algorithms

MDPI

Article

Adding Matrix Control: Insertion-Deletion Systems with Substitutions III

Martin Vu [1,*] and Henning Fernau [2,*]

[1] Fachbereich 3, Universität Bremen, 28359 Bremen, Germany
[2] Fachbereich 4, Universität Trier, 54296 Trier, Germany
* Correspondence: martin.vu@uni-bremen.de (M.V.); fernau@uni-trier.de (H.F.)

Abstract: Insertion-deletion systems have been introduced as a formalism to model operations that find their counterparts in ideas of bio-computing, more specifically, when using DNA or RNA strings and biological mechanisms that work on these strings. So-called matrix control has been introduced to insertion-deletion systems in order to enable writing short program fragments. We discuss substitutions as a further type of operation, added to matrix insertion-deletion systems. For such systems, we additionally discuss the effect of *appearance checking*. This way, we obtain new characterizations of the family of context-sensitive and the family of recursively enumerable languages. Not much context is needed for systems with appearance checking to reach computational completeness. This also suggests that bio-computers may run rather traditionally written programs, as our simulations also show how Turing machines, like any other computational device, can be simulated by certain matrix insertion-deletion-substitution systems.

Keywords: computational completeness; matrix control; insertions; deletions; substitutions

Citation: Vu, M.; Fernau, H. Adding Matrix Control: Insertion-Deletion Systems with Substitutions III. *Algorithms* **2021**, *14*, 131. https://doi.org/10.3390/a14050131

Academic Editors: Frank Werner, Riccardo Dondi and Florian Sikora

Received: 31 March 2021
Accepted: 19 April 2021
Published: 22 April 2021

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 Lila Kari [1]. Corresponding to the mismatched annealing of DNA sequences, both the insertion and deletion operations have a strong biological background, which led to their study in the molecular computing framework (cf. [2]). Insertion rules add a substring to a string, given a specified left and right context, while deletion rules remove a substring from a string, again taking a specified left and right context into consideration. The replacement of single letters (possibly within some context) by other letters by an operation, called *substitution*, is discussed in [3,4], again from a bio-computing background.Interestingly, all of the theoretical studies on grammatical mechanisms involving insertions and deletions (except [5]) omitted, including the substitution operation in their studies. In [6–8], we started out a project to formally and systematically study insertion-deletion systems with substitutions as an additional operation, leading to ins-del-sub systems (for short).

It can be argued that the potentially most error-prone part of a bio-computing implementation of ins-del-sub systems are context checks concerning the site where the operation (be it an insertion, a deletion or a substitution operation) may be applied. Therefore, it is interesting to study how much context dependency is necessary for achieving computational completeness, given the ability to add a substring of length n and to delete a substring of length p.

Conversely, assuming that one can prove that ins-del-sub systems with limited context checks exist that can simulate arbitrary Turing machines (or phrase structure grammars, or any arbitrary computational mechanism that can be used to define terms like *computability*), then this means that there is some hope that some day one could build computers that are no longer silicon-based, but that compute with RNA, DNA, or protein structures.

It should be noted that all of our computational completeness proofs are constructive, which means that algorithms exist that will finally turn any program written in some high-level programming language of your choice into an ins-del-sub system that executes this program based on insertion, deletion, or substitution operations.

As the following definitions will show, the 'derivation relation', although formally defined as a sequential series of insertion, deletion, or substitution operations, can be easily seen to allow an inherent parallelism, as all operations that are 'far enough' from each other can be executed in parallel. Because the potentially high degree of parallelism in bio-computing is argued as one of the main attractive features of this form of computation, this gives another reason as to why one should strive for operations that are as context-independent as possible, as context dependencies could be seen as semaphore-like synchronization points. Namely, on the level of bio-computing, checking certain strings means building up some structures (mostly proteins) that read and check 'matching structures' by forming chemical links between molecules. This means that 'checking from one side' prevents and, hence, rules out 'checking from the other side' in parallel.

Ins-del systems can be extended with some form of control to further reduce their context dependency. Matrix insertion-deletion systems, or matrix ins-del systems for short, were introduced in [9,10]. These systems group insertion and deletion rules in sequences, called *matrices*; either the whole sequence of operations is applied consecutively, or no rule is applied at all, thus resembling traditional *matrix grammars*, originally introduced with a linguistic motivation [11]. From the perspective of bio-computing, the matrices correspond to small program fragments without jumps that are easier to implement than longer and more involved ones. Allowing such program fragments should also make a better fit for finally implementing compilers that produce executable 'bio-code' from traditional high-level programming languages, as the sequential execution of commands is one of the cornerstones of basically any of the programming languages of today. There is certainly a certain trade-off between the potentially high degree of parallelism and these new sequential program fragments. This is one of the motivations to limit (on top of context lengths) the length of these program fragments (i.e., technical speaking the length of the matrices).

Additionally, we discuss *appearance checking* in the context of matrix ins-del-sub systems. In the case of matrix grammars, it is known that allowing certain rules of a matrix to be skipped if not applicable increases the computational power [12]. In this paper, we investigate the effect of appearance checking on matrix ins-del-sub systems. We show that the context dependency of ins-del systems can be greatly reduced if matrices, appearance checking, and substitution rules are allowed. For instance, it is shown that a matrix ins-del-sub system that only allows context-free single letter insertions and two-letter deletions in addition to context-free substitution is sufficient for generating any recursively enumerable language. In addition, we show that a 'normal form' for matrix ins-del-sub systems exists, in which only matrices of size at most 2 occur. On the downside, it can be argued that appearance checks is a particularly expensive feature when it comes to implementing it in bio-computing devices, as many sites of potential rule applications have to be checked before being able to execute the next command in the matrix. However, one clearly sees a trade-off in our results between the necessity to have larger contexts and the necessity to have appearance checks. Because it is not clear which of these mechanisms is really harder to implement when it comes to build real bio-computers, it appears to be reasonable to study the general possibilities of these mechanisms, hence paving the way to future generations of new computing devices.

2. Definitions

We assume the reader to be familiar with the standard notations in formal language theory. By λ we denote the empty string. Let w be an arbitrary string. We denote by w^R the *reversal* or *mirror image* of w. By L^R and $\mathcal{L}^R$, we denote the *reversal* of a language L and a language family $\mathcal{L}$, respectively. We denote, by RE, CS, CF, and REG, the families of

recursively enumerable, context-sensitive, context-free, and regular languages, respectively. We are interested in computational completeness results, i.e., in describing RE with matrix ins-del-sub systems with little resources as formally explained next.

2.1. Matrix Grammars

A *matrix grammar* is a tuple $G = (N, T, M_G, S)$ where N, T and S are the finite set of nonterminals, the finite set of terminals and the start symbol, respectively. M_G is a finite set of sequences of the form $m = [r_1, r_2, \ldots, r_n]$, $n \geq 1$, with rewriting rules $r_i = \alpha_i \to \beta_i$ with $\alpha_i \in (N \cup T)^* N (N \cup T)^*$ and $\beta_i \in (N \cup T)^*$. Such a sequence m is called a matrix [12]. The relation $\Rightarrow$ induced by G is defined, as follows. For words $w_1, w_2 \in (N \cup T)^*$, $w_1 \Rightarrow w_2$ holds if a matrix $m = [r_1, r_2, \ldots, r_n]$ and $w'_0, \ldots, w'_n \in (N \cup T)^*$ with $w'_0 = w_1$ and $w'_n = w_2$ exist, such that $w'_j \Rightarrow_{r_{j+1}} w'_{j+1}$ holds for all $0 \leq j < n$. The language that is generated by G is $\mathcal{L}(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. We denote, by $\mathcal{L}(\mathrm{M}, \mathrm{CF})$, the language family that is generated by matrix grammars with context-free rewriting rules [12]. A *matrix grammar with appearance checking* is a tuple $G_{ac} = (N, T, M_G, S, F)$, where N, T, M_G, and S are defined as in usual matrix grammars. F is a set of rewriting rules occurring in matrices of M_G. All of the rules in F may be skipped in a transition of G_{ac}, if not applicable. Thus, the absence of symbols can be checked.

2.2. Insertion-Deletion Systems

An *insertion-deletion system* (*ins-del system* for short) is a five-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 $u, v \in V^*$ and $a \in V^+$. We call elements occurring in T *terminal* symbols, while referring to elements of $V \setminus T$ as *nonterminals*. Elements of A are called *axioms*.

Let $w_1 uvw_2$ and $w_1 uavw_2$, with $w_1, u, v, w_2 \in V^*$, $a \in V^+$, be strings. The application of an insertion rule $(u, a, v) \in I$ (also written $(u, a, v)_{\mathrm{ins}}$) to $w_1 uvw_2$ corresponds to inserting the string $a \in V^*$ between u and v, which results in the string $w_1 uavw_2$. The application of a deletion rule $(u, a, v) \in D$ (also written $(u, a, v)_{\mathrm{del}}$) to $w_1 uavw_2$ results in the removal of a substring a from the context (u, v), which results in the string $w_1 uvw_2$. The relation $\Longrightarrow$ is defined, as follows: Let $x, y \in V^*$. Afterwards, we write $x \Longrightarrow y$ iff y is the result of applying an insertion or deletion rule to x. We write $\Longrightarrow_{\mathrm{ins}} / \Longrightarrow_{\mathrm{del}}$ if y is obtained via an insertion/ a deletion rule. We denote, by $\Longrightarrow^+$ and $\Longrightarrow^*$, the transitive and the reflexive and transitive closure, respectively. The language that is generated by ID is defined by $L(ID) = \{w \in T^* \mid \exists \alpha \in A : \alpha \Longrightarrow^* w\}$. Consider $(u, a, v)_{\mathrm{ins}}$ or $(u, a, v)_{\mathrm{del}}$. We refer to u as the left context and v as the right context of $(u, a, v)_{\mathrm{ins}} / (u, a, v)_{\mathrm{del}}$. A *sentential form* of ID is a string over V. The *size* of ID describes its complexity and it is defined by a vector $(n, m, m'; p, q, q')$, where $n = \max\{|a| \mid (u, a, v) \in I\}$, $p = \max\{|a| \mid (u, a, v) \in D\}$, $m = \max\{|u| \mid (u, a, v) \in I\}$, $q = \max\{|u| \mid (u, a, v) \in D\}$, $m' = \max\{|v| \mid (u, a, v) \in I\}$ and $q' = \max\{|v| \mid (u, a, v) \in D\}$.

By $\mathrm{INS}_n^{m,m'} \mathrm{DEL}_p^{q,q'}$, we denote the family of all insertion-deletion systems of size $(n, m, m'; p, q, q')$ [13,14]. Depending on the context, we also denote the family of languages that can be generated by insertion-deletion systems of size $(n, m, m'; p, q, q')$ by $\mathrm{INS}_n^{m,m'} \mathrm{DEL}_p^{q,q'}$.

We first study a concrete example now to clarify the definitions and also to return to some of the general discussions of the introduction.

Example 1. *Consider the following ins-del system $ID = (V, T, A, I, D)$ where $V = T = \{a, b, c\}$, $A = \{acb\}$, $I = \{(a, a, c), (c, b, b)\}$, and $D = \{(\lambda, c, \lambda)\}$. Clearly this system is of size $(1, 1, 1; 1, 0, 0)$ and the generated language is $L(ID) = a^+ b^+ \cup a^+ c b^+$.*

As mentioned in the introduction, ins-del systems offer a high degree of parallelism. The system ID, for instance, exhibits this trait when considering the rules (a, a, c) and (c, b, b). It is easy to see that the order in which these rules are applied is insignificant and, thus, these rules do

not affect each other. Hence, (a, a, c) *and* (c, b, b) *are "far enough" from each other to be applied in parallel without affecting the computation. However, note that the deletion rule* (λ, c, λ) *cannot be applied in parallel with any insertion rule as the order does matter in this case. More precisely, the deletion* (λ, c, λ) *cannot be applied before applying an insertion, as it removes necessary context information.*

Finally, observe that the amount of parallelism that is observable in generating the language $a^+b^+ \cup a^+cb^+$ *can be further significantly increased by adding the insertion rules* (a, a, a) *and* (b, b, b).

2.3. Combining Ideas: Matrix Insertion-Deletion Systems

The idea of regulating ins-del systems with matrix control goes back to [10,15]. A *matrix ins-del system* [10] is a construct $MID = (V, T, A, M)$ where V, T and A are defined as in usual ins-del systems. $M = \{m_1, \ldots, m_t\}$, $t \geq 1$, is a finite set of sequences, called matrices, of the form $m_i = [r_{i,1}, \ldots, r_{i,k_i}]$, where $k_i \geq 1$. $r_{i,j}$, with $1 \leq i \leq t$, $1 \leq j \leq k_i$, is either an insertion or a deletion rule. A sentential form of *MID* is a string $w \in V^*$. Consider a matrix $m_i = [r_{i,1}, \ldots, r_{i,k_i}]$. A transition $w \Longrightarrow_{m_i} w'$ is performed if there exist strings $w_1, \ldots, w_{k_i+1} \in V^*$ such that $w_j \Longrightarrow_{r_{i,j}} w_{j+1}$ with $w_1 = w$ and $w_{k_i+1} = w'$. Let $\Longrightarrow := \bigcup_{m \in M} \Longrightarrow_m$. The language that is generated by *MID* is defined as

$$L(MID) = \{w \in T^* \mid \exists \alpha \in A : \alpha \Longrightarrow^* w\}.$$

We say that *MID* has *matrices of size k* if $k = \max_{1 \leq i \leq t} k_i$. If *MID* is an ins-del system of size $(n, m, m'; p, q, q')$ with matrices of size k , we also say that *MID* is of size $(k; n, m, m'; p, q, q')$. We denote by $\mathrm{MAT}_k\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}$ either the family of languages that are generated by ins-del systems of size $(k; n, m, m'; p, q, q')$ or the family of ins-del systems of size $(k; n, m, m'; p, q, q')$, depending on the context. Denote, by $\mathrm{MAT}_*\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}\mathrm{SUB}^{r,r'}$, the family of matrix ins-del-sub systems with matrices of arbitrary size and insertion rules and deletion rules, of size (n, m, m') and (p, q, q'), respectively. The following matrix ins-del systems are known to describe RE:

- $\mathrm{MAT}_3\mathrm{INS}_1^{1,1}\mathrm{DEL}_1^{0,0}$ and $\mathrm{MAT}_3\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{1,1}$ [16]
- $\mathrm{MAT}_3\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{2,0}$, $\mathrm{MAT}_3\mathrm{INS}_1^{2,0}\mathrm{DEL}_1^{0,0}$ and $\mathrm{MAT}_2\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0}$ [17]
- $\mathrm{MAT}_3\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{0,1}$, $\mathrm{MAT}_2\mathrm{INS}_1^{1,0}\mathrm{DEL}_2^{0,0}$ and $\mathrm{MAT}_2\mathrm{INS}_2^{0,0}\mathrm{DEL}_1^{1,0}$ [10].

The incompleteness results for matrix ins-del sub systems include$\mathrm{MAT}_*\mathrm{INS}_2^{0,0}\mathrm{DEL}_2^{0,0}$, for instance. More precisely, the following theorem holds.

Theorem 1. *$REG \setminus MAT_*INS_2^{0,0}DEL_2^{0,0} \neq \emptyset$, testified by a^*b.*

This result follows from [18], as stated as in [10]. For reasons of completeness, we give a formal proof of this result in the following.

Proof. Before we begin with our proof, we introduce 'markings' [14,19] in the following paragraph. We explain the details of the marking approach with the following example. Consider the derivation

$$\lambda \Longrightarrow_{\mathrm{ins}} aA \Longrightarrow_{\mathrm{ins}} aAaA \Longrightarrow_{\mathrm{ins}} abCAaA \Longrightarrow_{\mathrm{ins}} abDECAaA$$
$$\Longrightarrow_{\mathrm{del}} abDAaA \Longrightarrow_{\mathrm{del}} abaA \Longrightarrow_{\mathrm{ins}} abaAbc \Longrightarrow_{\mathrm{del}} abac$$

of an ins-del system of size $(2, 0, 0; 2, 0, 0)$. Then we introduce 'markings', as follows: Two symbols that have been introduced together are joined with an overline, while two symbols that are deleted together are joined with an underline. The overlines and underlines mark the insertion pairs and the deletion pairs, respectively. The marking to a word *abac*, which is derived in the manner presented above, is shown in Figure 1.

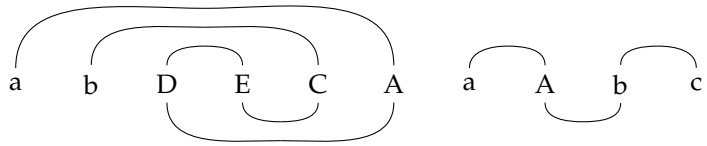


Figure 1. Marking corresponding to a derivation of *abac* [14,19].

Interpreting all of the symbols as labelled nodes and all lines as edges, it is easy to see that any word w that is generated by an arbitrary ins-del system of size $(2,0,0;2,0,0)$ corresponds to a graph, which consists of disjoint paths and/or cycles. If a symbol A is deleted by a deletion rule (λ, A, λ), then we replace the corresponding node in the graph with $\not{A}$. The node $\not{A}$ is interpreted as λ. Additionally, the application of an insertion rule (λ, A, λ) corresponds to adding a path, which consists of a single edge with nodes λ and A, to the graph at the corresponding position. We assume that all letters of the axiom have been introduced by such insertion rules. There is clearly no interaction between two symbols of two different paths/cycles or between a symbol of a cycle and a symbol of a path. We remark that all of the symbols/nodes, which have a degree of two, are deleted and hence only symbols with a degree of one contribute to the final word w. Note that all of the symbols of degree one only have an 'overline' edge, while nodes of degree two have both types of edges. Therefore, it is clear that at most two symbols of a path can contribute to w. We refer to [14,19] for more details on the 'marking' approach. Clearly, the marking approach can be applied to matrix-ins-del systems of size $(*;2,0,0;2,0,0)$, as well.

We now show that the language a^*b cannot be generated by any matrix ins-del systems of size $(*;2,0,0;2,0,0)$ by contradiction. Assume that there is a matrix insertion-deletion system *MID*, which generates the language a^*b. Subsequently, clearly the word a^mb, where $m > 2n+1$ and n is the length of the longest axiom, can also be generated. Consider the graph that corresponds to a^mb. It is clear that this graph consist of more than $n+1$ paths. Then there is at least one path which does not involve any symbols of the axiom and contributes at least one letter a (and no b) to a^mb. Because this path does not involve any letters of the axiom, all symbols of this path have been introduced with context-free insertion rules. We denote this path as P. Consider the derivation of a^mb. Because there is no interaction between two symbols of different paths and /or cycles, it is clear that there is a derivation that applies all of the matrices used in the derivation of a^mb in the same order, but in which all insertions and deletions corresponding to the path P occur right of the final b. This, in turn, means that *ID* also generates a word in a^*ba^+. □

2.4. Adding Substitutions

With substitution rules, we now introduce the central notion of this paper. 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. Afterwards, 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, which results in the string w_1ubvw_2.

Formally, we define an *insertion-deletion-substitution system*, or *ins-del-sub system* for short, to be specified by a six-tuple $ID_\varsigma = (V, T, A, I, D, S)$, 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.

Let $x = w_1uavw_2$ and $y = w_1ubvw_2$ be the strings over V. The substitution rules define a relation $\Longrightarrow_{\text{sub}}$, as follows: $x \Longrightarrow_{\text{sub}} y$ if 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 $\hat{\Longrightarrow}^*$ and $\hat{\Longrightarrow}^+$ as usual, denoting the reflexive-transitive and transitive closure of $\hat{\Longrightarrow}$, respectively.

The language that is generated by an ins-del-sub system ID_ς is defined as

$$L(ID_\varsigma) = \{w \in T^* \mid \alpha \hat{\Longrightarrow}^* w,\ \alpha \in A\}.$$

As with usual ins-del system, we measure the complexity of an ins-del-sub system $ID_\varsigma = (V, T, A, I, D, S)$ via its *size*, which is, an eight-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 and 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 at 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 $\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}\mathrm{SUB}^{r,r'}$.

As with ins-del systems, ins-del-sub systems can be regulated with matrix control introducing *matrix ins-del-sub systems*. A *matrix ins-del-sub system* is a construct $MID_\varsigma = (V, T, A, M_\varsigma)$, where V, T and A are defined as in usual ins-del systems. $M_\varsigma = \{m_1, \ldots, m_t\}$, $t \geq 1$, is a finite set of sequences, called matrices, of the form $m_i = [r_{i,1}, \ldots, r_{i,k_i}]$, where $k_i \geq 1$. $r_{i,j}$, with $1 \leq i \leq t$, $1 \leq j \leq k_i$, is either an insertion, a deletion or a substitution rule.

We define the relation between the strings w and w' over V $w \hat{\Longrightarrow}_{m_i} w'$, as well as the generated language of MID_ς, analogously to the case without substitution rules. We say that matrix ins-del-sub systems, which have insertion rules, deletion rules, substitution rules, and matrices of size (n, m, m'),(p, q, q'), (r, r'), and k, respectively, is of size $(k; n, m, m'; p, q, q'; r, r')$. By $\mathrm{MAT}_k\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}\mathrm{SUB}^{r,r'}$, denote the family of matrix ins-del-sub systems of size $(k; n, m, m'; p, q, q'; r, r')$, as well as the family of languages generated by such systems, depending on the context. Consider a language family $\mathrm{MAT}_k\mathrm{INS}_n^{m,m'}\mathrm{DEL}_p^{q,q'}\mathrm{SUB}^{r,r'}$. Concerning the reversal operator (that reads words from right to left), the following lemma holds.

Lemma 1. *Let $\mathcal{L}$ be a family of languages that is closed under reversal. Then:*

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

Proof. These claims follow analogously to [17]. □

By definition, it is also clear that the following lemma holds.

Lemma 2. $MAT_kINS_n^{m,m'}DEL_p^{q,q'} \subseteq MAT_kINS_n^{m,m'}DEL_p^{q,q'}SUB^{r,r'}$.

2.5. Appearance Checking: An Additional Feature

It is known that matrix grammars with context-free production are not computationally complete, but they can reach computational completeness if used with *appearance checking*. Transferring this idea to matrix ins-del-sub systems, we introduce matrix ins-del-sub systems with appearance checking and show that, similar to the matrix grammar case, formerly computationally incomplete matrix ins-del-sub systems can reach computational completeness if they are used in conjunction with appearance checking. We begin by defining matrix ins-del-sub systems and appearance checking. A matrix ins-del-sub systems and appearance checking is a tuple $MID_{\varsigma,ac} = (V, T, A, M_\varsigma, F)$, where V, T, A, and M_ς are defined as in usual matrix ins-del-sub systems. F is a subset of all rules occurring in M_ς.

Let $w_1, w_2 \in V^*$ and z be an arbitrary rule occurring in M_ς. We define the relation $\hat{\Longrightarrow}_z^{ac}$, as follows: $w_1 \hat{\Longrightarrow}_z^{ac} w_2$ if one of the following conditions hold: (a) the rule z is applicable to w_1, such that $w_1 \hat{\Longrightarrow}_z w_2$ or (b) the rule z is not applicable to w_1, $z \in F$ and $w_1 = w_2$. Basically, this means that, if some rule in F is not applicable, we can skip that rule. Let $m = [r_1, \ldots, r_n] \in M_\varsigma$ and $x, y \in V^*$. Then $x \hat{\Longrightarrow}_m^{ac} y$ iff

$x = w_0 \hat{\Longrightarrow}^{ac}_{r_1} w_1 \hat{\Longrightarrow}^{ac}_{r_2} \dots \hat{\Longrightarrow}^{ac}_{r_n} w_n = y$. The language that is generated by $MID_{\varsigma,ac}$ is defined as

$$L(MID_{\varsigma,ac}) = \{w \in T^* \mid \exists\, m_1, \dots, m_n \in M_\varsigma, \alpha \in A : \alpha \hat{\Longrightarrow}^{ac}_{m_1} \dots \hat{\Longrightarrow}^{ac}_{m_n} w\}$$

We define the *size* of $MID_{\varsigma,ac}$ analogously to matrix ins-del-sub systems without appearance checking. We denote the language family that is generated by matrix ins-del-sub systems with appearance checking of size $(k; n, m, m'; p, q, q'; r, r')$ by the term $\mathrm{MAT}^{ac}_k \mathrm{INS}^{m,m'}_n \mathrm{DEL}^{q,q'}_p \mathrm{SUB}^{r,r'}$.

Again, we like to clarify these definitions by presenting a concrete example of a matrix ins-del-sub system that makes use of appearance checking.

Example 2. *Consider the matrix ins-del-sub systems with appearance checking* $MID_{\varsigma,ac} = (V, T, A, M_\varsigma, F)$ *of size* $(3; 1, 0, 0; 2, 0, 0; 0, 0)$ *with* $V = \{X, a, b\}$, $T = \{a, b\}$, $A = \{b\}$, $M_\varsigma = \{[(\lambda, X, \lambda)_{ins}, (\lambda, bX, \lambda)_{del}, (\lambda, X \to a, \lambda)]\}$ *and* $F = \{(\lambda, bX, \lambda)_{del}\}$. *The language that is generated by* $MID_{\varsigma,ac}$ *is* a^*b. *This can be shown, as follows.*

Let $m = [(\lambda, X, \lambda)_{ins}, (\lambda, bX, \lambda)_{del}, (\lambda, X \to a, \lambda)]$, $w_1 \in a^*b$ *and* $w_2 \in \{a, b\}^*$, *such that* $w_1 \hat{\Longrightarrow}^{ac}_m w_2$.

Assume that during the application of m the nonterminal X is inserted to the right of b. Subsequently, clearly the deletion rule (λ, bX, λ) *is applicable after the insertion and we delete X along with b. However, now the substitution rule* $(\lambda, X \to a, \lambda)$ *cannot be applied anymore and, as* $(\lambda, X \to a, \lambda) \notin F$, *this means that the matrix as a whole cannot be applied.*

Assume that, during the application of m, the nonterminal X is inserted somewhere left of b. Clearly, $(\lambda, bX, \lambda)_{del}$ *is not applicable and, as* $(\lambda, bX, \lambda)_{del} \in F$, *we skip this rule and proceed with the application of* $(\lambda, X \to a, \lambda)$. *Applying m in this way effectively inserts the letter a somewhere left of b and, therefore,* $w_2 \in a^*b$.

Using the argument above inductively yields our claim. We remark that, for any word w derived from the axiom b, $|w|_{\{X\}} = 0$ *holds, as any X introduced during the application of m is resolved at the end of m.*

The example above clarifies that appearance checking can result in an increase in computational power, as $a^*b \notin \mathrm{MAT}_* \mathrm{INS}^{0,0}_2 \mathrm{DEL}^{0,0}_2 \mathrm{SUB}^{0,0}$ is known (Theorem 1).

3. Computational (In-)Completeness Results

In this section, we present the main results of our research.

3.1. A Normal Form Theorem

We begin by introducing a (binary) normal form for matrix ins-del-sub systems that are similar to the two-normal form (also known as binary normal form) for matrix grammars ([12] Def. 1.2.1). Recall our discussion in the introductory section concerning possible applications of matrix ins-del-sub systems: there, we argued that short matrices offer advantages, as they help allow for parallel execution in this type of computational devices. However, when looking at the proof of the next theorem, one sees that it enforces some sequentialization by introducing a shared resource in the form of specific nonterminals. Therefore, in essence, the question of parallelizability within computational devices, like (matrix) ins-del-sub systems, remains an interesting topic of future research.

A matrix ins-del-sub systems $MID_\varsigma = (V, T, A, M_\varsigma)$ is said to be in *normal form* if all the matrices are either of the form $[r, (\lambda, A \to B, \lambda)]$ or $[r, (\lambda, A, \lambda)_{\mathrm{del}}]$, where r is some insertion, deletion, or substitution rule and $A, B \in V$. Clearly, all matrix ins-del-sub systems in normal form are included in $\mathrm{MAT}_2 \mathrm{INS}^{m,m'}_n \mathrm{DEL}^{q,q'}_p \mathrm{SUB}^{r,r'}$. We show that, for every matrix ins-del-sub system, there is a matrix ins-del-sub system in normal form that generates the same language.

Theorem 2. *For every* $MID_\varsigma \in MAT_k INS_n^{m,m'} DEL_p^{q,q'} SUB^{r,r'}$, $p > 0$, *there is a system* $MID'_\varsigma \in MAT_2 INS_n^{m,m'} DEL_p^{q,q'} SUB^{r,r'}$, *such that* $L(MID_\varsigma) = L(MID'_\varsigma)$.

Proof. Let $MID_\varsigma = (V, T, A, M_\varsigma)$ and all matrices of MID_ς be labelled in a one-to-one manner, i.e., a bijection from M_ς to a set of labels exists. Subsequently, we define $MID'_\varsigma = (V', T, A', M'_\varsigma)$, where $V' = V \cup \{(i,j) \mid i$ is the label of a matrix of MID_ς and $j \leq k\}$ and $A' = \{\alpha(i,1) \mid \alpha \in A$ and i is the label of a matrix of $MID_\varsigma\} \cup \{\alpha \mid \alpha \in A \cap T^*\}$. Without a loss of generality we assume $V \cap \{(i,j) \mid i$ is the labelof a matrix of MID_ς and $j \leq k\} = \emptyset$. For every matrix $m = [r_1, r_2, \ldots, r_n]$ of MID_ς, where r_j, with $j = 1, \ldots, n$, is some insertion, deletion, or substitution rule and i is the label of m, we add the following matrices

$$
\begin{aligned}
&[r_1, (\lambda, (i,1) \rightarrow (i,2), \lambda)],\\
&[r_2, (\lambda, (i,2) \rightarrow (i,3), \lambda)],\\
&\ldots,\\
&[r_{n-1}, (\lambda, (i,n-1) \rightarrow (i,n), \lambda)] \text{ and}\\
&[r_n, (\lambda, (i,n), \lambda)_{\mathrm{del}}]
\end{aligned}
$$

to MID'_ς. For every label i' of some matrix of MID_ς and every matrix $m = [r_1, r_2, \ldots, r_n]$ of MID_ς, we add a matrix $[r_n, (\lambda, (i,n) \rightarrow (i',1), \lambda)]$ to MID'_ς. By definition, the second component of every matrix of MID'_ς is either a context-free deletion rule of the form $(\lambda, (i,j), \lambda)_{\mathrm{del}}$ or a context-free substitution rule of the form $(\lambda, (i,j) \rightarrow (i',j'), \lambda)$, where i and i' are the labels of some matrices of MID_ς and $j, j' \leq k$. Furthermore, the first rule of any matrix of MID'_ς does not involve any nonterminals in $V' \setminus V$.

Consider a sentential form $w_1(i,j)w_2$ with $w_1, w_2 \in V^*$. It can be shown that all the sentential forms of MID'_ς are either of this form or of the form $w \in V^*$. The basic idea is that the nonterminal (i,j) serves as an indicator where the matrix is to be applied next. For instance, the occurrence of (i,j) in $w_1(i,j)w_2$ signifies that the next rule to be applied is either $(r_j, (\lambda, (i,j) \rightarrow (i,j+1), \lambda))$ if the length of the matrix of MID_ς labelled by i is greater than j or $(r_j, (\lambda, (i,j) \rightarrow (i',1), \lambda))$ / $(r_j, (\lambda, (i,j), \lambda)_{\mathrm{del}})$, otherwise. We note that, in every derivation of MID'_ς, a matrix of the form $(r_j, (\lambda, (i,j), \lambda)_{\mathrm{del}})$ is applied at most once. Furthermore, we remark that, if a sentential form $w \in V^*$ occurs during a derivation of MID'_ς, then the derivation cannot proceed as the second rule of any matrix cannot be applied. We now prove the correctness of the construction, i.e., $L(MID_\varsigma) = L(MID'_\varsigma)$, by showing both inclusion directions separately.

'$\supseteq$': Consider the following derivation $w_1(i,j)w_2 \hat{\Longrightarrow}_{[r,(\lambda,(i,j)\rightarrow(i',j'),\lambda)]} w'_1(i',j')w'_2$, where r is some insertion, deletion, or substitution rule and $w_1, w_2 \in V^*$. because r does not involve any nonterminals in $V' \setminus V$, clearly $w_1w_2 \hat{\Longrightarrow}_r w'_1w'_2$ holds. We now extend this result. Consider a matrix $m = [r_1, \ldots, r_n]$ labelled by i. Subsequently, clearly

$$
w_1(i,1)w_2 \underset{[r_1,(\lambda,(i,1)\rightarrow(i,2),\lambda)]}{\hat{\Longrightarrow}} \cdots \underset{[r_{n-1},(\lambda,(i,n-1)\rightarrow(i,n),\lambda)]}{\hat{\Longrightarrow}} w'_1(i,n)w'_2 \underset{[r_n,(\lambda,(i,n)\rightarrow(i',1),\lambda)]}{\hat{\Longrightarrow}} w''_1(i',1)w''_2
$$

or

$$
w_1(i,1)w_2 \underset{[r_1,(\lambda,(i,1)\rightarrow(i,2),\lambda)]}{\hat{\Longrightarrow}} \cdots \underset{[r_{n-1},(\lambda,(i,n-1)\rightarrow(i,n),\lambda)]}{\hat{\Longrightarrow}} w'_1(i,n)w'_2 \qquad \underset{[r_n,(\lambda,(i,n),\lambda)_{\mathrm{del}}]}{\hat{\Longrightarrow}} w''_1w''_2
$$

implies $w_1w_2 \hat{\Longrightarrow}_{[r_1,\ldots,r_n]} w''_1w''_2$.

'$\subseteq$': Conversely, it can be shown that $w \hat{\Longrightarrow}_{[r_1,\ldots,r_n]} w''$ implies

$$
w(i,1) \underset{[r_1,(\lambda,(i,1)\rightarrow(i,2),\lambda)]}{\hat{\Longrightarrow}} \cdots \underset{[r_{n-1},(\lambda,(i,n-1)\rightarrow(i,n),\lambda)]}{\hat{\Longrightarrow}} w'(i,n) \qquad \underset{[r_n,(\lambda,(i,n)\rightarrow(i',1),\lambda)]}{\hat{\Longrightarrow}} w''(i',1)
$$

and

$$w(i,1) \underset{[r_1,(\lambda,(i,1)\to(i,2),\lambda)]}{\hat{\Longrightarrow}} \cdots \underset{[r_{n-1},(\lambda,(i,n-1)\to(i,n),\lambda)]}{\hat{\Longrightarrow}} w'(i,n) \underset{[r_n,(\lambda,(i,n),\lambda)_{\mathrm{del}}]}{\hat{\Longrightarrow}} w''$$

We remark that, in the simulation of the application a matrix of MID_ς, we can assume that the leftmost symbol of any sentential form of a derivation of MID'_ς is of a nonterminal $(i,j) \in V' \setminus V$ (unless a matrix of the form $[r_n, (\lambda,(i,n),\lambda)_{\mathrm{del}}]$ is applied). □

Similarly, for every matrix ins-del-sub system $MID_\varsigma \in \mathrm{MAT}_k\mathrm{INS}_n^{m,m'}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{r,r'}$, one can construct $MID'_\varsigma \in \mathrm{MAT}_2\mathrm{INS}_n^{m,m'}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{r,r'}$ in normal form, such that $L(MID'_\varsigma) = L(MID_\varsigma)$.

Theorem 3. *Let $MID_\varsigma \in MAT_kINS_n^{m,m'}DEL_0^{0,0}SUB^{r,r'}$. Afterwards, it is possible to construct a system $MID'_\varsigma \in MAT_2INS_n^{m,m'}DEL_0^{0,0}SUB^{r,r'}$ in normal form, such that $L(MID_\varsigma) = L(MID'_\varsigma)$.*

Proof. Let $MID_\varsigma = (V,T,A,M_\varsigma)$ and all matrices of MID_ς be labelled in a one-to-one manner, i.e., a bijection M_ς to a set of labels exists. We define the set of symbols as

$$V' = V \cup \{a_{i,j} \mid a \in V,\ i \text{ is the label of a matrix of } MID_\varsigma \text{ and } j \le k\},$$

where k denotes the maximal length of a matrix in M_ς. We now describe how to construct an equivalent matrix ins-del-sub system MID'_ς of the same size, such that MID'_ς is in normal form.

For every $v \in V$, every label i' of a matrix of MID_ς and every matrix $m = [r_1, r_2, \ldots, r_n]$ of MID_ς, where r_j, with $j = 1, \ldots, n$, is some insertion or substitution rule and i is the label of m, we add the following matrices

$$\begin{aligned}
&[r_1, (\lambda, v_{i,1} \to v_{i,2}, \lambda)]\\
&[r_2, (\lambda, v_{i,2} \to v_{i,3}, \lambda)]\\
&\ldots\\
&[r_{n-1}, (\lambda, v_{i,n-1} \to v_{i,n}, \lambda)]\\
&[r_n, (\lambda, v_{i,n} \to v_{i',1}, \lambda)] \text{ and } [r_n, (\lambda, v_{i,n} \to v, \lambda)],
\end{aligned}$$

to MID'_ς. Intuitively, any letters of any sentential form of MID may be substituted or used as a context of some rule. Hence, to simulate MID, any letter of the form $v_{i,j}$ may have to be substituted or used as a context. Therefore, if one of the matrices added to MID'_ς is of the form $[(w_1, v \to v', w_2), (\lambda, v_{i,j} \to v_{i',j'}, \lambda)]$, we add the matrix $[(w_1, v_{i,j} \to v'_{i,j}, w_2), (\lambda, v'_{i,j} \to v'_{i',j'}, \lambda)]$ to MID'_ς, as well. Additionally, we add $[(w_1, v_{i,j} \to v'_{i,j}, w_2), (\lambda, v'_{i,j} \to v', \lambda)]$ to MID'_ς if $[(w_1, v \to v', w_2), (\lambda, v_{i,j} \to v, \lambda)] \in MID'_\varsigma$. Furthermore, if one of the matrices added to MID'_ς is of the form $[(w_{1,1}vw_{1,2}, w_3, w_2)_{\mathrm{ins}}, (\lambda, v_{i,j} \to v_{i',j'}, \lambda)]$ with $w_{1,1}, w_{1,2}, w_2, w_3 \in V^*$, then we add the matrix $[(w_{1,1}v_{i,j}w_{1,2}, w_3, w_2)_{\mathrm{ins}}, (\lambda, v_{i,j} \to v_{i',j'}, \lambda)]$. Additionally, if a matrix of the form $[(w_{1,1}vw_{1,2}, w_3, w_2)_{\mathrm{ins}}, (\lambda, v_{i,j} \to v, \lambda)]$ occurs in MID'_ς, then $[(w_{1,1}v_{i,j}w_{1,2}, w_3, w_2)_{\mathrm{ins}}, (\lambda, v'_{i,j} \to v', \lambda)]$ is also added to MID'_ς.

The cases

- $[(w_1, w_3, w_{2,1}vw_{2,2})_{\mathrm{ins}}, (\lambda, v_{i,j} \to v_{i',j'}, \lambda)]$
- $[(w_1, w_3, w_{2,1}vw_{2,2})_{\mathrm{ins}}, (\lambda, v_{i,j} \to v, \lambda)]$
- $[(w_{1,1}vw_{1,2}, a \to b, w_2), (\lambda, v_{i,j} \to v_{i',j'}, \lambda)]$
- $[(w_{1,1}vw_{1,2}, a \to b, w_2), (\lambda, v_{i,j} \to v, \lambda)]$
- $[(w_1, a \to b, w_{2,1}vw_{2,2}), (\lambda, v_{i,j} \to v_{i',j'}, \lambda)]$
- $[(w_1, a \to b, w_{2,1}vw_{2,2}), (\lambda, v_{i,j} \to v, \lambda)]$

are treated analogously. The set of axioms is defined as

$$A' = \{\alpha_{1_{i,1}}\alpha_2 \dots \alpha_n \mid \alpha_1, \dots, \alpha_n \in V,\ n \geq 1,\ \alpha_1 \dots \alpha_n \in A \text{ and } i \text{ is the label of a matrix of } MID_\varsigma\}.$$

Additionally, if $\lambda \in A$, for every matrix of MID_ς, which has the form

$$m = [(\lambda, a_1 \dots a_n, \lambda)_{\text{ins}}, r_2, \dots, r_n]$$

with $a_1, \dots, a_n \in V$, we add $a_{1_{i,2}}a_2 \dots a_n$ to A', where i is the label of m. This simulates the case that λ is the axiom of a derivation of MID_ς and the matrix m is applied.

The basic overall idea is the same as in Theorem 2. Here, nonterminals $v_{i,j}$ control the derivations. Hence, we can hence forego a formal inductive proof. □

3.2. One-Sided Context Dependence

Our previous computational completeness results either required two-sided contexts for insertion or two-sided contexts for deletions. As argued in the introduction, there are good motivations to try to reduce the context dependence. Hence, we are now looking at one-sided contexts for deletions and insertions. We are going to prove two main results in this subsection: first, we show that uni-directional (for instance, left) single-symbol context in insertions and deletions of single symbols suffice in achievinf computational completeness, which this is in contrast with the second result that tells that we cannot completely forego using context information: we do need one-sided context dependency for both deletions and for insertions.

3.2.1. Computational Completeness

As a consequence of the previously introduced normal form for matrix ins-del-sub systems, we obtain the following result:

Corollary 1. *The following statements hold.*

1. $MAT_2INS_1^{0,0}DEL_1^{1,1}SUB^{0,0} = RE.$
2. $MAT_2INS_1^{1,1}DEL_1^{0,0}SUB^{0,0} = RE.$
3. $MAT_2INS_1^{1,0}DEL_1^{1,0}SUB^{0,0} = RE.$
4. $MAT_2INS_1^{1,0}DEL_1^{0,1}SUB^{0,0} = RE.$
5. $MAT_2INS_2^{0,0}DEL_1^{1,0}SUB^{0,0} = RE.$
6. $MAT_2INS_1^{1,0}DEL_2^{0,0}SUB^{0,0} = RE.$

This follows easily with Theorem 2, as computational completeness has been shown for $\text{MAT}_3\text{INS}_1^{1,1}\text{DEL}_1^{0,0}$ and for $\text{MAT}_3\text{INS}_1^{0,0}\text{DEL}_1^{1,1}$, see [16]. Furthermore, computational completeness for $\text{MAT}_3\text{INS}_1^{1,0}\text{DEL}_1^{1,0}$, $\text{MAT}_3\text{INS}_1^{1,0}\text{DEL}_1^{0,1}$, $\text{MAT}_3\text{INS}_2^{0,0}\text{DEL}_1^{1,0}$, and $\text{MAT}_3\text{INS}_1^{1,0}\text{DEL}_2^{0,0}$ has been shown in [10]. Clearly, context-free substitution rules improve the existing completeness results by reducing the complexity of matrices.

3.2.2. Computational Incompleteness

Though ins-del systems with matrix control and substitution rules are powerful devices, they are not always sufficient for achieving computational completeness.

Lemma 3. *Let MID_ς be a a matrix ins-del-sub systems of size $(*; 1, 1, 0; 1, 0, 0; 0, 0)$. Subsequently, $L(MID_\varsigma) \in \mathcal{L}(M, CF)$.*

Proof. Let $MID_\varsigma = (V, T, A, M_\varsigma)$ be in normal form and constructed according to the construction in Theorem 2. Subsequently, we construct the following matrix grammar $G = (N, T, M_G, S)$ with $N = \{N_a \mid a \in V\}$. For every $\alpha_1 \dots \alpha_n \in A$, we add a matrix

$[S \to N_{\alpha_1} \dots N_{\alpha_n}]$ to M_G.

For every matrix of the form X of MID_ς, we add a matrix of the form Y to M_G.

form X	form Y
$[(\lambda, b, \lambda)_{\text{del}}, (\lambda, c \to d, \lambda)]$	$[N_b \to \lambda, N_c \to N_d]$
$[(\lambda, b, \lambda)_{\text{del}}, (\lambda, c, \lambda)_{\text{del}}]$	$[N_b \to \lambda, N_c \to \lambda]$
$[(\lambda, b \to b', \lambda), (\lambda, c \to d, \lambda)]$	$[N_b \to N_{b'}, N_c \to N_d]$
$[(\lambda, b \to b', \lambda), (\lambda, c, \lambda)_{\text{del}}]$	$[N_b \to N_{b'}, N_c \to \lambda]$
$[(a, b, \lambda)_{\text{ins}}, (\lambda, c \to d, \lambda)]$	$[N_a \to N_a N_b, N_c \to N_d]$
$[(a, b, \lambda)_{\text{ins}}, (\lambda, c, \lambda)_{\text{del}}]$	$[N_a \to N_a N_b, N_c \to \lambda]$

For every matrix of the form X' of MID_ς and every $N_a \in N$, we add matrices of the form Y' to M_G.

form X'	form Y'
$[(\lambda, b, \lambda)_{\text{ins}}, (\lambda, c \to d, \lambda)]$	$[N_a \to N_a N_b, N_c \to N_d], [N_a \to N_b N_a, N_c \to N_d]$
$[(\lambda, b, \lambda)_{\text{ins}}, (\lambda, c, \lambda)_{\text{del}}]$	$[N_a \to N_a N_b, N_c \to \lambda], [N_a \to N_b N_a, N_c \to \lambda]$

Furthermore, we add matrices of the form $[N_a \to a]$ to M_G if $a \in T$.
By induction, it can be shown that $a_1 \dots a_n \Rightarrow_G b_1 \dots b_m$ if and only if

$$N_{a_1} \dots N_{a_n} \hat{\Longrightarrow}_{MID_\varsigma} N_{b_1} \dots N_{b_m}.$$

Whenever a matrix $[(\lambda, b, \lambda)_{\text{ins}}, (\lambda, c \to d, \lambda)]$ is applicable to $a_1 \dots a_n \in V^*$, some matrix of the form $[N_a \to N_a N_b, N_c \to N_d]$ is applicable to $N_{a_1} \dots N_{a_n} \in N^*$ and vice versa.

We remark that, if a sentential form λ occurs during either a derivation of MID_ς or G, the derivation cannot proceed, as no matrix is applicable any more. (see Theorem 2). Hence, the case that the axiom of a derivation of MID_ς is λ is covered by an application of $S \to \lambda$.
Therefore, it is easy to see that $L(G) = L(MID_\varsigma)$ holds.
Moreover, $L(G) \in \mathcal{L}(\text{M}, \text{CF})$. □

Although systems of size $(2; 1, 1, 0; 1, 0, 0; 0, 0)$ do not reach computational completeness, they do characterize matrix grammars.

Lemma 4. *Let G be a matrix grammar, such that $L(G) \in \mathcal{L}(M, CF)$. Subsequently, there exists a matrix ins-del system with substitution rules MID_ς of size $(2; 1, 1, 0; 1, 0, 0; 0, 0)$, such that $L(G) = L(MID_\varsigma)$.*

Proof. Let $G = (N, T, M_G, S)$ be a matrix grammar with context-free production rules. Afterwards, we construct $MID_\varsigma = (V, T, A, M_\varsigma)$ as follows: We define $V = N \cup T \cup \{X\}$ and $A = \{S\}$.

Consider a context-free production rule of the form $A \to \lambda$. Clearly, this rule is equivalent to a deletion rule (λ, A, λ). Analogously, a production rule $A \to B$ is essentially the same as a substitution rule $(\lambda, A \to B, \lambda)$.

Consider a production rule of the form $A \to B_1 \dots B_n$, $n \geq 2$, $B_1, \dots, B_n \in N \cup T$, and the following sequence of substitution and deletion rules

$$(\lambda, A \to X, \lambda), (X, B_n, \lambda), (X, B_{n-1}, \lambda), \dots, (X, B_2, \lambda), (\lambda, X \to B_1, \lambda).$$

Clearly, applying this sequence to a word $w \in (N \cup T)^*$ is the same as applying the production rule $A \to B_1 \dots B_n$.

Hence, we add the matrices that were obtained by the Algorithm 1 to M_ς. □

Lemmas 3 and 4 yield the following result.

Algorithm 1 Generate_$M_\varsigma(M)$

Require: set M_G of matrices with context-free production rules
for all $m \in M_G$ **do**
1. replace every occurrence of a rule of the form $A \to \lambda$ in m with $(\lambda, A, \lambda)_{\text{del}}$
2. replace every occurrence of a rule of the form $A \to B$ in m with $(\lambda, A \to B, \lambda)$
3. replace every occurrence of a rule of the form $A \to B_1 \dots B_n$ in m with the sequence

$$(\lambda, A \to X, \lambda), (X, B_n, \lambda)_{\text{ins}}, (X, B_{n-1}, \lambda)_{\text{ins}}, \dots, (X, B_2, \lambda)_{\text{ins}}, (\lambda, X \to B_1, \lambda)$$

4. add the resulting matrix to M_ς
end for

Theorem 4. *$L \in \mathcal{L}(M, CF)$ if and only if there is a matrix ins-del-sub systems MID_ς of size $(*; 1, 1, 0; 1, 0, 0; 0, 0)$, such that $L(MID_\varsigma) = L$.*

Because matrix grammars with context-free production are not computationally complete [20], matrix ins-del-sub systems of size $(*; 1, 1, 0; 1, 0, 0; 0, 0)$ are not computationally complete either. It is known [12] that $\mathcal{L}(\mathrm{M}, \mathrm{CF})$ is closed under reversal. With Lemma 1, we can conclude:

Corollary 2. $MAT_* INS_1^{1,0} DEL_1^{0,0} SUB^{0,0} \cup MAT_* INS_1^{0,1} DEL_1^{0,0} SUB^{0,0} \subsetneq RE.$

We will now show that $\mathrm{MAT}_* \mathrm{INS}_1^{0,0} \mathrm{DEL}_1^{1,0} \mathrm{SUB}^{0,0}$ is not computationally complete, either. Consequently, we arrive at the conclusion that $\mathrm{MAT}_* \mathrm{INS}_1^{0,0} \mathrm{DEL}_1^{1,0}$ is not computationally complete either.

Consider the following construction for the proof. For each derivation of a matrix ins-del-sub system $MID_\varsigma = (V, T, A, M_\varsigma)$ of size $(*; 1, 0, 0; 1, 1, 0; 0, 0)$, we construct a group of trees that represents the structure of the derivation.

Each tree node is labelled by a string over V, such that reading the rightmost symbols of all root labels of the corresponding group of trees from left to right yields w (we refer to Figure 2).

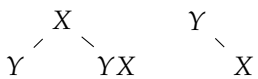


Figure 2. The tree group corresponding to a sentential form $w = XY$.

If an insertion rule (λ, a, λ) adds the letter a at some position of the sentential form, we add a new tree with a single node labelled a at the corresponding position in the group of trees (see Figure 3).

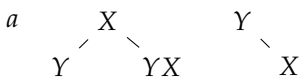


Figure 3. The tree group following the application of $(\lambda, a, \lambda)_{\text{ins}}$.

Applying a deletion rule (a, X, λ) has the following effect on the group of trees: the node corresponding to X becomes the rightmost child of the node corresponding to a (see Figure 4).

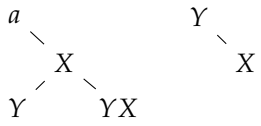


Figure 4. The tree group following the application of $(a, X, \lambda)_{\text{del}}$.

Let w_Y be the string of the node corresponding to a letter Y. If a substitution rule $(\lambda, Y \to b, \lambda)$ is applied, then we concatenate b right of w_Y (see Figure 5).

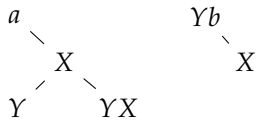


Figure 5. The tree group following the application of $(\lambda, Y \to b, \lambda)$.

Let the axiom of the derivation be $\alpha_1, \ldots, \alpha_n$. Subsequently, the group of trees consists initially of n trees with single nodes, each being labelled by a symbol of the axiom, such that reading the labels of the respective roots from left to right yields $\alpha_1, \ldots, \alpha_n$. Each root node corresponds to a letter of the current sentential form.

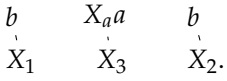
By construction, it is clear that only (the rightmost letter of) root labels contribute letters to the final word, i.e., each tree contributes, at most, one letter to the final word. Furthermore, it is clear that there is no interaction between letters of two different trees, i.e., a letter belonging to certain tree is not a context for some operation on a letter of another tree.

Before explaining the weaknesses of matrix ins-del-sub systems with their size limited to $(*; 1, 0, 0; 1, 1, 0; 0, 0)$, we illustrate their power by presenting a concrete example.

Example 3. *Consider a matrix ins-del-sub system MID_ς of size $(*; 1, 0, 0; 1, 1, 0; 0, 0)$, which has the axiom X_1X_2. Let*

$$m_1 = [(\lambda, b, \lambda)_{ins}, (b, X_1, \lambda)_{del}, (\lambda, X_3, \lambda)_{ins}]$$
$$m_2 = [(\lambda, b, \lambda)_{ins}, (b, X_2, \lambda)_{del}, (\lambda, X_a, \lambda)_{ins}]$$
$$m_3 = [(\lambda, X_a \to a, \lambda), (\lambda, a, \lambda)_{ins}, (a, X_3, \lambda)_{del}]$$

be matrices of MID_ς. Clearly $X_1X_2 \hat{\Longrightarrow}_{m_1m_2m_3} bab$ holds and the corresponding group of trees is



Note that all of the letters corresponding to the eventual a-tree originate from context-free insertions. Furthermore, because there is no interaction between letters of two different trees, inserting all letters of the eventual a-tree (in the order specified by the a-tree) left or right of all letters belonging to eventual b-trees does not affect the b-trees. Thus, $X_1X_2 \hat{\Longrightarrow}_{m_1m_2m_3} abb$ and $X_1X_2 \hat{\Longrightarrow}_{m_1m_2m_3} bba$ also hold.

Theorem 5. $REG \setminus MAT_*INS_1^{0,0}DEL_1^{1,0}SUB^{0,0} \neq \emptyset$.

Proof. We show that there is no matrix ins-del-sub system of size $(1, 0, 0; 1, 1, 0; 0, 0)$ generating the regular language a^+b^+. Assume to the contrary that $MID_\varsigma = (V, T, A, M_\varsigma)$ generates a^+b^+. Subsequently, MID_ς generates the word a^nb^n, $n > \gamma$, as well, where γ is the length of the longest axiom of MID_ς. Consider the group of trees corresponding to a derivation of a^nb^n starting from the axiom α. Because $n > \gamma$, there exists a tree t with the following properties: (1) the tree contributes a letter a to a^nb^n and (2) all nodes of the tree originate from the application of some insertion rule. Consider the derivation from

α to $a^n b^n$. Subsequently, MID_ς also generates $a^{n-1}b^n a$. The string $a^{n-1}b^n a$ is generated by applying the same matrices used in the derivation from α to $a^n b^n$ in the same order. All of the insertion rules corresponding to nodes of the tree t that are specified above are applied right of all letters belonging to (eventual) b-trees. Because there is no interaction between letters of two different trees, none of the letters that correspond to nodes of t are used as context to delete symbols not affiliated with nodes of t. Thus, inserting these letters right of all letters belonging to (eventual) b-trees changes nothing for the other trees. Note that the tree t specifies the position of the inserted letters in relation to each other as well as how the rest of the rules concerning symbols of t are applied. □

Interestingly, while neither $\mathrm{MAT}_*\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{0,0}$ nor $\mathrm{MAT}_*\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{1,0}\mathrm{SUB}^{0,0}$ are computationally complete, by Theorem 4 at least the context-free languages are included in $\mathrm{MAT}_*\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{0,0}$. $\mathrm{MAT}_*\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{1,0}\mathrm{SUB}^{0,0}$ does not even include all regular languages. Consequently, we can also state:

Corollary 3. *$REG \setminus MAT_*INS_1^{0,0}DEL_1^{1,0} \neq \emptyset$.*

We remark that Corollary 2 and Theorem 5 show that the result of Corollary 1 is optimal, i.e., the context dependency cannot be reduced any further without losing computational power.

3.3. Context-Free Substitutions Do Not Always Help

We now show that extending context-free matrix ins-del systems with context-free substitution rules does not result in an increase in computational power.

Theorem 6. *$MAT_*INS_n^{0,0}DEL_p^{0,0}SUB^{0,0} = MAT_*INS_n^{0,0}DEL_p^{0,0}$ with $n, p \geq 2$.*

Proof. Because $\mathrm{MAT}_*\mathrm{INS}_n^{0,0}\mathrm{DEL}_p^{0,0} \subseteq \mathrm{MAT}_*\mathrm{INS}_n^{0,0}\mathrm{DEL}_p^{0,0}\mathrm{SUB}^{0,0}$ holds by definition, we only prove the converse.

Let $MID_\varsigma = (V, T, A, M_\varsigma)$ be of size $(*; n, 0, 0; p, 0, 0; 0, 0)$. Subsequently, there exists a system $MID = (V \cup \{X\}, T, A, M')$ of size $(*; n, 0, 0; p, 0, 0)$ where X is a new symbol not in V, which simulates MID_ς. It is sufficient to prove that context-free substitution rules can be simulated by MID. Let $(\lambda, a \to b, \lambda)$ be a context-free substitution rule. Consider the sequence $(\lambda, Xb, \lambda)_{\mathrm{ins}}$, $(\lambda, aX, \lambda)_{\mathrm{del}}$. Applying this sequence to $w_1 a w_2$, $w_1, w_2 \in V^*$, is equivalent to applying the substitution rule $(\lambda, a \to b, \lambda)$, i.e., $w_1 a w_2 \Longrightarrow\!\!\!\!\!\!\wedge\ \ w_1 a X b w_2 \Longrightarrow\!\!\!\!\!\!\wedge\ \ w_1 b w_2$. Note that the string Xb has to be directly inserted right of a letter a, as, otherwise, $(\lambda, aX, \lambda)_{\mathrm{del}}$ cannot be applied.

Therefore, the set M' is constructed, as follows: let $m \in M$, then we replace all of the occurrences of substitution rules $(\lambda, a \to b, \lambda)$ in m with the sequence $(\lambda, Xb, \lambda)_{\mathrm{ins}}$, $(\lambda, aX, \lambda)_{\mathrm{del}}$. The resulting matrix is added to M'. This procedure is applied to all $m \in M$. □

By Theorem 1, we deduce that the matrix ins-del systems of size $(*; 2, 0, 0; 2, 0, 0; 0, 0)$ are not computationally complete.

Corollary 4. *$MAT_*INS_2^{0,0}DEL_2^{0,0}SUB^{0,0} \subsetneq RE$.*

3.4. One-Sided Substitutions

We now consider matrix ins-del systems with one-sided substitution rules. In particular, the families of systems that are discussed in detail now are $\mathrm{MAT}_2\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$ and $\mathrm{MAT}_2\mathrm{INS}_1^{0,0}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{1,0}$.

Theorem 7. *$MAT_*INS_1^{1,0}DEL_1^{1,0} \subseteq MAT_*INS_1^{0,0}DEL_1^{0,0}SUB^{1,0}$.*

Proof. Let $MID = (V, T, A, M) \in \mathrm{MAT}_*\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0}$. Subsequently, $MID_\varsigma = (V', T, A, M')$ is defined as follows: Let $V' := V \cup \{X\}$. Without a loss of generality, we assume $V \cap \{X\} = \emptyset$.
The following procedure is applied to all matrices of MID. Consider an arbitrary matrix m of MID. We replace every occurrence of an insertion rule of the form (a, b, λ) with an insertion rule (λ, X, λ) and a substitution rule $(a, X \to b, \lambda)$. Additionally, any deletion rule of the form (a, b, λ) is replaced with a substitution rule $(a, b \to X, \lambda)$ and a deletion rule (λ, X, λ). The matrix that is obtained by these replacements is added to M'.

Clearly, $MID_\varsigma \in \mathrm{MAT}_*\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$. The basic idea of this proof is that the nonterminal X is immediately resolved after being introduced. It is easy to see that applying the substitution rule $(a, b \to X, \lambda)$ immediately after the insertion rule (λ, X, λ) is essentially the same as applying a rule of the form (a, b, λ). Likewise, applying the deletion rule (λ, X, λ) immediately after the substitution rule $(a, b \to X, \lambda)$ is basically the same as applying the deletion rule (a, b, λ).

Therefore, clearly $L(MID) = L(MID_\varsigma)$. □

It has been shown in [10] that $\mathrm{MAT}_3\mathrm{INS}_1^{1,0}\mathrm{DEL}_1^{1,0} = \mathrm{RE}$ holds. Therefore, $\mathrm{RE} = \mathrm{MAT}_*\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{0,0}\mathrm{SUB}^{1,0}$. We can improve this result by using the result that is presented in Theorem 2. Together with Lemma 1, the next corollary follows.

Corollary 5. $RE = MAT_2INS_1^{0,0}DEL_1^{0,0}SUB^{1,0} = MAT_2INS_1^{0,0}DEL_1^{0,0}SUB^{0,1}$.

We now show that omitting deletion rules in the systems mentioned above yields a characterization of context-sensitive languages, which is quite rare with ins-del systems.

Lemma 5. $MAT_2INS_1^{0,0}DEL_0^{0,0}SUB^{0,1} \subseteq CS$

Proof. Let $MID = (V, T, A, M_\varsigma) \in \mathrm{MAT}_*\mathrm{INS}_1^{0,0}\mathrm{DEL}_0^{0,0}\mathrm{SUB}^{0,1}$ be constructed according to Theorem 3. The basic idea is the same as in Lemma 3. We define the matrix grammar $G = (N, T, P, M)$ with $N = \{N_a \mid a \in V\}$, as in Lemma 3 with the following addition: For every matrix of the form $[\lambda, b \to b', a), (\lambda, c \to d, \lambda)]$ in M_ς we add a matrix of the form $[N_bN_a \to N_{b'}N_a, N_c \to N_d]$ to M (we remark that the second component of every matrix of MID is a context-free substitution rule).

Clearly, G is a matrix grammar with context-sensitive production rules. In ([12] [Theorem 1.2.1]), it is shown that $\mathcal{L}(\mathrm{M}, \mathrm{CS}) = L(\mathrm{CS})$ holds. Therefore, our claim holds. □

We now prove the converse.

Lemma 6. $CS \subseteq MAT_2INS_1^{0,0}DEL_0^{0,0}SUB^{0,1}$

Proof. Let $G = (V, T, S, P)$ be a context-sensitive grammar in Penttonen normal form [21]. Subsequently, we construct the following matrix ins-del-sub systems to simulate G:

$$MID_\varsigma = (V', T, A, M_\varsigma)$$

with $V' = V \cup \{X_1, X_2\}$ and $A = \{S\}$. Without a loss of generality, we assume $V \cap \{X_1, X_2\} = \emptyset$.

The simulation of a production rule of the form $AB \to AC$ is carried out by the following matrix

$$[(\lambda, B \to X_1, \lambda), (\lambda, A \to X_2, X_1), (\lambda, X_1 \to C, \lambda), (\lambda, X_2 \to A, \lambda)].$$

A production rule of the form $A \to BC$ is simulated by the matrix

$$[(\lambda, A \to X_1, \lambda), (\lambda, X_2, \lambda)_{\mathrm{ins}}, (\lambda, X_2 \to B, X_1), (\lambda, X_1 \to C, \lambda)],$$

while a production rule of the form $A \to a$ is simulated by a matrix $[(\lambda, A \to a, \lambda)]$. We make the following observation: all nonterminals X_1, X_2, which are introduced by a matrix m, are resolved at the end of m.

In the following paragraph, we prove the equality $L(MID_\varsigma) = L(G)$ by induction. We begin by proving that, if there are matrices $m_1, \ldots, m_n \in M_\varsigma$ with $n \in \mathbb{N}$, such that

$$S \hat{\Longrightarrow}_{m_1} \ldots \hat{\Longrightarrow}_{m_n} w,$$

then $S \Rightarrow_G^n w$ holds. The base case, i.e., $n = 0$, is clear. We now consider the inductive step. Let

$$S \hat{\Longrightarrow}_{m_1} \ldots \hat{\Longrightarrow}_{m_n} w \hat{\Longrightarrow}_{m_{n+1}} w'.$$

Let the matrix that is used in the derivation step $w \hat{\Longrightarrow}_{m_{n+1}} w'$ be a matrix of the form

$$[(\lambda, B \to X_1, \lambda), (\lambda, A \to X_2, X_1), (\lambda, X_1 \to C, \lambda), (\lambda, X_2 \to A, \lambda)].$$

Subsequently, $w = w_1 ABw_2$ and $w' = w_1 ACw_2$ follow. Consider an application of m_{n+1}. Clearly $|w|_{\{X_1, X_2\}} = 0$ due to our observation. Therefore, the substitution rule $(\lambda, B \to X_1, \lambda)$ of m_{n+1} must be applied to a letter B whose left context is A. Otherwise, the substitution rule $(\lambda, A \to X_2, X_1)$ of m_{n+1} (and m_{n+1} itself) cannot be applied. Hence, it is easy to see that $w = w_1 ABw_2$ and $w' = w_1 ACw_2$ hold. Because of our induction hypothesis $S \Rightarrow_G^n w = w_1 ABw_2$ holds. Because of our construction, the existence of a matrix of the form m_{n+1} implies the existence of a production rule $AB \to AC$. Clearly,

$$w = w_1 ABw_2 \Rightarrow_{AB \to AC} w_1 ACw_2 = w'$$

holds. The case

$$m_{n+1} = [(\lambda, A \to X_1, \lambda), (\lambda, X_2, \lambda)_{\text{ins}}, (\lambda, X_2 \to B, X_1), (\lambda, X_1 \to C, \lambda)]$$

is handled analogously (clearly, the insertion rule $(\lambda, X_2, \lambda)_{\text{ins}}$ must insert X_2 left of X_1, otherwise $(\lambda, X_2 \to B, X_1)$ cannot be applied), while the case $m_{n+1} = [(\lambda, A \to a, \lambda)]$ is obvious.

The converse follows analogously. □

More specifically, with Lemmas 1, 5 and 6, we can state the following result.

Theorem 8. $CS = MAT_2 INS_1^{0,0} DEL_0^{0,0} SUB^{0,1} = MAT_2 INS_1^{0,0} DEL_0^{0,0} SUB^{1,0}$.

3.5. Adding Appearance Checking

We have previously shown that there are even regular languages not included in $\text{MAT}_* \text{INS}_2^{0,0} \text{DEL}_2^{0,0} \text{SUB}^{0,0}$. We now show that expanding these systems with appearance checking yields computational completeness. More precisely, we show that the expanded systems can simulate type-0 grammars in Penttonen normal form, which means that all of the production rules are of the form

$$AB \to AC \text{ or}$$
$$A \to BC \text{ or}$$
$$A \to a \text{ or}$$
$$A \to \lambda,$$

where A, B, C are nonterminal symbols and a is a terminal symbol.

Theorem 9. $MAT_*^{ac} INS_1^{0,0} DEL_2^{0,0} SUB^{0,0} = RE$.

Proof. Because the inclusion $\mathrm{MAT}^{ac}_{*}\mathrm{INS}^{0,0}_{2}\mathrm{DEL}^{0,0}_{2}\mathrm{SUB}^{0,0} \subseteq \mathrm{RE}$ is clear, we now proceed to prove the converse by simulating a type-0 grammar $G = (N, T, P, S)$ in Penttonen normal form. The matrix ins-del-sub systems with appearance checking simulating G is defined as $MID_{\varsigma,ac} = (V, T, \{\$S\}, M_{\varsigma}, F)$, with $V = N \cup T \cup \{X, X_1, X_2, \$, \$'\}$ and $(N \cup T) \cap \{X, X_1, X_2, \$, \$'\} = \emptyset$. The nonterminal \$ is an auxiliary symbol that marks the beginning of a sentential form, and it is eventually deleted by the matrix $[(\lambda, \$, \lambda)_{\mathrm{del}}]$. We now describe how the rules of G are simulated.

For every rule of G of the form $A \to a$, we add a matrix $[(\lambda, A \to a, \lambda)]$ to M_{ς}, and, for every rule $A \to \lambda$, we add the matrix $[(\lambda, A, \lambda)_{\mathrm{del}}]$ to M_{ς}. We remark that the following matrices that are introduced to M_{ς} require the sentential form to have \$ as the leftmost symbol. Hence, these rules cannot be applied if \$ is absent. For every rule of the form $AB \to AC$, to M_{ς}, we add a matrix

$$\begin{aligned}&[(\lambda, \$ \to \$', \lambda), (\lambda, B \to X, \lambda),\\ &(\lambda, A_1X, \lambda)_{\mathrm{del}}, \ldots, (\lambda, A_nX, \lambda)_{\mathrm{del}},\\ &(\lambda, X \to C, \lambda), (\lambda, \$' \to \$, \lambda)]\end{aligned}$$

with $V \setminus \{A\} = \{A_1, \ldots, A_n\}$. Add $\{(\lambda, A_1X, \lambda)_{\mathrm{del}}, \ldots, (\lambda, A_nX, \lambda)_{\mathrm{del}}\}$ to F. These deletion rules are used to check whether the left context of the B, which is substituted by X, has been A. The basic idea is as follows: consider the application of the matrix above to $\$w$, where w is a string over $V \setminus \{X, X_1, X_2, \$, \$'\}$. It is clear that the letter B, which is substituted by X, must have some symbol as its left context, i.e., this B cannot be the leftmost symbol. Furthermore, if the matrix above has been successfully applied, neither of the deletion rules in F has been applicable, as, otherwise, the substitution rule $(\lambda, X \to C, \lambda)$ could not have been applied. Therefore, the application of these deletion rules has been skipped during the processing of the matrix above. Because the letter B (which is eventually substituted by X) must have some left context, but neither of the deletion rules from F has been applicable, this means that the left context of this B could not have been a letter from $\{A_1, \ldots, A_n\} = V \setminus \{A\}$. Hence, the left context of this B has been A. Therefore, it is easy to see that the matrix above correctly simulates $AB \to AC$.

For every production rule of the form $A \to BC$, we add a matrix

$$\begin{aligned}&[(\lambda, \$ \to \$', \lambda), (\lambda, A \to X_1, \lambda), (\lambda, X_2, \lambda)_{\mathrm{ins}},\\ &(\lambda, A_1'X_2, \lambda)_{\mathrm{del}}, \ldots, (\lambda, A_n'X_2, \lambda)_{\mathrm{del}}, (\lambda, X_2\$', \lambda)_{\mathrm{del}}),\\ &(\lambda, X_1 \to B, \lambda)(\lambda, X_2 \to C, \lambda), (\lambda, \$' \to \$, \lambda)]\end{aligned}$$

with $\{A_1', \ldots, A_n'\} = V \setminus \{X_1\}$ to M_{ς}. The deletion rules $(\lambda, A_1'X_1, \lambda), \ldots, (\lambda, A_n'X_1, \lambda)$ and $(\lambda, X_2\$', \lambda)$ are added to F.

Consider a string w over $V \setminus \{X, X_1, X_2, \$, \$'\}$. Let the matrix above be applied to the string $\$w$. Using the same argumentation as before, $(\lambda, A_1'X_2, \lambda), \ldots, (\lambda, A_n'X_2, \lambda)$ in F ensure that the left context of the inserted X_2 is not an element of $\{A_1', \ldots, A_n'\} = V \setminus \{X_1\}$. Additionally, if the matrix above has been successfully applied, then the deletion rule $(\lambda, X_2\$', \lambda)_{\mathrm{del}}$ could not have been applicable, as, otherwise, the substitution rule $(\lambda, X_2 \to C, \lambda)$ could not have been be applied. Thus, X_2 has not been inserted left of \$'. This in turn means that X_2 must have been inserted left of X_1 and, therefore, it is clear that the above matrix simulates $A \to BC$.

We remark that applying any of the matrices, which simulate a production rule of G, to a sentential form of $MID_{\varsigma,ac}$, whose leftmost symbol is \$, results in a string whose leftmost symbol remains \$. By induction, we can show that $S \Rightarrow_G^* w$ iff $\$S \hat{\Longrightarrow}^{ac*} \$w$. □

Additionally, we can show that $\mathrm{MAT}^{ac}_{*}\mathrm{INS}^{0,0}_{1}\mathrm{DEL}^{1,0}_{1}\mathrm{SUB}^{0,0}$ is also computationally complete.

Theorem 10. $MAT^{ac}_{*}INS^{0,0}_{1}DEL^{1,0}_{1}SUB^{0,0} = RE$.

Proof. The idea is the same as in Theorem 9. Replacing all the deletion rules of the form (λ, ab, λ) in the matrices of the system constructed in Theorem 9 with deletion rules of the form (a, b, λ) yields our claim. □

This result shows that appearance checking is indeed powerful, as we have previously seen that $\mathrm{MAT}_{*}\mathrm{INS}_1^{0,0}\mathrm{DEL}_1^{1,0}\mathrm{SUB}^{0,0}$ does not even include all regular languages. Furthermore, we can conclude the following from this result.

Theorem 11. $MAT_{*}^{ac}INS_2^{0,0}DEL_2^{0,0} = RE$.

Proof. All of the substitution rules occurring in matrices of the construction in Theorem 9 can be replaced, as specified in the proof of Theorem 6. □

Additionally, the following can be derived from [22], based on ideas on P systems. This is interesting, as P systems (also known as membrane systems, introduced in [23]) are another formalization of a bio-computing device, when considering a cell (abstractly viewed as a membrane structure) as a computing mechanism.

An insertion-deletion P system is a construct $\Pi = (O, T, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n)$ where

- O is a finite alphabet,
- $T \subseteq O$ is the terminal alphabet,
- μ is the tree structure of the system which has n nodes,
- M_i, $1 \leq i \leq n$, is a finite language associated to the membrane i, and
- R_i, $1 \leq i \leq n$, is a set of insertion and deletion rules with target indicators of the node i. The rules are of the form: $(u, x, v; \mathrm{tar})$, where (u, x, v) is an insertion rule or a deletion rule, and $\mathrm{tar} \in \{\mathrm{here}, \mathrm{in}_j, \mathrm{out} \mid 1 \leq j \leq n\}$.

A configuration of Π is an n-tuple $(N_1, \ldots, N_n)$ of finite languages over O. The transition between two configurations consists of applying rules in parallel to all possible strings, non-deterministically with respect to the target indications associated with the rules. A sequence of transitions between configurations of a given insertion-deletion P system Π starting from the initial configuration is called a computation with respect to Π. The result of Π's computations is collected in the language $L(\Pi)$ that consists of all strings over T that are sent out of the root node during its computations.

An insertion-deletion P system has the priority of deletion rules over insertion rules if a rule $(u_1, x_1, v_1; \mathrm{tar}_1)$, where (u_1, x_1, v_1) is an insertion rule, is only allowed to be applied if no rule $(u_2, x_2, v_2; \mathrm{tar}_2)$, where (u_2, x_2, v_2) is a deletion rule, is applicable.

Theorem 12. $MAT_{*}^{ac}INS_1^{0,1}DEL_1^{0,0} = RE$.

Proof. Consider an insertion-deletion P system with the priority of deletion rules over insertion rules. It is known that such systems with insertion rules of size $(1, 0, 1)$ and deletion rules of size $(1, 0, 0)$ are computationally complete, see [22]. We now show that such an insertion-deletion P system with a priority of deletion rules over insertion rules can be simulated by an matrix ins-del system with the appearance checking of size $(*; 1, 0, 1; 1, 0, 0)$.

Let $\Pi = (O, T, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n)$ be such a system. An equivalent matrix ins-del system with appearance checking $MID_{\varsigma} = (O \cup \{1, \ldots, n\} \cup \{X\}, T, A, M_{\varsigma}, F)$, where X is a trap symbol, is constructed, as follows. We define $A = \{wi \mid w \in M_i\}$. For every deletion rule $(\lambda, x, \lambda; \mathrm{tar}) \in R_i$, we add a matrix $[(\lambda, i, \lambda)_{\mathrm{del}}, (\lambda, x, \lambda), (\lambda, k, \lambda)_{\mathrm{ins}}]$ to M_{ς}, where $k = i$ if $\mathrm{tar} = \mathrm{here}$, $k = j$ if $\mathrm{tar} = \mathrm{in}_j$ and $k = i'$ if $\mathrm{tar} = \mathrm{out}$ and i' is the parent node of i. Furthermore, if the node i has no parent node and if the deletion rule $(\lambda, x, \lambda; \mathrm{out})$ is in R_i, then we add the matrix $[(\lambda, i, \lambda)_{\mathrm{del}}, (\lambda, x, \lambda)]$ to M_{ς}.

Let $K_i = \{(\lambda, x_1, \lambda), \ldots, (\lambda, x_m, \lambda)\}$ denote the set of all deletion rules, which satisfy $(\lambda, x_\tau, \lambda; \text{tar}) \in R_i$ if $(\lambda, x_\tau, \lambda) \in K_i$. For every insertion rule $(\lambda, x, v; \text{tar}) \in R_i$, we add the matrix

$$[(\lambda, i, \lambda)_{\text{del}}, (\lambda, X, x_1)_{\text{ins}}, \ldots, (\lambda, X, x_m)_{\text{ins}}, (\lambda, x, v), (\lambda, k, \lambda)_{\text{ins}}]$$

to M_ς, where $k = i$ if tar = here, $k = j$ if tar = in$_j$ and $k = i'$ if tar = out and i' is the parent node of i. In the case the node i has no parent node, we add

$$[(\lambda, i, \lambda)_{\text{del}}, (\lambda, X, x_1)_{\text{ins}}, \ldots, (\lambda, X, x_m)_{\text{ins}}, (\lambda, x, v), (\lambda, k, \lambda)_{\text{ins}}]$$

to M_ς. All insertion rules, which introduce the trap symbol X are added to F.
The basic idea is as follows. By definition of Π, an insertion rule $(\lambda, x, v; \text{tar}) \in R_i$ can only be applied if no deletion rule $(\lambda, x_\tau, \lambda; \text{out}) \in R_i$ is applicable. In other words, $(\lambda, x, v; \text{tar})$ can only be applied if the current sentential form does not have any occurrence of $x_1, \ldots, x_m$. This is simulated via the insertion rules $(\lambda, X, x_1)_{\text{ins}}, \ldots, (\lambda, X, x_m)_{\text{ins}}$ in our matrices. Clearly, if any of these rules is applicable, then a trap symbol is introduced.
Becayse the language generated by Π consists of all words over T sent outside of the system during the computation, it can be shown that $L(\Pi) = L(MID_\varsigma)$. □

4. Conclusions

Our results complement the results that were obtained in the long versions of [6,7]. In particular, in these papers we have shown that extending ordinary ins-del systems with substitution rules yields, in most cases, an increase in computational power. In some cases, this increase can be quite significant. To give an overview, the main characterization results for RE and for CS of [6] and [7] have been, as follows.

Previous Result		**Extended with Substitution**	
Size	Family	Size	Family
$(2,0,0;2,0,0)$	$\subset$ CF	$(2,0,0;2,0,0;1,0)$	$=$ RE
$(1,0,0;1,0,0)$	$\subset$ REG	$(1,0,0;1,0,0;1,1)$	$=$ RE
$(1,0,0;0,0,0)$	$\subset$ REG	$(1,0,0;0,0,0;1,1)$	$=$ CS
$(1,0,1;1,0,0)$	$\subset$ RE	$(1,0,1;1,0,0;1,0)$	$=$ RE

The incompleteness results of systems without substitution rules can be found in [19] ([Th. 3.5] and [Th. 4.2]) and [24] [Th. 7], respectively.

We remark that deletion rules are essential for computational completeness results, as we have shown that, even with the addition of substitution rules, systems that do not have deletion rules lack computational power beyond CS. Furthermore, we have shown that $\text{INS}_n^{0,0}\text{DEL}_m^{0,0} = \text{INS}_n^{0,0}\text{DEL}_m^{0,0}\text{SUB}^{0,0}$, i.e., ins-del-sub systems require some context information; otherwise, substitution rules do not offer any benefits in regards to the size complexity of those systems.

We have shown that matrix ins-del systems do not need much context to reach computational completeness if substitution rules and appearance checking are used. For instance, we have shown that, in this setting, no context other than single symbol context for deletion rules is necessary for computational completeness. In the case of no context other than single symbol context for insertion rules, we have shown that appearance checking is a necessary and sufficient feature for ensuring computational completeness. Notice that the term "appearance checking" might, indeed, be a bit misleading, as it usually applies to checking the non-applicability of certain rules. Because we (always) test for the absence of single symbols, it might be better to formalize or interpret our results in the context of (forbidden) random context [25–29], or, more generally speaking, to semi-conditional (matrix) grammars. So-called *generalized forbidding matrix grammars* (GFM) have been presented at ICMC 2020 (the corresponding paper will appear in the Springer LNCS

volume 12687 of CMC21) and cover the idea of associating tests for the absence of symbols (or even words) as a filter prior to applying a matrix. The mentioned paper uses (more traditional) context-free rules within the matrices. In a sense, the results of our paper could also be seen as initializing the study of generalized forbidding matrix ins-del-sub systems. Subsequently, also studies on random context with respect to ins-del systems are also interesting for comparison, see [30,31]. Another interpretation could be in terms of ordered variants of matrix ins-del-sub systems, a topic so far explored only in connection with context-free grammars [32], because ordered grammars and forbidden context grammars mostly coincide, also see [12,33,34]. To provide an overview, our completeness results are collected in the following table.

Normal Matrix ins-del Systems		**Extended with Substitution and Appearance Checking**		
Size	Family	Size	Family	ac?
$(2;1,0,0;1,0,0)$	$\subsetneq$ RE	$(2;1,0,0;1,0,0;1,0)$	$=$ RE	-
$(2;1,0,0;0,0,0)$	$\subsetneq$ CS	$(2;1,0,0;0,0,0;1,0)$	$=$ CS	-
$(*;1,0,0;2,0,0)$	$\subsetneq$ RE	$(*;1,0,0;2,0,0;0,0)$	$=$ RE	✓
$(*;1,0,0;1,1,0)$	$\subsetneq$ RE	$(*;1,0,0;1,1,0;0,0)$	$=$ RE	✓
$(*;2,0,0;2,0,0)$	$\subsetneq$ RE	$(*;2,0,0;2,0,0)$	$=$ RE	✓
$(*;1,1,0;1,0,0)$	$\subsetneq$ RE	$(*;1,1,0;1,0,0)$	$=$ RE	✓

The incompleteness reults for normal matrix ins-del systems follow from [10] and Theorem 1, Theorems 8, Theorem 5, and Corollary 2.

Although matrix ins-del systems extended with substitution rules are powerful devices, this extension is not always sufficient for reaching computational completeness. More precisely, our incompleteness results have been thus as follows.

Incompleteness Results	
Size	Family
$(*;1,1,0;1,0,0;0,0)$	$= \mathcal{L}(\mathrm{M},\mathrm{CF}) \subsetneq$ RE
$(*;1,0,0;1,1,0;0,0)$	$\subsetneq$ RE
$(*;2,0,0;2,0,0;0,0)$	$\subsetneq$ RE

Coming back to the original motivation of our study, that of exploring the limits of computability with bio-computing devices, it is not that clear whether appearance checks (which always mean that the whole, say, RNA string has to be checked for possible sites where a rule might apply) can be efficiently implemented in real bio-computing mechanisms. Yet, they may serve as a yardstick, showing what is possible and also indicating in which way one might want to modify the formalisms in order to achieve computational completeness with smaller context dependencies for the different types of rules. Conversely, we suppose that, if bio-computing devices based on processing RNA or other large molecules leaves the experimental laboratories one day, turning into an industrial-strength technology, then computational completeness results, as presented in this paper on matrix ins-del-sub systems, as well as in previous papers for ins-del-sub systems, may serve as a basis of implementing compilers and so forth to master these future machines.

Author Contributions: Conceptualization, H.F.; writing—original draft preparation, M.V.; writing—review and editing, H.F., M.V.; supervision, H.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
RE	Recursively enumerable (languages)
CS	Context-sensitive (languages)
CF	Context-free (languages)
REG	Regular (languages)
ac	Appearance checking
INS/ins	Insertion
DEL/del	Deletion
SUB/sub	Substitution
MAT/M	Matrix
P system	Păun (membrane) system

References

1. Kari, L. On Insertions and Deletions in Formal Languages. Ph.D. Thesis, University of Turku, Turku, Finland, 1991.
2. Kari, L.; Păun, Gh.; Thierrin, G.; Yu, S. At the crossroads of DNA computing and formal languages: Characterizing recursively enumerable languages using insertion-deletion systems. In *DNA Based Computers III*; the Center for Discrete Mathematics and Theoretical Computer Science: Piscataway, NJ, USA; the Association for Computer Machinery (ACM): NewYork, NY, USA,1999; Volume 48, pp. 329–338.
3. Beaver, D. Computing with DNA. *J. Comput. Biol.* **1995**, *2*, 1–7. [CrossRef] [PubMed]
4. Kari, L. DNA computing: Arrival of biological mathematics. *Math. Intell.* **1997**, *19*, 9–22.
5. Freund, R.; Rogozhin, Y.; Verlan, S. Generating and accepting P systems with minimal left and right insertion and deletion. *Nat. Comput.* **2014**, *13*, 257–268. [CrossRef]
6. Vu, M.; Fernau, H. *Insertion-Deletion Systems With Substitutions I. Computability in Europe, CiE*; Anselmo, M., Vedova, G.D., Manea, F., Pauly, A., Eds.; Springer Nature Switzerland AG, 2020; Volume 12098, pp. 366–378.
7. Vu, M.; Fernau, H. Insertion-Deletion Systems With Substitutions II. In Proceedings of the Descriptional Complexity of Formal Systems—22nd International Conference, DCFS, Vienna, Austria, 24–26 August 2020; Jiraskova, G., Pighizzini, G., Eds.; Springer Nature Switzerland AG, 2020; Volume 12442, pp. 231–243.
8. Vu, M. On Insertion-Deletion Systems with Substitution Rules. Master's Thesis, Informatikwissenschaften, Universität Trier, Trier, Germany, 2019.
9. Kuppusamy, L.; Mahendran, A.; Krishna, S.N. On Representing Natural Languages and Bio-molecular Structures using Matrix Insertion-deletion Systems and its Computational Completeness. In Proceedings of the 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology (ICAART 2011), Rome, Italy, 28–30 January 2011; pp. 47–56.
10. Petre, I.; Verlan, S. Matrix insertion-deletion systems. *Theor. Comput. Sci.* **2012**, *456*, 80–88. [CrossRef]
11. Ábrahám, S. Some questions of phrase-structure grammars, I. *Comput. Linguist.* **1965**, *4*, 61–70.
12. Dassow, J.; Păun, Gh. *Regulated Rewriting in Formal Language Theory*; EATCS Monographs in Theoretical Computer Science; Springer-Verlag: Berlin/Heidelberg, Germany, 1989; Volume 18.
13. Alhazov, A.; Krassovitskiy, A.; Rogozhin, Y.; Verlan, S. Small size insertion and deletion systems. In *Applications of Language Methods*; Martin-Vide, C., Ed.; Imperial College Press: River Edge, NJ, USA, 2010; pp. 459–515.
14. Verlan, S. Recent Developments on Insertion-Deletion Systems. *Comput. Sci. J. Mold.* **2010**, *18*, 210–245.
15. Kuppusamy, L.; Mahendran, A.; Krishna, S.N. Matrix Insertion-Deletion Systems for Bio-Molecular Structures. In Proceedings of the Distributed Computing and Internet Technology—7th International Conference, ICDCIT, Bhubaneshwar, India, 9–12 February 2011; Natarajan, R., Ojo, A.K., Eds.; Springer-Verlag: Berlin/Heidelberg, Germany, 2011; Volume 6536, pp. 301–312.
16. Fernau, H.; Kuppusamy, L.; Raman, I. Investigations on the power of matrix insertion-deletion systems with small sizes. *Nat. Comput.* **2018**, *17*, 249–269. [CrossRef]
17. Fernau, H.; Kuppusamy, L.; Raman, I. On Matrix Ins-Del Systems of Small Sum-Norm. In *SOFSEM: Theory and Practice of Computer Science*; Catania, B., Královič, R., Nawrocki, J., Pighizzini, G., Eds.; Springer Nature Switzerland AG, 2019; Volume 11376, pp. 192–205.
18. Krassovitskiy, A.; Rogozhin, Y.; Verlan, S. Computational power of insertion-deletion (P) systems with rules of size two. *Nat. Comput.* **2011**, *10*, 835–852. [CrossRef]
19. Verlan, S. On Minimal Context-Free Insertion-Deletion Systems. *J. Autom. Lang. Comb.* **2007**, *12*, 317–328.
20. Hauschildt, D.; Jantzen, M. Petri net algorithms in the theory of matrix grammars. *Acta Inform.* **1994**, *31*, 719–728. [CrossRef]
21. Penttonen, M. One-sided and two-sided context in formal grammars. *Inf. Control (Now Inf. Comput.)* **1974**, *25*, 371–392. [CrossRef]

22. Alhazov, A.; Krassovitskiy, A.; Rogozhin, Y.; Verlan, S. P systems with minimal insertion and deletion. *Theor. Comput. Sci.* **2011**, *412*, 136–144. [CrossRef]
23. Păun, G. Computing with Membranes. *J. Comput. Syst. Sci.* **2000**, *61*, 108–143. [CrossRef]
24. Matveevici, A.; Rogozhin, Y.; Verlan, S. Insertion-Deletion Systems with One-Sided Contexts. In Proceedings of the Machines, Computations, and Universality, 5th International Conference, MCU, Orléans, France, 10–13 September 2007; Durand-Lose, J.O., Margenstern, M., Eds.; Springer-Verlag: Berlin/Heidelberg, Germany, 2007; Volume 4664, pp. 205–217.
25. Fernau, H.; Kuppusamy, L.; Raman, I. Descriptional Complexity of Matrix Simple Semi-conditional Grammars. In Proceedings of the Descriptional Complexity of Formal Systems—21st IFIP WG 1.02 International Conference, DCFS; Hospodár, M., Jirásková, G., Konstantinidis, S., Eds.; Springer Nature Switzerland AG, 2019; Volume 11612, pp. 111–123.
26. Fernau, H.; Kuppusamy, L.; Oladele, R.O.; Raman, I. Improved descriptional complexity results on generalized forbidding grammars. *Discret. Appl. Math.* **2021**. [CrossRef]
27. Meduna, A. Generalized forbidding grammars. *Int. J. Comput. Math.* **1990**, *36*, 31–39. [CrossRef]
28. Păun, Gh. A variant of random context grammars: semi-conditional grammars. *Theor. Comput. Sci.* **1985**, *41*, 1–17. [CrossRef]
29. van der Walt, P.J. Random context languages. In *Processing IFIP Congress*; North-Holland Publishing Company: Amsterdam, The Netherlands, 1972; pp. 66–68.
30. Fernau, H.; Kuppusamy, L.; Raman, I. Computational Completeness of Simple Semi-conditional Insertion-Deletion Systems. In *Unconventional Computation and Natural Computation, UCNC*; Stepney, S., Verlan, S., Eds.; Springer International Publishing AG, part of Springer Nature, 2018; Volume 10867, pp. 86–100.
31. Ivanov, S.; Verlan, S. Random Context and Semi-conditional Insertion-deletion Systems. *Fundam. Inform.* **2015**, *138*, 127–144. [CrossRef]
32. Dassow, J.; Păun, G. On ordered variants of some regulated grammars. *J. Inf. Process. Cybern. EIK* **1985**, *21*, 491–504.
33. Fernau, H. Closure properties of ordered languages. *EATCS Bull.* **1996**, *58*, 159–162.
34. Freund, R. A General Framework for Sequential Grammars with Control Mechanisms. In Proceedings of the Descriptional Complexity of Formal Systems—21st International Conference, DCFS, Košice, Slovakia, 17–19 July 2019; Hospodár, M., Jirásková, G., Konstantinidis, S., Eds.; Springer Nature Switzerland AG: Cham, Switzerland, 2019; Volume 11612, pp. 1–34.