteq TSL(CF)$ (Lemma 4). □

We shall strenghten this result in the following section.

6 The Power of Layered Grammars

First, we show that all families of languages generated by layered grammars with linear rules can generate non-context-free languages. (Note that this does not follow from the proof of Theorem 1, because we use the non-linear rule $S_2' \to AS_2$ in P_2'.)

Theorem 2. *$YL(LIN) - CF \neq \emptyset$, $Y \in \{TS, TNS, NTS, NTNS\}$.*

Proof. Let us consider the following layered grammar

$$\begin{aligned}
\gamma &= (\{S_1, S_1', S_1'', S_2, S_2'\}, \{a, b, c, d, e\}, t, (S_1, P_1), (S_2, P_2)),\\
P_1 &= \{S_1 \to tS_1',\ S_1' \to S_1''d,\ S_1'' \to aS_1''t,\ S_1'' \to et\},\\
P_2 &= \{S_2 \to ttS_2,\ S_2 \to tS_2,\ S_2 \to dS_2',\ S_2' \to bS_2'c,\ S_2' \to te\}.
\end{aligned}$$

A non-synchronized derivation in γ is equivalent (modulo the order of some steps) to a derivation of the following form. After using the rule $S_1 \to tS_1'$ in the upper layer, the rule $S_2 \to ttS_2$ must be used in the lower one in order to "get free" this component:

$$[S_1, S_2] \Longrightarrow_{ns} [tS_1', S_2] \Longrightarrow_{ns} [tS_1', ttS_2].$$

In the second component we can derive freely:

$$\begin{aligned}
[tS_1', ttS_2] &\Longrightarrow_{ns}^* [tS_1', t^m S_2], \text{ for } m \geq 2,\\
&\Longrightarrow_{ns} [tS_1', t^m dS_2'] \Longrightarrow_{ns}^* [tS_1', t^m db^n S_2' c^n], \text{ for } n \geq 0,\\
&\Longrightarrow_{ns} [tS_1', t^m db^n tec^n].
\end{aligned}$$

At any time, we can also start deriving in the upper component, where a string $ta^rett^rd, r \geq 0$, can be produced. If we consider also the regular language

$$R = ta^+edb^+dec^+,$$

and we look only for strings in $L_{t,ns}(\gamma) \cap R$, then we have to stop by producing a layered string

$$[ta^rett^rd, t^mdb^ntec^n],$$

such that $r + 2 = m, r = n$. Therefore, the obtained string is

$$(ta^rett^rd) \diamond (t^mdb^ntec^n) = ta^nedb^ndec^n.$$

Consequently,

$$L_{t,ns}(\gamma) \cap R = \{ta^nedb^ndec^n \mid n \geq 0\},$$

which is not a context-free language. Because the intersection with R asks for having the terminal rule $S_2' \to te$ used, we have $L_{t,ns}(\gamma) \cap R = L_{nt,ns}(\gamma) \cap R$. Therefore, $TNSL(LIN) - CF \neq \emptyset, NTNSL(LIN) - CF \neq \emptyset$. With Lemma 4, also $TSL(LIN) - CF \neq \emptyset, NTSL(LIN) - CF \neq \emptyset$. □

We consider now the case of right-linear layered grammars. When they work in the synchronized mode, they can generate non-regular languages. (However, we do not know whether or not also non-context-free languages can be generated in this way.)

Theorem 3. *$TSL(RL) - REG \neq \emptyset$, $NTSL(RL) - REG \neq \emptyset$.*

Proof. Consider the layered grammar

$$\begin{aligned}
\gamma &= (\{S_1, S_1', S_2\}, \{a, b, c, d\}, t, (S_1, P_1), (S_2, P_2)),\\
P_1 &= \{S_1 \to tS_1,\ S_1 \to b^2tb^2S_1',\ S_1' \to b^2tb^2S_1',\ S_1' \to b^2tc\},\\
P_2 &= \{S_2 \to a^3S_2, S_2 \to a^3d\},
\end{aligned}$$

as well as the regular language

$$R = a^+(bbabb)^+bbdc.$$

In order that a terminal synchronized derivation in γ will produce a string in R, it has to proceed as follows. After the first step, S_2 is observable; it runs faster than S_1 as long as the upper component uses the rule $S_1 \to tS_1$; after starting to use another rule, the upper component runs faster and eventually it catches up the lower one. This must indeed happen when looking for strings in R, because we must obtain a string containing both the symbol c (introduced by the upper component) and the symbol d (introduced by the lower component), in neighboring positions. Thus, we have to follow derivations of the form

$$\begin{aligned}
[S_1, S_2] &\Longrightarrow_s [tS_1, S_2] \Longrightarrow_s^* [tt^nS_1, a^{3n}S_2], \text{ for } n \geq 0,\\
&\Longrightarrow_s [tt^nbbtbbS_1', a^{3n}a^3S_2]\\
&\Longrightarrow_s^* [tt^n(bbtbb)^mS_1', a^{3n}a^{3m}S_2], \text{ for } m \geq 0.
\end{aligned}$$

The second component can stop in any moment, but the upper one must derive until catching up (in order to produce the substring dc). Therefore, we have to produce a layered string of the form

$$\Longrightarrow_s^* [tt^n(bbtbb)^{m+p}bbtc, a^{3(n+m)}d],\ p \geq 0,$$

such that

$$n + 1 + 5(m + p) + 3 = 3(n + m) + 1 \tag{1}$$

(in order to have d adjacent to c). This implies that $2m + 5p + 3 = 2n$; because $p \geq 0$, we obtain

$$2m + 3 \leq 2n. \tag{2}$$

Consequently, we get

$$L_{t,s}(\gamma) \cap R = \{a^{n+1}(bbabb)^m bbdc \mid n \geq 0, 2n \geq 2m + 3\}.$$

The intersection $L_{t,s}(\gamma) \cap R$ is infinite. More precisely, this intersection contains strings $a^{n+1}(bbabb)^m bbdc$ with arbitrarily large m: consider the values

$$n = 3s + 3,\ p = 1,\ m = 3s - 1,$$

for any integer $s \geq 1$. Conditions (1), (2) are fulfilled, hence the strings $a^{3s+3+1}(bbabb)^{3s-1}bbdc$ are in $L_{t,s}(\gamma) \cap R$ for all $s \geq 1$. This means that $L_{t,s}(\gamma) \cap R$ (hence $L_{t,s}(\gamma)$, too) is not a regular language: by pumping a substring of the suffix $(bbabb)^{3s-1}bbdc$ we get strings not in $L_{t,s}(\gamma) \cap R$.

Because the occurrences symbols c, d are introduced by the terminal rules of P_1, P_2, respectively, it follows that the strings in R are obtained by terminal derivations, that is, $L_{t,s}(\gamma) \cap R = L_{nt,s}(\gamma) \cap R$. Therefore, $L_{nt,s}(\gamma) \cap R \notin REG$, which implies that $L_{nt,s}(\gamma) \notin REG$ either. □

The synchronization is essential in the result above:

Theorem 4. *$TNSL(RL) \subseteq REG, NTNSL(RL) \subseteq REG$.*

Proof. Consider the layered grammar $\gamma = (N, T, t, (S_1, P_1), (S_2, P_2))$ and examine the derivations performed in the terminal non-synchronized mode. In each layer there is one nonterminal only, which moves from left to right. If the lower nonterminal is behind the upper one, then the derivation in the lower level depends on the transparent symbols in the upper layer (the lower level is blocked if its nonterminal is placed under a terminal symbol in the upper level). If the lower nonterminal goes ahead the upper one, then no restriction on the derivation is imposed. The two nonterminals can work freely, but the result is obtained by superposition. Because of the non-synchronization, we can apply a rule in any of the two layers. Therefore, (1) *only the relative position of the two nonterminals is important*, and (2) *as long as both nonterminals are present we can keep them at a bounded distance.* The distance between the two nonterminals can be bounded by $2m$, where

$$m = \max\{|u| \mid A \to u \in P_1 \cup P_2\}.$$

(If the distance between the two nonterminals is smaller than m, then we rewrite the nonterminal which is ahead; if the distance is between m and $2m$, then we rewrite the nonterminal which is behind. Note that we cannot bound this distance by m by always rewriting the nonterminal which is behind: if the upper nonterminal is behind, by rewriting it we can cover the lower nonterminal, which can modify the language, because the derivation is not synchonized. By first rewriting the lower nonterminal, we go at a distance at most $2m$, and the lower nonterminal continues to be observable after one more rule used in the upper layer.)

Therefore, the work of γ can be controlled by a "window" of length at most $2m$. Initially, this is $[S_1, S_2]$, it possibly grows to some $[w_1A_1, A_2]$ or $[A_1, w_2A_2]$, with $|w_1|, |w_2|$, terminal strings of length at most $2m - 1$, and continues in this way until ending the derivation in one component; after that, only one nonterminal is enough in order to control the derivation.

We construct a right-linear grammar

$$G = (N', T \cup \{t\}, (S_1, S_2), P)$$

with

$$\begin{aligned} N' = \{ & (w_1, w_2) \mid w_1 \in T^*(N \cup \{\lambda\}), w_2 \in T^*(N \cup \{\lambda\}), \\ & 0 \leq |w_1|, |w_2| \leq 2m, \text{ at least one of } w_1, w_2 \text{ contains a nonterminal} \\ & \text{and at most one of them contains terminal symbols}\} \end{aligned}$$

and with the following rules.

We distinguish several cases, according to the type (terminal or nonterminal) of the strings in the two layers and to the length of these strings.

1. Both layers contain a nonterminal symbol and these symbols are superposed, that is, the sentential form ends with (A, B) and only A can be rewritten.
   (a) If $A \to uC$ is in $P_1, u \in (T \cup \{t\})^*$, then
   $$(A, B) \to (uC, B)$$
   is a production in P.
   (b) If $A \to u$ is in $P_1, u \in (T \cup \{t\})^*$, then
   $$(A, B) \to (u, B)$$
   is a production in P.

2. Both layers contain a nonterminal and the nonterminal in the second layer is behind, that is, the sentential form ends with a nonterminal (uA, B). The string u must be of the form $u = tu'$ in order to be able to apply a rule in the second layer. Note that it is not necessary to rewrite in the upper layer before rewriting the lower string and getting a longer string in the lower layer (the rewriting in the upper layer does not depend on the contents of the lower layer). Thus, we do not consider rules for modifying the symbol A in (uA, B).

(a) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(uA, B) \to u \diamond v_1(A, v_2D),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(b) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(uA, B) \to u_1 \diamond v(u_2A, D),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

(c) If $B \to v$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(uA, B) \to u \diamond v_1(A, v_2),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(d) If $B \to v$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(uA, B) \to u_1 \diamond v(u_2A, \lambda),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

3. Both layers contain a nonterminal and the nonterminal in the first layer is behind, that is, the sentential form ends with a nonterminal (A, vB), with $v \in (T \cup \{t\})^+$. (This time we need rules for modifying both symbols A and B – see again the mode of obtaining the bound $2m$ as a maximal distance between nonterminals in the two layers.)

(a) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(A, vB) \to u \diamond v_1(C, v_2B),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(b) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(A, vB) \to u_1 \diamond v(u_2C, B),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

(c) If $B \to wD$ is in P_2, and $|vwD| \leq 2m$, then

$$(A, vB) \to (A, vwD)$$

is a production in P.

(d) If $A \to u$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(A, vB) \to u \diamond v(\lambda, B)$$

is a production in P.

(e) If $A \to u$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(A, vB) \to u_1 \diamond v(u_2, B),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

(f) If $B \to w$ is in P_2 and $|vw| \leq 2m$, then

$$(A, vB) \to (A, vw)$$

is a production in P.

4. Only the first layer contains a nonterminal, that is, the sentential form ends with a nonterminal (A, v).

(a) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(A, v) \to u \diamond v_1(C, v_2),$$

for $v = v_1v_2$ with $|u| = |v_1|$, is a production in P.

(b) If $A \to uC$ is in P_1, $u \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(A, v) \to u \diamond v(C, \lambda)$$

is a production in P.

(c) If $A \to u$ is in P_1 and $u \in (T \cup \{t\})^*$, then

$$(A, v) \to u \diamond v$$

is a production in P.

5. Only the second layer contains a nonterminal, that is, the sentential form ends with a nonterminal (u, B); the string u must be of the form $u = tu'$ or $u = \lambda$ in order to be able to apply a rule in the second layer.

(a) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| \leq |v|$, then

$$(u, B) \to u \diamond v(\lambda, D)$$

is a production in P.

(b) If $B \to vD$ is in P_2, $v \in (T \cup \{t\})^*$, and $|u| > |v|$, then

$$(u, B) \to u_1 \diamond v(u_2, D),$$

for $u = u_1u_2$ with $|u_1| = |v|$, is a production in P.

(c) If $B \to v$ is in P_2 and $v \in (T \cup \{t\})^*$, then

$$(u, B) \to u \diamond v$$

is a production of G.

This completes the construction.

One can easily check that we obtain $L(G) = L_{t,ns}(\gamma)$, which proves the inclusion $TNSL(RL) \subseteq REG$.

For the non-terminal case we can finish the derivation with the lower nonterminal placed under a symbol different from t. The necessary modifications are left to the reader. □

Theorems 3 cannot be improved by replacing RL by REG: even synchronized, layered grammars with regular components can generate only regular languages.

Theorem 5. $TSL(REG) \subseteq REG, NTSL(REG) \subseteq REG.$

Proof. Let us consider a layered grammar $\gamma = (N, T, t, (S_1, P_1), (S_2, P_2))$ with the sets P_1, P_2 containing regular rules. Because the work of γ is synchronized, whenever the nonterminal in the lower level is observable, it has to be rewritten. At the first step, when the upper level uses a rule different from $A \to tB, A \to t$ (that is a terminal in T is introduced), this will cover the nonterminal in the lower level, hence it is no longer rewritten. Thus, the derivation of γ starts by a number of steps of using rules of the form $A \to tB$ (this number may be zero). Synchronously, the lower level nonterminal advances, one step behind the nonterminal in the upper level, at any step it can be replaced by a terminal, and this ends the derivation in the lower level. When the upper level introduced a symbol in T, the lower one should use a terminal rule in the case of terminal derivation, or any rule in the case of non-terminal derivation and it stops. The upper level can continue without any restriction. To sum up, a window of length two suffices in order to control the work of γ in a way similar to that in the proof of Theorem 4. Consequently, $L_{t,s}(\gamma) \in REG, L_{nt,s}(\gamma) \in REG$. □

Corollary 5. *$YL(REG) = REG$ for all $Y \in \{TS, TNS, NTS, NTNS\}$.*

Proof. Combine Lemma 5 (it gives the inclusion $\supseteq$) with Theorem 4 (the converse inclusion for the non-synchronized case) and Theorem 5 (the converse inclusion for the synchronized case). □

Thus, the regular layered grammars need no further investigation (in what concerns the generative capacity). The results above deserve to be emphasized: in the synchronized case, the regular rules are strictly less powerful than right-linear rules. This does not happen in many situations in formal language theory (for instance, in regulated rewriting area, [6]). However, a similar result has been recently proved for parallel communication grammar systems: in the centralized returning case, the regular rules are strictly weaker than the right-linear rules [7].

7 A Characterization of RE Languages

We now increase the degree of synchronization in a layered grammar, by specifying the pair of rules to be used by the two components, at steps when both of them have observable nonterminals.

A *matrix layered grammar* is a construct

$$\gamma = (N, T, t, S_1, S_2, M),$$

where N, T, t, S_1, S_2 are as in a usual layered grammar (the nonterminal and the terminal alphabet, the transparent symbol, the axioms of the two layers, respectively), and M is a finite set of pairs (we call them *matrices*) of the forms

$$(A_1 \to u_1, A_2 \to u_2),\ (A_1 \to u_1, \#),\ (\#, A_2 \to u_2),$$

where $A_1 \to u_1, A_2 \to u_2$ are context-free rules over $N \cup T \cup \{t\}$. In a derivation step $[x_1, x_2] \to [y_1, y_2]$ we have to use a pair $(A_1 \to u_1, A_2 \to u_2)$ if both x_1 and x_2 contain observable nonterminals (hence $x_1 = x_1' A_1 x_1'', y_1 = x_1' u_1 x_1'', x_2 = x_2' A_2 x_2'', y_2 = x_2' u_2 x_2''$), a pair $(A_1 \to u_1, \#)$ if x_2 contains no observable nonterminal and A_1 appears in x_1 (hence $x_1 = x_1' A_1 x_1'', y_1 = x_1' u_1 x_1'', x_2 = y_2$), and a pair $(\#, A_2 \to u_2)$ if x_1 is a terminal string and A_2 is an observable nonterminal in x_2 (hence $x_1 = y_1, x_2 = x_2' A_2 x_2'', y_2 = x_2' u_2 x_2''$). We denote by $L_t(\gamma)$ the language generated by γ in the terminal mode and by $L_{nt}(\gamma)$ the language generated by γ in the non-terminal mode. The corresponding families of languages are denoted by $TML(X), NTML(X), X \in \{REG, RL, LIN, CF\}$.

Remark. Because we may always assume that the nonterminals used in the two layers are distinct, we can consider matrices as pairs of rules without specifying where each rule is used: we have just to use the two rules in two different layers. We prefer here to work with the previous definition because we find it more natural.

Rather surprisingly, the following characterization of recursively enumerable languages can be obtained.

Theorem 6. *For every language $L \in RE$ there are a projection h, a regular language R, and a language $L' \in TML(RL) \cap NTML(RL)$ such that $L = h(L' \cap R)$.*

Proof. We use the following variant (proved in [8]) of the characterization of recursively enumerable languages by means of equality sets of morphisms (see [5], [12], [13]). For two morphisms $h_1, h_2 : V^* \longrightarrow U^*$ we denote

$$EQ(h_1, h_2) = \{w \in V^* \mid h_1(w) = h_2(w)\}.$$

For every language $L \in RE, L \subseteq V^*$, there are two alphabets V_1, V_2 such that $V \subseteq V_2$, two λ-free morphisms $h_1, h_2 : V_1^* \longrightarrow V_2^*$, a regular set $R \subseteq V_2^*$, and a projection $h_3 : V_2^* \longrightarrow V^*$, such that $L = h_3(h_1(EQ(h_1, h_2)) \cap R)$. Consider also the alphabet of new symbols $V_2' = \{a' \mid a \in V_2\}$. We now construct the matrix layered grammar

$$\gamma = (\{S_1, S_1', S_2\}, V_2 \cup V_2' \cup \{c, d\}, t, S_1, S_2, M),$$

with the following matrices of rules:

$$\begin{aligned}
&(1)\ (S_1 \to tS_1', \#),\\
&(2)\ (S_1' \to tb_{i_1}tb_{i_2}\dots tb_{i_k}tS_1',\ S_2 \to b'_{j_1}tb'_{j_2}t\dots tb'_{j_l}tS_2),\\
&(3)\ (S_1' \to tb_{i_1}tb_{i_2}\dots tb_{i_k}td,\ S_2 \to b'_{j_1}tb'_{j_2}t\dots tb'_{j_l}tc),\\
&\quad \text{for } h_1(a) = b_{i_1}b_{i_2}\dots b_{i_k}, k \geq 1, b_{i_s} \in V_2, 1 \leq s \leq k,\\
&\quad \text{and } h_2(a) = b'_{j_1}b'_{j_2}\dots b'_{j_l}, l \geq 1, b'_{j_s} \in V_2', 1 \leq s \leq l.
\end{aligned}$$

Consider also the regular language

$$R_0 = \{b'b \mid b \in V_2\}^*\{cd\}$$

and the projection $g : (V_2 \cup V_2')^* \to V_2^*$ defined by $g(b) = b, b \in V_2$, and $g(b') = \lambda, b' \in V_2'$. It is easy to see that we have

$$g(L_t(\gamma) \cap R_0) = g(L_{nt}(\gamma) \cap R_0) = h_1(EQ(h_1, h_2)).$$

Indeed, because the upper component introduces the symbol t in each odd position and the nonterminal of the lower component appears always in an odd position, the non-terminal derivations should be terminal. Moreover, the intersection with R_0 ensures the fact that we end in the two layers with two identical strings modulo the primes appearing in the lower layer, namely the two strings correspond to some $h_1(w_1) = h_2(w_2)$; the matrices in M ensure the fact that $w_1 = w_2$: after using the only matrix of type (1), always is S_2 observable, and the two components should stop at the same time, by using a matrix of type (3).

Now, let us consider the morphism $h : V_2^* \longrightarrow (V_2 \cup V_2')^*$ defined by $h(b) = b'b, b \in V_2$, the regular language $R' = h(R)cd$, and extend the projection h_3 to $h_3' : (V_2 \cup V_2')^* \to V^*$ by $h_3'(b) = h_3(b), b \in V_2$, and $h_3'(b') = \lambda, b' \in V_2'$, $h_3'(c) = h_3'(d) = \lambda$. Then we have

$$L = h_3'(L_t(\gamma) \cap h(R')) = h_3'(L_{nt}(\gamma) \cap h(R')).$$

Indeed, R' plays at the same time the roles of both R and R_0, while h_3' plays at the same time the roles of h_3 and g. □

Corollary 6. *For every family of languages $F \subset RE$ which is closed under intersection with regular languages and projections we have $TML(RL) - F \neq \emptyset$, $NTML(RL) - F \neq \emptyset$.*

Proof. An inclusion $TML(RL) \subseteq F, NTML(RL) \subseteq F$ would imply that the closure of $TML(RL), NTML(RL)$ under intersection with regular languages and projections is included into the closure of F under these operations. This implies the inclusion $RE \subseteq F$, contradicting the strict inclusion $F \subset RE$. □

Important families F as above are MAT^λ, of languages generated by matrix grammars with arbitrary context-free rules but without appearance checking, and $ET0L$, the family of languages generated by extended tabled interactionless Lindenmayer systems, [10]. The proof of Theorem 5 remains valid also for matrix layered grammars, that is the next result holds:

Theorem 7. $TML(REG) = NTML(REG) = REG$.

8 Final Remarks; Variants

Several variants of layered grammars can be naturally defined. We only mention some of them as a proof of the richness of this notion. First, as it is the case with the icons observable in windows superposed on the computer screen, we can assume that some nonterminals are "more active" than others. When several nonterminals are observable, the "most active" of them is rewritten. This idea can be implemented, for instance, under the form of a partial order relation on the nonterminal alphabet of the grammar, or under the form of a leftmost restriction on the derivation (the first observable nonterminal from the left of the layer should be rewritten). Note that in the case of linear grammars (hence also of right-linear and regular grammars) these variants coincide with the one investigated here, hence all Theorems 1 – 7 from the previous sections remain true also for these variants. Second, we can consider a variant which removes the apparent contrast between the parallel character of activating symbols by observability and the sequential mode of rewriting. That is, it is quite natural to derive all observable symbols at the same time. This leads to considering parallel derivations, either in a context-free grammar as here (this reminds the so-called Indian and Russian parallel grammars in regulated rewriting area, see [6]), or in a pure grammar, where there is no distinction between terminal and nonterminal symbols (this corresponds to Lindenmayer systems; in particular, layered 0L or D0L systems look attractive). Third, we can consider layered grammars of any order, not only with two levels as we have done here. The definition of observability can be obviously extended to n-layered strings; similarly the definition of a layered grammar of degree $n, n \geq 1$, is an obvious extension of the definition here. Such systems will probably have a rather intricate behavior. They correspond in a better way to a parallel grammar system, with a particular type of cooperation among components: they work synchronously, on their separate sentential forms (this is similar to a parallel communication grammar system, [9], [4]), but they do not communicate by sending messages, rather they just influence each other through observable symbols; however, the result is highly integrated: it is the superposition (in the sense of the operation $\diamond$, extended to n-layered strings) of the strings generated by the component grammars. Then, usual derivation, leftmost, parallel derivations can be considered also in this case. Of course, such a system with n layers can be simulated by a system with $n+1$ layers (just add a component which contains only the rule $S \rightarrow t$). Whether or not the systems of degree $n+1$ are strictly more powerful than those of degree n becomes a fundamental problem (in general, not solved in grammar systems area, [4]). By the various motivations of the model, by the results given here (especially the possibility of generating non-regular languages by layered grammars with right-linear rules and the characterization of recursively enumerable languages by matrix layered grammars with rules of the same type — right-linear), and by the wealth of problem raised by the variants mentioned above, the layered grammars prove to be a research area of definite interest. Of course, it is however premature to inquiry about the practical relevance of these grammars.

Acknowledgements. Gh. Păun was supported by a grant under the program for visiting professors of the Group for Mathematical Informatics (GNIM) of the Italian National Research Council (CNR).

Thanks are due to an anonymous referee, who made many comments on an earlier version of the paper; the structuring of the construction in the proof of Theorem 4 is also due to him/her.

References

[1] M. Andraşiu, J. Dassow, Gh. Păun, A. Salomaa, Language-theoretic problems arising from Richelieu cryptosystems, *Theor. Comput. Sci.*, 116 (1993), 339 – 357.

[2] P. Bottoni, M. F. Costabile, S. Levialdi, P. Mussio, Defining visual languages for interactive computing, *IEEE Transactions on Systems, Man, and Cybernetics – A*, 27 (1997), 773 – 782.

[3] P. Bottoni, M. F. Costabile, S. Levialdi, P. Mussio, Specification of Visual Languages as Means for Interaction, *Theory of Visual Languages*, K. Marriott, B. Meyer eds., Springer-Verlag, 1997, to appear.

[4] E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[5] K. Culik II, A purely homomorphic characterization of recursively enumerable sets, *Journal of the ACM*, 26 (1979), 345 – 350.

[6] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

[7] S. Dumitrescu, Gh. Păun, On the power of parallel communicating grammar systems with right-linear components, *Rev. Fr. Aut. Inform. Theor., RAIRO, Theor. Informatics*, 31, 4 (1997), 331 – 354.

[8] L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa, S. Yu, DNA computing, sticker systems, and universality, *Acta Informatica*, 35 (1998), 401 – 420.

[9] Gh. Păun, L. Sântean, Parallel communicating grammar systems: the regular case, *Ann. Univ. Buc., Matem.-Inform. Series*, 38, 2 (1989), 55 – 63.

[10] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1980.

[11] G. Rozenberg, A. Salomaa, Eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Heidelberg, 1997.

[12] A. Salomaa, Equality sets for homomorphisms of free monoids, *Acta Cybernetica*, 4 (1978) 127 - 139.

[13] A. Salomaa, *Jewels of Formal Language Theory*, Computer Science Press, Rockville, Maryland, 1981.

Received October, 1997