Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 2 (June), pp. 266-277

On the Power of Small Size Insertion P Systems

A. Krassovitskiy

Alexander Krassovitskiy
University Rovira i Virgili
Research Group on Mathematical Linguistics
Av. Catalunya, 35, Tarragona 43002, Spain
E-mail: alexander.krassovitskiy@estudiants.urv.cat

Abstract: In this article we investigate insertion systems of small size in the framework of P systems. We consider P systems with insertion rules having one symbol context and we show that they have the computational power of context-free matrix grammars. If contexts of length two are permitted, then any recursively enumerable language can be generated. In both cases a squeezing mechanism, an inverse morphism, and a weak coding are applied to the output of the corresponding P systems. We also show that if no membranes are used then corresponding family is equal to the family of context-free languages.
Keywords: P systems, insertion-deletion systems, context-free languages, matrix grammars, computational completeness

1 Introduction

The study of insertion-deletion operations on strings has a long history; we just mention [2,5,11,16,20]. Insertion-deletion systems motivated from molecular computing have been studied in [1, 3, 10, 19, 21]. With some linguistic motivation they may be found in [9].

In general form, an insertion operation means adding a substring to a given string in a specified (left and right) context, while a deletion operation means removing a substring of a given string from a specified (left and right) context. A finite set of insertion/deletion rules, together with a set of axioms provide a language generating device: starting from the set of initial strings and iterating insertion-deletion operations as defined by the given rules we get a language. The number of axioms, the length of the inserted or deleted strings, as well as the length of the contexts where these operations take place are natural descriptional complexity measures of the insertion-deletion systems.

Inspired by the structure and the functioning of a living cell, especially by the local information processing, P systems are brought to light few years ago [15]. This is a highly distributed computational model which combines local processing (in membranes) and communication (between them). It is natural to consider insertion and deletion operations in the framework of P systems and it was firstly done by Gh. Paun in [17]. Such a combination permits do define programmed-like insertion-deletion systems, which, as expected, increase their computational power. Some combinations of parameters for pure insertion-deletion lead to systems which are not computationally complete [6, 12] or even decidable [22], while in [7, 8] it was shown that P systems framework can easily increase the computational power comparing to ordinary insertion-deletion systems.

Traditionally, language generating devices having only insertion or only deletion rules were studied. Early computational models based only on insertion appear already in [9], and are discussed in [19] and [17] (with membrane tree structure). It was proved that pure insertion systems having one letter context are always context-free. Yet, there are insertion systems with two letter context which generate nonsemilinear languages (see Theorem 6.5 in [19]). On the

other hand, it appears that by using only insertion operations the obtained language classes with contexts greater than one are incomparable with many known language classes. For example, there is a simple linear language $\{a^n b a^n \mid n \geq 1\}$ which cannot be generated by any insertion system (see Theorem 6.6 in [19]).

In order to overcome this obstacle one can use some codings to "interpret" the generated strings. The computational power of insertion systems with morphisms and intersection with special languages was investigated in [14] and [18]. In [11] there were used two additional mapping relations: a morphism h and a weak coding φ. The strings of a language are considered being the products $h^{-1} \circ \varphi$ over the generated strings. More precisely, the squeezing mechanism selects only those output strings of the corresponding (P) systems to which h^{-1} and φ are being applied. As expected, the language generating mechanisms have greater expressivity, and the corresponding language family is larger. It appears that with the help of morphisms and codings one can obtain every RE language if insertion rules have sufficiently large context. It is proved in [11] that for every recursively enumerable language L there exists a morphism h, a weak coding φ and a language L' generated by an insertion system with rules using the length of the contexts at most 7, such that, $L = h(\varphi^{-1}(L'))$. The result was improved in [13], showing that rules having at most 5 letter contexts are sufficient to encode every recursively enumerable language. Recently, in [4] it was shown that the same result can be obtained with the length of contexts equal to 3.

Our aim is to reduce the length of the contexts in insertion rules by regulating derivations in terms of membranes. Unlike the previous works, our article considers the encoding as a part of insertion P systems. The obtained model is quite powerful and has the power of matrix languages if contexts of length one are used. We also show that if no encoding is used, then the corresponding family is strictly included into the family of matrix languages and is equal to the family of context-free languages if no membranes are used. If an insertion of two symbols in two letters contexts is used, then all recursively enumerable languages can be generated (using of course the inverse morphism, the weak coding, and the squeezing mechanism).

2 Prerequisites

Here we use standard formal language theoretic notions and notations. The reader can consult any of the many monographs in this area (see, e.g., in [20] for the unexplained details).

We denote by $|w|$ the length of a word w and by $card(A)$ the cardinality of the set A, while ε denotes the empty string.

By CF and RE we denote the classes of context-free and recursively enumerable languages, respectively. A language L is context-free if there exists a context-free grammar G such that $L(G) = L$. A context-free grammar is a construct $G = (N, T, S, P)$, where N and T are disjoint alphabets, $S \in N$, and P is a finite set of context-free rules of the form $A \longrightarrow v$, where $A \in N$ and $v \in (N \cup T)^*$. We say that a context-free grammar $G = (N, T, P, S)$ is in Chomsky normal form, if each rule in P has one of the forms $A \longrightarrow \alpha$, or $A \longrightarrow BC$, for $A, B, C \in N, \alpha \in T \cup \{\varepsilon\}$.

A language L is recursively enumerable if there exists a type 0 grammar G such that $L(G) = L$. A type 0 grammar is a construct $G = (N, T, S, P)$, where N and T are disjoint alphabets, $S \in N$ and P is a finite set of rules of the form $u \longrightarrow v$, where $u \in (N \cup T)^* N (N \cup T)^*$ and $v \in (N \cup T)^*$.

We say that a type 0 grammar $G = (N, T, P, S)$ is in Penttonen normal form, if each rule in P can be written either as $A \longrightarrow \alpha$, or $AB \longrightarrow AC$, for $A, B, C \in N, \alpha \in (T \cup N)^*, |\alpha| \leq 2$. It is well known that for every type 0 language there is a modified Penttonen normal form where each rule can be written as either: $A \longrightarrow \alpha$, or $A \longrightarrow AC$, or $A \longrightarrow CA$, or $AB \longrightarrow AC$, or $AB \longrightarrow CB$, for $A, B, C \in N, \alpha \in T \cup \{\varepsilon\}$.

We also recall the following definition from [17]. A *context-free matrix grammar* (without appearance checking) is a construct $G = (N, T, S, M)$, where N, T are disjoint alphabets (of non-terminals and terminals, respectively), $S \in N$ (axiom), and M is a finite set of matrices, that is sequences of the form $(A_1 \longrightarrow x_1, \ldots, A_n \longrightarrow x_n), n \geq 1$, of context-free rules over $N \cup T$. For a string w, a matrix $m : (r_1, \ldots, r_n)$ is executed by applying the productions $r_1, \ldots, r_n$ one after the other, following the order in which they appear in the matrix. Formally, we write $w \Longrightarrow_m u$ if there is a matrix $m : (A_1 \longrightarrow u_1, \ldots, A_n \longrightarrow u_n) \in M$ and the strings $w_1, w_2, \ldots, w_{n+1} \in (N \cup T)^*$ such that $w = w_1, w_{n+1} = u$, and for each $i = 1, 2, \ldots, n$ we have $w_i = w' A_i w''$ and $w_{i+1} = w' u_i w''$. If the matrix m is understood, then we write $\Longrightarrow$ instead of $\Longrightarrow_m$. As usual, the reflexive and transitive closure of this relation is denoted by $\Longrightarrow^*$. Then, the language generated by G is $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages generated by context-free matrix grammars is denoted by MAT^λ (the superscript indicates that erasing rules are allowed). It is well known fact that every language from MAT^λ can be generated by a modified binary normal form, (similarly to the binary normal form, see e.g., [17]) having each matrix m of the following form $m : (A \longrightarrow \alpha, A' \longrightarrow \alpha')$, for $A, A' \in N, \alpha, \alpha' \in (N \cup T)^*, max(|\alpha|, |\alpha'|) \leq 2$.

An *insertion system* is a construct $\Gamma = (V, A, I)$, where V is an alphabet, A is a finite language over V, and I is finite set of triples of the form (u, α, v), where u, α, and v are strings from V^*. The elements of A are called *axioms*, the triples in I are *insertion rules*. An insertion rule $(u, \alpha, v) \in I$ indicates that the string α can be inserted in between u and v. Stated otherwise, $(u, \alpha, v) \in I$ corresponds to the rewriting rule $uv \to u\alpha v$. We denote by $\Longrightarrow$ the relation defined by an insertion rule (formally, $x \Longrightarrow y$ iff $x = x_1 uv x_2, y = x_1 u\alpha v x_2$, for some $(u, \alpha, v) \in I$ and $x_1, x_2 \in V^*$). We denote by $\Longrightarrow^*$ the reflexive and transitive closure of $\Longrightarrow$ (as usual, $\Longrightarrow^+$ is its transitive closure).

The language generated by Γ is defined by

$$L(\Gamma) = \{w \in V^* \mid x \Longrightarrow^* w, x \in A\}.$$

We say that an insertion system (V, A, I) has *weight* (n, m, m') if

$$m = \max\{|u| \mid (u, \alpha, v) \in I\},\ m' = \max\{|v| \mid (u, \alpha, v) \in I\},$$
$$n = \max\{|\alpha| \mid (u, \alpha, v) \in I\}.$$

We denote by $INS_n^{m,m'}$ the corresponding families of languages generated by insertion systems.

An *insertion P system* is a construct:

$$\Pi = (V, \mu, M_1, \ldots, M_k, R_1, \ldots, R_k),$$

where

- V is a finite alphabet,
- μ is a (cell-like, i.e., hierarchical) membrane structure with k membranes. This structure will be represented by a word containing correctly nested marked parentheses, i.e., by a word from the Dyck language. The skin membrane is labeled with "1".
- M_i, for each $1 \leq i \leq k$ is a finite language associated to the membrane i.
- R_i, for each $1 \leq i \leq k$ is a set of insertion rules with target indicators associated to membrane i and which have the following form: $(u, x, v; tar)$, where (u, x, v) is an insertion rule, and tar, called the *target indicator*, is from the set $\{here, in_j, out\}, 1 \leq j \leq k$.

A configuration of Π is a k-tuple $(N_1, \ldots, N_k)$ of finite languages over V. For two configurations $(N_1, \ldots, N_k)$ and $(N'_1, \ldots, N'_k)$ of Π we write $(N_1, \ldots, N_k) \Longrightarrow (N'_1, \ldots, N'_k)$ if we can pass from $(N_1, \ldots, N_k)$ to $(N'_1, \ldots, N'_k)$ by applying nondeterministically the insertion rules, to all strings which can be rewritten from the corresponding regions, and following the target indications associated with the rules. More specifically, $w' \in N'_j$ if either $w' \in N_j$ or there is a word $w \in N_i$ and $w \Longrightarrow_r w'$, where $r = (u, x, v; tar) \in R_i$. Moreover, the membrane labeled by j is immediately outside the membrane labeled by i if $tar = out$; the membrane labeled by j is immediately below the membrane labeled by i if $tar = in_j$; and $i = j$ if $tar = here$. No other words are present in $N'_j, 1 \leq j \leq k$. We say that a word w' is *sent out* of the system if there is a configuration $(N_1, \ldots, N_k)$, a word $w \in N_1$, and $w \Longrightarrow^r w'$ where $r = (u, x, v; out) \in R_1$.

We use the definition of the language generated by Π according to [17]. This language is denoted by $L(\Pi)$ and it is defined as follows: we start from the initial configuration $(M_1, \ldots, M_k)$ of the system and proceed iteratively, by transition steps performed by applying the rules in parallel, to all strings which can be rewritten. All strings over the alphabet V sent out of the system (*i.e.* sent from the skin membrane) during any step of any computation form the language $L(\Pi)$.

Insertion tissue P systems are defined in an analogous manner. As the tissue P systems use arbitrary graph structures we write the target indicator in the form $tar = go_j, j = 1, \ldots, k$. We remark, that the result of a computation consists of all strings over V which are sent to one selected output cell.

The *weight* of insertion rules (n, m, m') and the membrane *degree* k describe the complexity of the system. We denote by $LSP_k(ins_n^{m,m'})$(see [17]) the family of languages $L(\Pi)$ generated by insertion P systems of degree at most $k \geq 1$, having weight at most (n, m, m'). If some of the parameters n, m, m', or k is not specified we write " * " instead.

We say that a language L' is from $MorINS_n^{m,m'}$ (from $MorLSP_k(ins_n^{m,m'})$, respectively) if there exist a morphism h, a weak coding φ and $L \in INS_n^{m,m'}$ ($L \in LSP_k(ins_n^{m,m'})$) such that $\varphi(h^{-1}(L)) = L'$.

For every language $L \in MorLSP_k(ins_n^{m,m'}), L = \varphi(h^{-1}(L(\Pi))$, we add h and φ to the system, and we write Π in the form

$$\Pi = (V, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n, h, \varphi),$$

Hereafter h and φ are naturally extended to strings: $h(a_1 a_2 \ldots a_t) = h(a_1)h(a_2)\ldots h(a_t)$, and $\varphi(a_1 a_2 \ldots a_t) = \varphi(a_1)\varphi(a_2)\ldots\varphi(a_t), a_i \in V$.

We insert "t'' before P to denote classes of languages corresponding to the tissue cases (e.g., $LStP$). We also write "[t]" (e.g., $LS[t]P$) if we do not distinguish between tissue and tree classes.

We say that a letter a is *marked* in a sentential form waw'' if it is followed by #, i.e., $|w''| > 0$, and # is the prefix of w''. In the following proofs we use a marking technique introduced in [17]. The technique works as follows: in order to simulate a rewriting production $A \longrightarrow B$ we add adjacently right from A the word $\#B$ specifying that letter A is already rewritten. As soon as the derivation of the simulated sentential form is completed, every nonterminal A is marked and the pair $A\#$ is subject to the inverse morphism.

3 Main results

Let us consider insertion systems (without membranes) with one letter context rules, i.e., the family $MorINS_*^{1,1}$. Applying the marking technique we get a characterization of context-free languages.

Theorem 3.1. $MorINS_*^{1,1} = CF$.

Proof: First we show that $CF \subseteq MorINS_3^{1,1}$.

Let $G = (V, T, S, P)$ be a context-free grammar in Chomsky normal form. Consider the following insertion system

$$\Pi = (T \cup V \cup \{\#\}, R, \{S\}, h, \varphi),$$

where $R = \{(A, \#\gamma, \alpha) \mid \alpha \in T \cup V, A \longrightarrow \gamma \in P, \gamma \in (T \cup V)^*, 1 \leq |\gamma| \leq 2\}$, the morphism h is defined as follows

$$h(a) = a\#, \text{ if } a \in V, \text{ and } h(a) = a, \text{ if } a \in T,$$

and the weak coding φ is defined as follows

$$\varphi(a) \longrightarrow \varepsilon, \text{ if } a \in V, \varphi(a) \longrightarrow a, \text{ if } a \in T.$$

It is clear that $L(\Pi) \in MorINS_3^{1,1}$. We claim that $L(\Pi) = L(G)$. Indeed, each rule $(A, \#\gamma, \alpha) \in R$ can be applied in the sentential form $wA\ alphaw'$ if A is unmarked (not rewritten). Hence, the production $A \longrightarrow \gamma \in P$ can be simulated by the corresponding derivation of G. Hence, by applying the counterpart rules we get equivalent derivations. At the end of the computation every nonterminal is marked, and no rules can be applied any more. (Indeed, if the system produces a word having some unmarked nonterminal then h^{-1} is not defined on this word.) At this point h^{-1} removes all marks, and φ removes all nonterminal symbols. Hence $L(\Pi) = L(G)$. We get $CF \subseteq MorINS_3^{1,1}$.

The equivalence of these two classes follows from Theorem 6.4 in [19] stating that $INS_*^{1,1} \subseteq CF$ and the fact that the family of context-free languages is closed under inverse morphisms and weak codings. □

Now we consider insertion P systems with the left and right contexts of at most one letter. It is known from Theorem 5.5.1 in [17] that $LSP_2(ins_2^{1,1})$ contains non context-free languages. We prove that the more general family $LSP_*(ins_*^{1,1})$ is bounded by the class of languages generated context-free matrix grammars:

Lemma 3.2. $LStP_*(ins_*^{1,1}) \subset MAT^{\lambda}$.

Proof: The proof uses a similar technique as in [19], Theorem 6.4 for context-free grammars.

Let $\Pi = (V, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n)$ be an insertion P system such that $L(\Pi) \in LStP_n(ins_*^{1,1})$ for some $n \geq 1$.

Consider the matrix grammar $G = (D \cup Q \cup \{S\}, V, S, P)$, where $Q = \{Q_i \mid i = 1, \ldots, n\}$, $D = \{D_{a,b} \mid a, b \in V \cup \{\varepsilon\}\}$, and P is constructed as follows:

1. For every rule $(a, b_1 \ldots b_k, c, go_j) \in R_i$, $a, c \in V \cup \{\varepsilon\}, b_1, \ldots, b_k \in V, k > 0$ we add to P $(Q_i \longrightarrow Q_j, D_{\overline{a},\overline{c}} \longrightarrow D_{\overline{a},b_1} D_{b_1,b_2} \ldots D_{b_{k-1},b_k} D_{b_k,\overline{c}})$, where

$$\overline{a} = \begin{cases} a, & \text{if } a \in V, \\ t, \forall t \in V \cup \{\varepsilon\}, & \text{if } a = \varepsilon \end{cases} \qquad \overline{c} = \begin{cases} c, & \text{if } c \in V, \\ t, \forall t \in V \cup \{\varepsilon\}, & \text{if } c = \varepsilon. \end{cases}$$

2. For every rule $(a, \varepsilon, c, go_j) \in R_i$, $a, c \in V \cup \{\varepsilon\}, k > 0$ we add to P $(Q_i \longrightarrow Q_j, D_{\overline{a},\overline{c}} \longrightarrow D_{\overline{a},\overline{c}})$, where $\overline{a}$ and $\overline{c}$ are defined as in the previous case.

3. Next, for every $w = b_1 \ldots b_k \in M_i, i = 1, \ldots, n, k > 0$ we add to P the matrix $(S \longrightarrow Q_i D_{\varepsilon,b_1} D_{b_1,b_2} \ldots D_{b_{k-1},b_k} D_{b_k,\varepsilon})$.

4. As a special case if $\varepsilon \in M_i$ we add $(S \longrightarrow Q_i D_{\varepsilon,\varepsilon})$ to P.

5. Also, for every $D_{a,b} \in D, a, b \in V \cup \{\varepsilon\}$ we add $(D_{a,b} \longrightarrow a)$ to P.

6. Finally, we add $(Q_1 \longrightarrow \varepsilon)$ to P (we assume that the first cell is the output cell).

The simulation of Π by the matrix grammar is straightforward. We store the label of current cell by means of nonterminals from Q. Every nonterminal $D_{a,c} \in D, a, c \in V \cup \{\varepsilon\}$ represents a pair of adjacent letters, so we can use them as a context. A rule $(a, b_1 \dots b_k, c, go_j) \in R_i, a, c \in V, b_1 \dots b_k \in V^k$ can be simulated by the grammar iff the sentential form contains both Q_i and $D_{a,c}$. As a result, the label of the current cell is rewritten to Q_j and $D_{a,c}$ is rewritten to the string $D_{a,b_1} D_{b_1,b_2} \dots D_{b_{k-1},b_k} D_{b_k,c}$. We note, that in order to simulate rules that have no context we introduce productions with every possible context symbols by writing $\overline{a} \in V \cup \{\varepsilon\}$ and $\overline{c} \in V \cup \{\varepsilon\}$. Clearly, since indexed symbols are duplicated for adjacent nonterminals, e.g., $D_{b_{i-1},b_i} D_{b_i,b_{i+1}}$, every nonterminal thus preserves one symbol right and left contexts.

The simulation of Π by the grammar starts with a nondeterministic choice of the axiom from $M_1, \dots, M_n$. Then, during the derivation any rule from $R_1, \dots, R_n$ having the context (a, b) is applied in one to one correspondence with grammar productions having $D_{a,b}$ in the left hand side. Finally, the string over V is produced by the grammar iff Q_1 has been deleted from the simulated sentential form. The deletion of Q_1 specifies that Π reached the output cell. So, we obtain $L(\Pi) = L(G)$. Hence, $LStP_*(ins_*^{1,1}) \subseteq MAT^\lambda$.

The strictness of the inclusion follows from the fact there are languages from MAT^λ which cannot be generated by any insertion P system from $LStP_*(ins_n^{1,1})$, for any $n \geq 1$. Indeed, consider the context-free language $L_a = \{ca^k ca^k c \mid k \geq 0\}$. Since every context-free language is a matrix language we have $L_a \in MAT^\lambda$. On the other hand, $L_a \notin LStP_*(ins_n^{1,1})$, for any $n \geq 1$. For the contrary, assume there is such a system Π'. We note, that the system cannot delete or rewrite any letter, so every insertion is terminal. As the languages of axioms are finite we need an insertion rule of letter a. Consider the final insertion step in a derivation which has at most one step and derives a word $w = ca^k ca^k c$, for some $k \geq n + 1$:

$$w_0 \Longrightarrow^* w' \Longrightarrow w,$$

where w_0 is an axiom. Since $|w_0|_c \leq 3$, c may be inserted by the last insertion. Assume, that $|w'|_c = 3$. In the latter case, let a^p be the inserted string, $p \leq n$. Because, we may insert a^p in the distinct positions of w' we get that either $ca^{k-p}ca^{k+p}c \in L(\Pi')$ or $ca^{k+p}ca^{k-p}c \in L(\Pi')$. This is a contradiction.

Now assume that c is inserted by the last insertion. We note that the insertion of two c is not possible, since $k \geq n + 1$. Consider three cases: (1) the last applied rule inserts c in the middle, (2) at the end, or (3) at the beginning of w'.
(1) Let $w_c = a^{p'}ca^{p''}$ be the inserted string, where $p' + p'' \leq n - 1$. Hence, $w' = ca^{k'+k''}c$, where $k' + p' = k'' + p'' = k$, and $k' + k'' = 2k - p' - p'' \geq 2n + 2 - n + 1 \geq 4$. Obviously, regardless of the contexts of the last insertion rule there are at least two positions at which w_c can be inserted. So, we get a contradiction because either $ca^{k'+p'+1}ca^{k''+p''-1}c \in L(\Pi')$, or $ca^{k'+p'-1}ca^{k''+p''+1}c \in L(\Pi')$.
(2) Let $a^q c$ be the inserted string, where $q \leq n - 1$. The corresponding insertion rule has one of the following forms: $(\varepsilon, a^q c, \varepsilon, go_j)$ or $(a, a^q c, \varepsilon, go_j)$, where j is an index of the final membrane. In ether case, $a^q c$ may be inserted in w' before the last letter a. This is a contradiction. The case (3) is a mirror to the case (2) and can be treated similarly.
So we proved $L_a \notin LStP_*(ins_n^{1,1})$, for any $n \geq 1$ and, hence, $LStP_*(ins_*^{1,1}) \subset MAT^\lambda$. □

Since trees are special cases of graphs we get the following result

Corollary 3.3. $LSP_*(ins_*^{1,1}) \subset MAT^\lambda$.

Lemma 3.4. $MAT^\lambda \subseteq MorLSP_*(ins_2^{1,1})$.

Proof: We prove the lemma by direct simulation of a context-free matrix grammar $G = (N, T, S, P)$. We assume that G is in the modified binary normal form, i.e., every matrix has the form $i : (A \longrightarrow BC, A' \longrightarrow B'C') \in P$, where $A, A' \in N; B, B', C, C' \in N \cup T \cup \{\varepsilon\}$, and $i = 1, \ldots, n$.

Consider a P insertion system Π defined as follows:

$$\Pi = (V, [_1\ [_2\ [_3\ [_4\ ]_4\ \ldots [_{n+3}\ ]_{n+3}\ ]_3\ ]_2\ ]_1, \{S\$\}, \emptyset, \ldots, \emptyset, R_1, \ldots, R_{n+3}, h, \varphi),$$

where $V = N \cup T \cup \{C_i, C_i' \mid i = 1, \ldots, n\} \cup \{\#, \$\}$.

For every matrix $i : (A \longrightarrow BC, A' \longrightarrow B'C')$ we add

$$\begin{array}{lll}
r.1.1 : (A, \#C_i, \alpha, in_2), & & \text{to } R_1;\\
r.2.1 : (C_i, BC, \alpha, in_3), & r.2.2 : (C_i', \#, \alpha, out) & \text{to } R_2;\\
r.3.1 : (C_i, \#, \alpha, in_{i+3}), & r.3.2 : (C_i', B'C', \alpha, out) & \text{to } R_3;\\
r.i+3.1 : (A', \#C_i', \alpha, out), & & \text{to } R_{i+3}
\end{array}$$

for every $\alpha \in V \backslash \{\#\}$. In addition we add $(\varepsilon, \$, \varepsilon, out)$ to R_1.

We define the morphism h and the weak coding φ by:

$$h(a) = \begin{cases} a, & \text{if } a \in T,\\ a\# & \text{if } a \in V \backslash (T \cup \{\#\}) \end{cases} \qquad \varphi(a) = \begin{cases} a, & \text{if } a \in T,\\ \varepsilon & \text{if } a \in V \backslash T. \end{cases}$$

Clearly, $L(\Pi) \in MorLSP_{n+3}(ins_2^{1,1})$. We claim that $L(\Pi) = L(G)$. To prove this it is enough to prove that $w \in L(G)$ iff $w' \in L(\Pi)$ and $w' = \varphi(h^{-1}(w))$.

First we show that for every $w \in L(G)$ there exists $w' \in L(\Pi)$ and $w' = \varphi(h^{-1}(w))$. Consider the simulation of the i-th matrix $(A \longrightarrow BC, A' \longrightarrow B'C') \in P$. It is controlled by letters C_i and C_i'. First, we insert $\#C_i$ in the context of unmarked A and send the obtained string to the second membrane. Then we use C_i as a context to insert adjacently right the word BC. After that, we mark the control letter C_i and send sentential form to the $(i+3)$-rd membrane. Here we choose nondeterministically one letter A', mark it, write adjacently right the new control letter C_i', and, after that, send the obtained string to the third membrane. (We remark, the third membrane is immediately outside of the $i+3$-rd membrane.) It is clear that it is not possible to apply the rule $r.i+3.1 : (A', \#C_i', \alpha; out)$ in the $(i+3)$-rd membrane and to reach the skin membrane if the sentential form does not contain unmarked A'. So, this branch of computation cannot influence the result and may be omitted in the consideration. Next, in the third membrane, $B'C'$ is inserted in the context of unmarked C_i' and the form is sent to the second membrane. Finally, we mark C_i' and send the resulting string back to the skin membrane.

At the beginning of the simulation the sentential form in the skin membrane does not contain unmarked C_i, C_i'. Hence, the insertions in the second and third membranes are deterministic. The derivation preserves this property, as after the sentential form is sent back to the skin membrane, introduced C_i, and C_i' are marked. At the end of the computation we send the resulting form out from the system by the rule $(\varepsilon, \$, \varepsilon, out)$.

Let w be a string in the skin region which contains some unmarked A and A'. If the letter A precedes A' then we can write $w = w_1 A \alpha_1 w_2 A' \alpha_2 w_3$. The simulation of the matrix is the following

$$\begin{aligned} w_1 A \alpha_1 w_2 A' \alpha_2 w_3 &\overset{r.1.1, r.2.1, r.3.1}{\Longrightarrow} w_1 A \#C_i \# BC \alpha_1 w_2 A' \alpha_2 w_3 \overset{r.3+i.1, r.3.2, r.2.2}{\Longrightarrow}\\ &w_1 A \#C_i \# BC \alpha_1 w_2 A' \#C_i' \#B'C' \alpha_2 w_3, \end{aligned}$$

where $w_1, w_2, w_3 \in V^*, \alpha_1, \alpha_2 \in V \backslash \{\#\}$. We can write a similar derivation if A' precedes A.

Hence, as result of simulation of i-th matrix we get both A and A' marked and BC, $B'C'$ inserted in the correct positions. The derivation in Π may terminate by the rule $(\varepsilon, \$, \varepsilon, out)$ only in the first membrane. Hence it guaranties that the simulation of each matrix has completed. According to the definition of $MorLSP$ the string w' belongs to the language if $w' = \varphi(h^{-1}(w))$, where w is the generated string. We may consider only the final derivations of Pi in which each nonterminal is marked. Hence, we have $L(G) \subseteq \varphi(h^{-1}(L(\Pi)))$.

The inverse inclusion is obvious since every rule in Π has its counterpart in G. Moreover the case when the derivation in Π is blocked corresponds to the case in which the simulation of a matrix cannot be completed.

Hence, we get $MAT^{\lambda} \subseteq MorLSP_*(ins_2^{1,1})$. □

Remark 3.5. One can mention that a similar result can be obtained with a smaller number of membranes at the price of the maximal length of inserted words. Precisely, for any context-free matrix grammar G' there is a P insertion system Π' such that $L(\Pi') \in MorLSP_{n+1}(ins_3^{1,1})$ and $L(G') = L(\Pi')$, where n is the number of matrices in G'. To prove this we can use the same argument as in the previous theorem and replace rules in $R_1, \ldots, R_{n+3}$ by

$$\begin{aligned} &(A, \#BC, \alpha, in_{i+1}) && \text{to } R_1, \\ &(A', \#B'C', \alpha, out) && \text{to } R_{i+1}, \end{aligned}$$

for every $\alpha \in V \backslash \{\#\}, i = 1, \ldots, n$.

Since trees are special cases of graphs we get the following result

Corollary 3.6. $MAT^{\lambda} \subseteq MorLStP_*(ins_2^{1,1})$.

Taking into account Lemma 3.4 and 3.2, and the fact that the class of languages generated by context-free matrix grammars is closed under inverse morphisms and weak codings we get a characterization of MAT :

Theorem 3.7. $MorLS[t]P_*(ins_*^{1,1}) = MAT^{\lambda}$.

Now we increase the maximal size of the context of insertion rules to two letters. It is known from [19] that $INS_2^{2,2}$ contains non-semilinear languages. By considering these systems with membrane regulation we obtain the following result

Theorem 3.8. $MorLSP_3(ins_2^{2,2}) = RE$.

Proof:

We prove the theorem by simulating a type 0 grammar in the modified Penttonen normal form. Let $G = (N, T, S, P)$ be such a grammar. Suppose that rules in P are ordered and $n = card(P)$.

Now consider the following insertion P system,

$$\Pi = (V, [_1\ [_2\ [_3\ ]_3\ ]_2\ ]_1, \{S\$\}, \emptyset, \emptyset, R_1, R_2, R_3, h, \varphi),$$

where $V = T \cup N \cup F \cup \overline{F} \cup \{\#, \overline{\#}, \$\}$, $F = \{F_A, \mid A \in N\}$, $\overline{F} = \{\overline{F}_A, \mid A \in N\}$.

We include into R_1 the following rules:

$$\begin{aligned} (AB, \#C, \alpha, here), &\text{ if } AB \longrightarrow AC \in P; \\ (A, \#C, B\alpha, here), &\text{ if } AB \longrightarrow CB \in P; \\ (A, C, \alpha, here), &\text{ if } A \longrightarrow AC \in P; \\ (\varepsilon, C, A\alpha, here), &\text{ if } A \longrightarrow CA \in P; \\ (A, \#\delta, \alpha, here), &\text{ if } A \longrightarrow \delta \in P; \\ (\$, \varepsilon, \varepsilon, out), & \end{aligned}$$

where $\alpha \in V\backslash\{\#\}$, $A, B, C \in N$, and $\delta \in T \cup \{\varepsilon\}$. It may happen that the pair of letters AB subjected to be rewritten by a production $AB \longrightarrow AC$ or $AB \longrightarrow CB \in P$ is separated by letters that have been marked. We use two additional membranes to transfer a letter over marked ones. In order to transfer $A \in N$ we add for each $\alpha \in V\backslash\{\#\}$

$$r.1.1 : (A, \#F_A, \alpha, in_2),$$

to the skin membrane. Then we add to the second membrane

$$\begin{aligned}
&r.2.1 : (F_A, \#A, \alpha', out), && r.2.2 : (\overline{F_A}, \overline{\#}A, \alpha', out),\\
&r.2.3 : (F_A X, \#\overline{F_A}, \#, in_3), && r.2.4 : (F_A\overline{F_B}, \overline{\#}F_A, \overline{\#}, in_3),\\
&r.2.5 : (F_A\overline{\#}, F_A, \alpha, in_3), && r.2.6 : (\overline{F_A}\#, F_A, \alpha, in_3),\\
&r.2.7 : (F_A\overline{\#}, F_A, \overline{\#}, in_3), && r.2.8 : (\overline{F_A}\#, F_A, \overline{\#}, in_3),\\
&r.2.9 : (F_A\overline{\#}, \overline{F_A}, \#, in_3), && r.2.10 : (\overline{F_A}\#, \overline{F_A}, \#, in_3),
\end{aligned}$$

for every

$$\begin{aligned}
&X \in F \cup N, \overline{F_B} \in \overline{F}, \alpha \in V\backslash\{\#, \overline{\#}\},\\
&\alpha' \in \{ab \mid a \in N \cup T, b \in N \cup T \cup \{\$\}\} \cup \{\$\}.
\end{aligned}$$

Finally, we add to the third membrane the rules

$$r.3.1 : (F_A, \#, \alpha, out), \qquad r.3.2 : (\overline{F_A}, \overline{\#}, \alpha', out),$$

for every $\alpha \in V\backslash\{\#\}, \alpha' \in V\backslash\{\overline{\#}\}$.

The morphism h and the weak coding φ are defined as

$$h(a) = \begin{cases} a, & \text{if } a \in V\backslash N,\\ a\# & \text{if } a \in N. \end{cases} \qquad \varphi(a) = \begin{cases} a, & \text{if } a \in T,\\ \varepsilon & \text{if } a \in V\backslash T. \end{cases}$$

We simulate productions in P by marking nonterminals from N and inserting corresponding right hand sides of the productions. This can be done with insertions in the skin membrane by rules of weight $(2, 2, 2)$ since the grammar has such a form that production rewrites/adds at most one letter.

The simulation of the transfer is done in the second and third membranes. The idea of the simulation is (1) to *mark* the nonterminal we want to transfer, (2) *jump* over the marked letters with the help of one special letter, at the end (3) *mark* the special letter and insert the original nonterminal. Since we use two letter contexts, in one step we can jump only over a single letter. We also need to jump over the marking letter # as well as over the marked nonterminals, and the letters inserted previously. In order to jump over # we introduce one additional marking symbol $\overline{\#}$. We mark letters from $\overline{F}$ by $\overline{\#}$, and all the other letters in $V\backslash\{\overline{\#}, \#\}$ by #, e.g., in a word $\overline{F_A}\#$, letter $\overline{F_A}$ is unmarked.

(1) The rule $r.1.1 : (A, \#F_A, \alpha, in_2)$, specifies that every unmarked letter from N may be subjected to the transfer.

(2) The rules $r.2.3 - r.2.10$ in the second membrane specify that F_A or $\overline{F_A}$ is copied to the right in such a way that inserted letters would not be marked. In order to do so, the appropriate rule chooses to insert either the overlined copy $\overline{F_A}$ or the simple copy F_A. The rules $r.2.3, r.2.4$ describe jumps over one letter not in $\{\#, \overline{\#}\}$, and $r.2.5 - r.2.10$ describe jumps over $\#$, $\overline{\#}$. Every rule $r.2.3 - r.2.10$ sends the sentential form to the third membrane, and the rules $r.3.1, r.3.2$ in the third membrane send the sentential form back to the second membrane after marking one symbol $F_A \in F$ or $\overline{F_A} \in \overline{F}$.

(3) The rules $r.2.1$ and $r.2.2$ may terminate the transferring procedure and send the sentential form to the first membrane if letter \$ or two letters from $\{ab \mid a \in N \cup T, b \in N \cup T \cup \{\$\}\}$ appear in the right context.

For example, consider the transfer of A in the string $AX\#C\$$ (here, we underline inserting substrings)

$$\begin{aligned} AX\#C\$ &\overset{r.1.1}{\Longrightarrow} A\underline{\#F_A}X\#C\$ \overset{r.2.3}{\Longrightarrow} A\#F_AX\underline{\#\overline{F_A}}\#C\$ \overset{r.3.1}{\Longrightarrow} \\ &A\#F_A\underline{\#}X\#\overline{F_A}\#C\$ \overset{r.2.6}{\Longrightarrow} A\#F_A\#X\#\overline{F_A}\underline{\#F_A}C\$ \overset{r.3.2}{\Longrightarrow} \\ &A\#F_A\#X\#\overline{F_A}\ \underline{\overline{\#}}\#F_AC\$ \overset{r.2.1}{\Longrightarrow} A\#F_A\#X\#\overline{F_A}\ \overline{\#}\#F_A\underline{\#A}C\$. \end{aligned}$$

The sentential form preserves the following property: (i) The first membrane does not contain unmarked letters from $F \cup \overline{F}$; there is exactly one unmarked letter from $F \cup \overline{F}$ in the second membrane; and there are always two unmarked letters from $F \cup \overline{F}$ in the third membrane.

We mention that property (i) is preserved by every derivation. Indeed, we start derivation from the axiom $S\$$ that satisfies the property, then one unmarked symbol is inserted by $r.1.1$. Rules $r.2.3 - r.2.12$ always add one more unmarked letter, whereas rules $r.2.1, r.2.2, r.3.1, r.3.2$ always mark one letter from $F \cup \overline{F}$.

In order to verify that Π generates the same language as G we note that every reachable sentential form in G will be reachable also in Π by simulating the same production.

We also note that the derivation in Π may terminate by the rule $(\$, \varepsilon, \varepsilon, out)$ only in the first membrane. Hence, every transfer will be completed. It follows from property (i) that the simulation of the transfer is deterministic in the second membrane. Also note, that there is a nondeterministic choice in the third membrane, where the rules $r.3.1, r.3.2$ may mark one of the two unmarked letters. In the case the rule marks the rightmost letter, the derivation has to "jump" again over the inserted letter.

In a special case if $r.1.1$ starts the transfer of a letter adjacently left from an unmarked one then the rules $r.1.1, r.2.1$ produce two marked symbols which do not affect the result of the simulation.

The output string w is in the language, iff $w' = \varphi(h^{-1}(w))$ is defined. Hence, the resulting output of Π does not contain unmarked nonterminals. On the other hand every final derivation in Π has its counterpart in G. By applying the inverse morphism h^{-1} we filter out every sentential form with unmarked nonterminals from N. Hence, the corresponding derivation in G is completed. Finally, the weak coding φ filter away all supplementary letters. Hence, we have $L(G) = L(\Pi)$. □

4 Conclusions

This article investigates the generative power of insertion P systems with encodings. The length of insertion rules and number of membranes are used as parameters of the descriptional complexity of the system. In the article we exploit the fact that a morphism and a weak coding are incorporated into insertion P systems. The obtained family $MorLS[t]P_*(ins_*^{1,1})$ characterizes the matrix languages. When no membranes are used, the class $MorINS_*^{1,1}$ is equal to the family of context-free languages. We proved also the universality result regarding the family $MorLSP_*(ins_*^{2,2})$. Also, for the family $LSP_*(ins_*^{2,2})$ one can get an analogous computational completeness result by applying right(left) quotient with respect to a regular language.

We recall the open problem posed in [4], namely, whether $MorINS_*^{2,2}$ is computationally complete. Our work gives a partial solution of the problem by using the membrane computing framework. One may see that in order to solve the problem completely (by the technique

promoted in the article), it is enough to find a concise way to transfer a letter over a marked context. In our case this can be reduced to the question whether it is possible to compute the membrane regulation in the skin membrane.

One may mention that there is a trade-off between the number of membranes and the maximal length of productions. By introducing additional nonterminals and fitting the grammars into the normal forms we decrease the amount of membranes used. On the other hand by raising the number of membranes we can simulate larger production rules. Moreover, the descriptional complexity used in the paper may be extended by such internal system parameters as, e.g., the size of the alphabet, the number of rules per membrane, etc. It may be promising to continue the research of the minimal systems regarding these parameters. We are also interested in the computational power of the insertion P systems having only one sided (right or left) contexts.

Acknowledgments

The author acknowledges the support PIF program from University Rovira i Virgili, and project no. MTM2007-63422 from the Ministry of Science and Education of Spain. The author sincerely thanks the help of Yurii Rogozhin and Seghey Verlan without whose supervision the work would not been done. Also the author warmly thanks Gheorghe Păun and Erzsèbet Csuhaj-Varjú who draw attention, during the meeting BWMC-09, to the point of using different normal forms of grammars to simulate P insertion-deletion systems.

Bibliography

[1] M. Daley, L. Kari, G. Gloor, R. Siromoney, Circular Contextual Insertions/Deletions with Applications to Biomolecular Computation, In: *Proc. SPIRE'99*, pp. 47-54, 1999.

[2] L. Kari, From Micro-soft to Bio-soft: Computing with DNA, In:*Proc. of Biocomputing and Emergent Computations*, 146-164, 1997.

[3] L. Kari, Gh. Păun, G. Thierrin, S. Yu, At the Crossroads of DNA Computing and Formal Languages: Characterizing RE Using Insertion-Deletion Systems. In: *Proc. of 3rd DIMACS*, pp. 318-333, 1997.

[4] L. Kari, P. Sosík, On the Weight of Universal Insertion Grammars, *TCS*, 396(1-3), pp. 264-270, 2008.

[5] L. Kari, G. Thierrin, Contextual Insertion/Deletion and Computability, *Information and Computation*, 131, 1, pp. 47-61, 1996.

[6] A. Krassovitskiy, Yu. Rogozhin, S. Verlan, Further Results on Insertion-Deletion Systems with One-Sided Contexts, In *Proc. of 2nd LATA08*, *LNCS*, 5196, pp. 333-344, 2008.

[7] A. Krassovitskiy, Yu. Rogozhin, S. Verlan, One-Sided Insertion and Deletion: Traditional and P Systems Case, *CBM08*, pp. 53-64, 2008.

[8] A. Krassovitskiy, Yu. Rogozhin, S. Verlan, Computational Power of P Systems with Small Size Insertion and Deletion Rules, In:*Proc of CSP08*, pp. 137-148, 2008.

[9] S. Marcus, Contextual Grammars, *Rev. Roum. Math. Pures Appl.*, 14, pp 1525-1534, 1969.

[10] M. Margenstern, Gh. Păun, Yu. Rogozhin, S. Verlan, Context-Free Insertion-Deletion Systems, *TCS*, 330, pp. 339-348, 2005.

[11] C. Martin-Vide, Gh. Păun, A. Salomaa, Characterizations of Recursively Enumerable Languages by Means of Insertion Grammars, *TCS*, 205, 1–2, pp. 195-205, 1998.

[12] A. Matveevici, Yu. Rogozhin, S. Verlan, Insertion-Deletion Systems with One-Sided Contexts, *LNCS*, 4664, pp. 205-217, 2007.

[13] M. Mutyam, K. Krithivasan, A. S. Reddy, On Characterizing Recursively Enumerable Languages by Insertion Grammars, *Fundamenta Informaticae*, 64(1-4), pp. 317-324, 2005.

[14] K. Onodera, New Morphic Characterization of Languages in Chomsky Hierarchy Using Insertion and Locality, In *Proc. of 3rd LATA09*, *LNCS*, 5457, pp. 648–659, 2009.

[15] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, Vol. 61, 1, pp. 108-143, 2000.

[16] Gh. Păun, Marcus Contextual Grammars, Kluwer, Dordrecht, 1997.

[17] Gh. Păun, Membrane Computing. An Introduction, Springer–Verlag, Berlin, 163, pp. 226-230, 2002.

[18] Gh. Păun, M. J. Pérez-Jiménez, T. Yokomori, Representations and Characterizations of Languages in Chomsky Hierarchy by Means of Insertion-Deletion Systems, *International Journal of Foundations of Computer Science*, 19, 4, pp. 859-871, 2008.

[19] Gh. Păun, G. Rozenberg, A. Salomaa, DNA Computing. New Computing Paradigms, Springer–Verlag, Berlin, 1998.

[20] G. Rozenberg, A. Salomaa, eds., Handbook of Formal Languages, Springer–Verlag, Berlin, 1997.

[21] A. Takahara, T. Yokomori, On the Computational Power of Insertion-Deletion Systems, *DNA 2002*, Sapporo, *LNCS*, 2568, pp. 269–280, 2003.

[22] S. Verlan, On Minimal Context-Free Insertion-Deletion Systems, *Journal of Automata, Languages and Combinatorics*, 12, 1/2, pp. 317–328, 2007.