Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 330 (2005) 339–348

Theoretical Computer Science

www.elsevier.com/locate/tcs

Context-free insertion–deletion systems

Maurice Margenstern[a], Gheorghe Păun[b, c], Yurii Rogozhin[d, *], Sergey Verlan[a]

[a] *Université de Metz, LITA, UFR MIM Ile du Saulcy, 57045 Metz Cedex, France*
[b] *Institute of Mathematics of the Romanian Academy, P.O. Box 1-764, 70700 Bucureşti, Romania*
[c] *Department of Computer Science and AI, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain*
[d] *Institute of Mathematics and Computer Science, Academy of Sciences of Moldova, Str. Academiei 5, 2028 Chişinău, Moldova*

Received 17 November 2003; received in revised form 5 April 2004; accepted 29 June 2004

Abstract

We consider a class of insertion–deletion systems which have not been investigated so far, those without any context controlling the insertion–deletion operations. Rather unexpectedly, we found that context-free insertion–deletion systems characterize the recursively enumerable languages. Moreover, this assertion is valid for systems with only one axiom, and also using inserted and deleted strings of a small length. As direct consequences of the main result we found that set-conditional insertion–deletion systems with two axioms generate any recursively enumerable language (this solves an open problem), as well as that membrane systems with one membrane having context-free insertion–deletion rules without conditional use of them generate all recursively enumerable languages (this improves an earlier result). Some open problems are also formulated.

Keywords: Formal grammars; Insertion–deletion systems; Membrane computing

1. Introduction

The operations of insertion and deletion are fundamental in formal language theory, and generative mechanisms based on them were considered (with linguistic motivation)

* Corresponding author.
E-mail addresses: margens@sciences.univ-metz.fr (M. Margenstern), gpaun@us.es (G. Păun), rogozhin@math.md (Y. Rogozhin), verlan@sciences.univ-metz.fr (S. Verlan).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.06.031

since "old times"—see, e.g., [8,2]. Related formal language investigations can be found in several places; we mention only [5,7,10,11]. In the last years, the study of these operations has received a new motivation, from molecular computing—see, e.g., [1,6,13,15].

In the general form, an insertion operation means adding a substring to a given string in a specified context, while a deletion operation means removing a substring of a given string from a specified context. A finite set of insertion–deletion rules, together with a set of axioms provide a language generating device (called an InsDel system): 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 in this framework. As expected, insertion and deletion operations with context dependence are very powerful, leading to characterizations of recursively enumerable languages. Most of the papers mentioned above contain such results, in many cases improving the complexity of insertion–deletion systems previously available in the literature (mainly from the point of view of the length of strings involved in the rules).

We contribute here to this effort with an unexpected result: context-free insertion–deletion systems with one axiom are already universal, they can generate any recursively enumerable language. We also show that this result can be obtained by inserting and deleting strings of a rather small length, at most three. (The case of context-free insertion–deletion rules dealing with strings of length at most two remains open.)

The main result of the paper has direct consequences for the so-called set-conditional InsDel systems (a rule is used only when certain associated strings are present in the currently generated set of strings) and for membrane systems with string-objects processed by means of insertion–deletion operations. In the first case we solve a problem left open in [9], in the latter case we improve a result from [12].

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—for instance, [14]—for the unexplained details.

An *InsDel system* is a construct $\gamma = (V, T, A, I, D)$, 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), of strings over V. The elements of T are *terminal* symbols (in contrast, those of $V - T$ 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). 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 $\Longrightarrow_{\text{ins}}$ the relation defined by an insertion rule (formally, $x \Longrightarrow_{\text{ins}} y$ 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 $\Longrightarrow_{\text{del}}$ the relation defined by a deletion rule (formally, $x \Longrightarrow_{\text{del}} y$ 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 $\Longrightarrow_{\text{ins}}$, $\Longrightarrow_{\text{del}}$, and denote by $\Longrightarrow^*$ the reflexive and transitive closure of $\Longrightarrow$ (as usual, $\Longrightarrow^+$ is the transitive closure of $\Longrightarrow$).

The language generated by γ is defined by $L(\gamma) = \{w \in T^* \mid x \Longrightarrow^* w$, for some $x \in A\}$.

An InsDel system $\gamma = (V, T, A, I, D)$ is said to be *of weight* $(n, m; p, q)$ if

$$n = \max\{|\alpha| \mid (u, \alpha, v) \in I\},$$
$$m = \max\{|u| \mid (u, \alpha, v) \in I \text{ or } (v, \alpha, u) \in I\},$$
$$p = \max\{|\alpha| \mid (u, \alpha, v) \in D\},$$
$$q = \max\{|u| \mid (u, \alpha, v) \in D \text{ or } (v, \alpha, u) \in D\},$$

where $|w|$ is the length of a string $w \in V^*$. The *total weight* of γ is the sum $m+n+p+q$.

We denote by $INS_n^m DEL_p^q$, for $n, m, p, q \geqslant 0$, the family of languages $L(\gamma)$ generated by InsDel systems of weight $(n', m'; p', q')$ such that $n' \leqslant n$, $m' \leqslant m$, $p' \leqslant p$, $q' \leqslant q$. If some of the parameters n, m, p, q is not specified, then we write instead the symbol $*$. Thus, $INS_*^0 DEL_*^0$ denotes the family of languages generated by *context-free InsDel systems*, i.e., with insertion rules of the form $(\lambda, \alpha, \lambda) \in I$ and deletion rules of the form $(\lambda, \alpha, \lambda) \in D$, where λ denotes the empty string.

InsDel systems of a "sufficiently large" weight can characterize *RE*, the family of recursively enumerable languages. For instance, in [7,10] one proves that $RE = INS_3^2 DEL_3^0$, while in [13] one can find proofs for the equalities $RE = INS_1^2 DEL_1^1 = INS_1^1 DEL_2^0$. The first equality above was improved in [15]: $RE = INS_1^1 DEL_1^1$. Note that we have two characterizations of *RE* by means of InsDel systems with the total weight equal to 4.

We will not improve here on the total weight, but on the size of contexts: context-free insertion and deletion rules suffice.

3. The main result

We give now the central theorem of the paper; in the subsequent sections we will infer several consequences of it.

Theorem 1. $RE = INS_*^0 DEL_*^0$.

Proof. Let $G = (N, T, S, P)$ be type-0 Chomsky grammar where N, T are disjoint alphabets, $S \in N$, and P is a finite subset of rules of the form $u \to v$ with $u, v \in (N \cup T)^*$ and u contains at least one letter from N. We assume all rules from P labelled in a one-to-one manner with elements of a set M, disjoint of $N \cup T$.

We construct the context-free InsDel system $\gamma = (N \cup T \cup M, T, \{S\}, I, D)$, where

$$I = \{(\lambda, vR, \lambda) \mid R\colon u \to v \in P,\ R \in M,\ u, v \in (N \cup T)^*\},$$
$$D = \{(\lambda, Ru, \lambda) \mid R\colon u \to v \in P,\ R \in M,\ u, v \in (N \cup T)^*\}.$$

Two rules $(\lambda, vR, \lambda) \in I$, $(\lambda, Ru, \lambda) \in D$ as above are said to be *M-related*.

We have the equality $L(G) = L(\gamma)$.

The inclusion $L(G) \subseteq L(\gamma)$ is obvious: each derivation step $x_1 u x_2 \Longrightarrow x_1 v x_2$, performed in G by means of a rule $R\colon u \to v$, can be simulated in γ by an insertion operation step $x_1 u x_2 \Longrightarrow_{\text{ins}} x_1 vRu x_2$ which uses the rule $(\lambda, vR, \lambda) \in I$, followed by the deletion operation $x_1 vRu x_2 \Longrightarrow_{\text{del}} x_1 v x_2$ which uses the rule $(\lambda, Ru, \lambda) \in D$.

Consider now the inclusion $L(\gamma) \subseteq L(G)$. The idea of the proof is to transform any terminal derivation in γ into one in which any two consecutive $\langle$odd, even$\rangle$ derivations steps simulate one production in G. Because the labels of rules from P precisely identify a pair of M-related insertion–deletion rules, and the elements of M are nonterminal symbols for γ, every terminal derivation with respect to γ must involve the same number of insertion steps and of deletion steps; moreover, these steps are performed by using pairs of M-related rules from I and D.

Consider an arbitrary terminal derivation in γ,

$$\delta : S \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \cdots \Longrightarrow w_{2k} = w \in T^*,$$

where $k \geqslant 1$ is the number of the pairs of M-related insertion–deletion rules used in this derivation. Let $w_i \Longrightarrow_{\text{ins}} w_{i+1} \Longrightarrow w_{i+2}$ be a subderivation of δ such that the step $w_i \Longrightarrow_{\text{ins}} w_{i+1}$ is performed by a rule $(\lambda, vR, \lambda) \in I$ and the step $w_{i+1} \Longrightarrow w_{i+2}$ is performed by using a rule different from the M-related rule $(\lambda, Ru, \lambda) \in D$. We say that the pair of rules used in the two mentioned steps *do not match*.

Assume now that the derivation δ contains $m > 0$ non-matching pairs of rules. Let us identify a pair of M-related rules $(\lambda, vR, \lambda) \in I$ and $(\lambda, Ru, \lambda) \in D$ which are used for the same occurrence of R in δ but not in consecutive steps (that is, this pair introduces a non-matching sequence of rules in δ):

$$\delta : S \Longrightarrow^* z_1 z_2 \Longrightarrow_{\text{ins}} z_1 v R z_2 \Longrightarrow^+ y_1 R u y_2 \Longrightarrow_{\text{del}} y_1 y_2 \Longrightarrow^* w,$$

for some $z_1, z_2, y_1, y_2 \in (N \cup T \cup M)^*$. Then we have:

$$S \Longrightarrow^* z_1 z_2, \tag{1}$$
$$z_1 z_2 \Longrightarrow_{\text{ins}} z_1 v R z_2, \tag{2}$$
$$z_1 v \Longrightarrow^* y_1, \tag{3}$$
$$z_2 \Longrightarrow^* u y_2, \tag{4}$$
$$y_1 y_2 \Longrightarrow^* w. \tag{5}$$

Clearly, in at least one of the relations (3), (4) we have $\Longrightarrow^+$ instead of $\Longrightarrow^*$, since the sequence of rules is non-matching.

We rearrange the previous derivations as follows. From (1) and (4) we have

$$S \Longrightarrow^* z_1 z_2 \Longrightarrow^* z_1 u y_2.$$

We can now apply the insertion rule as in (2) and we have

$$z_1 u y_2 \Longrightarrow_{\text{ins}} z_1 v R u y_2,$$

and then the deletion rule $(\lambda, Ru, \lambda) \in D$:

$$z_1 v R u y_2 \Longrightarrow_{\text{del}} z_1 v y_2.$$

From (3) and (5) we can now obtain

$$z_1 v y_2 \Longrightarrow^* y_1 y_2 \Longrightarrow^* w.$$

Consequently, we have obtained a derivation

$$\delta' : S \Longrightarrow^* z_1 z_2 \Longrightarrow^* z_1 u y_2 \Longrightarrow_{\text{ins}} z_1 v R u y_2 \Longrightarrow_{\text{del}} z_1 v y_2 \Longrightarrow^* y_1 y_2 \Longrightarrow^* w$$

which produces the same terminal string w and has at most $m - 1$ non-matching pairs of rules.

Continuing in this way, for every terminal derivation in γ we can get an equivalent derivation, using the same rules in a different order, and having only matching pairs of consecutive rules. Clearly, two consecutive steps of a derivation in γ which use M-related rules $(\lambda, vR, \lambda) \in I$, $(\lambda, Ru, \lambda) \in D$, correspond to a derivation step in G which uses the rule R: $u \to v$. This implies the inclusion $L(\gamma) \subseteq L(G)$. □

Note 1. The InsDel system constructed in the previous proof has only one axiom, hence this complexity parameter has an optimal value.

Note 2. The context control of a type 0 grammar does not really disappear in the corresponding InsDel system (as constructed in Theorem 1 above). It rather changes its form, becoming a rigid synchronization of insertions and deletions. In other terms, if a word u represents the context of a word v in a "context-sensitive production" R: $u \to v$, then in the corresponding InsDel system the word v will also be conditioned by the later occurrence of u in a successful derivation (hence u is yet again the context of v). This condition is enforced by the newly introduced symbol R which acts as a "remote context binder". The fact that the context u "seems" to appear after the context-controlled v is of no importance, reflecting the reversal of generative process of the grammar.

We illustrate the construction from the proof with a simple **example**: consider the context-sensitive grammar $G = (\{S, X, Y\}, \{a, b, c\}, S, P)$ with the set of productions

$$P = \{R_1\colon\ S \to aSX,\ R_2\colon\ S \to aY,\ R_3\colon\ YX \to bYc,\\ R_4\colon\ cX \to Xc,\ R_5\colon\ Y \to bc\}.$$

It is easy to see that it generates the non-context-free language $L(G) = \{a^i b^i c^i \mid i \geqslant 1\}$.

The obtained InsDel system is

$$\gamma = (V, \{a, b, c\}, \{S\}, I, D),$$

where

$$V = \{S, X, Y, a, b, c, R_1, R_2, R_3, R_4, R_5\},$$
$$I = \{(\lambda, aSXR_1, \lambda),\ (\lambda, aYR_2, \lambda),\ (\lambda, bYcR_3, \lambda),\\ (\lambda, XcR_4, \lambda),\ (\lambda, bcR_5)\},$$
$$D = \{(\lambda, R_1S, \lambda),\ (\lambda, R_2S, \lambda),\ (\lambda, R_3YX, \lambda),\\ (\lambda, R_4cX, \lambda),\ (\lambda, R_5Y, \lambda)\}.$$

Consider a derivation for the word $a^3b^3c^3$ in grammar G:

$S \Longrightarrow aSX \Longrightarrow aaSXX \Longrightarrow aaaYXX \Longrightarrow aaabYcX \Longrightarrow aaabYXc \Longrightarrow aaabbYcc \Longrightarrow aaabbbccc$.

One of the corresponding derivations in γ is as follows:

$S \Longrightarrow_{\text{ins}} aSXR_1S \Longrightarrow_{\text{ins}} aaSXR_1SXR_1S \Longrightarrow_{\text{ins}} aaaYR_2SXR_1SXR_1S \Longrightarrow_{\text{del}} aaaYR_2SXXR_1S \Longrightarrow_{\text{del}} aaaYXXR_1S \Longrightarrow_{\text{ins}} aaabYcR_3YXXR_1S \Longrightarrow_{\text{del}} aaabYcXR_1S \Longrightarrow_{\text{ins}} aaabYXcR_4cXR_1S \Longrightarrow_{\text{ins}} aaabbYcR_3YXcR_4cXR_1S \Longrightarrow_{\text{del}} aaabbYcR_3YXcR_1S \Longrightarrow_{\text{del}} aaabbYccR_1S \Longrightarrow_{\text{ins}} aaabbbcR_5YccR_1S \Longrightarrow_{\text{del}} aaabbbcR_5Ycc \Longrightarrow_{\text{del}} aaabbbccc.$

In the proof of Theorem 1, the length of inserted or deleted strings is not bounded, but a bound can be easily found by controlling the length of strings appearing in the rules of the starting type-0 grammar:

Corollary 2. $RE = INS_3^0DEL_3^0$.

Proof. Let $G = (N, T, S, P)$ be type-0 Chomsky grammar in Kuroda normal form, that is, containing rules of the following forms: $A \to a$, $A \to BC$, $A \to \lambda$, $AB \to CD$, where $A, B, C, D \in N$ and $a \in T$.

Then, the rules of the context-free InsDel system constructed in the proof of Theorem 1 are of the form $(\lambda, \alpha, \lambda)$ with $|\alpha| \leqslant 3$, hence $RE \subseteq INS_3^0DEL_3^0$. □

4. Improving the total weight

The total weight of the InsDel system provided by the proof of Theorem 1 and Corollary 2 is 6. We can improve by one this result, by decreasing by one either the length of the inserted strings or the length of the deleted strings.

Theorem 3. $RE = INS_3^0DEL_2^0$.

Proof. Consider again a type-0 grammar $G = (N, T, S, P)$ in Kuroda normal form, with the productions of P injectively labelled with elements of a set M, where $M \cap (N \cup T) = \emptyset$.

We construct the InsDel system

$$\gamma = (N \cup \{A' \mid A \in N\} \cup T \cup M \cup \{R', R'' \mid R \in M\}, T, \{S\}, I, D)$$

as follows.

For each context-free rule R: $u \to v \in P$ we introduce the insertion rule (λ, vR, λ) in I and the deletion rule (λ, Ru, λ) in D.

For each non-context-free rule R: $AB \to CD \in P$ we introduce the insertion rules (λ, CDR', λ), $(\lambda, R''B'A', \lambda)$ in I and the deletion rules $(\lambda, A'A, \lambda)$, $(\lambda, B'B, \lambda)$, $(\lambda, R'R'', \lambda)$ in D (we say that these rules are M-related).

The context-free productions of the grammar G are simulated in γ in the same way as it is done in Theorem 1. A non-context-free production R: $AB \to CD$ can be simulated in the following way: we first perform two insertions,

$$x_1ABx_2 \Longrightarrow_{\text{ins}} x_1CDR'ABx_2 \Longrightarrow_{\text{ins}} x_1CDR'R''B'A'ABx_2,$$

and after that we successively delete $A'A$, $B'B$, and $R'R''$. Clearly, the string we obtain is x_1CDx_2. This proves the inclusion $L(G) \subseteq L(\gamma)$.

The converse inclusion can be proved as in the proof of Theorem 1: the insertion rules introduce nonterminals of the form R, R', R'' for $R \in M$, or A', for $A \in N$; the nonterminals R, R', R'', A' relate the insertion and the deletion rules; each derivation in γ which does not consist of consecutive steps which use M-related rules can be reordered in such a way to obtain an equivalent derivation composed of matching consecutive steps; such a derivation corresponds to a derivation in G.

We leave the details as a task for the reader and only note that the rules of γ have the weight as requested in the statement of the theorem. □

A counterpart of this result is also true: we can trade-off the length of inserted and deleted strings.

Theorem 4. $RE = INS_2^0DEL_3^0$.

Proof. Consider a type-0 grammar $G = (N, T, S, P)$ in Kuroda normal form. Each production of the form R: $AB \to CD \in P$, for $A, B, C, D \in N$, is replaced by the productions $A \to CD_R$, $D_R \to DX_B$, $X_BB \to \lambda$, where D_R, X_B are new symbols associated with D, R, and B, respectively. Clearly, we get an equivalent grammar. Thus, without loss of generality, we may assume that the rules of G are of the forms $A \to a$, $A \to \lambda$, $A \to BC$, and $AB \to \lambda$, for $a \in T$ and $A, B, C \in N$. Assume also that the rules from P are injectively labelled with elements of a set M disjoint of N and T.

We construct the InsDel system

$$\gamma = (N \cup \{A' \mid A \in N\} \cup T \cup M, T, \{S\}, I, D)$$

with the following rules.

For each erasing production $u \to \lambda \in P$ we introduce the deletion rule (λ, u, λ) in D.

For each context-free rule R: $A \to a \in P$ we introduce the insertion rule (λ, aR, λ) in I and the deletion rule (λ, RA, λ) in D.

For each context-free rule R: $A \to BC \in P$ we introduce the insertion rules (λ, BB', λ), (λ, CC', λ) in I and the deletion rule $(\lambda, C'B'A, \lambda)$ in D (we say that these rules are M-related).

Thus, the erasing productions are simulated directly by the deletion rules, the terminal productions $A \to a$ are simulated in the same way as it is done in Theorem 1, while a production of the form R: $A \to BC$ is simulated by a sequence of two insertions ($x_1Ax_2 = \Rightarrow_{\text{ins}} x_1BB'Ax_2 \Longrightarrow_{\text{ins}} x_1BCC'B'Ax_2$) and one deletion (of the string $C'B'A$). This proves the inclusion $L(G) \subseteq L(\gamma)$.

The converse inclusion can be proved again as in the previous proofs by counting the consecutive steps of a derivation in γ which do not use M-related rules. □

We do not know whether or not the total weight of InsDel systems which are able to characterize *RE* can be further decreased. We *conjecture* that this is not the case. More precisely, we believe that the language a^+b^+ does not belong to the family $INS_2^0DEL_2^0$. A characterization of this family also remains to be found.

We have considered here only two descriptional complexity measures, the number of axioms and, mainly, the weight of insertion–deletion rules. Of course, further parameters are of interest, such as the number of rules, or the total length of the rules (the total number of symbols used in writing the rules). The study of such measures remains to be carried out; trade-off results are expected, as usual in the descriptional complexity area—see [4,3].

5. Final remarks

First, for the sake of readability, we collect all known results about families $INS_n^m DEL_p^q$ with the total weight at most 6 in the table below.

Rows 11 and 12 indicate open problems (that from row 11 was already formulated in [15]).

No.	Total weight	$(n, m; p, q)$	Family generated	References
1	6	(3, 0; 3, 0)	*RE*	Corollary 2
2	5	(1, 2; 1, 1)	*RE*	[13]
3	5	(1, 2; 2, 0)	*RE*	[13]
4	5	(2, 1; 2, 0)	*RE*	[13]
5	5	(1, 1; 1, 2)	*RE*	[15]
6	5	(2, 1; 1, 1)	*RE*	[15]
7	5	(2, 0; 3, 0)	*RE*	Theorem 3
8	5	(3, 0; 2, 0)	*RE*	Theorem 4
9	4	(1, 1; 2, 0)	*RE*	[13]
10	4	(1, 1; 1, 1)	*RE*	[15]
11	4	(1, 2; 1, 0)	?	
12	4	(2, 0; 2, 0)	?	

The results in our Theorems 1, 3, 4 and Corollary 2 have direct consequences for the so-called *set-conditional InsDel systems* from [9]. Such a system is a usual context-free InsDel system with each insertion/deletion rule r having associated a string, in the form (r, w); the rule r can be applied to a string x only if the current set of strings contains, besides the string x, also the string w (we start from a set of axioms, hence after each step we continue to have a set of strings, obtained by simultaneously evolving the available strings). Set-conditional InsDel systems with three axioms were proven in [9] to characterize *RE*, but systems with only one axiom generate only singleton languages (no insertion or deletion step is possible, because we cannot promote any rule by a string different from the string to rewrite).

The case of two axioms was left open. However, as a consequence of the results in the previous sections of the paper, set-conditional context-free InsDel systems with two axioms can generate all recursively enumerable languages: just take a usual context-free InsDel system with one axiom and add a dummy axiom, in the form of a nonterminal symbol, which promotes all insertion/deletion rules.

Characterizations of recursively enumerable languages can be obtained also in the framework of membrane systems (P systems) with insertion/deletion rules applied to string-objects. One of the results of this type reported in [12] refers to insertion/deletion rules with a total weight equal to 4, hence better from this point of view than our results here (the distributed architecture of P systems helps in decreasing the weight of rules). Two other theorems from [12], 5.5.2 and 5.5.4, are improved by the results from the previous sections. Specifically, Theorem 5.5.2 from [12] characterizes *RE* by means of a one-membrane P systems with rules of weight (3, 1; 2, 0), while Theorem 5.5.4 uses rules of weight $(*, 0; 2, 0)$ applied in the conditional manner (in the same way as in set-conditional InsDel systems; the star indicates that no upper bound is imposed on the length of the inserted strings).

As a consequence of our results, the contexts used in the former theorem are not necessary, while the length of the inserted rules can be bound by 3 in the latter theorem, where, moreover, the rules can be applied in the free mode, without conditioning them.

We conclude by our belief that the insertion/deletion operations and the InsDel systems deserve further investigations, both from the mathematical point of view and with respect to their possible applications to molecular computing.

Acknowledgements

The authors acknowledge Satoshi Okawa, Artiom Alhazov and all anonymous referees for very helpful suggestions, most of them incorporated in the present version of the paper. The authors acknowledge also INRIA Lorraine, PST.CLG.976912 NATO project, IST-2001-32008 project "MolCoNet", and the French Ministry of Education project, for providing a challenging and fruitful framework for cooperation.

References

[1] M. Daley, L. Kari, G. Gloor, R. Siromoney, circular contextual insertions/deletions with applications to biomolecular computation, Proc. of Sixth Internat. Symp. on String Processing and Information Retrieval, SPIRE'99, Cancun, Mexico, IEEE Computer Society Press, Los Alamitos, CA, 1999, pp. 47–54.

[2] B.S. Galiukschov, Semicontextual grammars, Matematika Logica i Matematika Linguistika, Tallin University, 1981, pp. 38–50 (in Russian).

[3] J. Goldstine, M. Kappes, C.M.R. Kintala, H. Leung, A. Malcher, D. Wotschke, Descriptional complexity of machines with limited resources, J. Univ. Comput. Sci. 8 (2002) 193–234.

[4] J. Gruska, Descriptional complexity of context-free languages, Proc. Symp. on Mathematical Foundations of Computer Science, MFCS High Tatras, 1973, pp. 71–83.

[5] L. Kari, On insertion and deletion in formal languages, Ph.D. Thesis, University of Turku, 1991.

[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. the Third DIMACS Workshop on DNA Based Computing, Philadelphia, 1997, pp. 318–333.

[7] L. Kari, G. Thierrin, Contextual insertion/deletion and computability, Inform. and Comput. 131 (1) (1996) 47–61.

[8] S. Marcus, Contextual grammars, Rev. Roumaine Math. Pures Appl. 14 (1969) 1525–1534.

[9] M. Margenstern, Gh. Păun, Yu. Rogozhin, On the power of the (molecular) crowd: set-conditional string processing, Preproc. AFL'02, Debrecen, Hungary, August 13–18, 2002.

[10] C. Martin-Vide, Gh. Păun, A. Salomaa, Characterizations of recursively enumerable languages by means of insertion grammars, Theoret. Comput. Sci. 205 (1–2) (1998) 195–205.

[11] Gh. Păun, Marcus Contextual Grammars, Kluwer, Dordrecht, 1997.
[12] Gh. Păun, Membrane Computing. An Introduction, Springer, Berlin, 2002.
[13] Gh. Păun, G. Rozenberg, A. Salomaa, DNA Computing. New Computing Paradigms, Springer, Berlin, 1998.
[14] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Springer, Berlin, 1997.
[15] A. Takahara, T. Yokomori, On the computational power of insertion–deletion systems, Proc. of Eighth Internat. Workshop on DNA-Based Computers, DNA8, Sapporo, Japan, June 10–13, 2002; Revised Papers, Lecture Notes in Computer Science, Vol. 2568 (2003) 269–280.