ON MINIMAL CONTEXT-FREE INSERTION-DELETION SYSTEMS

Sergey Verlan
LACL, University of Paris XII
61, av. Général de Gaulle, 94010, Créteil, France
e-mail: verlan@univ-paris12.fr

ABSTRACT

We investigate the class of context-free insertion-deletion systems. It is known that such systems are universal if the length of the inserted/deleted string is at least three. We show that if this length is bounded to two, then the obtained systems are not universal. We characterise the obtained class and we introduce a new complexity measure for insertion-deletion systems, which permits a better explanation of the obtained results.

Keywords: Formal languages, Insertion-deletion systems, Decidability

1. Introduction

The operations of insertion and deletion are fundamental in formal language theory, and generative mechanisms based on them have been considered (with linguistic motivation) since a long time ago, see [6] and [2]. Related formal language investigations can be found in several places; we mention only [3], [5], [8], [9]. In the last years, the study of these operations has received a new motivation from molecular computing, see [1], [4], [10], [12].

In 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 (an InsDel system): starting from the set of initial strings and iterating insertion-deletion operations as defined by the given rules we obtain 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.

The paper [7] contains an unexpected result: context-free insertion-deletion systems with one axiom are already universal, they can generate any recursively enumerable

language. Moreover, this result can be obtained by inserting and deleting strings of a rather small length, at most three. The same paper stated an open question about context-free insertion-deletion systems having rules dealing with strings of length at most two.

In this paper we answer this open question and we show that if the length of the inserted and deleted string is at most two, then such systems generate a particular subset of the family of context-free languages. We also show that the traditional complexity measures for insertion-deletion systems, in particular the size of contexts, need a revision and we propose new measures based on the total weight.

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, [11] – for the unexplained details.

We label positions between letters of a string $a_1a_2\dots a_n$ as follows:

$$\begin{array}{ccccccc} & a_1 & a_2 & \dots & & a_n & \\ 0 & & 1 & 2\dots & n-1 & & n \end{array}$$

We denote by $|w|$ the length of word w. If A is a set of words, then we put $|A| = \max_{w\in A}(w)$. Finally, we denote by $card(A)$ the cardinality of the set A.

The *Dyck language*, D_n, over $T_n = \{a_1, a'_1, \dots, a_n, a'_n\}$, $n \geq 1$, is the context-free language generated by the grammar

$$G = (\{S\}, T_n, S, \{S \to \varepsilon, S \to SS\} \cup \{S \to a_iSa'_i \mid 1 \leq i \leq n\}, S).$$

Intuitively, the pairs $(a_i, a'_i), 1 \leq i \leq n$, can be viewed as parentheses, left and right, of different kinds. Then D_n consists of all strings of correctly nested parentheses.

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 $\Longrightarrow_{ins}$ 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 the transitive closure of $\Longrightarrow$).

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

Example 2.1 Let us consider the following InsDel system $ID = (T, T, A, I, \emptyset)$, with $T = \{a, b\}$, $A = \{ab\}$ and $I = \{(a, ab, b)\}$. It is easy to see that at each derivation step the string ab is inserted in the middle of the word obtained on the previous step because this is the only place where the context ab occurs. Since we start from ab it is clear that $L(ID) = \{a^n b^n \mid n \geq 1\}$.

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

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

The *total weight* of γ is the sum $m + n + p + q$.

Example 2.2 The system ID from the previous example is of weight $(2, 1, 0, 0)$ and has the total weight equal to 3.

We denote by $INS_n^m DEL_p^q$, for $n, m, p, q \geq 0$, the family of languages $L(\gamma)$ generated by InsDel systems of weight $(n', m'; p', q')$ such that $n' \leq n$, $m' \leq m$, $p' \leq p, q' \leq q$. 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 $(\varepsilon, \alpha, \varepsilon) \in I$ and deletion rules of the form $(\varepsilon, \alpha, \varepsilon) \in D$, where ε denotes the empty string. In this case we may treat I and D as finite languages containing strings from the middle part of rules. We also denote by $CFINSDEL_{m,p}$ the family of context-free InsDel systems having the length of the inserted string at most m and the length of the deleted string at most p. For a system $ID \in CFINSDEL_{m,p}$ we shall also say that ID is of size (m, p).

Context-free InsDel systems have an interesting particularity: insertions and deletions are uncontrolled and may happen at any time at any place in a string. This fact can give an impression that such systems cannot be controlled in order to perform computations. However, this affirmation is not true. In [7] it is shown that, in spite of the above remark, such systems are able to simulate an arbitrary Chomsky grammar. The next example partially shows the technique used.

Example 2.3 Let us consider a context-free InsDel system $ID_2 = (V, T, \{S\}, I, D)$ of size $(3, 2)$ with $T = \{a, b\}$, $V = T \cup \{S, S'\}$, $I = \{S'aSb, S'ab\}$ and $D = SS'$. The computation in this system goes as follows. Given a word $w_1 S w_2$ (initially $w_1, w_2 = \varepsilon$) the string $S'aSb$ is inserted after S, which produces $w_1 SS'aSbw_2$. After that, the deletion rule eliminates SS' and word $w_1 aSbw_2$ is obtained. It is easy to see that this corresponds to rewriting rule $S \to aSb$. If the string $S'aSb$ is not inserted immediately after S, then symbol S' cannot be eliminated, hence it will not be possible to generate a terminal string. A more detailed proof of this assertion may be found in [7].

Therefore, system ID_2 simulates the grammar with productions $\{S \to aSb, S \to ab\}$. Hence, $L(ID_2) = \{a^n b^n \mid n \geq 1\}$.

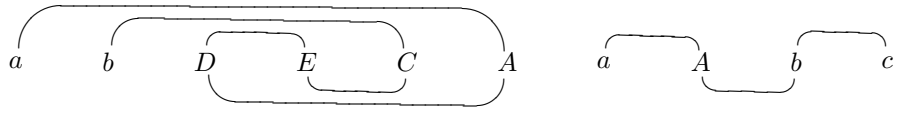
InsDel systems of a "sufficiently large" weight can characterize RE, the family of recursively enumerable languages. A collection of these results may be found in Section 5.

3. Main results

In this section we present results about context-free insertion-deletion systems having the length of the inserted/deleted string at most 2. The following lemma shows that the non-terminal alphabet is not relevant if we start from an empty word.

Lemma 3.1 *Let $ID = (V, T, A, I, D)$ be a context-free InsDel system of size $(2, 2)$. Suppose also that $A = \varepsilon$. Then there is a system $ID_2 = (T, T, \varepsilon, I_2, D_2)$ of size $(2, 2)$ such that $L(ID) = L(ID_2)$.*

Proof. Consider a derivation of $w \in T^*$. Let us mark corresponding insertion pairs by an overline and corresponding deletion pairs by an underline. For example, suppose that we insert aA, after that bC in position 1, DE in position 2, aA in position 6 and bc in position 8. After that suppose that we delete EC, DA and Ab. Then the corresponding marking will be as follows ($w = abac$):



We may interpret symbols as labeled graph nodes and lines as edges. In this case we obtain a graph. It is easy to observe that this graph is composed from a set of linear paths, or cycles. Indeed, for each node, at most two edges corresponding to an insertion and a deletion may be drawn. Let us also label edges corresponding to insertions by i and edges corresponding to deletions by d. If we take the example above, we obtain:

$$a \overset{i}{\text{——}} A \overset{d}{\text{——}} D \overset{i}{\text{——}} E \overset{d}{\text{——}} C \overset{i}{\text{——}} b$$

$$a \overset{i}{\text{——}} A \overset{d}{\text{——}} b \overset{i}{\text{——}} c$$

We may suppose that the first and the last edge of a path are marked with i. If this is not the case, we add an additional node labeled by ε and we connect this node with the last node by a path labeled by i. In particular, a path containing only one letter a (corresponding to an insertion of a) will be written as $\varepsilon \overset{i}{\text{——}} a$. Hence, each path consists of sequences of one insertion followed by one deletion.

We observe that if we consider a derivation of $w \in T^*$ then there are paths of 4 types:

1. Paths that start with a letter $a \in T$ and that end with a letter $b \in T$.
2. Paths that have at one end a terminal letter a and at the other end ε.
3. Paths that have ε at both ends.
4. Circular paths.

We remark that in case 1 the path leads to the word ab (*i.e.*, contributed to the production of the subword ab of w), in the second case the path produces the letter a and in the last two cases the path generates the empty word.

More exactly, let p be a path. We denote by $yield(p)$ the word produced by p:

$$yield(p) = \begin{cases} ab, \text{ if } p = a \overset{i}{-} \cdots \overset{i}{-} b,\ a, b \in T \cup \{\varepsilon\}. \\ \varepsilon, \quad \text{if } p \text{ is circular.} \end{cases}$$

Let $xy \neq \varepsilon$, $x, y \in V \cup \{\varepsilon\}$ be a word. We denote by $yield(xy)$ the set of all words that may be produced by the path containing xy:

$yield(xy) = \{yield(p),\ \text{where } x \overset{i}{-} y \in p\}$.

We remark that $|yield(xy)| \leq |xy|$.

Without loss of generality we may suppose that there are no paths of type 3 and 4, because by eliminating corresponding insertions and deletions we obtain the same word.

Suppose that we have a path marked by over- and underlines as above. We shall understand by an interior of the path the set of all positions that are underlined. In the example above, all positions between D and the first A form the interior of the path. It is clear that no other path (of type 1 and 2) may be situated in the interior of some path, because in this case the corresponding deletion cannot be performed. Consequently, all paths are independent of each other and we may group rules corresponding to each path and compute paths one after another. Moreover, each path contributes to at most two terminal symbols of the resulting word. Therefore, the computation consists of insertion of terminal symbols corresponding to paths ends as well as of deletion of terminal symbols.

Now, in order to prove the lemma, it is enough to show that me may precompute all possible paths. This may be done by using the following observation. We may assume that each path p has the following property: if $A \overset{i}{-} B$ belongs to p, then p does not contain an insertion that has A in the left-hand side ($A \overset{i}{-} X$) or B in the right-hand side ($Y \overset{i}{-} B$). This affirmation is obvious, because if p contains such a pair, for example $p = \cdots \overset{d}{-} A \overset{i}{-} X \overset{d}{-} \cdots \overset{d}{-} A \overset{i}{-} B \cdots$, then we may eliminate the subpath between two A's by obtaining an equivalent path (that leads to same ends) $p' = \cdots \overset{d}{-} A \overset{i}{-} B \cdots$. Hence, the length of each path is bounded by $2 * card(V)$ and we may precompute all possible paths.

Finally, let I' be the subset of I that contains only terminal insertion rules, *i.e.*, any rule of I' is of form ab or a, where $a, b \in T$. Analogously, let D' be the subset of terminal deletion rules of D. Let us also precompute all possible paths that end with a terminal letter or ε (of type 1 and 2). Let us denote by I_1 the set of words produced

by these paths. More exactly, we define $I_1 = \{yield(xy) \mid xy \in I\}$. It is clear that the system $ID_2 = (T, T, \varepsilon, I_2, D_2)$, where $I_2 = I' \cup I_1$ and $D_2 = D'$ generates the same language as ID. It is also clear that the converse inclusion holds. Hence, our assertion is proved. □

Now we prove that the non-terminal alphabet is not relevant even in general case.

Lemma 3.2 *Let $ID = (V, T, A, I, D)$ be a context-free InsDel system of size $(2, 2)$. Then there is a system $ID_2 = (T, T, A_2, I_2, D_2)$ of size $(2, 2)$ such that $L(ID) = L(ID_2)$.*

Proof. If we take a look at the proof of previous lemma, we may see that we eliminated insertion productions having nonterminals as follows. All possible paths were precomputed. For each insertion production xy of I, the set of paths that contain $x \overset{i}{-} y$ was selected. Finally, the production xy was replaced by the set $yield(xy)$.

A similar idea holds in the case of this lemma. We shall show that we may replace each axiom w_i by the set $YIELD(w_i)$ consisting of only terminal words.

Let $w =$ be a word. We construct the set $YIELD(w)$ incrementally as follows.

$YIELD_0(w) = \{w\}$.

$YIELD_{n+1}(w) = YIELD_n(w) \cup YIELD'_{n+1}(w) \cup YIELD''_{n+1}(w)$,

where

$YIELD'_{n+1}(w) = \{w_1 t w_2 \mid w_1 x w_2 \in YIELD_n(w), |x| = 1 \text{ and } t \in yield(x)\}$,

$YIELD''_{n+1}(w) = \{w_1 w_2 \mid w_1 x w_2 \in YIELD_n(w), x \neq \varepsilon \text{ and } x \in D\}$.

And finally,

$YIELD(w) = \cup_{n \geq 0} YIELD_n(w) \cap T^*$.

Informally, set $YIELD''_{n+1}$ contains words that are obtained by applying a deletion rule on words from $YIELD_n$, while in order to obtain set $YIELD'_{n+1}$ each letter x of a word from $YIELD_n$ is tried to be replaced by another end of an insertion/deletion path that starts with this letter ($\varepsilon \overset{i}{-} x \overset{d}{-} \ldots \overset{i}{-} t$).

It is easy to observe that there is $n \geq 0$ such that $YIELD_k(w) = YIELD_n(w)$ for all $k \geq n$. This follows from the fact that at each step we do not increase the length of words because $|YIELD'_n(w)| \leq |w|$, hence $|YIELD(w)| \leq |w|$.

From the construction of the set $YIELD(w)$ it is clear that if we replace an axiom $w_i \in A$, $1 \leq i \leq n$, by the set $YIELD(w_i)$ then the generated language is the same. Indeed, let $w_0 \in A$ and $w_0 \Longrightarrow^* w \in T^*$, $w_0 = a_1 \ldots a_n$. We may suppose without loss of generality that $a_1 \in V \setminus T$ and $a_2, \ldots, a_n \in T$. Since a_1 is not terminal, then it shall be erased during the derivation. If a_1 is erased by a deletion rule a_1 from D, then it is clear that $w' \Longrightarrow^* w$, where $w' = a_2 \ldots a_n$. It is also clear that $w' \in YIELD(w_0)$. A similar argument holds if a_1 is erased by a deletion rule $a_1 a_2$. Now, if a_1 is erased by a rule $a_1 b$ from D ($b \neq a_2$), then this b shall be inserted by some insertion rule, hence it shall be at the end of an insertion/deletion path ($b \overset{i}{-} \ldots \overset{i}{-} c$). Hence, a_1 is replaced by c. It is clear, that if $c \in V \setminus T$, then we may repeat the above process. Otherwise, $ca_2 \ldots a_n \Longrightarrow^* w$ and, by construction, $ca_2 \ldots a_n$ is in $YIELD(w_0)$.

Now, consider the system $ID_2 = (T, T, A_2, I_2, D_2)$, where A_2 is defined by $A_2 = \{\, YIELD(w) \mid w \in A\}$, and I_2 and D_2 are constructed as in the previous lemma. It is clear that $L(ID_2) = L(ID)$. □

Following lemma shows that we can eliminate the deletion operation.

Lemma 3.3 *Let $ID = (T, T, A, I, D)$ be a context-free InsDel system of size $(2, 2)$. Then there is a system $ID_2 = (T, T, A_2, I_2, \emptyset)$ of size $(2, 2)$ such that $L(ID) = L(ID_2)$.*

Proof. Let us define the set INS as follows.
$INS_0 = I$.
$INS_{n+1} = INS_n \cup \{\, YIELD(w) \mid w \in INS_n\}$.
$INS = \cup_{n\geq 0} INS_n$.
Following the same arguments as in previous lemma, it is clear that there is $n \geq 0$ such that $INS_k = INS_n$, for all $k \geq n$ (because $|INS_k| \leq 2$).
It is clear that if we substitute set I by set INS then the new system generates the same language. Indeed, for each path $a \overset{i}{-} \cdots \overset{i}{-} b$ of ID there is an insertion rule ab in INS. We can also replace each axiom w by $YIELD(w)$. Hence, we do not need any deletion rules any more. So, if we consider $ID_2 = (T, T, A_2, INS, \emptyset)$, $A_2 = \{\, YIELD(w) \mid w \in A\}$, then $L(ID) = L(ID_2)$. □

We now show that a context-free insertion-deletion system of size $(2, 2)$ without non-terminal alphabet and without deletion rules may be described by a context-free grammar.

Lemma 3.4 *Let $ID = (T, T, A, I, \emptyset)$ be a context-free InsDel system of size $(2, 2)$. Then there is a context-free grammar $G = (N, T, Z, P)$ such that $L(ID) = L(G)$.*

Proof. We construct G as follows.
Consider $N = \{Z, S\}$ and put T be the terminal alphabet of ID.
Define $P = P_A \cup P_I \cup \{S \to \varepsilon\}$, where
$P_A = \{Z \to Sa_1Sa_2S \dots Sa_nS \mid a_1a_2 \dots a_n \in A\}$ and
$P_I = \{S \to SaSbS \mid ab \in I\} \cup \{S \to SaS \mid a \in I\}$.
It is clear that $L(G) = L(ID)$. Indeed, symbol S marks all possible insertion positions and it permits to simulate insertion rules as well. □

Consequently, we obtain:

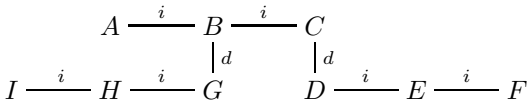
Theorem 3.5 $CFINSDEL_{2,2} \subset CF$.

Proof. The strictness of the inclusion follows from the fact that the language $\mathcal{L} = \{a^*b^* \mid n \geq 0\}$ cannot be generated by a context-free insertion-deletion system of size $(2, 2)$. Indeed, consider an arbitrary system $ID = (T, T, A, I, \emptyset)$. It is easy to observe that for each word w that belong to $L(ID)$ words I^*wI^* belong to $L(ID)$. Therefore, if we suppose that $L(ID)$ is not finite, in this case $I \neq \emptyset$, then for any word $w \in L(ID)$, there are words I^*wI^* in $L(ID)$. It is easy to see that $\mathcal{L}$ does not have such a property. □

Theorem 3.6 *$CFINSDEL_{2,2}$ is incomparable with REG.*

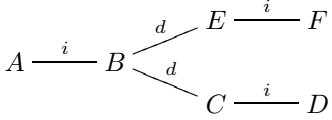
Proof. From the previous theorem we obtain that $REG \setminus CFINSDEL_{2,2} \neq \emptyset$. It is also clear that the Dyck language D_n may be generated by a context-free insertion system having insertion rules $a_i a_i'$, $1 \leq i \leq n$. Hence, the assertion is proved. □

Remark 3.1 The proof of Lemma 3.1 shows why the computational power increases in the case when the size of inserted or deleted string is increased to 3. It is easy to observe that if we construct a similar insertion/deletion graph – by connecting inserted, respectively deleted, symbols pairwise by edges labeled with i, respectively d, and following the ordering of symbols in the string – then this graph may contain trees. For example, suppose that we insert ABC, after that DEF in position 3, GHI in position 2 and finally delete CD and BG. Then, the corresponding graph will be the following:



The obtained graph no longer has a linear structure, hence sets *YIELD* defined above do not have the property $|YIELD(w)| \leq |w|$, *i.e.*, we may increase the length of a word.

A similar thing happens when the deletion of strings of length 3 is permitted. In the example below, we inserted AB, CD in position 2, FE in position 1 and deleted EBC.



4. Systems with insertion or deletion of less than two symbols

In this section we shall consider context-free insertion-deletion systems having the length of the inserted or deleted context equal to one.

Theorem 4.1 *For any $m > 0$, $CFINSDEL_{m,1} = CFINSDEL_{m,0} \subset CF$.*

Proof. It is clear that if we consider a context-free insertion-deletion system of size $(m, 1)$, then we can use same construction as in Lemma 3.3, see also Remark 3.1, and reduce it to a system without any deletion (as $|INS_n| \leq m$ and $|YIELD(w)| \leq |w|$).

Moreover, a system $ID = (T, T, A, I, \emptyset) \in CFINSDEEL_{m,0}$ can be simulated by the following grammar $G = (N, T, Z, P)$, where $N = \{Z, S\}$ and

$P = P_A \cup P_I \cup \{S \to \varepsilon\}$, with

$P_A = \{Z \to Sa_1Sa_2S \dots Sa_kS \mid a_1a_2 \dots a_k \in A\}$ and

$P_I = \{S \to Sa_1Sa_2S \dots Sa_nS \mid a_1a_2 \dots a_n \in I\} \quad (n \leq m)$.

It is clear that $L(ID) = L(G)$, see also Lemma 3.4. □

If we consider systems where only insertions of size at most 1 are permitted, then the power of such systems is smaller.

Theorem 4.2 *For any $p > 0$, $CFINSDEL_{1,p} \subset REG$.*

Proof. Consider a system $ID = (N, T, A, I, D) \in CFINSDEL_{1,p}$. Using same construction as in Lemma 3.3, see also Remark 3.1, we may reduce it to the system $ID_2 = (T, T, A_2, I_2, \emptyset)$ (as $|INS_n| = 1$ and $YIELD(w) \leq |w|$). The last system generates the regular language $\{I_2^* a_1 I_2^* a_2 \dots a_n I_2^* \mid a_1 \dots a_n \in A_2\}$. □

5. Complexity measures

We summarize the known results on insertion-deletion systems in the table below.

Nb.	total weight (γ)	$(n, m; p, q)$	family generated	references
1	6	$(3, 0; 3, 0)$	RE	[7]
2	5	$(1, 2; 1, 1)$	RE	[4, 10]
3	5	$(1, 2; 2, 0)$	RE	[4, 10]
4	5	$(2, 1; 2, 0)$	RE	[4, 10]
5	5	$(1, 1; 1, 2)$	RE	[12]
6	5	$(2, 1; 1, 1)$	RE	[12]
7	5	$(2, 0; 3, 0)$	RE	[7]
8	5	$(3, 0; 2, 0)$	RE	[7]
9	4	$(1, 1; 2, 0)$	RE	[10]
10	4	$(1, 1; 1, 1)$	RE	[12]
11	4	$(2, 0; 2, 0)$	$\subset CF$	Theorem 3.5
12	$m + 1$	$(m, 0; 1, 0)$	$\subset CF$	Theorem 4.1
13	$p + 1$	$(1, 0; p, 0)$	$\subset REG$	Theorem 4.2

From this table it is clear that the notion of the *total weight* (γ) shall be revised, because it cannot distinguish between universality and non-universality cases (rows 10 and 11). We think that the main problem consists in the fact that in relation $m = \max\{|u| \mid (u, \alpha, v) \in I \text{ or } (v, \alpha, u) \in I\}$, m is the maximum of the length of both contexts.

We suggest to use the length of each context instead of the maximum. More exactly,

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

Hence we shall describe the complexity of insertion/deletion by the vector $(n, m, m'; p, q, q')$. We also denote by $INS_n^{m,m'} DEL_p^{q,q'}$ corresponding families of insertion-deletion systems. Moreover, we define the total weight of the system as the sum of all numbers above: $\psi = n + m + m' + p + q + q'$. In this case we may distinguish rows 10 and 11 by the total weight, because the total weight of 10 is equal to 6 while the total weight of 11 is equal to 4; corresponding systems are of size $(1, 1, 1; 1, 1, 1)$ and $(2, 0, 0; 2, 0, 0)$ respectively.

We give below the previous table without lines 12 and 13 where we indicate the old (γ) and the new (ψ) measures.

Nb.	γ	$(n, m; p, q)$	family	ψ	$(n, m, m'; p, q, q')$
1	6	$(3, 0; 3, 0)$	RE	6	$(3, 0, 0; 3, 0, 0)$
2	5	$(1, 2; 1, 1)$	RE	8	$(1, 2, 2; 1, 1, 1)$
3	5	$(1, 2; 2, 0)$	RE	7	$(1, 2, 2; 2, 0, 0)$
4	5	$(2, 1; 2, 0)$	RE	6	$(2, 1, 1; 2, 0, 0)$
5	5	$(1, 1; 1, 2)$	RE	8	$(1, 1, 1; 1, 2, 2)$
6	5	$(2, 1; 1, 1)$	RE	7	$(2, 1, 1; 1, 1, 1)$
7	5	$(2, 0; 3, 0)$	RE	5	$(2, 0, 0; 3, 0, 0)$
8	5	$(3, 0; 2, 0)$	RE	5	$(3, 0, 0; 2, 0, 0)$
9	4	$(1, 1; 2, 0)$	RE	5	$(1, 1, 1; 2, 0, 0)$
10	4	$(1, 1; 1, 1)$	RE	6	$(1, 1, 1; 1, 1, 1)$
11	4	$(2, 0; 2, 0)$	$\subset CF$	4	$(2, 0, 0; 2, 0, 0)$

The value of ψ describes in some sense the amount of cooperation between symbols in the system. We think that in order to achieve the universality, a cooperation at least equal to 5 is required.

Conjecture 5.1 Consider systems of size $(n, m, m'; p, q, q')$. Assume that $n, p \geq 1$, and that both $n + m + m'$ and $p + q + q'$ are at least equal to 2. We conjecture that such systems having total weight at least 5 are universal, while those whose weight is at most 4 are not universal.

6. Conclusions

In this article we investigated context-free insertion-deletion systems having a minimal length of the inserted or deleted string (at most 2). We showed that systems of size $(2, 2)$ generate a subset of context-free languages incomparable with the family of regular languages. We also provided a context-free grammar that describes the obtained family. A similar result holds for systems of size $(m, 1)$, $m > 0$. Systems of size $(1, p)$, $p > 0$, are less powerful and they generate only a subset of regular languages. The obtained results together with results from [7], where it is shown that systems of size $(2, 3)$ and $(3, 2)$ generate all recursively enumerable languages, answer

completely the question about the generative power of context-free insertion-deletion systems.

The obtained results also show that one of the measures considered before, the total weight, is not accurate because it cannot distinguish between decidability and non-decidability cases. We propose to redefine this measure and to base it on the sum of all parameters of the system. Since we consider two additional parameters, there are left several open questions about the power of corresponding systems, in particular about systems having only one-sided context.

Acknowledgments

The author acknowledges Rudolf Freund and Jean Néraud for their important remarks concerning the above result. The author also acknowledges Yurii Rogozhin and Artiom Alhazov for very helpful suggestions, most of them incorporated in the present version of the paper.

References

[1] M. Daley, L. Kari, G. Gloor, R. Siromoney, Circular contextual insertions/Deletions with applications to biomolecular computation. In: *Proc. of 6th Int. Symp. on String Processing and Information Retrieval, SPIRE'99* (Cancun, Mexico, 1999), 47–54.

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

[3] L. Kari, *On insertion and deletion in formal languages*, PhD Thesis, University of Turku, 1991.

[4] L. Kari, Gh. Păun, G. Thierrin, S. Yu, At the crossroads of DNA computing and formal languages: characterizing RE using insertion-deletion systems. In: *Proceedings of 3rd DIMACS Workshop on DNA Based Computing*, Philadelphia, 1997, 318–333.

[5] L. Kari, G. Thierrin, Contextual insertion/deletion and computability, *Information and Computation*, **131**, 1 (1996), 47–61.

[6] S. Marcus, Contextual grammars, *Rev. Roum. Math. Pures Appl.*, **14** (1969), 1525–1534.

[7] M. Margenstern, Gh. Paun, Yu. Rogozhin, S. Verlan, Context-free insertion-deletion systems. *Theoretical Computer Science*, **330** (2005), 339–348.

[8] C. Martin-Vide, Gh. Păun, A. Salomaa, Characterizations of recursively enumerable languages by means of insertion grammars, *Theoretical Computer Science*, **205**, 1–2 (1998), 195–205.

[9] Gh. Păun, *Marcus contextual grammars*. Kluwer, Dordrecht, 1997.

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

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

[12] A. Takahara, T. Yokomori, On the computational power of insertion-deletion systems. In: *Proceedings of 8th International Workshop on DNA-Based Computers, DNA8* (Sapporo, Japan, June 10–13, 2002), *Revised Papers, LNCS*, **2568** (2003), 269–280.