On the Power of Insertion P Systems of Small Size

Alexander Krassovitskiy

Research Group on Mathematical Linguistics,
Rovira i Virgili University
Av. Catalunya, 35, Tarragona 43002, Spain
alexander.krassovitskiy@estudiants.urv.cat

Summary. 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 matrix grammars. If contexts of length two are permitted, then any recursively enumerable language can be generated. In both cases an inverse morphism and a weak coding were applied to the output of the corresponding P systems.

1 Introduction

The study of insertion-deletion operations on strings has a long history. We just mention [18], [5], [7], [13], [15]. Motivated from molecular computing they have been studied in [1], [6], [17], [19], [12]. With some linguistic motivation they may be found in [11] and [3].

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.

Some combinations of parameters lead to systems which are not computationally complete [14], [8] or even decidable [20]. It was shown that the operations of insertion and deletion considered in P systems framework can easily increase the computational power with respect to ordinary insertion-deletion systems [9], [10].

Traditionally, language generating devices having only insertion rules were studied. Early computational models based only on insertion appear already in [11], and are discussed in [17] and [16] (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 non-semilinear languages (see Theorem 6.5 in [17]). On the other hand, it appears that by using only insertion operation the obtained language classes with context greater than one are incomparable with many known language classes. For example there is a simple linear language $\{a^n ba^n \mid n \geq 1\}$ which cannot be generated by any insertion system (see Theorem 6.6 in [17]).

In order to overcome this obstacle one can use some codings to "interpret" generated strings. In [17], in a natural way, there were used two additional string operations: a morphism h and a weak coding φ. The result is considered as a product of application $h^{-1} \circ \varphi$ on the generated strings. Clearly, the obtained languages have greater expressivity and the corresponding language class is more powerful. 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 [17] 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 [2], showing that rules having at most 5 letter context 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.

In this article we consider 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 in MAT and equals CF 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 (of course, using the inverse morphism and the weak coding).

2 Prerequisites

All formal language notions and notations we use here are elementary and standard. The reader can consult any of the many monographs in this area (see details, e.g., in [18]).

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

An *insertion-deletion system* is a construct $ID = (V, T, A, I, D)$, where V is an alphabet, $T \subseteq V$, A is a finite language over V, and I, D are finite sets of triples of the form (u, α, v), where u, α, and v are strings over V. The elements of T are *terminal* letters (in contrast, those of $V \backslash T$ are called nonterminals), those of A are *axioms*, the triples in I are *insertion rules*, and those from D are *deletion rules*. An insertion rule $(u, \alpha, v) \in I$ indicates that the string α can be inserted in between u and v, while a deletion rule $(u, \alpha, v) \in D$ indicates that α can be

removed from the context (u, v). As stated otherwise, $(u, \alpha, v) \in I$ corresponds to the rewriting rule $uv \to u\alpha v$, and $(u, \alpha, v) \in D$ corresponds to the rewriting rule $u\alpha v \to uv$. We denote by $\Longrightarrow_{ins}$ the relation defined by an insertion rule (formally, $x \Longrightarrow_{ins} y$ iff $x = x_1uvx_2, y = x_1u\alpha vx_2$, for some $(u, \alpha, v) \in I$ and $x_1, x_2 \in V^*$) and by $\Longrightarrow_{del}$ the relation defined by a deletion rule (formally, $x \Longrightarrow_{del} y$ iff $x = x_1u\alpha vx_2, y = x_1uvx_2$, for some $(u, \alpha, v) \in D$ and $x_1, x_2 \in V^*$). We refer by $\Longrightarrow$ to any of the relations $\Longrightarrow_{ins}, \Longrightarrow_{del}$, and denote by $\Longrightarrow^*$ the reflexive and transitive closure of $\Longrightarrow$ (as usual, $\Longrightarrow^+$ is its transitive closure).

The language generated by ID is defined by

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

A (pure) *insertion systems* of weight (n, m, m') is a construct $ID = (V, A, I)$, where V is a finite alphabet, $I \subseteq V^*$ is a finite set of axioms, I is a finite set of insertion rules of the form (u, α, v), for $u, \alpha, v \in V^*$, and

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

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

An *insertion P system* is the following construct:

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

where

- V is a finite alphabet,
- μ is the membrane (tree) structure of the system which has n membranes (nodes). This structure will be represented by a word containing correctly nested marked parentheses.
- 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 $\{here, in_j, out\}, 1 \leq j \leq k$.

Any k-tuple $(N_1, \ldots, N_k)$ of languages over V is called a configuration of Π. 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 possible strings from the corresponding regions, and following the target indications associated with the rules. We assume that every string represented in a membrane has arbitrary many copies. Hence, by applying a rule to a string we get both arbitrary many copies of resulted string as well as old copies of the same string. More specifically, if $w \in N_i$

and $r = (u, x, v; tar) \in R_i$, such that $w \Longrightarrow^r w'$ then w' will go to the region indicated by tar. If $tar = here$, then the string remains in N_i, if $tar = out$, then the string is moved to the region immediately outside the membrane i (maybe, in this way the string leaves the system), if $tar = in_j$, then the string is moved to the region j.

A sequence of transitions between configurations of a given insertion P system Π, starting from the initial configuration $(M_1, \ldots, M_n)$, is called a computation with respect to Π. The result of a computation consists of all strings over V which are sent out of the system at any time during the computation. We denote by $L(\Pi)$ the language of all strings of this type. We say that $L(\Pi)$ is generated by Π.

The *insertion-deletion tissue P systems* are defined in an analogous manner. As the tissue P systems use arbitrary graph structure we write the target indicator in the form $tar = go_j, j = 1, \ldots, k$. 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 an insertion P system. We denote by $LSP_k(ins_n^{m,m'})$ (see, for example [16]) the family of languages $L(\Pi)$ generated by insertion P systems of degree at most $k \geq 1$ having the 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 $hINS_n^{m,m'}$ (from $hLSP_k(ins_n^{m,m'})$, correspondingly) if there exist a morphism h, weak coding φ and $L(\Pi) \in INS_n^{m,m'}$ ($L(\Pi) \in LSP_k(ins_n^{m,m'})$ such that $\varphi(h^{-1}(L(\Pi))) = L'$.

We write an instance of the system $hLSP$ in the form

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

where

- h is a morphism $h : V \to V^+$,
- φ is a weak coding $\varphi : T \to T \cup \{\varepsilon\}$,
- other components are defined as for insertion P system.

We insert "t" before P to denote classes corresponding to the tissue cases (e.g., $hLStP$). Insertion (t) holds for both tissue and tree membrane structure.

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 [16]. The technique works as follow: in order to simulate rewriting production $A \to B$ we add adjacently right from A the word $\#B$ specifying that letter A is already rewritten. And, as soon as the derivation is completed, every pair $A\#$ in the sentential form is subject to the inverse morphism.

3 Main Results

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

Theorem 1 $hINS_*^{1,1} = CF$

Proof. First we show that $CF \subseteq hINS_3^{1,1}$.

Let $G = (V, T, S, P)$ be a context-free grammar in Chomsky normal form. Consider the following system from $hINS_3^{1,1}$

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

where $R = \{(A, \#BC, \alpha) \mid \alpha \in T \cup V, A \to BC \in P\}$, the morphism h

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

and the weak coding φ

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

We claim that $L(\Pi) = L(G)$. Indeed, each rule $(A, \#BC, \alpha) \in R$ can be applied in the sentential form $wA\alpha w'$ if A is unmarked (not rewritten). Thus, the production $A \to BC \in P$ can be applied in the corresponding derivation G. Hence, by applying the counterpart rules we get equivalent derivations.

At the end of derivation, by applying the inverse morphism h^{-1} we warranty that every nonterminal is marked. Finally, we delete every nonterminal by the weak coding φ. Hence $L(\Pi) = L(G)$, and we get $CF \subseteq hINS_3^{1,1}$.

The equivalence of the two classes follows from Theorem 6.4 in [17] stating $INS_*^{1,1} \subseteq CF$ and the fact that context-free languages are closed under inverse morphisms and weak codings.

Now we consider insertion systems with contexts controlled by membranes (P systems). It is known from Theorem 5.5.1 in [16] that $LSP_2(ins_2^{1,1})$ contains non context-free languages. We show that this class is bounded by matrix grammars:

Lemma 2 $LStP_*(ins_*^{1,1}) \subset MAT$.

Proof. The proof uses the similar technique presented in [17], Theorem 6.4 for context-free grammars.

Let $\Pi = (V, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n)$ be a system from $LStP_n(ins_*^{1,1})$ for some $n \geq 1$.

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

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

2. For every rule $(a, \varepsilon, c, in_j) \in R_i, a, c \in V \cup \{\varepsilon\}, k > 0$ we add to P $(Q_i \to Q_j, D_{\overline{a},\overline{c}} \to D_{\overline{a},\overline{c}})$, where $\overline{a}$ and $\overline{c}$ defined as in the previous case.
3. Next, for every $w = b_1 \cdots_k \in M_i, i = 1, \dots, n, k > 0$ we add to P the matrix $(S \to Q_i D_{\varepsilon,b_1} D_{b_1,b_2} \dots D_{b_{k-1},b_k} D_{b_k,\varepsilon})$.
4. In special case if $\varepsilon \in M_i$ we add $(S \to Q_i D_{\varepsilon,\varepsilon})$ to P.
5. Also, for every $D_{a,b} \in D, a, b \in V \cup \{\varepsilon\}$ we add $(D_{a,b} \to a)$ to P.
6. Finally, we add $(p_1 \to \varepsilon)$ 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 nonterminals 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 $(a, b_1 \dots b_k, c, in_j) \in R_i, a, c \in V, b_1 \dots b_k \in V^k$, can be simulated by the grammar iff the sentential form contains both Q_i and $D_{a,c}$. It results the label of current cell is rewritten to Q_j and $D_{a,c}$ is rewritten to the string $D_{a,b_1} D_{b_1,b_2} \dots D_{b_{k-1},b_k} D_{b_k,a}$. Clearly, the string preserves one symbol context. In order to treat those rules which have no context we introduce productions that preserve arbitrary context ($\overline{a} \in V \cup \{\varepsilon\}$ and $\overline{c} \in V \cup \{\varepsilon\}$).

The simulation of the grammar starts with a nondeterministic choice of the axiom. Then, during the derivation any rule corresponding to the context (a, b) have to be applied (in a one to one correspondence with grammar productions). 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, $LStP_*(ins_*^{1,1}) \subseteq MAT$.

The strictness of the inclusion follows from the fact there are languages from MAT which cannot be generated by any insertion P system from $LStP_*(ins_n^{1,1})$, for any $n \geq 1$. Indeed, consider $L_a = \{ca^k ca^k c \mid k \geq 1\}$. One may see that the matrix grammar $(\{S_l, S_r, S\}, \{a, b\}, S, P')$ generates L_a, where

$$\begin{aligned} P' = \{&(S \to cS_l cS_r c), \\ &(S_l \to \varepsilon, S_r \to \varepsilon), \\ &(S_l \to aS_l, S_r \to aS_r)\}. \end{aligned}$$

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. We note that the system cannot delete or rewrite any letter so every insertion is terminal. And, as the language of axioms is finite, we need an insertion rule of letter a. Consider alternatives for a final insertion step in a derivation which has at most one step and derives a word $ca^k ca^k c$, for some $k > n$: (1) the last applied rule inserts the central letter c, or (2) it does not insert the

central letter c. (1) The central c can be inserted between any two letters a. So we get a contradiction because the prefix ca^pc may not be equal to the suffix ca^qc.
(2) The last applied rule can (2.1) either insert the letter c(at the end or start of the string) or, (2.2) no c is inserted by the final rule.
(2.1) We get a contradiction because c can be alternatively inserted in between two a as we assumed $k > n$.
(2.2) The last rule cannot distinguish whether to insert a before the central c or after the central c. So, again, we get a contradiction because the prefix ca^pc may not be equal to the suffix ca^qc.

So we proved $L_a \notin LStP_*(ins_n^{1,1})$, for any $n \geq 1$ and hence $LStP_*(ins_*^{1,1}) \subset MAT$.

Corollary 3 $LSP_*(ins_*^{1,1}) \subset MAT$.

Proof. A tree is a special case of a graph.

Lemma 4 $MAT \subseteq hLSP_*(ins_2^{1,1})$.

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

Consider a system $\Pi \in hLSP_{n+3}(ins_2^{1,1})$,

$$\Pi = (V, [_1\ [_2\ [_3\ \left(\prod_{i=1,\ldots,n} [_{i+3}\ ]_{i+3}\right] _3\ ]_2\ ]_1, \{S\$\}, \emptyset, \ldots, \emptyset, R_1, \ldots, R_{n+3}, h, \varphi),$$

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

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

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

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

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

$$h(a) = \begin{cases} a, & \text{if } a \in T, \\ a\# & \text{if } a \in V \end{cases}$$

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

We claim that $L(\Pi) = L(G)$. To do so 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 $(A \to BC, A' \to B'C') \in P$. The simulation is controlled by letters C_i and C'_i. First, we insert $\#C_i$ in the context of a 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 the sentential form to the $i+3$ membrane. Here we choose nondeterministically one letter A', mark it, write adjacently right new control letter C'_i, and, after that, send the obtained string to the third membrane. We mention that it is not possible to apply the rule $r.i+3.1 : (A', \#C'_i, \alpha; out)_e$ in the $i+3$ membrane and to reach the skin membrane if the sentential form does not contain the 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 sentential form is sent to the second membrane. Finally, we mark C'_i and send the resulting string back to the skin membrane.

We assume that at the beginning of this 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 the assumption, as after the sentential form is sent back to the skin membrane the introduced C_i, and C'_i are marked. At the end of computation we send the obtained sentential form out of 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 B, then we can write $w = w_1 A \alpha_1 w_2 A' \alpha_1 w_3$. The simulation of the matrix is the following

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

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

Hence, as a result of the simulation of i-th matrix we get both A and A' marked and BC, $B'C'$ inserted in the right positions. The derivation in Π may terminate by the rule $(\varepsilon, \$, \varepsilon, out)$ only in the first membrane. This guarantees that the simulation of each matrix has been finished. According to the definition of Π the string w' belongs to the language if $w' = \varphi(h^{-1}(w))$, where w is the generated string. This is the case only if the resulting output of Π does not contain unmarked nonterminals. Hence we proved $L(G) \subseteq \varphi(h^{-1}(L(\Pi)))$.

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

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

Remark 5 *One can mention that a similar result can be obtained with a smaller number of membranes at the cost of increasing the maximal length of inserted*

words. I.e., for any grammar G' from MAT there is a P insertion system Π' corresponding to $hLSP_{n+1}(ins_3^{1,1})$ such that $L(G) = L(\Pi')$, and 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 $(r..*)$ by*

$$\begin{aligned} &(A, \#BC, \alpha, in_{i+1}), \alpha \in V\backslash\{\#\} && \text{to } R_1 \\ &(A', \#B'C', \alpha, in_1), \alpha \in V\backslash\{\#\} && \text{to } R_{i+1}. \end{aligned}$$

Corollary 6 $MAT \subseteq hLStP_*(ins_2^{1,1})$.

Proof. Obvious, since a tree is a special case of a graph.

Taking into account Lemma 4, Lemma 2, and the fact that the class of matrix grammars is closed under inverse morphisms and weak codings we get the following characterization of MAT:

Theorem 7 $hLS(t)P_*(ins_*^{1,1}) = MAT$.

Next we consider computationally complete insertion P systems. In order to use concise representations of productions in 0-type grammars we need an auxiliary lemma.

Lemma 8 *For every 0-type grammar $G' = (N', T, S', P')$ there exists a grammar $G = (N, T, S, P)$ such that $L(G) = L(G')$ and every production in P has the form*

$$\begin{aligned} AB \to AC \text{ or} \qquad & AB \to CB \text{ or} && (1) \\ A \to AC \text{ or} \qquad & A \to CA \text{ or} && (2) \\ A \to \delta, & && (3) \end{aligned}$$

where A, B and C are from N and $\delta \in T \cup N \cup \{\varepsilon\}$.

Proof. To prove the lemma it is enough to show that for any grammar in Penttonnen normal form there is an equivalent grammar having productions of the form (1)–(3). To do so it is enough to simulate the context-free productions $A \to BC$ by productions of the form (1)–(3).

Let $G = (N, T, S, P)$ be a grammar in Penttonnen normal form whose production rules in P are of the form:

$$\begin{aligned} AB \to AC \text{ or} \\ A \to BC \text{ or} \\ A \to \alpha \end{aligned}$$

where A, B, C and D are from N and $\alpha \in T \cup N \cup \{\varepsilon\}$.

Let $P_{CF} \subseteq P$ denotes the set of all context-free productions $A \to BC \in P$ such that $B \neq C$. Suppose that rules in P_{CF} are ordered and $n = card(P_{CF})$.

Consider a grammar $G' = (N', T, S, P')$, where $N' = N \cup \{X_i, Y_i, Z_i \mid i = 1, \ldots, n\}$, $P' = (P\backslash P_{CF}) \cup P'_{CF}$, and P'_{CF} is constructed as follows: for every $i : A \to BC \in P_{CF}$ add to P'_{CF} the following productions

$$r.i.1 : A \to X_i, \qquad r.i.2 : X_i \to X_iY_i,$$
$$r.i.3 : X_iY_i \to Z_iY_i, \qquad r.i.4 : Z_iY_i \to Z_iC$$
$$r.i.5 : Z_iC \to BC$$

Clearly, the obtained grammar has the form specified by (1)–(3). Now we prove that $L(G) = L(G')$. The inclusion $L(G) \subseteq L(G')$ is obvious as for every derivation in G we use its counterpart derivation in G' replacing i-th context-free production from P_{CF} by the sequence of productions $r.i.1, r.i.2, r.i.3, r.i.4, r.i.5$:

$$wAw' \overset{r.i.1}{\Longrightarrow} wX_iw' \overset{r.i.2}{\Longrightarrow} wX_iY_iw' \overset{r.i.3}{\Longrightarrow} wZ_iY_iw' \overset{r.i.4}{\Longrightarrow} wZ_iCw' \overset{r.i.5}{\Longrightarrow} wBCw'.$$

In order to prove that $L(G') \subseteq L(G)$ we show that for every terminal derivation in G' we can construct a derivation in G so that they both produce the same word. We use the counterpart productions from $P \backslash P_{CF}$ to mimic analogous production. For the productions $P_{CF'}$ we show that any deviation from the above defined sequence does not produce any new terminal derivation. First, we mention that the sequence of productions corresponding to $i : A \to BC$ starts by rewriting A on the new nonterminal, so, other productions not in $r.i.*$ cannot interfere the sequence. Yet, the production rule $r.i.2$ may generate extra Y_i (for simplicity, we assume one extra Y_i generated).

$$wAw' \overset{r.i.1}{\Longrightarrow} wX_iw' \overset{(r.i.2)^2}{\Longrightarrow} wX_i(Y_i)^2w' \overset{r.i.3,r.i.4}{\Longrightarrow} \overset{r.i.5}{\Longrightarrow} wBCY_iw'.$$

As we need to consider only terminal derivations we may assume that Y_i will be necessary rewritten. The only rule to rewrite Y_i is $r.i.4$. In order to perform it the letter Z_i must precede by the letter Y_i. It implies that the letter A must appear adjacently left from the letter Y_i. Then the sequence of productions $r.i.1, r.i.3, r.i.4, r.i.5$ results to the same sentential form as if $r.i.2$ is applied once per every rewriting of A.

$$wBCY_iw' \Longrightarrow^* w_1AY_iw_1' \overset{r.i.1}{\Longrightarrow} wX_iY_iw_1' \overset{r.i.3,r.i.4,r.i.5}{\Longrightarrow} w_1BCw_1'.$$

Therefore, it can be produced by rules from P_{CF}. We also mention that in $r.i.1 - r.i.5$ we start rewriting letters from $N \backslash N'$ by corresponding letters from N only after the letter X_i is rewritten. This imply that after $r.i.4$, we cannot insert additional Y_i adjacently left from C_i. So, Z_i can be rewritten unambiguously.

Finally, consider the case when the production $r.i.4$ is followed by some production from $P \backslash P_{CF}$ which rewrites C.

$$\ldots \overset{r.i.4}{\Longrightarrow} wZ_iC_iw' \Longrightarrow^* w_2Z_iC_iw_2' \overset{r.i.5}{\Longrightarrow} w_2BC_iw_2'.$$

As Z_i can be rewritten only if C_i appears to the right we may consider the equivalent derivation with the production $r.i.5$ applied directly after $r.i.4$. So the derivation is equivalent to the derivation with $i : A \to BC \in P_{CF}$. Hence we proved $L(G') \subseteq L(G)$, and hence $L(G') = L(G)$.

Every grammar in the normal form has the following property: every production can rewrite/add at most one (nonterminal) letter.

Now we increase the maximal size of the context of insertion rules to two letters. It is known from [17] that the class $INS_2^{2,2}$ contains non-semilinear languages. Considering these systems with membrane regulation we get

Theorem 9 $hLSP_3(ins_2^{2,2}) = RE$.

Proof. We prove the theorem by simulating a 0-type grammar in the normal form from Lemma 8. 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,

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

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

We include into R_1 the following rules:

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

where $\alpha \in V \backslash \{\#\}$. It may happen that the pair of letters AB subjected to be rewritten by a production $AB \to AC$ or $AB \to CB \in R$ 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

$$r.1.1 : (A, \#F_A, \alpha, in_2), \alpha \in V \backslash \{\#\}$$

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

$$\begin{aligned}
&r.2.1 : (F_A, \#A, \alpha', out), && r.2.2 : (\overline{F_A}, \overline{\#}A, \alpha', out),\\
&r.2.3 : (F_A X, \#\overline{F_A}, \#, in_3), && r.2.4 : (F_A\overline{F_B}, \overline{\#}F_A, \overline{\#}, in_3)\\
&r.2.5 : (\overline{F_A}X, \#\overline{F_A}, \#, in_3), && r.2.6 : (\overline{F_A}\ \overline{F_B}, \overline{\#}F_A, \overline{\#}, in_3)\\
&r.2.7 : (F_A\overline{\#}, F_A, \alpha, in_3), && r.2.8 : (\overline{F_A}\#, F_A, \alpha, in_3),\\
&r.2.9 : (F_A\overline{\#}, F_A, \overline{\#}, in_3), && r.2.10 : (\overline{F_A}\#, F_A, \overline{\#}, in_3),\\
&r.2.11 : (F_A\overline{\#}, \overline{F_A}, \#, in_3), && r.2.12 : (\overline{F_A}\#, \overline{F_A}, \#, in_3),
\end{aligned}$$

for every

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

Finally, we add to the third membrane the rules

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

The morphism h is defined by

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

The weak coding φ is defined by

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

We simulate the productions of P in the skin membrane by marking nonterminals from N and inserting corresponding letters of the productions. This is possible to do with insertion rules of weight $(2, 2, 2)$ since the grammar has such a form so every production rewrite/add at most one letter.

The simulation of the transfer is done in the second and the third membranes. The idea of the simulation is (1) to *mark* nonterminal we want to transfer, (2) *jump* over the marked letters with 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. Also we need to jump over the marking letter # as well as over marked nonterminals, and the letters inserted previously. We use letters $F_A \in F$ and $\overline{F_A} \in \overline{F}$ to keep information about the letter A we want to transfer. In order to jump over # we introduce one additional marking symbol $\overline{\#}$. We mark letters from $\overline{F}$ by $\overline{\#}$, and all other letters in $V \backslash \{\overline{\#}, \#\}$ by #. E.g., in a words $\overline{F_A}\#$, 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 used for the transfer.

(2) The rules $r.2.3 - r.2.12$ in the second membrane specify that F_A or $\overline{F_A}$ is copied to the right in such a way that the inserted letter 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 $r.2.3 - r.2.6$ describe the possible jumps over one letter not in $\{\#, \overline{\#}\}$, and $r.2.7 - r.2.12$ describe the possible jumps over the marking letters #, $\overline{\#}$. These rules send the sentential form to the third membrane. The rules in the third membrane mark one symbol $F_A \in F$ or $\overline{F_A} \in \overline{F}$ and send the sentential form back to the second membrane.

(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 the right context.

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

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

The sentential form preserves the following *invariant*:

- The first membrane does not contain unmarked letters from $F \cup \overline{F}$.
- There is exactly one unmarked letter $F \cup \overline{F}$ in the second membrane.
- There are always two unmarked letters from $F \cup \overline{F}$ in the third membrane.

We mention that the invariant is preserved by every derivation. Indeed, we start derivation from the axiom $S\$$ that satisfies the invariant, then one unmarked symbol is inserted by $r.1.1$. Rules $r.2.3 - r.2.12$ always add one more unmarked letter. And 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 simulates 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.

Also we note that the derivation in Π may terminate by the rule $(\$, \varepsilon, \varepsilon, out)$ only in the first membrane. Hence it guarantees that every transfer will be completed. It follows from the invariant that the simulation of the transfer is deterministic in the second membrane. There is a nondeterministic choice in the third membrane, where corresponding rules 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. The transfer satisfies the property that every terminating sequence replaces a nonterminal via arbitrary large string of marked letters, if it starts (by $r.1.1$) adjacently left from it. And in case the rule $r.1.1$ starts the transfer of a letter next to unmarked letter then it produces two marked symbols which do not affect on the result of the simulation.

The output string w is in the language only if $w' = \varphi(h^{-1}(w))$ is defined. This is the case only if 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 terminated. Finally, the weak coding φ filters away supplementary letters. Hence we have $L(G) = L(\Pi)$.

Remark 10 *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 grammar into the normal form we decrease the amount of used membranes. It is also the case in the other way: by growing the number of membranes we can simulate larger production rules.*

4 Conclusion

This article investigates the expressive power of insertion P systems with encodings. The length of insertion rules and number of membranes are used as a measure of descriptional complexity of the system. In the article we use the fact that morphisms and weak codings are incorporated to insertion P systems. The obtained family $hLS(t)P_*(ins_*^{1,1})$ serves to characterize matrix languages. When no membranes are used, the class $hINS_*^{1,1}$ equals the family of context-free languages. We proved the universality for the family $hLSP_*(ins_*^{2,2})$. More precisely, for every recursively enumerable language we can construct an insertion P system of weight $(2, 2, 2)$ and degree 3, so that applying an inverse morphism and a weak coding we generate the same language. Also, we want to mention that computational completeness of considered families indicates some trade-offs between descriptional complexity measures. Moreover, the descriptional complexity used in the paper may be extended by internal system parameters as, e.g., the size of alphabet, the number of rules per membrane, etc.

Also, it seems quite promising to investigate decidable computational properties of the language family $LS(t)P_*(ins_*^{*,*})$. We conjecture that it is incomparable with many known language families.

We recall the open problem posed in [4], namely, whether $hLSP_1(ins_*^{2,2})$ equals RE One may see that in order to solve the problem by the technique used 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.

Acknowledgments

The author acknowledges the support PIF program of University Rovira i Virgili, and projectno. 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. The author also warmly thanks Gheorghe Păun and Erzsèbet Csuhaj-Varjú who draw attention, during BWMC-09, to the possibility of using different normal forms of grammars to simulate P insertion-deletion systems.

References

1. M. Daley, L. Kari, G. Gloor, R. Siromoney: Circular contextual insertions/deletions with applications to biomolecular computation. *Proc. SPIRE'99*, 1999, 47–54.
2. M. Mutyam, K. Krithivasan, A. Siddhartha Reddy: On characterizing recursively enumerable languages by insertion grammars. *Fundam. Inform.*, 64, 1-4 (2005), 317–324
3. B.S. Galiukschov: Semicontextual grammars. *Matematika Logica i Matematika Linguistika*, Tallin University, 1981 38–50 (in Russian).

4. L. Kari, Petr Sosík: On the weight of universal insertion grammars. *Theoretical Computer Sci.*, 396, 1-3 (2008), 264–270.
5. L. Kari: From micro-soft to bio-soft. Computing with DNA. *Proc. on Biocomputing and Emergent Computations*, 1997, 146–164.
6. L. Kari, Gh. Păun, G. Thierrin, S. Yu: At the crossroads of DNA computing and formal languages: characterizing RE using insertion-deletion systems. *Proc. of 3rd DIMACS*, 1997, 318–333.
7. L. Kari, G. Thierrin: Contextual insertion/deletion and computability. *Information and Computation*, 131, 1 (1996), 47–61.
8. A. Krassovitskiy, Yu. Rogozhin, S. Verlan: Further results on insertion-deletion systems with one-sided contexts. *LNCS* 5196, 2008, 333–344.
9. A. Krassovitskiy, Yu. Rogozhin, S. Verlan: One-sided insertion and deletion. Traditional and P systems case. *Proc. CBM08*, 2008, Austria, 53–64.
10. A. Krassovitskiy, Yu. Rogozhin, S. Verlan: Computational power of P systems with small size insertion and deletion rules. *Proc. CSP08*, Ireland, 2008, 137–148.
11. S. Marcus: Contextual grammars. *Rev. Roum. Math. Pures Appl.*, 14 (1969), 1525–1534.
12. M. Margenstern, Gh. Păun, Yu. Rogozhin, S. Verlan: Context-free insertion-deletion systems. *Theoretical Computer Sci.*, 330 (2005), 339–348.
13. C. Martin-Vide, Gh. Păun, A. Salomaa: Characterizations of recursively enumerable languages by means of insertion grammars. *Theoretical Computer Sci.*, 205, 1–2 (1998), 195–205.
14. A. Matveevici, Yu. Rogozhin, S. Verlan: Insertion-deletion systems with one-sided contexts. *LNCS* 4664, 2007, 205–217.
15. Gh. Păun: *Marcus Contextual Grammars*. Kluwer, Dordrecht, 1997.
16. Gh. Păun: *Membrane Computing. An Introduction*. Springer, Berlin, 2002.
17. Gh. Păun, G. Rozenberg, A. Salomaa: *DNA Computing. New Computing Paradigms*. Springer, Berlin, 1998.
18. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*. Springer, Berlin, 1997.
19. A. Takahara, T. Yokomori: On the computational power of insertion-deletion systems. *LNCS*, 2568, 2003, 269–280.
20. S. Verlan: On minimal context-free insertion-deletion systems. *Journal of Automata, Languages and Combinatorics*, 12, 1-2 (2007, 317-328.