# On the Lower Bounds for Leftist Insertion-Deletion Languages

Sergiu Ivanov and Sergey Verlan

**Abstract -** This article investigates languages generated by insertion-deletion systems that insert or delete one symbol using only a left context. We show that if the context of insertion or deletion is greater than one, then the corresponding families are all equal to each other and properly include the family of regular languages. We also show that the obtained family contains non-context-free languages.

**Key words and phrases :** insertion-deletion, one-sided context, regular languages.

## 1 Introduction

*Context adjoining*, an insertion operation on strings, was first considered by S. Marcus in [20] with a linguistic motivation and later developed in [25]. These references investigate Marcus contextual grammars which capture many interesting linguistic properties like ambiguity and duplication. Following these ideas the *insertion* operation was introduced in [6]. This is an intermediate operation between context adjoining and string rewriting.

Another motivation for studying insertion can be found in [7, 8] where this operation and its iterated variant are introduced as a generalization of Kleene's operations of concatenation and closure [17] which may happen anywhere in the string. In [14] the *deletion* operation is defined as a right quotient operation which does not necessarily happen at the rightmost end of the string. Both operations were first considered together in [16].

Yet another inspiration for insertion and deletion operations comes from the field of molecular biology: they correspond to mismatched annealing of DNA sequences, see [26] for more details. Such operations are also present in evolution processes under the form of point mutations as well as in RNA editing, see the discussions in [2, 3, 26, 28]. This biological motivation of insertion and deletion operations led to their study in the framework of molecular computing, see, for example, [4, 15, 26, 29].

Intuitively, an insertion means adding a substring to a given string at a site having a specified left, right, or both contexts, while a deletion means removing a substring of a given string from a site having a specified left,

right, or both contexts. A finite set of insertion and deletion rules, together with a set of axioms, provide a language-generating device: by starting from a set of initial strings and iterating insertion and deletion operations as defined by the supplied rules, one obtains a language.

Insertion-deletion systems with relatively small rules are already able to produce all recursively enumerable languages; the works [1, 31] provide an overview of known results. Furthermore, as it was shown in [21], context dependence may be replaced by insertion and deletion of strings of sufficient length, in a context-free manner. If, however, the length is not sufficient (less than or equal to two), then such systems are not able to generate more than context-free languages. A detailed characterization of context-free insertion-deletion systems is given in [30].

In [18, 19, 22], similar investigations were continued on insertion-deletion systems with one-sided contexts, in which the (insertion and deletion) rules have all either a left or a right context, i.e., the asymmetry in context dependence is system-wide. The cited papers give several computational completeness results as a function of the size of the rules. We recall the fact that certain of the discussed structures are not computationally complete, i.e., there are recursively enumerable languages they cannot generate. For these variants, similarly to the case of context-free rewriting, it is possible to consider regulated variants of insertion-deletion systems. The papers [1, 5, 9, 10, 24] present results on insertion-deletion systems with graph, matrix, semi-conditional, and random context controls.

In this paper we consider a particular variant of insertion-deletion systems, *leftist* systems, where only one symbol can be inserted or deleted using only a left context. This model is closely related to the model of leftist grammars used to express the accessibility problem in some general protection systems [23]. We show that the hierarchy of corresponding systems with respect to the context size collapses immediately and we show that it properly includes the family of regular languages. Using the results from [13] we also show that leftist insertion-deletion systems contain non-context-free languages.

## 2 Definitions

We do not present here definitions concerning standard concepts of the theory of formal languages and we refer to [27] for more details. We denote by $|w|$ the length of a word $w$ and by $REG$, $CF$, $CS$, and $RE$ the families of regular, context-free, context-sensitive and recursively enumerable languages, respectively.

An *insertion-deletion system* is a construct $ID = (V, T, A, I, D)$, where:

- $V$ is an alphabet;

- $T \subseteq V$ is the *terminal* alphabet (the symbols from $V \setminus T$ are called *non-terminals*);
- $A \subseteq V^*$ is the set of *axioms*;
- $I, D$ are finite sets of triples of the form $(u, \alpha, v)$, where $u$, $\alpha$ ($\alpha \neq \lambda$), and $v$ are strings over $V$.

The triples in $I$ are *insertion rules*, and those in $D$ are *deletion rules*. An insertion rule $(u, \alpha, v)_{ins} \in I$ indicates that the string $\alpha$ can be inserted between $u$ and $v$ (which corresponds to the rewriting rule $uv \to u\alpha v$), while a deletion rule $(u, \alpha, v)_{del} \in D$ indicates that $\alpha$ can be removed from between the contexts $u$ and $v$ (which corresponds to the rewriting rule $u\alpha v \to uv$). By $\Rightarrow$ we denote the relation defined by the insertion or deletion rules and by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$.

The language generated by $ID = (V, T, A, I, D)$ is defined by

$$L(ID) = \{w \in T^* \mid x \Rightarrow^* w \text{ for some } x \in A\}.$$

The complexity of an insertion-deletion system $ID = (V, T, A, I, D)$ is described by the vector $(n, m, m'; p, q, q')$ called *size*, where

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

We also denote by $INS_n^{m,m'} DEL_p^{q,q'}$ all the languages generated by the families of insertion-deletion systems of size at most $(n, m, m'; p, q, q')$. Moreover, we define the total size of the system as the sum of all numbers above: $\psi = n + m + m' + p + q + q'$.

If some of the parameters $n, m, m', p, q, q'$ is not bounded, then we write instead the symbol $*$. If one of the numbers from the pairs $m$, $m'$ and/or $q$, $q'$ is equal to zero (while the other is not), then we say that the corresponding families have a one-sided context.

We would like to note that the notion of leftist grammars [23] corresponds to a semi-Thue system having rewriting rules of two types: $a \to ab$ and $ab \to a$. It is not difficult to see that this corresponds to an insertion-deletion system of size $(1, 1, 0; 1, 1, 0)$ having the terminal alphabet equal to the alphabet of the system. We remark that most of the results from this area [11, 12, 13] study the properties of the intersection of leftist languages with a regular or a context-free language.

We also recall that the family of insertion-deletion languages of size $(1, 1, 0; 1, 1, 0)$ is incomparable with $REG$: $(CF \setminus REG) \cap INS_1^{1,0} DEL_0^{0,0} \neq \emptyset$ and $(ba)^+ \notin INS_1^{1,0} DEL_1^{1,0}$ [19].

In this article we use the following results from [22] (Lemmas 2 and 6):

**Lemma 2.1** *For any insertion-deletion system $ID = (V, T, A, I, D)$ of size $(n, m, m'; p, q, q')$ there is a system $ID' = (V \cup V', T, A \cup A', I \cup I', D')$ having the same size such that $L(ID') = L(ID)$. Moreover, for any rule $(\alpha, \beta, \gamma) \in D'$ it holds that $\beta$ does not contain letters from $T$.*

**Lemma 2.2** *For any insertion-deletion system $ID = (V, T, A, I, D)$ having the size $(n, m, m'; p, q, q')$ it is possible to construct an insertion-deletion system $ID_2 = (V \cup \{X, Y\}, T, A_2, I_2, D_2 \cup D_2')$ having the same size such that $L(ID_2) = L(ID)$. Moreover, all rules from $I_2$ have the form $(u, \alpha, v)$, where $|u| = m$, $|v| = m'$, and all rules from $D_2$ have the form $(u', \alpha, v')$, where $|u'| = q$, $|v'| = q'$ and $D_2' = \{(\varepsilon, X, \varepsilon), (\varepsilon, Y, \varepsilon)\}$.*

# 3 Main Results

The first result from this section shows that $REG$ is strictly included in the family of insertion-deletion languages of size $(1, 1, 0; 1, 2, 0)$.

**Theorem 3.1** $REG \subsetneq INS_1^{1,0} DEL_1^{2,0}$.

**Proof.** Consider an arbitrary finite automaton $FA = (Q, T, q_0, F, \delta)$. We define the insertion-deletion system $\Gamma = (V, T, A, I, D)$ in the following way:

$$
\begin{aligned}
V &= \{Q_i \mid q_i \in Q\} \cup \{A, B, E\} \cup T, \\
A &= \{ABE\}, \\
I &= \{(B,\, Q_i,\, \lambda)_{ins} \mid q_i \in Q\} \\
&\quad \cup \{(B,\, a,\, \lambda)_{ins} \mid a \in T\}, \\
D &= \{(Q_f,\, E,\, \lambda)_{del} \mid q_f \in F\} \\
&\quad \cup \{(Q_i a,\, Q_j,\, \lambda)_{del} \mid q_j \in \delta(q_i, a)\} \\
&\quad \cup \{(A,\, B,\, \lambda)_{del}, (A,\, Q_0,\, \lambda)_{del}, (\lambda,\, A,\, \lambda)_{del}\}.
\end{aligned}
$$

We claim that the only way for $\Gamma$ to generate a terminal string is to correctly simulate a trajectory of $FA$. Indeed, in order to erase $E$, $B$ has to insert at least one $Q_f$, for $q_f \in F$. Then, for any $Q_j$, $j > 0$, to be erased, there must be such a subword $Q_i a$ to its left that $q_j \in \delta(q_i, a)$. Note that, while several instances of a state symbol $Q_j$ can be inserted one next to the other, the contexts of deletion rules assure that there is exactly one instance of a terminal symbol between two state symbols. Since $Q_0$ is the only state symbol whose deletion does not depend on other state symbols, $B$ is guaranteed to insert a sequence of the form $Q_0^* a_{i_0} Q_{i_1}^* a_{i_1} \dots Q_{i_n}^* a_{i_n} Q_f^*$ which corresponds to a trajectory of $FA$ accepting the word $a_{i_0} a_{i_1} \dots a_{i_n}$.

Remark now that the only symbol which can erase $Q_0$ is $A$, so if the following derivation step occurs: $\alpha B Q_0 \beta \Rightarrow \alpha B N Q_0 \beta$, $N \in V \setminus \{Q_0\}$, $\alpha, \beta \in V^*$, the highlighted instance of $Q_0$ will never be erased. Indeed, $\Gamma$ deletes no terminals, and if $N = Q_j$, $j > 0$, an insertion of a terminal will

be later required to erase $N$. Furthermore, because $A$ is situated to the left of $B$, it will have to delete $B$ before it can delete $Q_0$, which means that, after $Q_0$ is deleted, no more insertions can happen and the only applicable rule $(\lambda, A, \lambda)_{del}$ deletes the last non-terminal from the string.

We finish the proof by pointing out that, if $A$ or $B$ is deleted before a complete trajectory of the automaton is generated, some of the state symbols will never be erased. Therefore, deleting $A$ or $B$ too early yields derivations which do not produce terminal strings.

The strictness of the inclusion follows from the fact that non-regular languages can be generated by systems of size $(1, 1, 0; 1, 1, 0)$ [18]. □

Next, we would like to show that the family of insertion-deletion systems of size $(1, 1, 0; 1, 2, 0)$ can generate non-context-free languages. In order to prove this we show that for any insertion-deletion system $\Gamma$ of size $(1, 1, 0; 1, 1, 0)$ and a regular language $R$ there exist an insertion-deletion system $\Gamma'$ of size $(1, 1, 0; 1, 2, 0)$ such that $L(\Gamma') = L(\Gamma) \cap R$. This, combined with the result from [13] showing that there exists an insertion-deletion system $\Gamma$ of size $(1, 1, 0; 1, 1, 0)$ such that $L(\Gamma) \cap (F_1F_0)^+(a_0a_1)^+ = L_{2^n} = \{(F_1F_0)^n(a_0a_1)^m \mid n \geq 2^{2m-2}\}$, would trivially yield the statement. We remark that we cannot use the result from [11], where a linear bounded automaton is simulated using leftist grammars, because an intersection with a context-free language is performed. We start with the following lemma.

**Lemma 3.1** *For an arbitrary insertion-deletion system $\Gamma = (V, T, A, I, D)$ of size $(1, 1, 0; 1, 1, 0)$ there exists an equivalent system $\Gamma' = (V', T, A', I', D')$ of the same size such that $L(\Gamma') = L(\Gamma)$ and no insertion rule of $\Gamma$ is of the form $(a, x, \lambda)_{del}$, with $a \in T$, $x \in V$.*

**Proof.** We define the alphabet of $\Gamma'$ to contain additional non-terminal symbols per each terminal: $V' = V \cup \{N_a, N'_a\}$. The new set of axioms is defined as $A' = \{h(w_0) \mid w_0 \in A\}$, where $h : V' \to V'$ is a morphism given by the following:

$$h(x) = \begin{cases} xN'_x, & \text{if } x \in T, \\ x, & \text{otherwise.} \end{cases}$$

We will use the notation $I_T$ to refer to those insertion rules of $\Gamma$ which have a terminal symbol in their context: $I_T = \{(a, x, \lambda)_{ins} \in I \mid a \in T, x \in V\}$. The insertion and deletion rules of $\Gamma'$ are

$$\begin{aligned} I' &= \{(N_a, x, \lambda)_{ins}, (N'_a, x, \lambda)_{ins} \mid (a, x, \lambda)_{ins} \in I\} \\ &\quad \cup \{(x, N_a, \lambda)_{ins} \mid (x, a, \lambda)_{ins} \in I\} \\ &\quad \cup I \setminus I_T, \\ D' &= \{(N_a, x, \lambda)_{del}, (N'_a, x, \lambda)_{del} \mid (a, x, \lambda)_{del} \in D\} \\ &\quad \cup \{(a, N_a, \lambda)_{del}, (a, N'_a, \lambda)_{del} \mid a \in T\} \\ &\quad \cup D. \end{aligned}$$

Remember that it is with no loss of generality that we can consider that $\Gamma$ does not ever delete terminal symbols (Lemma 2.1).

$\Gamma'$ directly simulates a derivation of $\Gamma$ by applying the rule $(x, N_a, \lambda)_{ins}$ right before any application of the rule $(x, a, \lambda)_{ins}$, and by replacing all insertions and deletions happening in the context of $a$ by insertions and deletions happening in the context of the corresponding $N_a$ or $N'_a$. At the very end, the rules $(a, N_a, \lambda)_{del}$ and $(a, N'_a, \lambda)_{del}$ are applied to finalise the clean-up of the string.

To see how $\Gamma$ can simulate any derivation of $\Gamma'$, remark that, to be erased, a symbol $N_a$ requires the presence of $a$ to its left, and consider the derivation

$$C : wv \Rightarrow wN_av \Rightarrow^* w'aN_av' \Rightarrow w'av',$$

where $w, v, w', v' \in V^*$. Because all contexts are of length at most 1 and are all to the left, it holds that $N_av \Rightarrow^* N_av'$, and $w \Rightarrow^* w'a$. But then it is possible to reorder $C$ in the following way:

$$wv \Rightarrow wN_av \Rightarrow^* w'aN_av \Rightarrow^* w'aN_av' \Rightarrow w'av'.$$

Therefore, it suffices to consider those derivations of $\Gamma'$ in which all symbols $N_a$ carry out operations only when there is a corresponding letter $a$ to the left of them. Remember also that, by the construction of $\Gamma'$, all symbols $N'_a$, are guaranteed to always be located immediately to the right of an instance of $a$, too. This means that, to simulate any derivation of $\Gamma'$, $\Gamma$ would just need to directly repeat applications of the rules which do not involve symbols $N_a$ and $N'_a$, and perform the operations happening in the context of $N_a$ or $N'_a$ in the context of the corresponding letters $a$.

We have therefore established that any terminal derivation of $\Gamma$ can be simulated by $\Gamma'$ and conversely, which implies that $L(\Gamma) = L(\Gamma')$ and proves the statement of the lemma. □

The previous lemma together with Lemma 2.1 effectively state that, in the case of insertion-deletion systems of size $(1, 1, 0; 1, 1, 0)$, any contiguous region of terminals is guaranteed to never change. This allows us to formulate the following lower bound for the power of systems of size $(1, 1, 0; 1, 2, 0)$.

**Theorem 3.2** *Consider an insertion-deletion system $\Gamma_1$ having the size $(1, 1, 0; 1, 1, 0)$ and a regular language $L$. Then there exists an insertion-deletion system $\Gamma_2$ of size $(1, 1, 0; 1, 2, 0)$ such that $L(\Gamma_2) = L(\Gamma) \cap L$.*

**Proof.** Consider an insertion-deletion system $\Gamma_1 = (V, T, A, I, D)$ with rules of size $(1, 1, 0; 1, 1, 0)$ and the finite automaton $FA = (Q, T, q_0, F, \delta)$ recognising the language $L$. Without losing generality, we may suppose that $\Gamma_1$ has no context-free rules (Lemma 2.2), never deletes terminal symbols (Lemma 2.1), and never inserts anything in the context of a terminal (Lemma 3.1). We will construct the insertion-deletion system $\Gamma_2 =$

$(V_2, T, A_2, I_2, D_2)$ in the following way:

$$\begin{aligned}
V_2 &= \{Q_i \mid q_i \in Q\} \cup \{B, E\} \cup V, \\
A_2 &= \{BQ_0w_0E \mid w_0 \in A\}, \\
I_2 &= \{(a,\, Q_i,\, \lambda)_{ins} \mid a \in T, q_i \in Q\} \cup I, \\
D_2 &= \{(Q_ia,\, Q_j,\, \lambda)_{del} \mid q_j \in \delta(q_i, a)\} \\
&\quad \cup \{(Q_f,\, E,\, \lambda)_{del} \mid q_f \in F\} \\
&\quad \cup \{(B,\, Q_0,\, \lambda)_{del}, (\lambda,\, B,\, \lambda)_{del}\} \cup D.
\end{aligned}$$

A terminal derivation of $\Gamma_2$ consists of two phases. In the first phase the rules from $I \cup D$ are applied, possibly resulting in a word of the form $BQ_0wE$, where $w \in T^*$. In the second phase, every terminal inserts a state symbol, and the deletion rules of the form $(Q_ia,\, Q_j,\, \lambda)_{del}$ check that a trajectory of $FA$ is correctly simulated. The rules of the form $(Q_f,\, E,\, \lambda)_{del}$ and $(B,\, Q_0,\, \lambda)_{del}$ assure that complete accepting trajectories of $FA$ are generated (cf. the proof of Theorem 3.1).

To see that $\Gamma_2$ cannot essentially deviate from this evolution scheme, consider the word $wQ_ju$, in which $w \in V_2^*$ and $u \in T^*$. According to our initial assumptions about $\Gamma_1$, the only symbols that may be inserted in $u$ are state symbols $Q_i$. However, since these symbols do not insert anything, they will only eventually get erased without influencing the form of the terminal word $u$. Remark now that the only way to erase $E$ is to insert $Q_f$ to the left of it. We can now inductively apply our observation about the words of the form $wQ_ju$ to conclude that, whenever $\Gamma_2$ succeeds in erasing all non-terminal symbols, the resulting terminal word belongs to the language recognised by the finite automaton $FA$. □

The following statement can be now be directly deduced from [13, Theorem 6].

**Theorem 3.3** *$INS_1^{1,0}DEL_1^{2,0}$ contains non-context-free languages.*

In what follows we will consider the relationship between families of leftist insertion-deletion systems. First we show that the contexts of the deletion do not add expressive power.

**Lemma 3.2** *$INS_1^{k,0}DEL_1^{k,0} \subseteq INS_1^{k,0}DEL_1^{1,0}$, $k \geq 1$.*

**Proof.** Consider an insertion-deletion system $\Gamma = (V, T, A, I, D)$ with rules of size $(1, k, 0; 1, k, 0)$. We construct the insertion-deletion system $\Gamma = (V', T, A, I', D')$ of size $(1, k, 0; 1, 1, 0)$ in the following way:

$$\begin{aligned}
V' &= \{X_r \mid r \in D\} \cup V, \\
I' &= \{(u,\, X_r,\, \lambda)_{ins} \mid (u,\, x,\, \lambda)_{del} \in D\} \cup I, \\
D' &= \{(X_r,\, x,\, \lambda)_{del}, (t,\, X_r,\, \lambda)_{del} \mid (u,\, x,\, \lambda)_{del} \in D, u = u't\}.
\end{aligned}$$

We immediately see that $L(\Gamma) \subseteq L(\Gamma')$, because any derivation step $w_i \overset{r}{\Rightarrow}_\Gamma w_{i+1}$ for $r \in I$ can be directly reproduced in $\Gamma'$, while a step with $r = (u,\, x,\, \lambda)_{del}$ can be simulated in three steps of $\Gamma'$:

$$w_i' u x w_i'' \Rightarrow_{\Gamma'} w_i' u X_r x w_i'' \Rightarrow_{\Gamma'} w_i' u X_r w_i'' \Rightarrow_{\Gamma'} w_i' u w_i'',$$

To see that $L(\Gamma') \subseteq L(\Gamma)$, consider the derivation

$$w_1 u v_1 \Rightarrow w_1 u X_r v_1 \Rightarrow^* w_2 X_r x v_2,$$

where $w_1, v_1, w_2, v_2 \in V^*$. Remember that one may require, without losing generality, that all rules in $\Gamma$ have contexts of length exactly $k$. This, together with the fact that $X_r$ is not included in the context of any insertion rule, implies that $x$ can appear to the right of $X_r$ in $w_2 X_r x v_2$ only if it was already there in $w_1 u X_r v_1$; in other words, $v_1 = x v_1'$. But then, to simulate a derivation of $\Gamma'$, $\Gamma$ has to directly reproduce the applications of the rules not involving $X_r$, and, instead of carrying out deletions in the context of $X_r$, perform them in the context of $u$ directly. This means that any terminal word produced by $\Gamma'$ can be also produced by $\Gamma$, which concludes the proof. □

The following lemma captures a similar inclusion property for systems having rules of size $(1, 1, 0; 1, k, 0)$.

**Lemma 3.3** *$INS_1^{k,0} DEL_1^{k,0} \subseteq INS_1^{1,0} DEL_1^{k,0}$, $k \geq 1$.*

**Proof.** Consider an insertion-deletion system $\Gamma = (V, T, A, I, D)$ with rules of size $(1, k, 0; 1, k, 0)$. We construct the system $\Gamma' = (V', T, A, I', D')$ of size $(1, 1, 0; 1, k, 0)$ in the following way:

$$\begin{aligned} V' &= \{X_r \mid r \in I\} \cup V, \\ I' &= \{(t,\, X_r,\, \lambda)_{ins}, (X_r,\, x,\, \lambda)_{ins} \mid (u,\, x,\, \lambda)_{ins} \in I, u = u't\}, \\ D' &= \{(u,\, X_r,\, \lambda)_{del} \mid (u,\, x,\, \lambda)_{ins} \in I\} \cup D. \end{aligned}$$

It is immediately clear that $L(\Gamma) \subseteq L(\Gamma')$, because all the rules from $D$ are included in $D'$ and the application of a rule $r = (u,\, x,\, \lambda)_{ins} \in I$ can be simulated as follows:

$$wuv \Rightarrow_{\Gamma'} wuX_r v \Rightarrow_{\Gamma'} wuX_r xv \Rightarrow_{\Gamma'} wuxv.$$

Consider now the following derivation of $\Gamma'$:

$$w_1 t v_1 \Rightarrow_{\Gamma'} w_1 t X_r v_1 \Rightarrow^*_{\Gamma'} w_2 X_r v_2 \Rightarrow_{\Gamma'} w_2 X_r x v_2 \Rightarrow^*_{\Gamma'} w_3 u X_r v_3 \Rightarrow_\Gamma w_3 u v_3,$$

where $w_i, v_i \in V^*$, $1 \leq i \leq 3$, and $w_2 X_r v_2$ is the first sentential form in which $X_r$ inserts an $x$. In this derivation the applications of the rules $(t,\, X_r,\, \lambda)_{ins}$, $(X_r,\, x,\, \lambda)_{ins}$, and $(u,\, X_r,\, \lambda)_{del}$ are interleaved with other operations. Yet,

since $X_r$ only appears in two other rules of $\Gamma'$ and because all rules in $\Gamma'$ only have left contexts, the following hold: $w_1 t \Rightarrow^* w_2 \Rightarrow^* w_3 u$, $v_1 \Rightarrow^* v_2$, and $X_r x v_2 \Rightarrow^* X_r v_3$. Therefore, the above derivation can be reordered as follows:

$$w_1 t v_1 \Rightarrow^* w_3 u v_2 \Rightarrow w_3 u X_r v_2 \Rightarrow w_3 u X_r x v_2 \Rightarrow w_3 u X_r v_3 \Rightarrow w_3 u v_3.$$

The possibility of such a reordering of any derivation involving $X_r$ and the fact that this symbol can only be erased in the context of a substring $u$ leads us to the conclusion that any such derivation can be reproduced in $\Gamma$ by performing the insertions of $x$ directly in the context of the corresponding substring $u$, and by directly carrying over the applications of other rules. This indicates that $L(\Gamma') \subseteq L(\Gamma)$ and concludes the proof. □

Lemmas 3.2 and 3.3 immediately imply the following statement.

**Theorem 3.4** $INS_1^{1,0}DEL_1^{k,0} = INS_1^{k,0}DEL_1^{1,0} = INS_1^{k,0}DEL_1^{k,0}$, $k \geq 1$.

Contrary to what one might expect, increasing the length of the left context in one-sided one-symbol insertion and deletion rules beyond 2 does not add expressive power. To prove this statement, we will first show that insertion-deletion systems of size $(1, k, 0; 1, k, 0)$ can simulate systems of size $(1, k, 0; 1, k{+}1, 0)$, for $k \geq 2$, and will then inductively apply this observation.

**Lemma 3.4** $INS_1^{k,0}DEL_1^{k+1,0} \subseteq INS_1^{k,0}DEL_1^{k,0}$, $k \geq 2$.

**Proof.** Consider the insertion-deletion system $\Gamma = (V, T, A, I, D)$ with rules of size $(1, k, 0; 1, k+1, 0)$. One can require without losing generality that all deletion contexts in $\Gamma$ are exactly of size $k+1$ and all insertion contexts are of size $k$. We will construct the system $\Gamma' = (V', T, A, I', D')$ of size $(1, k, 0; 1, k, 0)$ in the following way:

$$\begin{aligned} V' &= \{X_r \mid r \in D\} \cup V, \\ I' &= \{(x_1 \dots x_k,\, X_r,\, \lambda)_{ins} \\ &\quad \mid (x_1 \dots x_k x_{k+1},\, t,\, \lambda)_{del} \in D\} \cup I, \\ D' &= \{(x_1 \dots x_k,\, X_r,\, \lambda)_{del}, (X_r x_{k+1},\, t,\, \lambda)_{del} \\ &\quad \mid (x_1 \dots x_k x_{k+1},\, t,\, \lambda)_{del} \in D\}. \end{aligned}$$

We obtain that $L(\Gamma) \subseteq L(\Gamma')$ because $\Gamma'$ can directly reproduce any application of an insertion rule from $I$, while any application of a deletion rule $(x_1 \dots x_k x_{k+1},\, t,\, \lambda)_{del} \in D$ can be simulated as follows:

$$\begin{aligned} & w\, x_1 \dots x_k x_{k+1}\, t\, v \Rightarrow w\, x_1 \dots x_k X_r x_{k+1}\, t\, v \\ \Rightarrow\ & w\, x_1 \dots x_k X_r x_{k+1}\, v \Rightarrow w\, x_1 \dots x_k x_{k+1}\, v. \end{aligned}$$

Remark now that, since $X_r$ inserts no symbols, whenever it is inserted the symbol $x_{k+1}$ must already be present to the right of the insertion site.

Moreover, the same is true for the erased instance of $t$, because we require that all insertion rules in $\Gamma$ have contexts of length $k \geq 2$, and, on the other hand, $X_{k+1}$ does not appear in the context of any insertion rule and can only participate in the deletion of $t$. Therefore any derivation of $\Gamma'$ in which $X_r$ is inserted and triggers the deletion of at least a $t$ has the following form:

$$C : w_1x_1 \dots x_kx_{k+1}\, t\, v_1 \Rightarrow w_1x_1 \dots x_kX_rx_{k+1}\, t\, v_1 \Rightarrow^* w_2X_rx_{k+1}\, t\, v_2$$
$$\Rightarrow\ w_2X_rx_{k+1}v_2 \Rightarrow^* w_3x_1 \dots x_kX_rx_{k+1}v_3 \Rightarrow w_3x_1 \dots x_kx_{k+1}v_3.$$

But then, because of the fact that the rules of $\Gamma'$ are one-sided and that $X_r$ only appears in the context of one rule, we know that $w_1x_1 \dots x_k \Rightarrow^* w_2 \Rightarrow^* w_3x_1 \dots x_k$, $X_rx_{k+1}tv_1 \Rightarrow^* X_rx_{k+1}tv_2$, and $X_rx_{k+1}v_2 \Rightarrow^* X_rx_{k+1}v_3$. With this in mind, one can reorder $C$ in the following way:

$$w_1x_1 \dots x_kx_{k+1}\, t\, v_1 \Rightarrow w_1x_1 \dots x_kX_rx_{k+1}\, t\, v_1 \Rightarrow^* w_1x_1 \dots x_kX_rx_{k+1}\, t\, v_2$$
$$\Rightarrow w_1x_1 \dots x_kX_rx_{k+1}v_2 \Rightarrow^* w_1x_1 \dots x_kX_rx_{k+1}v_3 \Rightarrow w_1x_1 \dots x_kx_{k+1}v_3$$
$$\Rightarrow^* w_2x_{k+1}v_3 \Rightarrow^* w_3x_1 \dots x_kx_{k+1}v_3.$$

In this new derivation $X_r$ is inserted and deleted within the same substring $x_1 \dots x_kx_{k+1}$, which means that $\Gamma$ can simulate a subderivation of $\Gamma'$ employing $X_r$ by directly applying the deletion rule $(x_1 \dots x_kx_{k+1},\, t,\, \lambda)_{del}$ in the corresponding context $x_1 \dots x_kx_{k+1}$. Together with the fact that the applications of all insertion rules from $I$ can carried over to $\Gamma$ directly, this implies that $\Gamma$ can generate any terminal word $\Gamma'$ can produce and concludes the argument. □

Note that since all rules of deletion rules of size $(1, k, 0)$ are also of size $(1, k + 1, 0)$, the previous lemma actually shows equality between the two classes of languages, rather than inclusion. Inductively applying that statement and using Theorem 3.4 yields the following result.

**Theorem 3.5** *For any $m, q \in \mathbb{N}$ the following relations hold:*

$$INS_1^{1,0}DEL_1^{2,0} = INS_1^{2,0}DEL_1^{1,0} = INS_1^{m,0}DEL_1^{q,0}.$$

# 4 Conclusions

In this paper we considered insertion-deletion systems of size $(1, m, 0; 1, q, 0)$, $m, q \geq 0$. We showed that if $m + q > 2$ then the corresponding families are all equal. We also showed that they properly include the family of regular languages and that they contain non-context-free languages. The characterization of the above family as well as its equality to $RE$ is left as an open question. However, even though the considered systems look powerful, the fact that they can only check contexts on one side and that only one symbol can be inserted or deleted at a time leads us to the supposition that such systems cannot generate all recursively enumerable languages.

# References


[1] A. Alhazov, A. Krassovitskiy, Y. Rogozhin, and S. Verlan. Small size insertion and deletion systems. In C. Martin-Vide, ed., *Scientific Applications of Language Methods*, vol. 2 of *Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory*, chapter 9, 459–524. World Scientific, 2010.

[2] R. Benne. *RNA Editing: The Alteration of Protein Coding Sequences of RNA*. Ellis Horwood, Chichester, West Sussex, 1993.

[3] F. Biegler, M. J. Burrell, and M. Daley. Regulated RNA rewriting: Modelling RNA editing with guided insertion. *Theoretical Computer Science*, 387(2):103 – 112, 2007.

[4] M. Daley, L. Kari, G. Gloor, and R. Siromoney. Circular contextual insertions/deletions with applications to biomolecular computation. In *SPIRE/CRIWG*, 47–54, 1999.

[5] R. Freund, M. Kogler, Y. Rogozhin, and S. Verlan. Graph-controlled insertion-deletion systems. In I. McQuillan et al., eds, *Proceedings Twelfth Annual Workshop on Descriptional Complexity of Formal Systems*, vol. 31 of *EPTCS*, 88–98, 2010.

[6] B. Galiukschov. Semicontextual grammars. *Matematicheskaya Logica i Matematicheskaya Lingvistika*, pages 38–50, 1981. Tallin University, (in Russian).

[7] D. Haussler. *Insertion and Iterated Insertion as Operations on Formal Languages*. PhD thesis, University of Colorado at Boulder, 1982.

[8] D. Haussler. Insertion languages. *Information Sciences*, 31(1):77–89, 1983.

[9] S. Ivanov and S. Verlan. Random context and semi-conditional insertion-deletion systems. *CoRR*, abs/1112.5947, 2011.

[10] S. Ivanov and S. Verlan. About one-sided one-symbol insertion-deletion P systems. In A. Alhazov et al., eds, *Membrane Computing - 14th International Conference, CMC 2013, Chişinău, Republic of Moldova, August 20-23, 2013, Revised Selected Papers*, vol. 8340 of *Lecture Notes in Computer Science*, 225–237. Springer, 2013.

[11] T. Jurdziński. On complexity of grammars related to the safety problem. In M. Bugliesi et al., eds, *Automata, Languages and Programming*, vol. 4052 of *Lecture Notes in Computer Science*, pages 432–443. Springer, 2006.

[12] T. Jurdziński. Leftist grammars are non-primitive recursive. In L. Aceto et al., eds, *Automata, Languages and Programming*, vol. 5126 of *Lecture Notes in Computer Science*, 51–62. Springer, 2008.

[13] T. Jurdziński and K. Loryś. Leftist grammars and the Chomsky hierarchy. In M. Liśkiewicz and R. Reischuk, editors, *Fundamentals of Computation Theory*, vol. 3623 of *Lecture Notes in Computer Science*, 293–304. Springer, 2005.

[14] L. Kari. *On Insertion and Deletion in Formal Languages*. PhD thesis, University of Turku, 1991.

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

[16] L. Kari and G. Thierrin. Contextual insertions/deletions and computability. *Information and Computation*, 131(1):47–61, 1996.

[17] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, eds., *Automata Studies*, 3–41. Princeton University Press, Princeton, NJ, 1956.

[18] A. Krassovitskiy, Y. Rogozhin, and S. Verlan. Further results on insertion-deletion systems with one-sided contexts. In C. Martín-Vide et al., eds., *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, vol. 5196 of *Lecture Notes in Computer Science*, 333–344. Springer, 2008.

[19] A. Krassovitskiy, Y. Rogozhin, and S. Verlan, Computational power of insertion-deletion (P) systems with rules of size two, *Natural Computing*, 10(2), 835–852, 2011.

[20] S. Marcus. Contextual grammars. *Revue Roumaine de Mathématiques Pures et Appliquées*, 14:1525–1534, 1969.

[21] M. Margenstern, G. Păun, Y. Rogozhin, and S. Verlan. Context-free insertion-deletion systems. *Theoretical Computer Science*, 330(2):339–348, 2005.

[22] A. Matveevici, Y. Rogozhin, and S. Verlan. Insertion-deletion systems with one-sided contexts. In J. O. Durand-Lose and M. Margenstern, eds., *Machines, Computations, and Universality, 5th International Conference, MCU 2007, Orléans, France, September 10-13, 2007, Proceedings*, vol. 4664 of *Lecture Notes in Computer Science*, 205–217. Springer, 2007.

[23] R. Motwani, R. Panigrahy, V. Saraswat, and S. Ventkatasubramanian. On the decidability of accessibility problems (extended abstract). In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, 306–315, 2000.

[24] I. Petre and S. Verlan. Matrix insertion-deletion systems. *Theoretical Computer Science*, 456:80 – 88, 2012.

[25] G. Păun. *Marcus Contextual Grammars*. Kluwer Academic Publishers, 1997.

[26] G. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing: New Computing Paradigms*. Springer, 1998.

[27] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer, Berlin, 1997.

[28] W. D. Smith. DNA computers in vitro and in vivo. In R. Lipton and E. Baum, editors, *Proceedings of DIMACS Workshop on DNA Based Computers*, DIMACS Series in Discrete Math. and Theoretical Computer Science, 121–185. American Mathematical Society, 1996.

[29] A. Takahara and T. Yokomori. On the computational power of insertion-deletion systems. In M. Hagiya and A. Ohuchi, eds., *DNA Computing, 8th International Workshop on DNA Based Computers, DNA8, Revised Papers*, vol. 2568 of *Lecture Notes in Computer Science*, 269–280. Springer, 2002.

[30] S. Verlan. On minimal context-free insertion-deletion systems. *Journal of Automata, Languages and Combinatorics*, 12(1-2):317–328, 2007.

[31] S. Verlan. *Study of Language-theoretic Computational Paradigms Inspired by Biology*. Habilitation thesis, Université Paris Est, 2010.

*Sergiu Ivanov*
LACL, Université Paris Est – Créteil Val de Marne
61, av. du général de Gaulle, 94010 Créteil Cedex, France
E-mail: `sergiu.ivanov@u-pec.fr`

*Serghei Verlan*
LACL, Université Paris Est – Créteil Val de Marne
61, av. du général de Gaulle, 94010 Créteil Cedex, France
E-mail: `verlan@u-pec.fr`