Insertion-Deletion Systems with Substitutions I

Martin Vu[1] and Henning Fernau[2](✉)

[1] FB 3 - Informatik, Universität Bremen, Bremen, Germany
martin.vu@uni-bremen.de
[2] Universität Trier, Fachber. 4 – Abteilung Informatikwissenschaften, 54286 Trier, Germany
fernau@uni-trier.de

Abstract. With good biological motivation, we add substitutions as a further type of operations to (in particular, context-free) insertion-deletion systems. This way, we obtain new characterizations of and normal forms for context-sensitive and recursively enumerable languages.

Keywords: Computational completeness · Context-sensitive · Insertions · Deletions · Substitutions

1 Introduction

Insertion-deletion systems, or ins-del systems for short, are well established as computational devices and as a research topic within Formal Languages throughout the past nearly 30 years, starting off with the PhD thesis of Kari [3].

However, from its very beginning, papers highlighting the potential use of such systems in modelling DNA computing also discussed the replacement of single letters (possibly within some context) by other letters, an operation called *substitution* in [2,4]. Interestingly, all theoretical studies on grammatical mechanisms involving insertions and deletions omitted including the substitution operation in their studies. With this paper, we are stepping into this gap by studying ins-del systems with substitutions, or ins-del-sub systems for short.

We put special emphasis on extending context-free ins-del systems with substitutions. We observe quite diverse effects, depending on whether the substitutions are context-free, one-sided or two-sided. We can characterize the context-sensitive languages by extending context-free insertion systems with substitutions, which can be seen as a new normal form for monotone Chomsky grammars. For omitted proofs and further results, see [13].

2 Basic Definitions and Observations

We assume the reader to be familiar with the basics of formal language theory.

An *ins-del system* is a 5-tuple $ID = (V, T, A, I, D)$, consisting of two alphabets V and T with $T \subseteq V$, a finite language A over V, a set of *insertion* rules I and a set of *deletion* rules D. Both sets of rules are formally defined as sets of triples of the form (u, a, v) with $a, u, v \in V^*$ and $a \neq \lambda$. We call elements occurring in T *terminal* symbols, while referring to elements of $V \backslash T$ as *nonterminals.* Elements of A are called *axioms.*

Let w_1uvw_2, with $w_1, u, v, w_2 \in V^*$, be a string. Applying the insertion rule $(u, a, v) \in I$ inserts the string $a \in V^*$ between u and v, which results in the string w_1uavw_2.

The application of a deletion rule $(u, a, v) \in D$ results in the removal of an substring a from the context (u, v). More formally let $w_1uavw_2 \in V^*$ be a string. Then, applying $(u, a, v) \in D$ results in the string w_1uvw_2.

We define the relation $\Longrightarrow$ as follows: Let $x, y \in V^*$. Then we write $x \Longrightarrow_{\text{ins}} y$ if y can be obtained by applying an insertion or deletion rule to x. We also write $(u, a, v)_{\text{ins}}$ or $(u, a, v)_{\text{del}}$ to specify whether the applied rule has been an insertion or a deletion rule. Consider $(u, a, v)_{\text{ins}}$ or $(u, a, v)_{\text{del}}$. Then we refer to u as the *left context* and to v as the *right context* of $(u, a, v)_{\text{ins}}/(u, a, v)_{\text{del}}$.

Let $ID = (V, T, A, I, D)$ be an ins-del system. The language generated by ID is defined by $L(ID) = \{w \in T^* \mid \alpha \Longrightarrow^* w, \alpha \in A\}$.

The *size* of ID describes its descriptional complexity and is defined by a tuple $(n, m, m'; p, q, q')$, where

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

By $\text{INS}_n^{m,m'}\text{DEL}_p^{q,q'}$ we denote the family of all ins-del systems of size $(n, m, m'; p, q, q')$ [1,12]. Depending on the context, we also denote the family of languages characterized by ins-del systems of size $(n, m, m'; p, q, q')$ by $\text{INS}_n^{m,m'}\text{DEL}_p^{q,q'}$. We call a family $\text{INS}_n^{0,0}\text{DEL}_p^{0,0}$ a family of *context-free* ins-del systems, while we call a family $\text{INS}_n^{m,m'}\text{DEL}_p^{q,q'}$ with $m + m' > 0 \wedge mm' = 0$ or $(q + q' > 0 \wedge qq' = 0)$ a family of *one-sided* ins-del systems. According to [1], an ins-del system

$$ID' = (V \cup \{\$\}, T, A', I', D' \cup \{(\lambda, \$, \lambda)\})$$

of size $(n, m, m'; p, q, q')$ is said to be *in normal form* if

- for any $(u, a, v) \in I'$, it holds that $|a| = n,\ |u| = m$ and $|v| = m'$, and
- for any $(u, a, v) \in D'$, it holds that $|a| = p,\ |u| = q$ and $|v| = q'$.

Alhazov *et al.* [1,12] have shown the following auxiliary result:

Theorem 1. *For every ins-del system ID, one can construct an insertion-deletion system ID′ in normal form of the same size with* $L(ID') = L(ID)$.

In the following sections, we use a modified normal form for ins-del systems of size $(1, 1, 1; 1, 1, 1)$. Given an arbitrary ins-del system of size $(1, 1, 1; 1, 1, 1)$, the construction of this modified normal form is as follows:

Construction 1. *Let $ID = (V, T, A, I, D)$ be an ins-del system of size $(1,1,1;1,1,1)$. Without loss of generality, we assume $\{\$, X\} \cap V = \emptyset$ and $\$ \neq X$. We construct $ID'' = (V \cup \{\$, X\}, T, A'', I'', D'')$ as follows:*

$$\begin{aligned}
A'' &= \{X\$\alpha\$X \mid \alpha \in A\}\\
I'' &= \{(z_1, s, z_2) \mid (r, s, t) \in I,\ z_1 = r \textit{ if } r \neq \lambda \textit{ and } z_1 = \$ \textit{ otherwise},\\
&\qquad z_2 = t \textit{ if } t \neq \lambda \textit{ and } z_2 = \$ \textit{ otherwise}\}\\
&\quad \cup \{(z_1, \$, z_2) \mid z_1, z_2 \in (\{\$\} \cup V)\}\\
D'' &= \{(z_1, a, z_2) \mid (u, a, v) \in D,\ z_1 = u \textit{ if } u \neq \lambda \textit{ and } z_1 = \$ \textit{ otherwise},\\
&\qquad z_2 = v \textit{ if } v \neq \lambda \textit{ and } z_2 = \$ \textit{ otherwise}\}\\
&\quad \cup \{(z_1, \$, z_2) \mid z_1, z_2 \in (\{\$\} \cup \{X\} \cup V)\} \cup \{(\lambda, X, \lambda)\}
\end{aligned}$$

The basic idea of Construction 1 is the same as in the usual normal form constructions (see Theorem 1): the symbol \$ is used as a padding symbol to ensure that the left and right contexts of all rules are of the required size. We can show that Construction 1 is equivalent to the usual normal form construction.

Theorem 2. *Let $ID' = (V \cup \{\$\}, T, A', I', D' \cup \{(\lambda, \$, \lambda)\})$ be an ins-del system of size $(1,1,1;1,1,1)$ in normal form and $ID'' = (V \cup \{\$, X\}, T, A'', I'', D'')$ be defined according to Construction 1. Then, $L(ID') = L(ID'')$.*

Unlike the usual normal form construction, context-free deletions can only occur at the beginning and the end of a sentential form in the case of Construction 1. This fact will prove useful below.

Ins-del systems have been extensively studied regarding the question if they can describe all of the recursively enumerable languages. Let us summarize these results first by listing the classes of languages known to be equal to RE: $\mathrm{INS}_1^{1,1}\mathrm{DEL}_1^{1,1}$ [10], $\mathrm{INS}_3^{0,0}\mathrm{DEL}_2^{0,0}$ and $\mathrm{INS}_2^{0,0}\mathrm{DEL}_3^{0,0}$ [7], $\mathrm{INS}_1^{1,1}\mathrm{DEL}_2^{0,0}$ [9, Theorem 6.3], $\mathrm{INS}_2^{0,0}\mathrm{DEL}_1^{1,1}$ [5], $\mathrm{INS}_2^{0,1}\mathrm{DEL}_2^{0,0}$ and $\mathrm{INS}_1^{1,2}\mathrm{DEL}_1^{1,0}$ [8], $\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,2}$ [5]. By way of contrast, the following language families are known not to be equal to RE, the first one is even a subset of CF: $\mathrm{INS}_2^{0,0}\mathrm{DEL}_2^{0,0}$ [12], $\mathrm{INS}_1^{1,1}\mathrm{DEL}_1^{1,0}$ [8], $\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,1}$ [5], $\mathrm{INS}_1^{1,0}\mathrm{DEL}_2^{0,0}$ and $\mathrm{INS}_2^{0,0}\mathrm{DEL}_1^{1,0}$ [6].

We define *substitution rules* to be of the form $(u, a \to b, v)$; $u, v \in V^*$; $a, b \in V$. Let w_1uavw_2; $w_1, w_2 \in V^*$ be a string over V. Then applying the substitution rule $(u, a \to b, v)$ allows us to substitute a single letter a with another letter b in the context of u and v, resulting in the string w_1ubvw_2. Formally, we define an ins-del-sub system to be a 6-tuple $ID_r = (V, T, A, I, D, S)$, where V, T, A, I and D are defined as in the case of usual ins-del systems and S is a set of substitution rules. Substitution rules define a relation $\Longrightarrow_{\text{sub}}$: Let $x = w_1uavw_2$ and $y = w_1ubvw_2$ be strings over V. We write $x \Longrightarrow_{\text{sub}} y$ iff there is a substitution rule $(u, a \to b, v)$. In the context of ins-del-sub systems, we write $\hat{\Longrightarrow}$ to denote any of the relations $\Longrightarrow_{\text{ins}}$, $\Longrightarrow_{\text{del}}$ or $\Longrightarrow_{\text{sub}}$. We define the closures $\hat{\Longrightarrow}^*$ and $\hat{\Longrightarrow}^+$ as usual. The language generated by an ins-del-sub system ID_r is defined as $L(ID_r) = \{w \in T^* \mid \alpha \hat{\Longrightarrow}^* w,\ \alpha \in A\}$.

As with usual ins-del system, we measure the complexity of an ins-del-sub system $ID_r = (V, T, A, I, D, S)$ via its *size*, which is defined as a tuple

$(n, m, m'; p, q, q'; r, r')$, where n, m, m', p, q and q' are defined as in the case of usual ins-del systems, while r and r' limit the maximal length of the left and right context of a substitution rule, respectively, i.e., $r = \max\{|u| \mid (u, a \to b, v) \in S\}$, $r' = \max\{|v| \mid (u, a \to b, v) \in S\}$. $\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}\mathrm{SUB}^{r,r'}$ denotes the family of all ins-del-sub systems of size $(n, m, m'; p, q, q'; r, r')$. Note that, as only one letter is replaced by any substitution rule, there is no subscript below SUB. Depending on the context, we also refer to the family of languages generated by ins-del-sub systems of size $(n, m, m'; p, q, q'; r, r')$ by $\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}\mathrm{SUB}^{r,r'}$. Expanding our previous terminology, we call substitution rules of the form $(\lambda, a \to b, \lambda)$ *context-free*, while substitution rules of the form $(u, a \to b, \lambda)$ or $(\lambda, a \to b, v)$ with $u \neq \lambda \neq v$ are called *one-sided.* Substitution rules of the form $(u, a \to b, v)$ with $u \neq \lambda \neq v$ are referred to as *two-sided.*

Let ^{R} be the the reversal (mirror) operator. For a language L and its mirror L^R the following lemma holds.

Lemma 1. *$L \in INS_n^{m,m'}DEL_p^{q,q'}SUB^{r,r'}$ iff $L^R \in INS_n^{m',m}DEL_p^{q',q}SUB^{r',r}$.*

We will now define the term *resolve.* Let $ID_r = (V, T, A, I, D, S)$ be an ins-del-sub system. We say that a nonterminal X of ID_r is resolved if X is either deleted or substituted. It is easy to see that in any terminal derivation of ID_r all nonterminals must be resolved at some point of the derivation. We remark that a nonterminal X may be resolved by being substituted with a nonterminal Y, which in turn must be resolved.

As in the case of ins-del systems without substitution rules, we define a *normal form* for ins-del-sub systems. An ins-del-sub system

$$ID_r = (V \cup \{\$\}, T, A, I, D \cup \{(\lambda, \$, \lambda)\}, S)$$

of size $(n, m, m'; p, q, q'; r, r')$ is said to be in normal form if

– for any $(u, a, v) \in I$, it holds that $|a| = n,\ |u| = m$ and $|v| = m'$;
– for any $(u, a, v) \in D$, it holds that $|a| = p,\ |u| = q$ and $|v| = q'$;
– for any $(u, a \to b, v) \in S$, it holds that $|u| = r$ and $|v| = r'$.

Theorem 3. *For every ins-del-sub system ID_r of size $(n, m, m'; p, q, q'; r, r')$, one can construct an ins-del-sub system ID_r' of the same size in normal form, with $L(ID_r') = L(ID_r)$.*

Proof. Let $ID_r = (V, T, A, I, D, S)$ be an ins-del-sub system of size $(n, m, m'; p, q, q'; r, r')$. The basic idea is similar to the normal form construction for ins-del systems in [1, 12]. In fact, the sets of insertion and deletion rules of $ID_r' = (V \cup \{\$\}, T, A', I', D' \cup \{(\lambda, \$, \lambda)\}, S')$ are constructed as in the ins-del system normal form construction. S' and A' are defined as follows:

$$S' = \{z_1, a \to b, z_2 \mid (u, a \to b, v) \in S,\ z_1 \in u \sqcup\!\sqcup \$^*,\ |z_1| = r,\ z_2 \in v \sqcup\!\sqcup \$^*,\ |z_2| = r'\},$$
$$A' = \{\$^i \alpha \$^t \$^j \mid \alpha \in A,\ i = \max\{m, q, r\},\ j = \max\{m', q', r'\},\ t = \max\{p - |w|, 0\}\}.$$

As in Theorem 1, \$ is a new symbol, that is, $\$ \notin V$, which is introduced to be the padding symbol.

Let $h : V \cup \{\$\} \to V$ be a homomorphism with $h(x) = x$ if $x \in V$ and $h(\$) = \lambda$. $\alpha \xRightarrow[ID_r]{\wedge}{}^* w,\ \alpha \in A$ if and only if $\$^i\alpha\$^t\$^j \xRightarrow[ID'_r]{\wedge}{}^* w'$ with $h(w') = w$ can be shown by induction. While the only-if part follows easily, consider the following for the if part. We can assume that in any derivation of ID'_r the first i and the last j letters of the axiom are not deleted until the very end of derivation. Hence, insertion rules of the form $(z_1, \$^n, z_2)$ with $z_1 \in (\$ \cup V)^m$, $z_2 \in (\$ \cup V)^{m'}$ are applicable until the very end. It is clear that due to insertion rules of the form $(z_1, \$^n, z_2)$ and the deletion rule $(\lambda, \$, \lambda)$ it is possible to generate an arbitrary number of \$ at an arbitrary position of a sentential form of ID'_r. □

The following result will be useful in subsequent proofs; compare to Lemma 1.

Lemma 2. *Let $\mathcal{L}$ be a family of languages that is closed under reversal. Then:*

1. $\mathcal{L} \subseteq INS_n^{m,m'} DEL_p^{q,q'} SUB^{r,r'}$ *iff* $\mathcal{L} \subseteq INS_n^{m',m} DEL_p^{q',q} SUB^{r',r}$.
2. $INS_n^{m,m'} DEL_p^{q,q'} SUB^{r,r'} \subseteq \mathcal{L}$ *iff* $INS_n^{m',m} DEL_p^{q',q} SUB^{r',r} \subseteq \mathcal{L}$.

Due to the definition of ins-del-sub systems, the following result is clear.

Lemma 3. $INS_n^{m,m'} DEL_p^{q,q'} \subseteq INS_n^{m,m'} DEL_p^{q,q'} SUB^{r,r'}$.

Whether this inclusion is proper, is the question, that will be addressed in the following sections. We will see that while in some cases an arbitrary system of size $(n, m, m'; p, q, q', r, r')$ can be simulated by a system of size $(n, m, m'; p, q, q')$, this is not the general case. Furthermore, we will see that families $\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}$, which are not computationally complete, may reach computational completeness via an extension with substitution rules. Additionally, we will see below that families of ins-del systems which are equally powerful may no longer be after being extended with the same class of substitution rules, i.e., we have $\mathrm{INS}_{n_1}^{m_1,m'_1}\mathrm{DEL}_{p_1}^{q_1,q'_1} = \mathrm{INS}_{n_2}^{m_2,m'_2}\mathrm{DEL}_{p_2}^{q_2,q'_2}$, but possibly $\mathrm{INS}_{n_1}^{m_1,m'_1}\mathrm{DEL}_{p_1}^{q_1,q'_1}\mathrm{SUB}^{r,r'} \subset \mathrm{INS}_{n_2}^{m_2,m'_2}\mathrm{DEL}_{p_2}^{q_2,q'_2}\mathrm{SUB}^{r,r'}$. The reverse case might occur, as well.

Because the application of an insertion rule (u, x, v) corresponds to the application of the monotone rewriting rule $uv \to uav$ and the application of a substitution rule $(u, a \to b, v)$ corresponds to the application of the monotone rewriting rule $uav \to ubv$, a monotone grammar can simulate derivations of an insertion-substitution system. (More technically speaking, we have to do the replacements on the level of pseudo-terminals N_a for each terminal a and also add rules $N_a \to a$, but these are minor details.) Hence, we can conclude:

Theorem 4. *For any integers $m, m', n, r, r' \geq 0$, $INS_n^{m,m'} DEL_0^{0,0} SUB^{r,r'} \subseteq CS$.*

3 Main Results

We will focus on context-free ins-del systems, which are extended with substitution rules. More precisely, we will analyze the computational power of the family of systems $\mathrm{INS}_n^{0,0}\mathrm{DEL}_p^{0,0}\mathrm{SUB}^{r,r'}$.

We are going to analyze substitution rules of the form $(\lambda, a \to b, \lambda)$, which means that letters may be substituted regardless of any context. We will show that extending context-free ins-del systems with context-free substitution rules does not result in a more powerful system. In fact, a context-free ins-del-sub system of size $(n, 0, 0; p, 0, 0; 0, 0)$ can be simulated by an ins-del system of size $(n, 0, 0; p, 0, 0)$.

Theorem 5. *Let $ID_r \in INS_n^{0,0} DEL_p^{0,0} SUB^{0,0}$, then there exists an ins-del system $ID \in INS_n^{0,0} DEL_p^{0,0}$ such that $L(ID_r) = L(ID)$.*

Proof (Sketch). Let $ID_r = (V, T, A, I, D, S) \in \mathrm{INS}_n^{0,0}\mathrm{DEL}_p^{0,0}\mathrm{SUB}^{0,0}$. It is clear that any letter a, that is to be replaced by a substitution rule $(\lambda, a \to b, \lambda)$, has been introduced by either an insertion rule $(\lambda, w_1 a w_2, \lambda)$ or as part of an axiom $w_1' a w_2'$ at some point before executing the substitution. As a serves no purpose other than to be replaced (i.e., it is not used as a context), the basic idea is to skip introducing a altogether and introduce b instead. More formally: instead of applying an insertion rule $(\lambda, w_1 a w_2, \lambda)$/an axiom $w_1' a w_2'$ and replacing a via $(\lambda, a \to b, \lambda)$ at a later point, we introduce a new insertion rule $(\lambda, w_1 b w_2, \lambda)$/a new axiom $w_1' b w_2'$, which we apply instead of $(\lambda, w_1 a w_2, \lambda)/w_1' a w_2'$. This idea can be cast into an algorithm to produce an ins-del system $ID = (V, T, A', I', D)$ with $L(ID_r) = L(ID)$. As only context-free insertion rules of size maximum $(n, 0, 0)$ are added to I', it is clear that $ID \in \mathrm{INS}_n^{0,0}\mathrm{DEL}_p^{0,0}$ holds. $\square$

Considering the question about the generative power of context-free ins-del systems with context-free substitution rules compared to usual context-free ins-del systems, Theorem 5 and Lemma 3 together yield:

Corollary 1. *$INS_n^{0,0} DEL_p^{0,0} SUB^{0,0} = INS_n^{0,0} DEL_p^{0,0}$.*

Example 1. Consider the ins-del-sub system

$$ID_r = (\{a, b, c\}, \{a, b, c\}, \{\lambda\}, \{(\lambda, aaa, \lambda)\}, \emptyset, S)$$

with $S = \{(\lambda, a \to b, \lambda), (\lambda, b \to c, \lambda)\}$. The language generated by ID_r is $L(ID_r) = \{w \mid w \in \{a, b, c\}^*,\ |w| = 3n,\ n \in \mathbb{N}\}$. Using the construction introduced in Theorem 5 yields the ins-del system $ID = (\{a, b, c\}, \{a, b, c\}, \{\lambda\}, I, \emptyset)$, with $I = \{(\lambda, x_1 x_2 x_3, \lambda) \mid x_1, x_2, x_3 \in \{a, b, c\}\}$. While it is clear that $L(ID) = L(ID_r)$, we remark that this example shows that the construction method of Theorem 5 may yield an ins-del system whose number of rules is exponentially greater than the number of rules of the system with substitutions.

3.1 Extension with One-Sided Substitution

Now, we will analyze the effect of one-sided substitution rules if used to extend a context-free ins-del system. We will show that using one-sided substitution rules can greatly increase the computational power of context-free insertion and deletion rules. In some cases, we even get computationally completeness results.

We will now construct an ins-del-sub system ID'_r of size $(1, 0, 0; 1, 0, 0; 1, 0)$ which simulates ID_r of size $(1, 1, 0; 1, 1, 0; 1, 0)$. The system ID'_r is constructed in the following manner:

Construction 2. *We assume the system $ID_r = (V, T, A, I, D, S)$ to be in normal form and any rule of ID_r to be labelled in a one-to-one manner, i.e., there is a bijection between a set of labels and the rule set. Let \$ be the padding symbol used in the construction of the normal form of ID_r. The system $ID'_r = (V', T, A, I', D', S \cup S')$ is constructed as follows. For each rule of ID_r, we introduce a new nonterminal X_i and define*

$$V' = V \cup \{X_i \mid i \textit{ is the label of a rule of } ID_r\}.$$

The set I' of insertion rules of ID'_r contains all (λ, X_i, λ), where i is the label of an insertion rule of ID_r, while the set D' of deletion rules as contains $(\lambda, \$, \lambda)$ and all (λ, X_i, λ), where i is the label of a deletion rule of ID_r. Furthermore, we define the set of substitution rules $S' = S_1 \cup S_2$, with

$$S_1 = \{(u, X_i \to a, \lambda) \mid i \textit{ is the label of an insertion rule } (u, a, \lambda) \textit{ of } ID_r\} \quad \textit{and}$$

$$S_2 = \{(u, a \to X_i, \lambda) \mid i \textit{ is the label of a deletion rule } (u, a, \lambda) \textit{ of } ID_r, u \neq \lambda\}.$$

Each deletion rule $(u, a, \lambda) \in D$ of ID_r, where i is the label of (u, a, λ), corresponds to a deletion rule $(\lambda, X_i, \lambda) \in D'$ and a substitution rule $(u, a \to X_i, \lambda) \in S_2$ of ID'_r. The basic idea of the construction is to simulate a deletion rule $(u, a, \lambda) \in D$ by substituting the letter a with left context u via $(u, a \to X_i, v) \in S_2$. The introduced nonterminal X_i is then deleted at some point by the deletion rule $(\lambda, X_i, \lambda) \in D'$. It is clear that a derivation of the form

$$w_1 u a w_2 \overset{\wedge}{\Longrightarrow} w_1 u X_i w_2 \overset{\wedge}{\Longrightarrow} w_1 u w_2,$$

in which the application of $(\lambda, X_i, \lambda) \in D'$ succeeds an application of $(u, a \to X_i, \lambda) \in S_2$ immediately, is equivalent to the application of a deletion rule $(u, a, v) \in D$. It needs much more care to prove the following converse:

Proposition 1. *Let $\alpha \in A$. Consider a derivation $\alpha \overset{\wedge}{\Longrightarrow}{}^* w \in T^*$ of ID'_r. Then, there is an alternative derivation of ID'_r, leading from α to w, in which all nonterminals $X_i \in V' \backslash V$ are resolved immediately after being introduced.*

This allows us to state:

Theorem 6. $INS_1^{0,0} DEL_1^{0,0} SUB^{1,0} = INS_1^{1,0} DEL_1^{1,0} SUB^{1,0}$.

Consider ins-del systems of size $(1, 0, 0; 1, 0, 0)$ extended with one-sided substitution rules; the increase in computational power is quite significant:

$$\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0} \subset \mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0} \subset \mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0}\mathrm{SUB}^{1,0} = \mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$$

Both inclusions are proper. First, observe that $ba^+ \in \mathrm{INS}_1^{1,0}\mathrm{DEL}_0^{0,0} \backslash \mathrm{INS}_1^{0,0} \mathrm{DEL}_1^{0,0}$. The system of size $(1, 1, 0; 0, 0, 0; 1, 0)$ presented in Example 2 generates $(ba)^+$. Verlan [12, Theorem 5.3] has shown that even ins-del systems of size $(1, 1, 0; 1, 1, 1)$ cannot generate the language $(ba)^+$.

Example 2. Consider the following ins-del-sub system: $ID_s = (V, T, I, \emptyset, S)$, with $V = \{X_1, X_2, X_3, a, b\}$, $T = \{a, b\}$ and $A = \{ba, bX_1X_3\}$. The set of insertion rules is defined as $I = \{(X_1, X_2, \lambda), (X_2, X_1, \lambda)\}$, while the set of substitution rules is $S = \{(b, X_1 \to a, \lambda), (a, X_2 \to b, \lambda), (b, X_3 \to a, \lambda)\}$. The generated language is $L(ID_s) = (ba)^+$, as we can easily see that any generated word begins with a letter b and ends with a letter a. Furthermore, any word generated by ID_s is not of the form w_1bbw_2, as the only way to introduce the terminal symbol b (except for the b introduced via the axiom) is by substituting a nonterminal X_2 with b. However, this substitution requires a left context a, which means that at some point the letter to the left of any b has been a. There are no insertion rules which can insert an additional b or X_2 between a and b. Furthermore, there are no deletion rules at all, which means that no a can be deleted. Therefore, the letter to the left of any b cannot be another b. Using the same argumentation, we can see, that any word generated by ID_s is not of the form w_1aaw_2, either.

It is easy to see that a result identical to Theorem 6 can be shown analogously for the mirrors of $\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$ and $\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0}\mathrm{SUB}^{1,0}$. Therefore:

Corollary 2. $INS_1^{0,0}DEL_1^{0,0}SUB^{0,1} = INS_1^{0,1}DEL_1^{0,1}SUB^{0,1}$.

We now analyze the computational power of ins-del-sub systems of size $(2, 0, 0; 2, 0, 0; 0, 1)$. While the family of ins-del systems of size $(2, 0, 0; 2, 0, 0)$ is known to be a proper subset of CF, see [11,12], we will show that an extension with substitution rules of the form $(\lambda, A \to B, C)$ results in a significant increase in computational power. More precisely, by simulating an ins-del systems of size $(2, 0, 1; 2, 0, 0)$, we will show that $\mathrm{INS}_2^{0,0}\mathrm{DEL}_2^{0,0}\mathrm{SUB}^{0,1} = \mathrm{RE}$ holds.

Construction 3. *Let $ID = (V, T, A, I, D)$ be a system of size $(2, 0, 1; 2, 0, 0)$ in normal form and all insertion rules of ID be labelled in a one-to-one manner. We construct the system $ID_r = (V', T, A, I', D, S')$, which simulates ID, as follows:*

$$V' = V \cup \{N_{i,2}, N_{i,1} \mid i \textit{ is the label of an insertion rule } (\lambda, ab, c) \in I\}$$
$$I' = \{(\lambda, N_{i,2}N_{i,1}, \lambda) \mid i \textit{ is the label of an insertion rule } (\lambda, ab, c) \in I\}$$
$$S' = \{(\lambda, N_{i,1} \to b, c), (\lambda, N_{i,2} \to a, b) \mid i \textit{ is the label of a rule } (\lambda, ab, c) \in I\}$$

The basic idea of Construction 3 is essentially the same as in Construction 2: as context-free insertion rules cannot scan for contexts (by definition), this task is handled by the corresponding substitution rules. Consider an insertion rule $(\lambda, N_{i,2}N_{i,1}, \lambda)$ of ID_r where i is the label of an insertion rule $(\lambda, ab, c) \in I$. Then the substitution rules, corresponding to this rule, are $(\lambda, N_{i,1} \to b, c)$ and $(\lambda, N_{i,2} \to a, b)$. This idea leads us to:

Theorem 7. $INS_2^{0,1}DEL_2^{0,0} \subseteq INS_2^{0,0}DEL_2^{0,0}SUB^{0,1}$.

As $\mathrm{INS}_2^{0,1}\mathrm{DEL}_2^{0,0} = \mathrm{RE}$ holds according to [8, Theorem 5], we conclude:

Corollary 3. $INS_2^{0,0}DEL_2^{0,0}SUB^{0,1} = RE$.

This is an interesting result as the families of ins-del systems of size $(2,0,0;2,0,0)$ and of size $(2,0,0;0,0,0)$ are known to be equal [12, Theorem 4.7], yet both classes extended with the same class of substitution rules differ in computational power. As RE and CS are closed under reversal, the next corollary follows with Lemma 2 and Theorem 4.

Corollary 4. *$INS_2^{0,0}DEL_0^{0,0}SUB^{1,0} \subseteq CS$ and $INS_2^{0,0}DEL_2^{0,0}SUB^{1,0} = RE$.*

3.2 Extension with Two-Sided Substitution

After analyzing the effect of context-free and one-sided substitution rules on context-free ins-del systems, we will now proceed to two-sided substitution rules, i.e., substitution rules with left and right context. Somehow surprisingly, this lifts the computational power of even the 'weakest' ins-del systems, that is, systems of size $(1,0,0;1,0,0)$, up to the level of RE. Let $ID \in \mathrm{INS}_1^{1,1}\mathrm{DEL}_1^{1,1}$. We will show that there is a system $ID_r \in \mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,1}$ capable of simulating ID. The basic idea is that the context checks, necessary for simulating rules with left and right context, are performed by the substitution rules. The system ID_r is constructed in the following manner:

Construction 4. *Let $ID = (V,T,A,I,D) \in INS_1^{1,1}DEL_1^{1,1}$ be in normal form according to Construction 1. For each rule of ID, we have a unique label, say, i, and we introduce a new nonterminal X_i. Define $ID_r = (V',T,A,I',D',S)$ with*

$$\begin{aligned}
V' &= V \cup \{X_i \mid i \text{ is the label of a rule in } I \text{ or } D\}\,,\\
I' &= \{(\lambda, X_i, \lambda) \mid i \text{ is the label of an insertion rule } (u,a,v)\}\,,\\
D' &= \{(\lambda, X_i, \lambda) \mid i \text{ labels a deletion rule } (u,a,v) \neq (\lambda, X, \lambda)\} \cup \{(\lambda, X, \lambda)\}\,,
\end{aligned}$$

where X is defined as in Construction 1. Finally, $S = S_1 \cup S_2$ with

$$\begin{aligned}
S_1 &= \{(u, X_i \to a, v) \mid i \text{ is the label of an insertion rule } (u,a,v)\} \text{ and}\\
S_2 &= \{(u, a \to X_i, v) \mid i \text{ is the label of a deletion rule } (u,a,v) \neq (\lambda, X, \lambda)\}\,.
\end{aligned}$$

The basic idea is similar to Construction 2. Each deletion rule $(u,a,v) \in D$ of ID, where i is the label of (u,a,v), corresponds to a deletion rule $(\lambda, X_i, \lambda) \in D'$ and a substitution rule $(u, a \to X_i, v) \in S_2$. We leave the context checks to the substitution rules. The same idea is applied to the insertion rules. With some technical effort, we can prove the following result.

Proposition 2. *Let $\alpha \in A$. Consider a derivation $\alpha \overset{\wedge}{\Longrightarrow}{}^* w \in T^*$. Then, there is an alternative derivation, leading from α to w, in which all nonterminals $X_i \in V' \backslash V$ are resolved immediately after being introduced.*

This property is the key to show that for a system ID_r of size $(1,0,0;1,0,0;1,1)$ constructed from a given ins-del system ID of size $(1,1,1;1,1,1)$ in normal form according to Construction 4, we find $L(ID) = L(ID_r)$. As such ins-del systems are known to be computational complete, we conclude:

Corollary 5. *$INS_1^{1,1}DEL_1^{1,1} = INS_1^{0,0}DEL_1^{0,0}SUB^{1,1} = RE$.*

We now analyze the power of ins-del-sub systems of size $(1,0,0;0,0,0;1,1)$. By definition, it is clear that $\mathrm{INS}_1^{0,0}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{1,1} \subseteq \mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,1}$ holds. In the following, we will show that this inclusion is proper. To be more precise, we will show that ins-del-sub systems of size $(1,0,0;0,0,0;1,1)$ characterize the context-sensitive languages. By Theorem 4, we are left to prove $\mathrm{CS} \subseteq \mathrm{INS}_1^{0,0}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{1,1}$.

For every context-sensitive language L, there is a linear bounded automaton (LBA) $LB = (Q, T, \Gamma, q_0, \delta, \Box, F)$ accepting L. We are going to construct an ins-del-sub system of size $(1,0,0;0,0,0;1,1)$ to simulate LB. We first give a brief sketch of the basic idea behind this simulation in the following paragraph. The simulation evolves around strings of the form

$$(u_1, \$v_1)(u_2, v_2)\dots(u_{i-1}, v_{i-1})(u_i, q_jv_i)(u_{i+1}, v_{i+1})\dots(u_{n-1}, v_{n-1})(u_n, v_n\#)$$

with $u_1, \dots, u_n \in T$; $q_j \in Q$ and $v_1, \dots, v_n \in \Gamma$. The concatenation of the first component of each tuple, that is, $u_1 \dots u_n$, is the input word of the linear bounded automaton LB, while the concatenation of the second component of each tuple, that is, $\$v_1v_2\dots v_{i-1}q_jv_iv_{i+1}\dots v_{n-1}v_n\#$, represents a configuration of LB running on the input word $u_1 \dots u_n$. The simulation of LB runs entirely on the second components of the tuples. If $\$v_1v_2\dots v_{i-1}q_jv_iv_{i+1}\dots v_{n-1}v_n\#$ is an accepting configuration, i.e., $q_i \in F$, we substitute all tuples with their respective first component. For instance (u_k, v_k) is substituted with u_k. In short, this means that if (the simulation of) LB running on $u_1 \dots u_n$ halts in an accepting configuration, we generate the word $u_1 \dots u_n$. More details follow:

Construction 5. *Consider an arbitrary LBA $LB = (Q, T, \Gamma, q_0, \delta, \Box, F)$ accepting $L \subseteq T^*$. Let \$ be the left and # be the right endmarker of LB. We define $\underline{L} := \{\lambda, \$\}$ and $\underline{R} := \{\lambda, \#\}$. Then, the ins-del-sub system $ID_r = (V \cup T, T, A, I, \emptyset, S)$ with $V = V_1 \cup V_2 \cup V_3$, where*

$$\begin{aligned}
V_1 =& \{X_0\} \cup \{X_a \mid a \in T\}\\
V_2 =& \{(a, q_ib), (a, \$q_ib), (a, q_i\$b), (a, q_ib\#), (a, bq_i\#) \mid a \in T, b \in \Gamma, q_i \in Q\}\\
&\cup \{(a, b), (a, \$b), (a, b\#) \mid a \in T, b \in \Gamma\}\\
V_3 =& \{X_{(a,b\underline{r});q_i;L} \mid a \in T, b \in \Gamma, q_i \in Q, \underline{r} \in \underline{R}\}\\
&\cup \{X_{(a,\underline{l}b);q_i;R} \mid a \in T, b \in \Gamma, q_i \in Q, \underline{l} \in \underline{L}\}\\
&\cup \{X_{(a,q_ib\underline{r}),r}, X_{(a,\underline{l}q_ib),l} \mid a \in T, b \in \Gamma, q_i \in Q, \underline{r} \in \underline{R}, \underline{l} \in \underline{L}\}
\end{aligned}$$

generates the language L by simulating LB. Strings of the form

$$(u_1, \$v_1)(u_2, v_2)\dots(u_{i-1}, v_{i-1})(u_i, q_jv_i)(u_{i+1}, v_{i+1})\dots(u_{n-1}, v_{n-1})(u_n, v_n\#)$$

consist of symbols in V_2, while the symbols in V_1 are auxiliary symbols used to generate such strings. The symbols in V_3 are used to simulate LB's transitions. We define $A = \{X_0(a, a\#) \mid a \in T\} \cup \{a \mid a \in L \cap T\}$ and $I = \{(\lambda, X_a, \lambda) \mid a \in$

$T\}$. If $\lambda \in L$, we add λ to the axiom. The set of substitution rules is defined as $S = S_{init} \cup S_N \cup S_R \cup S_L \cup S_{endmarker,L} \cup S_{endmarker,R} \cup S_{final}$. In

$$S_{init} = \{(X_0, X_a \to (a,a), \lambda) \mid a \in T\} \cup \{(\lambda, X_0 \to (a, \$q_0a), \lambda) \mid a \in T\},$$

we collect substitution rules used to initialize the simulation.

The substitution rules in the set S_N are used to simulate the application of a transition $\delta(q_i, b) \ni (q_j, c, N)$. S_N consists of substitution rules of the form

$$(\lambda, (a, q_ib) \to (a, q_jc), \lambda),\ (\lambda, (a, \$q_ib) \to (a, \$q_jc), \lambda),\ (\lambda, (a, q_ib\#) \to (a, q_jc\#), \lambda),$$

with $a \in T,\ q_i, q_j \in Q,\ b, c \in \Gamma$ and $\delta(q_i, b) \ni (q_j, c, N)$.

The substitution rules in the set S_L are used to simulate left moves. For each transition $\delta(q_i, b) \ni (q_j, c, L)$ of LB, we add substitution rules

$$(\lambda, (a, q_ib\underline{r}) \to X_{(a,c\underline{r});q_j;L}, \lambda),\ (\lambda, (d, \underline{l}e) \to X_{(d,\underline{l}q_je),l}, X_{(a,c\underline{r});q_j;L}),$$
$$(X_{(d,\underline{l}q_je),l}, X_{(a,c\underline{r});q_j;L} \to (a, c\underline{r}), \lambda),\ (\lambda, X_{(d,\underline{l}q_je),l} \to (d, \underline{l}q_je), (a, c\underline{r}))$$

to S_L with $a, d \in T,\ b, c, e \in \Gamma,\ q_i, q_j \in Q,\ \underline{l} \in \underline{L},\ \underline{r} \in \underline{R}$. Similarly, the substitution rules in S_R can simulate right moves $\delta(q_i, b) \ni (q_j, c, R)$ with:

$$(\lambda, (a, \underline{l}q_ib) \to X_{(a,\underline{l}c);q_j;R}, \lambda),\ (X_{(a,\underline{l}c);q_j;R}, (d, e\underline{r}) \to X_{(d,q_je\underline{r}),r}, \lambda)$$
$$(\lambda, X_{(a,\underline{l}c);q_j;R} \to (a, \underline{l}c), X_{(d,q_je\underline{r}),r}),\ ((a, \underline{l}c), X_{(d,q_je\underline{r}),r} \to (d, q_je\underline{r}), \lambda)\,.$$

The set $S_{endmarker,L}$ consists of substitution rules of the form

$$(\lambda, (a, \$q_ib) \to (a, q_j\$c), \lambda),\ (\lambda, (a, q_i\$b) \to (a, \$q_jb), \lambda)$$

with $a \in T,\ b, c \in \Gamma,\ q_i, q_j \in Q,\ \delta(q_i, b) \ni (q_j, c, L)$ and $\delta(q_i, \$) \ni (q_j, \$, R)$.
The set $S_{endmarker,R}$ consists of substitution rules of the form

$$(\lambda, (a, q_ib\#) \to (a, cq_j\#), \lambda),\ (\lambda, (a, bq_i\#) \to (a, q_jb\#), \lambda)$$

with $a \in T,\ b, c \in \Gamma,\ q_i, q_j \in Q,\ \delta(q_i, b) \ni (q_j, c, R)$ and $\delta(q_i, \#) \ni (q_j, \#, L)$. Both sets are used for the simulation of $\delta(q_i, b) \ni (q_j, c, L)$ and $\delta(q_i, b) \ni (q_j, c, R)$ as well, in the case the read/write head moves to/from an endmarker. The set $S_{final} = S_{f_1} \cup S_{f_2}$ is used to generate a word $w \in T^*$ if w has been accepted by the simulated linear bounded automaton LB. S_{f_1} consists of the substitutions

$$(\lambda, (a, q_fb) \to a, \lambda), (\lambda, (a, \$q_fb) \to a, \lambda), (\lambda, (a, q_f\$b) \to a, \lambda),$$
$$(\lambda, (a, q_fb\#) \to a, \lambda), (\lambda, (a, bq_f\#) \to a, \lambda)$$

and S_{f_2} consists of

$$(\lambda, (a, b) \to a, c), (c, (a, b) \to a, \lambda), (\lambda, (a, \$b) \to a, c), (c, (a, b\#) \to a, \lambda)$$

with $a, c \in T, b \in \Gamma, q_f \in F$.

Working out the correctness of this construction, we can show:

Theorem 8. $INS_1^{0,0} DEL_0^{0,0} SUB^{1,1} = CS$.

As a consequence of Theorem 8, we can formulate the following Penttonen-style normal form theorem for context-sensitive languages. We believe that this could be useful in particular when dealing with variations of insertion systems.

Theorem 9. *For every context-sensitive language L, $\lambda \notin L$, there is a context-sensitive grammar $G = (N, T, P, S)$, such that $L = L(G)$, with rules of the forms*

$$\begin{aligned} &A \to AB \\ &AB \to AC, AB \to CB \\ &A \to a, \ \textit{with } a \in T, \ A, B, C \in N. \end{aligned}$$

Allowing erasing productions on top, we also arrive at a new characterization of the family of recursively enumerable languages. By different methods, we could even prove that either of the two non-context-free forms suffices to achieve RE.

4 Summary and Main Open Questions

We have shown that the addition of substitution rules to ins-del systems yields new characterizations of RE and CS. In particular we have shown the following equalities: $\mathrm{INS}_2^{0,0}\mathrm{DEL}_2^{0,0}\mathrm{SUB}^{1,0} = \mathrm{RE}$, $\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,1} = \mathrm{RE}$ and $\mathrm{INS}_1^{0,0}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{1,1} = \mathrm{CS}$. Additionally we have shown $\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0}\mathrm{SUB}^{1,0} = \mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$. While in the above cases an extension with (non-context-free) substitution rules leads to an increase in computational power, we have also shown that the addition of context-free substitution rules to context-free ins-del systems does not affect the computational power.

The main open question is if $\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$ is computationally complete. We conjecture this not to be the case, as with only left context rules, information can only be propagated in one direction. Yet, should $\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$ equal RE, this would provide an interesting new normal form. A minor open question is the strictness of the inclusion $\mathrm{INS}_2^{0,0}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{0,1} \subseteq \mathrm{CS}$.

References

1. Alhazov, A., Krassovitskiy, A., Rogozhin, Y., Verlan, S.: Small size insertion and deletion systems. In: Martin-Vide, C. (ed.) Applications of Language Methods, pp. 459–515. Imperial College Press (2010)
2. Beaver, D.: Computing with DNA. J. Comput. Biol. **2**(1), 1–7 (1995)
3. Kari, L.: On insertions and deletions in formal languages. Ph.D. thesis, University of Turku, Finland (1991)
4. Karl, L.: DNA computing: arrival of biological mathematics. Math. Intell. **19**(2), 9–22 (1997). https://doi.org/10.1007/BF03024425
5. Krassovitskiy, A., Rogozhin, Y., Verlan, S.: Further results on insertion-deletion systems with one-sided contexts. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 333–344. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88282-4_31

6. Krassovitskiy, A., Rogozhin, Y., Verlan, S.: Computational power of insertion-deletion (P) systems with rules of size two. Nat. Comput. **10**, 835–852 (2011). https://doi.org/10.1007/s11047-010-9208-y
7. Margenstern, M., Păun, G., Rogozhin, Y., Verlan, S.: Context-free insertion-deletion systems. Theor. Comput. Sci. **330**(2), 339–348 (2005)
8. Matveevici, A., Rogozhin, Y., Verlan, S.: Insertion-deletion systems with one-sided contexts. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 205–217. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74593-8_18
9. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer, Heidelberg (1998). https://doi.org/10.1007/978-3-662-03563-4
10. Takahara, A., Yokomori, T.: On the computational power of insertion-deletion systems. Nat. Comput. **2**(4), 321–336 (2003). https://doi.org/10.1023/B:NACO.0000006769.27984.23
11. Verlan, S.: On minimal context-free insertion-deletion systems. J. Autom. Lang. Comb. **12**(1–2), 317–328 (2007)
12. Verlan, S.: Recent developments on insertion-deletion systems. Comput. Sci. J. Moldova **18**(2), 210–245 (2010)
13. Vu, M.: On insertion-deletion systems with substitution rules. Master's thesis, Informatikwissenschaften, Universität Trier, Germany (2019)