On the generative capacity of matrix insertion-deletion systems of small sum-norm

Henning Fernau1 · Lakshmanan Kuppusamy2 · Indhumathi Raman3

Accepted: 21 June 2021 / Published online: 8 September 2021 © The Author(s) 2021

Abstract
A matrix insertion-deletion system (or matrix ins-del system) is described by a set of insertion-deletion rules presented in matrix form, which demands all rules of a matrix to be applied in the given order. These systems were introduced to model very simplistic fragments of sequential programs based on insertion and deletion as elementary operations as can be found in biocomputing. We are investigating such systems with limited resources as formalized in descriptional complexity. A traditional descriptional complexity measure of such a matrix ins-del system is its size $s = (k; n, i', l''; m, j', j'')$, where the parameters from left to right represent the maximal matrix length, maximal insertion string length, maximal length of left contexts in insertion rules, maximal length of right contexts in insertion rules; the last three are deletion counterparts of the previous three parameters. We call the sum $n + i' + l'' + m + j' + j''$ the sum-norm of $s$. We show that matrix ins-del systems of sum-norm 4 and sizes (3; 1, 0, 0; 1, 2, 0), (3; 1, 0, 0; 1, 0, 2), (2; 1, 2, 0; 1, 0, 0), (2; 1, 0, 2; 1, 0, 0), and (2; 1, 1; 1; 1, 0, 0) describe the recursively enumerable languages. Moreover, matrix ins-del systems of sizes (3; 1, 1, 0; 1, 0, 0), (3; 1, 0, 1; 1, 0, 0) and (2; 2, 0, 1; 1, 0, 0) can describe at least the regular closure of the linear languages. In fact, we show that if a matrix ins-del system of size $s$ can describe the class of linear languages LIN, then without any additional resources, matrix ins-del systems of size $s$ also describe the regular closure of LIN. Finally, we prove that matrix ins-del systems of sizes (2; 1, 1, 0; 1, 1, 0) and (2; 1, 0, 1; 1, 0, 1) can describe at least the regular languages.

Keywords Matrix control · Insertion-deletion systems · Descriptional complexity · Computational completeness · Regular closure of linear languages

1 Introduction

Two common operations while processing natural languages are inserting and deleting words in between parts of sentences; such insertions and deletions are usually based on context information. The (context-free) insertion operation was first considered in Haussler (1983), but also recently in Verlan et al. (2020). The deletion operation as a basis of a grammatical derivation process was introduced in Kari (1991), where the deletion was motivated as a variant of the right-quotient operation that does not necessarily happen at the right end of the string. Insertion and deletion were considered with a linguistic motivation in Galiukschov (1981) and together were first studied in Kari and Thierrin (1996). The corresponding grammatical mechanism is called an insertion-deletion system (abbreviated as ins-del system). Informally, the insertion and deletion operations of an ins-del system are defined as
follows: if a string $\eta$ is inserted between two parts $w_1$ and $w_2$ of a string $w_1w_2$ to get $w_1\eta w_2$, we call the operation insertion, whereas if a substring $\delta$ is deleted from a string $w_1\delta w_2$ to get $w_1w_2$, we call the operation deletion. Suffixes of $w_1$ and prefixes of $w_2$ are called contexts.

Several variants of ins-del systems have been considered in the literature, imposing regulation mechanisms on top, motivated by classical formal language theory (Dassow and Päun 1989). Some interesting variants (from our perspective) are ins-del P systems (Alhazov et al. 2011), tissue P systems with ins-del rules (Kuppusamy and Rama 2003), context-free ins-del systems (Margenstern et al. 2005), graph-controlled ins-del systems (Fernau et al. 2017b; Freund et al. 2010; Ivanov and Verlan 2017), matrix insertion systems (Marcus and Päun 1990), matrix ins-del systems (Kuppusamy et al. 2011; Petre and Verlan 2012; Kuppusamy and Mahendran 2016), random context and semi-conditional ins-del systems (Ivanov and Verlan 2015), etc. We refer to the survey (Verlan 2010) for more details of several variants of ins-del systems. In this paper, we focus on matrix ins-del systems. In Table 1, we report on what resources are needed for a matrix ins-del system of sum-norm 3 or 4 to generate the class specified there, also giving a short literature survey. Further races for smaller sizes are described when we discuss the particularities of our results below.

Also, the open questions that we list are sometimes connected, sometimes not. For instance, by the closure of RE under reversal, $\text{MAT}(\{;1,0,0;1,1,0\}) = \text{RE}$ iff $\text{MAT}(\{;1,0,0;1,0,1\}) = \text{RE}$ (here all the parameters of insertions are swapped with deletion parameters) seem to be independent of each other.

Matrix ins-del systems of sizes $(3; 1, 1, 0; 1, 1, 0)$ and $(3; 1, 0, 1; 1, 0, 1)$ are shown to RE (here all the parameters of insertions are swapped with deletion parameters) seem to be independent of each other.
regular languages. This is clearly only a weak replacement of the earlier claimed computational completeness result.

A further technical contribution consists in formulating a new normal form, called time separating special Geffert normal form (tsSGNF), that allows to simplify some derivable sentential form. Hence, such mixed cases need not be considered when proving correctness of simulation results based on tsSGNF. This is important, as the non-existence of such mixed forms is often tacitly assumed in several proofs that use SGNF; replacing SGNF by tsSGNF should help to easily fix these results. In this respect, matrix control differs from graph control \(^2\) (GCID) in connection with ins-del systems since the latter control demands that if \( \text{LIN} \subseteq \text{GCID}(k; n, i', i''; m, j', j'') \), then \( \mathcal{L}_{\text{reg}}(\text{LIN}) \subseteq \text{GCID}(k + 2; n, i', i''; m, j', j'') \); see Fernau et al. (2017b), Fernau et al. (2017d).

Parts of this work have been presented at the conference SOFSEM 2019; see Fernau et al. (2019). As an easy reference, the following results were showcased in Fernau et al. (2019).

1. \( \text{MAT}(3; 1, 0, 0; 1, 2, 0) = \text{MAT}(3; 1, 0, 0; 1, 0, 2) = \text{RE} \).
2. \( \text{MAT}(2; 1, 1, 0; 1, 1, 0) = \text{MAT}(2; 1, 0, 1; 1, 0, 1) = \text{RE} \).
3. \( \text{MAT}(3; 1, 2, 0; 1, 0, 0) = \text{MAT}(3; 1, 0, 2; 1, 0, 0) = \text{RE} \).
4. \( \mathcal{L}_{\text{reg}}(\text{LIN}) \subseteq \text{MAT}(3; 1, 1, 0; 1, 0, 0) \cap \text{MAT}(3; 1, 0, 1; 1, 0, 0) \).

---

### Table 1
Generative power of \( \text{MAT}(k; n, i', i''; m, j', j'') \) with sum-norm 3 or 4

| Sum-norm | ID size \((n, i', i''; m, j', j'')\), where \(n, m \in \{1, 2\}\) and \(i', i'', j', j'' \in \{0, 1, 2\}\) | Matrix length \(k\) | Language family relation | Remarks |
|----------|-------------------------------------------------|-----------------|------------------------|--------|
| 3        | \((1, 0, 0; 2, 0, 0), (2, 0, 0; 1, 0, 0)\)        | \(\geq 1\)     | \(\neq \text{RE}\)    | Inferred from Krassovitskiy et al. (2011) |
| 3        | \((1, 0, 0; 1, 1, 0), (1, 0, 0; 1, 0, 1)\)       | \(1\)          | \(\neq \text{RE}\)    | Inferred from Krassovitskiy et al. (2011) |
| 3        | \((1, 1, 0; 1, 0, 0), (1, 0, 1; 1, 0, 0)\)       | \(\geq 2\)     | ?                      | OPEN   |
| 3        | \((1, 1, 0; 1, 1, 0), (1, 1, 1; 1, 0, 0)\)       | \(3\)          | \(\supset\) MLIN      | Fernau et al. (2018b) |
| 4        | \((1, 0, 0; 1, 1, 1)\)                          | \(3\)          | \(\supset\) \(\mathcal{L}_{\text{reg}}(\text{LIN})\) | Fernau et al. (2019), Corollary 1 |
| 4        | \((1, 0, 0; 1, 1, 0), (1, 0, 0; 1, 0, 1)\)       | \(2\)          | \(\neq \text{RE}\)    | Fernau et al. (2018b) |
| 4        | \((1, 1, 0; 1, 1, 0), (1, 1, 0; 1, 0, 1)\)       | \(2\)          | \(\supset\) REG       | Fernau et al. (2018b) |
| 4        | \((1, 1, 0; 1, 1, 0), (1, 0, 1; 1, 1, 0)\)       | \(3\)          | \(\neq \text{RE}\)    | Petre and Verlan (2012) and Fernau et al. (2018b) |
| 4        | \((1, 1, 0; 1, 0, 1), (1, 1, 0; 1, 1, 0)\)       | \(3\)          | \(\supset\) CF        | Krassovitskiy et al. (2011) |
| 4        | \((1, 1, 0; 1, 0, 1), (1, 0, 1; 1, 1, 0)\)       | \(1\)          | \(\neq \text{RE}\)    | Krassovitskiy et al. (2011) |
| 4        | \((1, 0, 1; 2, 0, 0), (1, 0, 1; 2, 0, 0)\)       | \(2\)          | \(\neq \text{RE}\)    | Krassovitskiy et al. (2011) |
| 4        | \((2, 1, 0; 1, 0, 0), (2, 0, 1; 1, 0, 0)\)       | \(2\)          | \(\neq \text{RE}\)    | Krassovitskiy et al. (2011) |
| 4        | \((2, 1, 0; 1, 0, 0), (2, 1, 0; 1, 0, 0)\)       | \(2\)          | \(\supset\) MLIN      | Fernau et al. (2018b) |
| 4        | \((2, 1, 0; 1, 0, 0), (2, 0, 1; 1, 0, 0)\)       | \(2\)          | \(\supset\) \(\mathcal{L}_{\text{reg}}(\text{LIN})\) | Fernau et al. (2019), Corollary 1 |

---

\(^2\) One should also keep in mind the slightly different meanings of the parameter lists.
However, most of the proofs have been suppressed in the conference paper. In this extended version of the paper, we retain the first one and the last two results stated above from Fernau et al. (2019), however with detailed proofs. We must also thank one of the referees of this manuscript for pointing us to a problem with the construction suggested in Fernau et al. (2019) for proving the above-stated second result. The matrix length in the above-stated third result is improved from 3 to 2 in this paper and a new result $\text{MAT}(2; 1, 1, 1; 1, 0, 0) = \text{RE}$ is also proved here. A summary of the results of this paper can be seen in Tables 1 and 2 below.

## 2 Preliminaries

We assume that the readers are familiar with the standard notations in formal language theory. We recall a few notations here to keep the paper self-contained.

Let $\Sigma^*$ denote the free monoid generated by the alphabet (finite set) $\Sigma$. The elements of $\Sigma^*$ are called strings or words; $\lambda$ denotes the empty string, $L^R$ and $L^L$ denote the reversal of language $L$ and language family $L$, respectively. RE and LIN denote the families of recursively enumerable languages and linear languages, respectively. Occasionally, we use the shuffle operator, written as $\shuffle$.

For the computational completeness results, we use the fact that type-0 grammars in Special Geffert Normal Form (SGNF) (Freund et al. 2010) are known to characterize the recursively enumerable languages and is extensively used in Fernau et al. (2017a, 2017c, 2018a) and Petre and Verlan (2012). In fact, we slightly extend this notion in order to simplify certain arguments below. These simplifications were often tacitly assumed in previous works, but the following new definition gives good ground for it.

### Definition 1

A type-0 grammar $G = (N, T, S, P)$ is said to be in time separating special Geffert normal form, tSGNF for short, if $N$ is the nonterminal alphabet, $T$ is the terminal alphabet, $S \in N$ is the start symbol and $P$ is the set of production rules satisfying the following conditions:

- $N$ decomposes as $N = N^{(0)} \cup N' \cup N''$, where $N'' = \{A, B, C, D\}$ and $S \in N^{(0)}$, $S' \in N'$,
- the only non-context-free rules in $P$ are the two erasing rules $AB \rightarrow \lambda$ and $CD \rightarrow \lambda$,
- the context-free rules are of one of the following forms:
  
  - $X \rightarrow Yb$ or $X \rightarrow b'Y$ where $X \in N^{(0)}$, $Y \in N^{(0)} \cup N'$, $b \in T$,
  - $X \rightarrow Yb''$ or $X \rightarrow b''Y$ where $X, Y \in N'$, or $S' \rightarrow \lambda$,
  - (a) possibly, there is also the rule $S \rightarrow \lambda$.

### Remark 1

Notice that as these context-free rules are more of a linear type, it is easy to see that there can be at most one nonterminal from $N^{(0)} \cup N'$ present in the derivation of $G$. We exploit this observation in our proofs. According to the construction of this normal form described in Freund et al. (2010) and Geffert (1991), the derivation of a string is performed in two phases. In Phase I, the context-free rules are applied repeatedly. More precisely, this phase splits into two stages: in stage one, rules from (a) have left-hand sides from $N^{(0)}$; this stage produces a string of terminal symbols to the right side of the only nonterminal from $N^{(0)}$ in the sentential form and codings thereof are put on the left side of the only nonterminal occurrence from $N^{(0)}$; the transition to stage two is performed by using rules with left-hand sides from $N^{(0)}$ and one symbol from $N'$ occurring on the right-hand sides; in stage two, rules from (b) with left-hand sides from $N'$ are applied; importantly, here (and later) no further terminal symbols are produced. This separation into non-interacting times of the derivation...
process was the reason to call this normal form time-separating. In the previous version of the normal form, it was possible that rules from the two stages could be applied also in different orders; this could lead to problems in the simulation. The two erasing rules $AB \rightarrow \lambda$ and $CD \rightarrow \lambda$ are not applicable during the first phase as long as there is a $S$ (or $S'$) in the middle. All the symbols $A$ and $C$ are generated on the left side of these middle symbols and the corresponding symbols $B$ and $D$ are generated on the right side. Phase I is completed by applying the rule $S$ as in Kuppusamy et al. (2011), Petre and Verlan (2008). Matrix insertion-deletion simulation. The two erasing rules also appear in different orders; this could lead to problems in the checking process. We call this normal form normal form.

On the generative capacity of matrix insertion-deletion systems of small sum-norm (or $S$) not applicable during the first phase as long as there is a $S$ and $S'$ in the normal form. We can alternate between even and odd derivation steps.

Let us remark that the idea of a time separating (special) Geffert normal form is useful even when the number of nonterminal symbols matter, as proven by us in a recent paper presented at ICMC 2020 (to appear in its LNCS volume).

Our reasoning shows in particular the following first result:

**Theorem 1** For any recursively enumerable language $L$, i.e., $L \in \text{RE}$, there exists a type-0 grammar $G$ in $\text{tsSGNF}$ with $L = L(G)$.

### 2.1 Matrix insertion-deletion systems

In this subsection, we describe matrix insertion-deletion systems as in Kuppusamy et al. (2011), Petre and Verlan (2012) and Kuppusamy and Mahendran (2016).

**Definition 2** A matrix insertion-deletion system is a construct $\Gamma = (V, T, A, R)$ where $V$ is an alphabet, $T \subseteq V$, $A$ is a finite language over $V$, $R$ is a finite set of matrices $\{r_1, r_2, \ldots, r_l\}$, where each $r_i$, $1 \leq i \leq l$, is a matrix of the form $r_i = [(u_1, x_1, v_1)_{i_1}, (u_2, x_2, v_2)_{i_2}, \ldots, (u_k, x_k, v_k)_{i_l}]$. For $1 \leq j \leq k$, $u_j, v_j \in V^*$, $x_j \in V^+$ and $t_j \in \{\text{ins}, \text{del}\}$.

The triplet $(u_j, v_j, x_j)_{i_j}$ is called an ins-del rule and the pair $(u_j, v_j)$ is termed the context with $u_j$ as the left and $v_j$ as the right context for $x_j$ in $t_j$; $x_j$ is called insertion string if $t_j = \text{ins}$ and deletion string if $t_j = \text{del}$. The elements of $A$ are called axioms. For all contexts of $r$ where $r \in \{\text{ins}, \text{del}\}$, if $u = \lambda$ or $v = \lambda$, then we call the context to be one-sided. If $u = v = \lambda$ for a rule, then the corresponding insertion/deletion can be done freely anywhere in the string and is called context-free insertion/deletion. An insertion rule is of the form $(u, \eta, v)_{\text{ins}}$, which means that the string $\eta$ is inserted between $u$ and $v$. A deletion rule is of the form $(u, \delta, v)_{\text{del}}$, which means that the string $\delta$ is deleted between $u$ and $v$. Applying $(u, \eta, v)_{\text{ins}}$ corresponds to applying the rewrite rule $uv \rightarrow u\eta v$, and applying $(u, \delta, v)_{\text{del}}$ corresponds to applying the rewriting rule $u\delta v \rightarrow uv$.

At this point, we make a note that in a derivation, the rules of a matrix are applied sequentially one after another in the given order and no rule is used in appearance checking, as it is often the case in more classical matrix grammars with rewriting rules; see Dassow and Păun (1989). For $x, y \in V^*$ we write $x \Rightarrow_y y$, if y can be obtained from $x$ by applying all the rules of a matrix $r_i$, $1 \leq i \leq l$, in order. The language $L(\Gamma)$ generated by $\Gamma$ is defined as $L(\Gamma) = \{w \in T^* | x \Rightarrow_* w, \text{for some } x \in A\}$, where $\Rightarrow_*$ (as usual with matrix ins-del systems) denotes the reflexive and transitive closure of $\Rightarrow := \bigcup_{R} \Rightarrow_*$.

If a matrix ins-del system has at most $k$ rules in a matrix and the size of the underlying ins-del system is $(n, i', i''; m, j', j'')$, then we denote the corresponding class of language by $\text{MAT}(k; n, i', i''; m, j', j'')$.

**Example 1** The language $L_1 = \{a^k b^p c^q d^m \mid m, n \geq 1\}$ of cross-serial dependencies can be generated by a binary matrix insertion-deletion system as follows: $\Gamma_1 = \{\{a, b, c, d\}, \{a, b, c, d\}, \{abcd\}, R\}$, where $R$ consists of two matrices: $m_1 = [\{a, a, \lambda\}_{\text{ins}}, \{c, c, \lambda\}_{\text{ins}}]$, $m_2 = [\{b, b, \lambda\}_{\text{ins}}, \{d, d, \lambda\}_{\text{ins}}]$. We note that the matrices $m_1' = [\{a, a, \lambda\}_{\text{ins}}, \{c, c, \lambda\}_{\text{ins}}]$ also generate $L_1$. Hence, $L_1 \subseteq \text{MAT}(2; 1, 1, 0; 0, 0, 0) \cap \text{MAT}(2; 1, 0, 1; 0, 0, 0)$.

We refer to Stabler (2004) for further variants and a discussion of the linguistic relevance of this type of example.

### 2.2 Regular closure of linear languages

Recall that a linear grammar is a context-free grammar $G = (N, T, S, P)$ whose productions are of the form $A \rightarrow x$, where $A$ is a nonterminal symbol, and $x$ is a word over $N \cup T$ with at most one occurrence of a nonterminal symbol. The language class LIN collects all languages that can be described by linear grammars. LIN can be characterized by linear grammars in normal form, which means that any rule $A \rightarrow x$ either obeys $x \in T \cup \{\lambda\}$ or $x \in NT$ or $x \in NT$. It is well known that LIN is not closed under concatenation and Kleene star. This motivates to consider the class $L_\text{reg}(\text{LIN})$ as the smallest class containing LIN that is closed under union, concatenation or Kleene star. Similarly, we can assume that any right-linear grammar that we consider is in normal form, i.e., it has only rules $A \rightarrow aB$ or $A \rightarrow \lambda$, with $A \in N$, $B \in N \setminus \{A\}$ and $a \in T$. The
following grammatical characterization for $L_{\text{reg}}(\text{LIN})$ was shown in Fernau et al. (2018d).

**Proposition 1** Fernau et al. (2018d) Let $L \subseteq T^*$ be some language. Then, $L \subseteq L_{\text{reg}}(\text{LIN})$ if and only if there is a context-free grammar $G = (N, T, S, P)$ with $L(G) = L$ that satisfies the following properties.

- $N$ can be partitioned into $N_0$ and $N'$.
- There is a right-linear grammar $G_R = (N_0, N', S, P_0)$.
- $N'$ can be further partitioned into $N_1, \ldots, N_k$ for some $k$, such that the restriction $P_i$ of $P$ involving symbols from $N_i \cup T$ are only linear rules, with $T$ serving as the terminal alphabet.
- $P$ can be partitioned into $P_0, P_1, \ldots, P_k$.

Notice that this characterization corresponds to a two-stage approach: First, the right-linear grammar $G_R$ is used to produce a sequence of symbols from $N'$ that both serve as terminal symbols for $G_R$ and as nonterminal symbols for the linear grammar $G_l$ that can be obtained from $G$ by using rules $P_l$ only. Here, it is not necessary but possible to insist on using $N'' \subseteq N'$ instead of $N'$ as the terminal alphabet of $G_R$, such that $N'' \cap N_i = \{S_i\}$ for each $i \in [1\ldots k]$, i.e., we can single out a start symbol $S_i$ for each $G_i$. Clearly, the linear rules mentioned in the previous proposition can be assumed to be in normal form. In order to simplify the proofs of some of our main results, the following observations from Fernau et al. (2018b); Fernau et al. (2018c) are helpful.

**Proposition 2** Fernau et al. (2018b) Let $L$ be a language class that is closed under reversal. Then, for all non-negative integers $k, n, i, i', m, j, j'$, we have that

- $\text{MAT}(k; n, i, i'; m, j, j') = [\text{MAT}(k; n, i', i; m, j', j)]^R$;
- $\mathcal{L} = \text{MAT}(k; n, i, i'; m, j, j')$ if and only if $\mathcal{L} = \text{MAT}(k; n, i', i'; m, j', j')$;
- $\mathcal{L} \subseteq \text{MAT}(k; n, i, i'; m, j, j')$ if and only if $\mathcal{L} \subseteq \text{MAT}(k; n, i', i'; m, j', j')$.

**Proposition 3** Fernau et al. (2018c) $L_{\text{reg}}(\text{LIN})$ is closed under reversal.

### 3 Computational completeness results

In this section, we show the computational completeness of matrix ins-del systems of sizes $(3; 1, 0, 0; 1, 2, 0), (3; 1, 0, 0; 1, 0, 2), (2; 1, 2, 0; 1, 0, 0), (2; 1, 0, 2; 1, 0, 0), (2; 1, 1, 0; 1, 1, 0), (2; 1, 0, 1; 1, 0, 1), (2; 1, 1, 1; 1, 0, 0),$ by providing matrix ins-del systems of the above said sizes that simulate type-0 grammars in tsSGNF. How these results improve/complement previously known results is explained in Table 2.

We often use labels from $[1\ldots|P|]$ to uniquely address the rules of a grammar in tsSGNF. Then, such labels (and possibly also primed version thereof) will be used as *rule markers* that are therefore part of the nonterminal alphabet of the simulating matrix ins-del system. For the ease of reference, we collect in $P_R$ the labels of the context-free rules of the form $X \rightarrow Yb$ (which resemble left-linear rules) and in $P_H$ the labels of the context-free rules of the form $X \rightarrow bY$ (which resemble right-linear rules).

Some of the key features in our construction of matrix ins-del systems in this section are the following ones:

- There is at least one deletion rule in most simulating matrices, mostly for reasons of control. In some cases, the axiom will have $\$ and we delete this in order to avoid repeating the matrix and to ensure that the matrix (containing the deletion of $\$) is applied only once until the intended rule simulation is completed. We insert $\$ again at the end of the simulation.
- In the majority of cases, at least one of the deletion rules of every matrix has a *rule marker* in the left context or the marker itself is deleted. A matrix of this type is said to be *guarded*. The importance of a matrix being guarded is that it can be applied only in the presence of the corresponding rule marker. This will avoid interference of any other matrix application.
- After successful application of every matrix, either a rule marker remains or the intended simulation is completed.
- As discussed in Remark 1, during Phase I, the symbols $A$ and $C$ are on the left of the middle nonterminal $S$ or $S'$ and the corresponding symbols $B$ and $D$ are on the right of $S$ or $S'$. When $S'$ is deleted from the center, the symbols from $\{A, C\}$ and $\{B, D\}$ may combine to be erased in Phase II.
- In the transition to Phase II, a special symbol $Z$ is introduced that is assumed to stay to the left of $AB$ or $CD$, whatever substring is to be deleted. Special matrices allow $Z$ to move in the sentential form or to be (finally) deleted. This is our novel idea not used in earlier papers.
- There is a subtlety if $\lambda \in L$. Then, we can assume that $S \rightarrow \lambda$ is in the simulated grammar, which would add one more erasing matrix, similar to $h1$ in Fig. 2, which deletes $S$ and introduces $Z$.

We now proceed to present our results.

**Theorem 2** \(\text{MAT}(3; 1, 0, 0; 1, 2, 0) = \text{MAT}(3; 1, 0, 0; 1, 0, 2) = \text{RE}\).

**Proof** Formally, consider a type-0 grammar $G = (N, T, P, S)$ in tsSGNF. The rules from $P$ are supposed to be labelled injectively with labels from the set $[1\ldots|P|]$, with label sets $P_R$ and $P_H$ as defined above. Also recall that
the nonterminal alphabet decomposes as $N = N^{(0)} \cup N' \cup N''$, $N^{(0)} = \{A, B, C, D\}$, $S \in N^{(0)}$, $S' \in N'$, according to the normal form. We construct a matrix insertion-deletion system $\Gamma = (V, T, \{S\}, M)$, where the alphabet of $\Gamma$ is

$$V = N \cup T \cup \{p, p', p'', p''| p \in P_{sl}\} \cup \{q, q', q''| q \in P_{sl}\} \cup \{Z\}.$$ 

The set of matrices $M$ of $\Gamma$ consists of the matrices described in the following.

We simulate a rule $p$: $X \rightarrow bY$, $X, Y \in N^{(0)} \cup N'$, $b \in N''$, i.e., $p \in P_{sl}$, by the four matrices displayed on the left-hand side of Fig. 1.

Similarly, we simulate the rule $q$: $X \rightarrow Yb$, $X, Y \in N^{(0)} \cup N'$, $b \in N'' \cup T$, i.e., $q \in P_{sl}$, by the four matrices shown on the right-hand side of Fig. 1 in a symmetrical manner.

Recall that applying the rule $h: S' \rightarrow \lambda$ starts Phase II within the working of $G$. In the simulation, the presence of a new symbol, $Z$, indicates that we are in Phase II. This motivates the introduction of the five matrices listed in Fig. 2.

We now proceed to prove that $L(\Gamma) = L(G)$. We initially prove that $L(G) \subseteq L(\Gamma)$ by showing that $\Gamma$ correctly simulates the application of the rules of the types $p, q, f, g, h$, as discussed above. We explain the working of the simulation matrices for the cases $p$ and $f$ mainly, as the working of $q$ and $g$ simulation matrices are similar, and as the working of the simulation of the $h$ rule is clear. Notice that the transition from Phase I to Phase II (as accomplished by applying $h$ in $G$) is now carried out by applying $h1$ and hence introducing $Z$ which will be always present when simulating Phase II with the system $\Gamma$.

**Simulation of $p : X \rightarrow bY$:** consider the string $zX\beta$ derivable from $S$ in $G$, with $X \in N^{(0)} \cup N'$ and $x \in \{A, C\}^*$, $\beta \in \{B, D\}^*T^*$ according to Remark 1. We now show that on applying the matrices introduced for simulating rules from $P_{sl}$, we can derive $zbY\beta$ within $\Gamma$, starting from $zX\beta$. First, we apply the rules of matrix $p1$. The markers $p$ and $p'$ are randomly inserted by the first two rules, leading to a string from $p \cup p' \cup \alpha X\beta$. However, the third rule of $p1$ is applicable only when $p$ is inserted before the nonterminal $X$. This shows that $zX\beta \Rightarrow_{p1} \gamma_1$ is possible if and only if $\gamma_1 = zp'p\beta$.

Now, on applying matrix $p2$, $p''$ and $p'''$ are inserted anywhere, so immediately we arrive at a string from $p'' \cup p''' \cup \alpha zp'p\beta$. Then, $p'$ is deleted in the left context of $p'' p'''$. So, we now arrive at the string $zp'' p''' p\beta$. This shows that $\gamma_1 = zp'p\beta \Rightarrow_{p2} \gamma_2$ is possible if and only if $\gamma_2 = zp'' p''' p\beta$. We now apply matrix $p3$. Hence, $b$ is first inserted randomly, leading to a string from $b \cup \alpha zp'' p''' p\beta$. The left context in the second rule of $p3$, enforces that, inevitably, we arrive at $\gamma_3 = zp'' b p\beta$. Finally, we apply matrix $p4$. Here, $Y$ is inserted anywhere by the first rule, but the second one enforces that we now look at $zp'' b Y \beta$, which yields $\gamma_4 = zp'' b Y \beta$ by the last rule. As $G$ is in tSGNF, we also know that $Y \in N''$. This shows that $\gamma_3 = zp'' b p\beta \Rightarrow_{p4} \gamma_4$ is possible if and only if $\gamma_4 = zp'' b Y \beta$. The intended sequence of derivations is hence:

$$zX\beta \Rightarrow_{p1} zp'p\beta \Rightarrow_{p2} zp''p'p\beta \Rightarrow_{p3} zp''b\beta \Rightarrow_{p4} zp''bY\beta.$$

This completes the simulation of rule $p$. Simulation of $q : X \rightarrow Yb$ is similar and symmetric to the working of the $p$ rule simulation and hence omitted.

**Simulation of $f : AB \rightarrow \lambda$ or $g : CD \rightarrow \lambda$:** consider the sentential form $zAB\beta$ derivable in $G$. This means that we are in Phase II. As said above, the symbol $Z$ will be present in the corresponding sentential form derivable in $\Gamma$. Any string from $Z \cup \alpha AB\beta$ can be transformed into $zAB\beta$ by using matrix move-$Z$. Now, $\alpha zAB\beta \Rightarrow_{f1} \alpha Z\beta$ correctly simulates one application of $f$.

Now, we prove $L(\Gamma) \subseteq L(G)$. Formally, this is an inductive argument that proves the following properties of a string $w \in \nu^*$ such that $S \Rightarrow^* w$ in $\Gamma$:

1. At most one symbol from $N^{(0)} \cup N'$ is occurring in $w$.
2. If one symbol $X$ from $N^{(0)}$ occurs in $w$, then $w = zXu$, where $x \in \{A, C\}^*$ and $u \in T^*$: $w$ is derivable in $G$;
3. If one symbol $X$ from $N'$ occurs in $w$, then $w = zXfu$, where $x \in \{A, C\}^*$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $w$ is derivable in $G$;
4. If no symbol from $N^{(0)} \cup N'$ occurs in $w$, then $Z$ occurs at most once in $w$.
5. If no symbol from $N^{(0)} \cup N' \cup \{Z\}$ occurs in $w$, then

   (a) either $w = zr'fu$, where $x \in \{A, C\}^*$, $r$ is some context-free rule from $G$ with left-hand side $X$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $zXfu$ is derivable in $G$;
   (b) or $w = zp''p'fu$, where $x \in \{A, C\}^*$, $p \in P_{sl}$ with $p : X \rightarrow bY$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $zXfu$ is derivable in $G$;

$\textbf{Fig. 1 Matrices of size (3; 1, 0, 0; 1, 2, 0) for simulating the context-free rules of tSGNF}$
Consider any sentential form \(w\) that satisfies 5(c). Hence, \(w = zp''bp\beta u\), where \(z \in \{A, C\}^*, p \in P_d\) with \(p : X \rightarrow bY, \beta \in \{B, D\}^*\) and \(u \in T^*\): \(zX\beta u\) is derivable in \(G\).

(d) \(w \in (\{A, B, C, D\} \cup T)^*\): \(w\) is derivable in \(G\).

6. If \(Z\) occurs in \(w\), then \(w \in Z \bigvee \alpha \beta u\), where \(\alpha \in \{A, C\}^*, \beta \in \{B, D\}^*\) and \(u \in T^*\): \(z\alpha \beta u\) is derivable in \(G\).

These properties are surely true at the very beginning, as the sentential form \(S\) satisfies 2. We are discussing the induction step in what follows.

Conditions 2 and 3 consider any sentential form \(w\) that satisfies 2 or 3. Hence, \(w = zX\beta u\), where \(z \in \{A, C\}^*, \beta \in \{B, D\}^*\) and \(u \in T^*\). In fact, the conditions are more specific about some details regarding \(X\) and \(\beta\). All rules but \(r1\) (for some context-free rule \(r\) or \(h1\) require the presence of some rule marker (of or \(Z\)) and are hence not applicable.

- If matrix \(r1\) is applied, then \(w' = \alpha' \beta' \gamma u\) results, satisfying 5(a). The left-hand side of rule \(r\) must have been \(X\). By induction, \(zX\beta u\) is derivable in \(G\).
- If \(h1\) is applied, then \(w'\) results, with \(w' \in Z \bigvee \alpha \beta u\), satisfying 6. Matrix \(h1\) checks the presence of \(X = S\). By induction, \(z\alpha \beta u\) is derivable in \(G\), so that applying \(h1\) corresponds to an application of \(h\) on \(z\alpha \beta u\) in \(G\).

Condition 5(a) Consider any sentential form \(w\) that satisfies 5(a). Hence, \(w = \alpha' \beta' \gamma u\), where \(z \in \{A, C\}^*, \beta \in \{B, D\}^*\) and \(u \in T^*\). Moreover, \(zX\beta u\) is derivable in \(G\), where \(X\) is the left-hand side of rule \(r\). As no other rule markers are present in \(w\), only matrix \(r1\) is applicable. Now, we have to distinguish between \(r = p \in P_d\) and \(r = q \in P_i\). In the first case, we inevitably derive \(w' = zp''p' \beta u\) [see 5(b)], and in the second case, we arrive at \(w' = zaq'q'' \beta u\) [see 5(d)].

In both cases, by induction hypothesis, the stated derivability conditions on \(G\) are true.

Condition 5(b) Consider any sentential form \(w\) that satisfies 5(b). Hence, \(w = zp''bp\beta u\), where \(z \in \{A, C\}^*, p \in P_d\) with \(p : X \rightarrow bY, \beta \in \{B, D\}^*\) and \(u \in T^*\): \(zX\beta u\) is derivable in \(G\). The only matrix that can cope with the presence of the rules markers \(p\) and \(p''\) (and only these) is \(p3\). Now, inevitably, \(w \Rightarrow p3 w' = zbY\beta u\), bringing us into Case 2 or 3, as \(G\) can derive this string from \(zX\beta u\) (induction hypothesis) by applying \(p\).

Condition 5(c) Consider any sentential form \(w\) that satisfies 5(c). Hence, \(w = zp''bp\beta u\), where \(z \in \{A, C\}^*, p \in P_d\) with \(p : X \rightarrow bY, \beta \in \{B, D\}^*\) and \(u \in T^*\): \(zX\beta u\) is derivable in \(G\). The presence of the rules markers \(p\) and \(p''\) enforces us to apply matrix \(p4\). Now, inevitably, \(w \Rightarrow p4 w' = zbY\beta u\), bringing us into Cases 2 or 3, as \(G\) can derive this string from \(zX\beta u\) (induction hypothesis) by applying \(p\).

Condition 5(d) As the reader may check, no matrix from \(\Gamma\) is applicable in such a situation.

Condition 6 Consider any sentential form \(w\) that satisfies 6. The only applicable matrices are move-\(Z\), del-\(Z\), fl1, g1. If we apply matrix move-\(Z\), this brings us back to another situation of Case 6, whose claims readily hold by induction. A similar easy case is the application of del-\(Z\), which leads us into Case 5(d). This is particularly interesting when \(w \in Z \bigvee T\), as now a terminal string was derived in \(\Gamma\) that is also generated by \(G\). Matrix fl1 can only be applied if \(Z\) sits immediately to the left of the substring \(AB\). The effect of applying fl1 corresponds to deleting this substring and hence to applying the rule \(f\) in \(G\). This brings us back to Case 6. A similar argument holds for applying matrix g1.

These considerations complete the proof due to Condition 5(d) that applies to \(w \in T^*\). The second equality follows by Proposition 2. □

It is shown that MAT(3; 1, 2, 0; 1, 0, 0) = MAT(3; 1, 0, 2; 1, 0, 0) = RE in Fernau et al. (2019), which is the conference version of this paper. We now improve the matrix length from 3 to 2 in the following theorem.

Theorem 3 MAT(2; 1, 2, 0; 1, 0, 0) = MAT(2; 1, 0, 2; 1, 0, 0) = RE.

Before we begin our proof, we highlight the key feature of the markers first. In order to simulate, say, \(AB \rightarrow \lambda\), we have to use deletion rules \(\langle \lambda, A, \lambda \rangle_{del}\) and \(\langle \lambda, B, \lambda \rangle_{del}\), as deletions cannot be performed under contexts. However, there is the danger that we are deleting unintended occurrences. So we have to carefully place markers before, after and between the chosen nonterminals \(A\) and \(B\) in order to check that they are neighbored. Also, the auxiliary nonterminal \(S\), which is present in the axiom itself, serves as a
Proof Consider a type-0 grammar $G = (N, T, P, S)$ in tsSGNF, with the rules uniquely labelled with $P_H \cup P_d$. Recall the decomposition $N = N^{(0)} \cup N' \cup N''$ by tsSGNF. We can construct a matrix ins-del system $\gamma = (V, T, \{SS\}, M)$ with alphabet $V = N \cup T \cup P_d \cup P_H \cup K$ where $K$ is the following set of markers:

(i) $h_1 = [(\lambda, S', \lambda)_{\text{del}}]$ to simulate the phase transition $h : S' \rightarrow \lambda$.

We now proceed to prove that $L(\Gamma) = L(G)$, starting with the inclusion $L(G) \subseteq L(\Gamma)$. Consider the string $x\lambda\beta$ derivable from $S$ in $G$, with $X \in N^{(0)} \cup N'$ and $x \in \{A, C\}^*$, $\beta \in \{B, D\}^*T^*$ according to Remark 1. We now show that on applying the matrices of Fig. 3a (introduced for simulating rules from $P_d$ of the type $X \rightarrow Yb$), we can derive some string $w^8 \in S \sqcup \alpha b Y \beta$ within $\Gamma$, starting from some $w^8 \in S \sqcup w^8$ simulating $w = x\lambda\beta \Rightarrow z b Y \beta = w'$ in $G$.

$$
S \sqcup \alpha X \beta \ni w^8 \Rightarrow p_1 \alpha X p_\beta \Rightarrow p_2 \alpha p \beta \Rightarrow p_3 \alpha p'' p' \beta \Rightarrow p_4 \alpha p'' p' b p'' \beta
$$

Quite similarly, applying the matrices of Fig. 3b (introduced for simulating rules from $P_H$ of the type $X \rightarrow Yb$), we can derive some string $w^8 \in S \sqcup \alpha Y \beta$ within $\Gamma$, starting from some $w^8 \in S \sqcup w^8$ simulating $w = x\lambda \beta \Rightarrow z b Y \beta = w'$ in $G$.

$$
S \sqcup \alpha X \beta \ni w^8 \Rightarrow q_1 \alpha X q_\beta \Rightarrow q_2 \alpha q \beta \Rightarrow q_3 \alpha q'' q' \beta \Rightarrow q_4 \alpha q'' q' Y q'' \beta
$$

The set of matrices $M$ is defined as follows. Rules $p : X \rightarrow Yb \in P_H$ and $q : X \rightarrow Yb \in P_H$ are simulated by the matrices shown in Fig. 3a, b, respectively. We simulate rule $f : AB \rightarrow \lambda$ by the matrices shown in Fig. 4. Rule $g : CD \rightarrow \lambda$ is simulated alike.

Recalling that our axiom is $SS$, we also have two additional matrices (i) $\tau = [(\lambda, S, \lambda)_{\text{del}}]$ for termination and (ii) $h_1 = [(\lambda, S', \lambda)_{\text{del}}]$ to simulate the phase transition $h : S' \rightarrow \lambda$.

Every time a context-free rule is simulated by either $p$ or $q$ rules, the marker $S$ is first deleted and finally re-inserted at a somewhat random position at the end of every simulation. In other words, the $S$ gets shuffled around the string during Phase I. The phase transition rule $h : S' \rightarrow \lambda$ is simulated by applying $h_1$, so that we can now speak about Phase II of $G$. As the working of the $g$ rule
simulation is similar to the working of the \( f \) rule simulation, we discuss only rule \( f \) below. To actually produce a terminal word, \( \Gamma \) has to apply \( \tau \) at the very end.

We now discuss Phase II in detail, focussing on \( f: AB \rightarrow \lambda \). Let \( w = \alpha \beta \delta \) be a sentential form derivable in \( G \), with \( A, B \in N'' \) and \( \alpha \in \{A, C\}^* \), \( \beta \in \{B, D\}^*T^* \), ensured by tsSGNF. This means that \( w^v \in \Sigma \cup \Sigma \) is derivable in \( \Gamma \) (by induction). We can now see that due to their similarity to the matrices simulating the \( f \) rules. Also, the matrices simulating the \( p \) rules and the \( q \) rules are very similar, so that we only discuss the first ones.

If we apply \( f1 \), this blocks any other simulation branch, as the marker \( S \) is deleted and the marker \( f \) is inserted, but no other marker is present in the string \( w \) that can be thought of being produced from \( w \) by randomly inserting some \( f \). As can be checked case-by-case, the only appli-

\[
\begin{align*}
w^v & \Rightarrow f_1 \alpha fA\beta \Rightarrow f_2 \alpha f' A\alpha' B\beta \Rightarrow f_3 \alpha f' A f'' Bf^5 \beta \\
& \Rightarrow f_4 \alpha f' f'' f^5 Bf^5 \beta \Rightarrow f_5 \alpha f' f^5 Bf^5 \beta \Rightarrow f_6 \alpha f' f^7 Bf^5 \beta \\
& \Rightarrow f_7 \alpha f' f^7 Bf^4 f^5 \beta \Rightarrow f_8 \alpha f' f^7 Bf^4 f^5 \beta \Rightarrow f_9 \alpha f' f^7 f^9 \beta \Rightarrow f_{10} \alpha f' f^7 f^9 \beta \\
& \Rightarrow f_{11} \alpha f' f^9 f^{10} \beta \Rightarrow f_{12} \alpha f^9 f^{10} \beta \Rightarrow f_{13} w^v \in \Sigma \cup \Sigma \beta .
\end{align*}
\]

The purpose of introducing a \( \$ \) in \( f13 \) is to enable another simulation of \( AB \rightarrow \lambda \) or of \( CD \rightarrow \lambda \). When all occurrences of \( AB \) and \( CD \) are deleted by repeated applications of the matrices designed for simulating the \( f \) and \( g \) rules, there is still a \( \$ \) at the end of every simulation. This \( \$ \) is deleted by applying rule \( \tau \), thereby terminating Phase II of tsSGNF. Inductively, this shows that \( L(G) \subseteq L(\Gamma) \).

Let us now prove the converse direction. Consider once more a string \( w = \alpha \beta \delta \) derivable from \( S \) in \( G \), with \( \alpha \in N''(0) \cup N'' \) and \( \alpha \in \{A, C\}^* \), \( \beta \in \{B, D\}^*T^* \) according to Remark 1. We know by our previous arguments that the variation \( w^v \in \Sigma \cup \Sigma \) can be derived in \( \Gamma \). In particular, the axiom fits into this description. We will now argue that, starting with such a string, we either terminate the first phase of the simulation by applying \( h1 \), which obviously corresponds to applying \( S' \rightarrow \lambda \) in \( G \), or we try to apply any of the other matrices. We will then show that any non-blocked derivation will again lead to a string of the form \( \$ \cup (A, C)^* (N''(0) \cup N'') (B, D)^*T^* \), which justifies our starting point inductively. As all matrices of the form p2, ..., p7 or q2, ..., q7 or f2, ..., f13 require the presence of a marker symbol from \( K \setminus \{\$\} \), we can focus on applying the matrices p1, q1 or f1. Here, we ignore the rules g1, ..., g13 due to their similarity to the matrices simulating the \( f \) rules.

\[
\begin{align*}
f1 &= ([\lambda, S, \lambda]_{det}, [\lambda, f, \lambda]_{ins}) \\
f2 &= ([fA, f'' B, \lambda]_{ins}, [f, f'' \lambda]_{ins}) \\
f3 &= ([f'' B, f'', \lambda]_{ins}, [f'' f^5, \lambda]_{ins}) \\
f4 &= ([\lambda, A, \lambda]_{det}, [\lambda f'' f^5, \lambda]_{ins}) \\
f5 &= ([\lambda, f, \lambda]_{det}, [\lambda f'' f^5, \lambda]_{det}) \\
f6 &= ([f', f^5, \lambda]_{ins}, [f', f^5, \lambda]_{ins}) \\
f7 &= ([f'' B, f', \lambda]_{det}, [f', f^5, \lambda]_{det}) \\
f8 &= ([\lambda, B, \lambda]_{det}, [f', f^5, f^8, \lambda]_{ins}) \\
f9 &= ([\lambda, f, \lambda]_{det}, [\lambda f', \lambda]_{det}) \\
f10 &= ([f', f^5, f^8, \lambda]_{ins}, [\lambda f', \lambda]_{ins}) \\
f11 &= ([f', f^5, f^{10}, \lambda]_{ins}, [\lambda f', \lambda]_{ins}) \\
f12 &= ([\lambda, f', \lambda]_{det}, [\lambda f', \lambda]_{det}) \\
f13 &= ([\lambda, f^{10}, \lambda]_{det}, [\lambda S, \lambda]_{ins})
\end{align*}
\]

Fig. 4 Simulating non-context-free rules by matrix rules of size (2; 1, 2, 0; 1, 0, 0)
Remark 1. Notice that by the discussions performed so far, there is one last catch in this simulation. Assuming that $w = zA\zeta B$ for some $z \in \{A, C\}^*$, $\zeta \in \{B, D\}$ and $\beta \in \{B, D\}^*T^*$, or $\zeta \in T$ and $\beta \in T^*$, or $w = zA$, then a sentential form derivable in $G$, corresponding to some string $w^8$ derivable in $G$. The only applicable rule is $f_1$ (or $g_1$, discussed at the end), since other matrices demand the presence of a marker (say, either $f$ or $f''$). Also, no rule from the simulation of Phase I is applicable anymore due to the absence of symbols from $N^{(0)} \cup N'$. Recall that we ignore the possible presence of symbols like $f'$ as discussed above. Also, by the very structure of the matrices simulating an $f$ rule, no progress on a sentential form $f$ with $\beta \in \{B, D\}^*T^*$ is observed beyond the possible application of $f_1$. One application of $f_1$ on $w$ deletes the symbol $\$ that was present in the axiom and introduces a marker $f$ randomly, hence leading to a string $w_1$ from $f \bot w$. All matrices simulating the $f$ rule require the presence of (multiply) primed versions of the marker $f$, apart from $f_2$ which has to be applied next. Matrix $f_2$ checks that an $A$ must be immediately to the right of $f$, which means that $f$ has been previously inserted within the prefix $zA$ of $w$. Hence, the resulting string $w_2$ can be (equivalently) obtained from $w$ by first picking one occurrence of $A$ within the prefix $zA$ and then inserting the string $f''$ immediately to the right of this $A$-occurrence and $f'$ immediately to the right of marker $f$. The introduction of $f'$ between $f$ and $A$ spoils the pattern $fA$, thus $f_2$ cannot be applied again. This leads to the string $z'f'f''f'''f''''f'''f''f''''$ with $zA = z'z''z''z'''z$ for some $z''$. All matrices $f_i$ but $f_3$ require at least one of the substrings $\$, $fA$, $f''''$ or $f''$ ($3 \leq k \leq 10$) to be present, which renders them inapplicable. However, matrix $f_3$ also checks that the $A$-occurrence that we picked within $w$ sees a $B$-occurrence to the immediate right of $Af'''$. This enforces $\zeta = B$ in our string $w = zA\zeta B$ and $z' = z''z'''$ (all other strings are stuck at this point). Therefore, after applying matrix $f_3$, we arrive at the string $w_3 = z'Af'''f''f$$f'Bf'B$. Now, none of the matrices $f_5 - f_{13}$ are applicable due to the absence of one of the markers $f', f''', f''', f''', f''$, $f^{10}$. The matrices $f_1, f_2, f_3$ are also inapplicable due to the absence of $\$, $fA$, and $f'B$, respectively. Hence, the only applicable rule is $f_4$. The first rule in $f_4$ demands the deletion of one occurrence of $A$. Initially, it might be tempting to delete some other unintended $A$-occurrence. However, only if the intended $A$-occurrence (part of the substring $f'f''f''f'f'''$) is deleted, then we will have the substring $f'f'''$ in order to apply the second rule of $f_4$. Hence, we get $w_4 = z'Af'''f''f'Bf'B$. Again, matrices $f_1 - f_3, f_6 - f_{13}$ still remain inapplicable due to the same
reason as above. The only applicable matrices on $w_4$ are $f/4$ (again) or $f/5$.

Suppose $f/4$ is applied again (repeatedly for $k \geq 0$ times) on $w_4$, then some $k$ number of As in $z$ are deleted freely and $(f^6)^k$ is inserted after $f^k f''$, leading to a string $w'_4 = z' f^k f'' (f^6)^k f^3 B f^5 \beta$, where $z'$ is obtained from $z$ by deleting $k$ occurrences of $A$. For further continuation of the derivation, the only other applicable matrix is $f/5$. Since there is a unique occurrence of $f''$, the matrix $f/5$ can be applied only once, which will delete only one $f''$ leading to the string $w''_4 = z' f^k f'' (f^6)^k f^3 B f^5 \beta$. Assuming $f/4$ is applied no more, no other matrix (including $f/6$), is applicable due to the absence of one of $S, f A, f'' B, f'' f^4, f^7, f^9, f^{10}$. Hence, the derivation is stuck here if $k > 0$. We hence apply matrix $f/5$ on $w_4 = z' f^k f'' (f^6)^k f^3 B f^5 \beta$ to get $w_5 = z' f^k f^3 B f^5 \beta$.

Due to the absence of one of the markers or substrings $S, f A, f'' B, f'' f^4, f^7, f^9, f^{10}$, no matrix other than $f/6$ is applicable. Applying $f/6$ on $w_5$ yields $w_6 = z' f^k f^3 B f^5 \beta$. The absence of $S, f A, f'' B, f'' f^4, f^7, f^9, f^{10}$ blocks the applicability of all $f$ matrices except $f/7$. Applying $f/7$ on $w_6$ leads to the string $w_7 = z' f^k f^3 B f^5 \beta$ as intended. Matrices $f/1 - f/7$ are not applicable, since the markers $S, f, f'' B, f'' f^3$ are not present in $w_7$. Similarly, matrices $f/9, f/11, f/12$ and $f/13$ are not applicable on $w_7$, since the markers $f^9, f^9$ and $f^{10}$ are not present. Additionally, $f/10$ is inapplicable due to the absence of $f^7 f^5$. Thus, the only applicable matrix is $f/8$. The first rule in $f/8$ demands the deletion of one occurrence of $B$. Initially, it might be tempting to delete some other unintended $B$-occurrence. However, only if the indented $B$ is deleted, then we will have the substring $f^7 f^4$, and the second rule of $f/8$ can be applied. Hence, we get $w_8 = z' f^k f^3 f^4 f^8 \beta$. The matrices $f/1 - f/7, f/11 - f/13$ are inapplicable due to the absence of one of the markers $S, f, f'' B, f'' f^3, f^9, f^{10}$. The matrix $f/10$ is inapplicable, since we do not have the context $f^7 f^5$ in $w_8$. The only applicable matrices on $w_8$ are $f/8$ (again) or $f/9$.

Suppose $f/8$ is applied again (repeatedly for $k \geq 0$ times) on $w_8$, then some $k$ number of $B$s in $\beta$ are deleted freely and $(f^8)^k$ is inserted after $f^7 f^4$, leading to a string $w'_8 = z' f^k f^3 f^4 (f^8)^k f^8 \beta$, where $\beta' = \beta$ without some $k$ occurrences of $B$. For further continuation of the derivation, the only other applicable matrix is $f/9$. Since there is a unique occurrence of $f^4$, the matrix $f/9$ can be applied only once, which will delete only one $f^8$, leading to the string $w''_8 = z' f^k f^3 f^4 (f^8)^k f^8 \beta$. Due to the absence of any of $S, f, f'' B, f'' f^4, f^9, f^{10}$, no matrix is applicable on $w'_8$ but $f/10$, but this requires the presence of the substring $f^7 f^5$. This is only possible of $k = 0$ (as intended), i.e., if $w_8 = w'_8$ and $w_9 = w''_8 = z' f^k f^3 f^5 \beta$.

The matrices $f/1 - f/9, f/11 - f/13$ are clearly inapplicable on $w_9$ due to the absence of the markers $S, f, f'', f^4, f^9, f^{10}$. So we could now apply $f/10$ on $w_9$, then we will have $w_{10} = z' f^k f^3 \beta$. Due to absence of essential markers, all matrices except $f/11$ and $f/12$ are inapplicable. If the latter is applied, then we get $w'_{10} = z' f^k \beta$ and the derivation is stuck as no continuation is possible. In particular, we cannot apply matrix $f/7$ due to the absence of $f$, a marker that we cannot introduce by applying $f/1$ due to the absence of the marker $S$. So, $f/11$ is applied onto $w_{10}$ and we get $w_{11} = z' f^k f^{9/10} \beta$. Now, the only applicable matrices are $f/12$ and $f/13$. Both the rules can be applied in any order (independently) to get $w'' = \xi f^k \beta$. Observe that there are also scenarios where only $f/13$ is applied, so that we stick with $w'' = \xi f^k \beta$. Now, we can indeed start a new cycle of simulation with $f/1$ and $f/2$, but then at latest, in order to find the substring $f'' B$ as required by $f/3$, we should apply $f/12$ to connect $x$ and $\beta$.

Let us finally return to the situation when we try to apply $g/1$ on $w_8$, with $w = \alpha A \beta$ for some $x \in \{A, C\}$, $\zeta \in \{B, D\}$ and $\beta \in \{B, D\}^* T^* \zeta$, or $\zeta \in T$ and $\beta \in T^*$, or $w = \alpha A$ (i.e., $\zeta = \beta = \lambda$), being a sentential form derivable in $G$, corresponding to some string $w_8$ derivable in $G$. If $w = g_1 w'_1$, then $w'_1 \in \{g \cup T\}$. If we now apply $g_2$, this means that we have split $x$ as $x' x''$, so that for $w_1 \Rightarrow g_2 w''$, $w'' = x' g_1^g C g'' x'' A \zeta \beta$, with $\zeta$ and $\beta$ satisfying the conditions stated above. However, now the derivation is stuck, as the string $w''$ does not contain a substring $CD$ as required by matrix $g/3$.

These arguments show that also the non-context-free rules are correctly simulated and hence the whole simulation is correct, as also no successful re-starts of simulations are possible on strings from $\{g \cup T\}$.

The second claimed computational completeness result follows by Proposition 2 and this concludes the proof. □

In the previous two theorems, the maximum length of the insertion/deletion context was two and the other operation, namely deletion/insertion is done in a context-free manner. If we restrict the parameters in the size to be binary (0 or 1), then we achieve computational completeness using matrices of maximum length two; however, insertion is now performed under a 2-sided context.

Theorem 4 MAT(2; 1, 1, 1; 1, 0, 0) = RE.

Proof Consider a type-0 grammar $G = (N, T, P, S)$ in tsSGNF. The rules from $P$ are supposed to be labelled injectively with labels from the set $[1, \ldots |P|]$, with label sets $P_{it}$ and $P_{rt}$ as defined above. Also recall that the nonterminal alphabet decomposes as $N = N^{(0)} \cup N' \cup N''$, $N'' = \{A, B, C, D\}$, $S \in N^{(0)}, S' \in N'$, according to the normal form. We construct a matrix insertion-deletion
Fig. 5 Matrices of size 
\( (2; 1, 1; 1, 0, 0) \) for
simulating context-free rules of

tsSGNF

\[
\begin{align*}
\text{p1} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, p, \lambda)_{\text{ins}}) \\
\text{p2} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, p', \lambda)_{\text{ins}}) \\
\text{p3} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, p', \lambda)_{\text{del}}) \\
\text{p4} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, p', \lambda)_{\text{ins}}) \\
\text{p5} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, p, \lambda)_{\text{del}}) \\
\text{q1} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, q, \lambda)_{\text{ins}}) \\
\text{q2} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, q, \lambda)_{\text{del}}) \\
\text{q3} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, q, \lambda)_{\text{del}}) \\
\text{q4} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, q, \lambda)_{\text{ins}}) \\
\text{q5} & = ([\lambda, \$, \lambda]_{\text{del}}, (X, q, \lambda)_{\text{ins}})
\end{align*}
\]

(a) Simulating \( p : X \rightarrow bY \)
(b) Simulating \( q : X \rightarrow bY \)

Fig. 6 Matrices with context-free deletion simulating \( f : AB \rightarrow \lambda \) and \( g : CD \rightarrow \lambda \)

\[
\begin{align*}
f1 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f2 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f3 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f4 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f5 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f6 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f7 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f8 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f9 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}}) \\
f10 & = ([\lambda, \$, \lambda]_{\text{del}}, (X, f, \lambda)_{\text{ins}})
\end{align*}
\]

\[ g1 = ([\lambda, \$, \lambda]_{\text{del}}, (X, q, \lambda)_{\text{ins}}) \]

The system \( \Gamma = (V, T, \{\$\}, M) \) with alphabet set

\[ V = N \cup T \cup K, \]

where

\[ K = \{p, p', q | p \in P_\lambda \} \cup \{q, q' | q \in P_\lambda \} \cup \{\$\} \]

\[ \cup \{f, f', f''; f^3, f^4, f^5, g, g', g'', g^3, g^4, g^5\} \]

The set of matrices \( M \) of \( \Gamma \) consists of the matrices described in Fig. 5a, b and in Fig. 6.

Specifically, we simulate a rule \( p : X \rightarrow bY, X \neq Y, X, Y \in N^0 \cup N', b \in N'', \) i.e., \( p \in P_\lambda \) by the matrices displayed on the left-hand side of Fig. 5a. Similarly, we simulate a rule \( q : X \rightarrow bY, X \neq Y, X, Y \in N^0 \cup N', b \in N'' \cup T, \) i.e., \( q \in P_\lambda \) by the matrices shown on the right-hand side of Fig. 5b. We simulate rule \( f : AB \rightarrow \lambda \) by the matrices shown in Fig. 6. The simulation of rule \( g : CD \rightarrow \lambda \) is done in a similar manner.

Recalling that our axiom is \( \$S \), we also have two additional matrices (i) \( \tau = ([\lambda, \$, \lambda]_{\text{del}}) \) for termination and (ii) \( h1 = ([\lambda, \$, \lambda]_{\text{del}}) \) to simulate the phase transition \( h : S' \rightarrow \lambda \).

We now proceed to prove that \( L(\Gamma) = L(G) \), starting with the inclusion \( L(G) \subseteq L(\Gamma) \). Consider the string \( w^8 \in S \cup \alphaX^\beta \) derivable from \( SS \) in \( G \), with \( X \in N^0 \cup N' \) and \( \alpha \in \{A, C\}^+, \beta \in \{B, D\}^+T^* \) according to Remark 1. We apply matrix \( p1 \), which deletes the only hanging marker \( \$ \) and then inserts \( p \) to the right of \( X \), leading to \( w1 = \alpha X^\beta \). On applying matrix \( p2 \), which inserts a \( p' \) between \( X \) and \( p \) and then deletes the nonterminal \( X \), we have \( w2 = \alpha X^\beta p \). Applying matrices \( p3, p4, p5 \) in the specified order, we have the following:

\[ w^8 \Rightarrow p1 \alpha X^\beta \Rightarrow p2 \alpha X^\beta p \Rightarrow p3 \alpha X^\beta p \Rightarrow p4 \alpha X^\beta p \Rightarrow p5 \alpha X^\beta p \]

We now consider the converse direction. Consider a string \( w = \alpha X^\beta \) derivable from \( S \) in \( G \), with \( X \in N^0 \cup N' \)

The purpose of introducing \( \$ \) in the last rule \( f10 \) or \( g10 \) is to enable another simulation of \( AB \rightarrow \lambda \) or of \( CD \rightarrow \lambda \). When all occurrences of \( AB \) and \( CD \) are deleted by repeated applications of the simulations of the \( f \) and \( g \) rules, there is still a \( \$ \) at the end of every simulation. This \( \$ \) is deleted by applying rule \( \tau \), thereby terminating Phase II of \( \text{tsSGNF} \). Inductively, this shows that \( L(G) \subseteq L(\Gamma) \).

Let us now consider the converse direction. Consider a string \( w = \alpha X^\beta \) derivable from \( S \) in \( G \), with \( X \in N^0 \cup N' \)
and $x \in \{A,C\}^*$, $\beta \in \{B,D\}^*$ $T^*$ according to Remark 1. We know by our previous arguments that the variation $w^8 \in S \cup \{w\}$ can be derived in $\Gamma$. We will now argue that, starting with such a string, we either terminate the first phase of the simulation by applying $h_1$, which obviously corresponds to applying $S' \rightarrow \lambda$ in $G$, or we try to apply any of the other matrices. As all matrices of the form $p2$, ..., $p5$ or $q2$, ..., $q5$ or $f2$, ..., $f10$ require the presence of a marker symbol from $K \setminus \{\}$, we can focus on applying $p1$, $q1$ or $f1$. Here, we ignore the matrices $g1$, ..., $g10$ due to their similarity to the $f$ matrices. Also, the simulations of the $p$ rules and $g$ rules are very similar, so that we only discuss the first ones.

We now start to apply $p1$ on $w^8$ which deletes the marker $\$ and then inserts a $p$ to the right of $X$. Hence, $w^8 \Rightarrow p1 x X p B \$. The absence of $x, p', Y$ blocks the application of rules $p1$, $p3$, $p4$, $p5$. Deterministically, the next matrix application is $p2$ on $w_1$ which inserts $p'$ after $X$ and then deletes the $X$ to yield $w_2 = x X p B \$. It is to be noted that to the left of $p$, the marker $p'$ is present and therefore, matrix $p4$ cannot be applied. Since there is no $X$, the matrices $p1$ and $p2$ cannot be applied. Further, the absence of $b Y p$ prevents the application of the rule of $p5$. We must apply matrix $p3$ now, introducing a $b$ between $p'$ and $p$ and then deleting the marker $p'$. This corresponds to applying the rewriting rule $p' \rightarrow b \rightarrow w_2$, yielding $w_3 = z x p B \$. The absence of $x, X, p', Y$ forces us to apply the only applicable matrix $p4$ on $w_3$ that inserts a $Y$ before $p$, yielding $w_4 = z b Y p B \$. The only applicable matrix is now $p5$ which inserts a $\$ between the recently introduced $b Y$ and deletes the marker $p$, yielding $w_5 = b \$ Y $.

Now, assume that Phase I was simulated as intended. This means that we consider some string $w$ that is derivable in Phase I by $G$ as well as $w^8 \in S \cup \{w\}$ by $\Gamma$. As we are using tsSGNF, this means that $w \in \{A,C\}^* \{B,D\}^* T^*$ by Remark 1.

First, assume that $w = z A \zeta B \$ for some $z \in \{A,C\}^*$ and $\beta \in \{B,D\}^* T^*$, or $\zeta \in T$ and $\beta \in T^*$, or $w = z A \$ is a sentential form derivable in $G$, corresponding to some string $w^8$ derivable in $\Gamma$. The only applicable rule is $f1$ (or $g1$, discussed at the end), since other matrices demand the presence of a marker (say, either $f$ or $f^\prime$). Also, no rule from the simulation of Phase I is applicable anymore due to the absence of symbols from $N^0 \cup N^\prime$. Also, by the very structure of the matrices simulating an $f$ rule, no progress on a sentential form $\beta$ with $\beta \in \{B,D\}^* T^*$ is observed beyond the possible application of $f1$. On applying $f1$ on $w$ deletes the symbol $\$ that was present in the axiom and introduces a marker $f$ randomly, hence leading to a string $w_1$ from $f \cup \{w\}$. Application of $f1$ again is not possible due to the absence of $\$. Apart from the matrices $f2$ and $f3$, all other matrices simulating the $f$ rule require the presence of (multiply) primed versions of the marker $f$ and therefore they cannot be applied.

Assume first we had applied matrix $f3$ on $w_1$. The obtained string $w'$ can be obtained from $w$ by inserting a $f''$ between $AB$ (which must form the central part). Since $f'$ is missing, $f4$ cannot be applied. But $f'$ is introduced by $f2$ only, which supposes $f$ to be presented, whose introduction assumes the presence of $\$. Can we get rid of $f''$ again? In order to do so, we must apply matrix $f6$, which assumes the presence of $f^\prime$. However, in order to introduce $f4$, we must apply matrix $f4$, which is impossible. Hence, the derivation is stuck.

So, to proceed further, one has to apply matrix $f2$ on $w_1$. The first rule in matrix $f2$ checks that an $A$ must be immediately to the right of $f$, which means that the previously inserted $f$ has been introduced within the prefix $z A$ of $w_1$, thus verifying the decomposition $w_1 = z A x A \zeta B$, with $z A = z A x A \zeta B$. A new marker $f'$ is introduced between $f$ and $A$ by the first rule of matrix $f2$. The second rule in $f2$ introduces the marker $f''$ after any $B$, thus obtaining $w_2 = z A x A \zeta B f B^3 \zeta B \$, with $\zeta \beta = B^3 \zeta B \$. The same matrix cannot be applied again, as the substring $A f$ is no longer present in the derived string. Now, only two matrices, namely $f3$ and $f8$, are applicable to $w_2$, since other matrices require markers which are not introduced yet. Applying $f8$ makes no sense, as the just introduced markers $f'$ and $f^\prime$ are deleted. Alternatively, the matrix $f3$ introduces the marker $f'''$ between $A$ and $B$, this ensures that the $B$ must be immediately to the right of $A$ in $w_2$, which enforces $\zeta = B$ in $w$ (also enforces that in $w_2$, $z A$ must end with $A$ and $\beta$ must start with a $B$) and also deletes the marker $f$. Notice that this introduction of $f'''$ also ensures that the central part of $w$ was properly formed, avoiding mismatches like $A D$.

Hence, we have $w_3 = z A x A \zeta B f B f f'' f'' B f B^3 \zeta B \$. The absence of the contexts $\$, $A$, $A B$, and $f^i$ for $i \geq 4$ blocks the application of matrices $f1 - f10$ except $f4, f5$ and $f8$. If we apply matrix $f8$ now, the derivation is stuck, as now also $f4$ and $f5$ are inapplicable. A similar problem appears if we apply first $f4$ or $f5$ and then $f8$. The matrix $f4$ (applied on $w_3$) demands that if a $A$ is deleted randomly, then we should get the substring $f' f''$ in the sentential form in order to apply the second rule of $f4$ and this is possible only if $z A x A \zeta B f B f f'' f'' B f B^3 \zeta B \$ in $w_3$, i.e., if $z A = \$. Similarly, the matrix $f5$ demands that if a $B$ is deleted randomly, then we should get the substring $f f f f f f f f''$ in the sentential form in order to apply the second rule of $f5$ and this is possible only if $f'' f'' f'' f f B f B f B^3 \zeta B \$ in $w_3$, i.e., if $\beta = \$. In summary, we find that necessarily $w_3 = z A x A \zeta B f B^3 \zeta B \$. Now, matrices $f4$ and $f5$ are applied in any order. They basically simulate the rewriting rules $f A \rightarrow f f A$ and $f'' B \rightarrow f'' f f$, so that we get $w_4 = z A x A f A f f f f f f f'' B f B f B^3 \zeta B \$. As the only matrix (namely, $f6$) that requires (apart from an occurrence of $f''$) the presence
of $f^4$ also requires the presence of $f^5$ (and vice versa), we have to apply both matrices $f_4$ and $f_5$ before proceeding.

Note that from now on the matrices $f_4$ or $f_5$ can be re-applied if and only if $f^n f^m$ or $f^n f^5$ is a substring of our sentential form, because deleting $A$ or $B$ cannot produce these substrings that are required by the second rules of the matrices $f_4$ or $f_5$, respectively. We call this observation $ob^*$. Now, what remains is the deletion of the markers that were introduced in this simulation in some order, however taking care that $f^n f^m$ and $f^n f^5$ is never a substring in the resulting sentential form, so as to avoid an unintended (re)application of rules $f_4$, $f_5$ which will delete random occurrences of $A$ or $B$ (see $ob^*$).

This danger is handled by replacing $f^n$ with $f^6$ in matrix $f_6$. One may wonder what if the matrix $f_6$ was never applied to the sentential form $w_4 = x'f'f_d f^n f^5 f^3 f^6 f^4$. Due to $ob^*$, the only matrices that are applicable to $w_4$ are $f_6$ and $f_6$. If $f_6$ was avoided and $f_8$ was applied, then the markers $f'_3$ are deleted, leaving behind $w_5 = x'f'f_d f^5 f^6 f^4$. At this point, no matrix is applicable except $f_6$. Hence the matrix $f_6$ is somehow enforced to be applied on $w_4$ or $w_5$ which replaces the marker $f^n$ as $f^6$. Due to absence of $f^n$ and $ob^*$, matrices $f_4$ and $f_5$ can never be re-applied. If $f^n$ has to be introduced again (using matrix $f_3$), then $AB$ need to be present as substring which is not possible as some markers are present in between them in the derived string $w_4$ or $w_5$. The purpose of matrix $f_7$ is to make sure that $f^4$ and $f^5$ are deleted in different matrices. The markers are deleted using matrices $f_8$ to $f_{10}$ in any desired (applicable) order yielding $w_9 = \Sigma \alpha ' \beta'$. Three such derivations are shown below.

- $w_4 = x'f'f_d f^n f^5 f^3 f^6 f^4 \Rightarrow f_6 x'f'f_d f^n f^5 f^3 f^6 f^4 \Rightarrow f_7 x'f'f_d f^6 f^4 f^5 f^3 f^6 f^4 \Rightarrow f_8 x'f'f_d f^6 f^5 f^3 f^6 f^4 \Rightarrow f_9 x'f'f_d f^5 f^6 f^4 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f^4 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f^3 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f^2 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f^1 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f^0 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_9 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_8 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_7 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_6 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_5 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_4 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_3 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_2 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_1 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d f_0 f^6 f^5 f^3 f^6 f^4 \Rightarrow f_{10} x'f'f_d \Rightarrow f_{10} x'f' \Rightarrow f_{10} x' \Rightarrow \emptyset$.

These arguments show that also the non-context-free rules are correctly simulated and hence completes the proof. □

The reader might think that the markers $f^n, f_7, f^8$ and their relevant insertion and deletion rules are not necessary. One idea would be to construct matrices like $f^7 = [(\lambda, f^3, f_3, \lambda)_{del}(\lambda, f^n, f_3)_{del}]$ and $f^8 = [(\lambda, f^3, f_3, \lambda)_{del}(\lambda, f^8, f_3)_{del}]$. The problem is that however, such ideas might not work as one can start to apply these matrices $f^7$ and $f^8$ soon after applying $f_3$, leading to unintended situations. This would make a correctness proof more tedious if not impossible at all. With the introduction of the markers $f^n, f_7, f^8$, the proof is simplified, to say the least.

### 4 Describing the regular closure of linear languages

We showed in Theorem 4 that matrix ins-del systems of size $(3; 2, 0, 1; 0, 0, 0)$ can describe RE. Further, it is shown in Fernau et al. (2018b) that, if we have a one-sided context for insertion, then matrix ins-del systems of size $(3; 1, 1, 0; 1, 0, 0)$ or $(3; 1, 0, 1; 1, 0, 0)$ and also $(2; 2, 0, 1; 0, 0, 0)$ or $(2; 2, 0, 1; 1, 0, 0)$ can simulate (meta-)linear grammars. However, whether or not one can simulate general context-free grammars with matrix ins-del systems of the above-mentioned sizes is still open. We summarize these results now. Example 1 shows that there are non-meta-linear languages that can be described by these matrix ins-del systems.

**Proposition 4** Fernau et al. (2018b) The following language relations are true.

- $\text{LIN} \subseteq \text{MAT}(3; 1, 1, 0; 1, 0, 0) \cap \text{MAT}(3; 1, 0, 1; 1, 0, 0)$.
- $\text{LIN} \subseteq \text{MAT}(2; 2, 1, 0; 1, 0, 0) \cap \text{MAT}(2; 2, 0, 1; 1, 0, 0)$.

For quick reference, we present the matrix ins-del rules of $\text{MAT}(3; 1, 1, 0; 1, 0, 0)$ that simulates the linear rules $p : X \rightarrow aY, q : X \rightarrow Ya, f : X \rightarrow \lambda$ in Fig. 7a and the matrix ins-del rules of $\text{MAT}(2; 2, 1, 0; 1, 0, 0)$ in Fig. 7b.

Initially, our main objective was to find how much beyond LIN can a matrix ins-del system (of the four sizes stated in Proposition 4) lead us to. However, we then succeeded to provide a general result showing that if there exists a matrix ins-del systems of size $(k; n, i^1, i^2; m, f^n)$ describing LIN, then the same system will describe $L_{reg}(\text{LIN})$.

**Theorem 5** For all integers $n, m \geq 1, \ t \geq 2$ and $i^1, i^2, f^n \geq 0$ with $t + n \geq 4$ and $i^1 + i^2 \geq 1$, if every $L \in \text{LIN}$ can be generated by a $\text{MAT}(t; n, i^1, i^2; m, f^n)$ system

\[ \text{ Springer] \]
with a single axiom that is identical to the start symbol \( S \) of a linear grammar describing \( L \), then \( \mathcal{L}_{\text{reg}}(\text{LIN}) \subseteq \text{MAT}(t; n, i', i''; m, j', j'') \), as well.\(^3\)

**Proof** Let \( L \in \mathcal{L}_{\text{reg}}(\text{LIN}) \) for some \( L \subseteq T^* \). By Proposition 1, we can assume that \( L \) is described by a context-free grammar \( G = (N, T, S, P) \) that basically consists of a right-linear grammar \( G_0 = (N_0, N^\alpha, S, P_0) \) and linear grammars \( G_i = (N_i, T, S_i, P_i) \) for \( 1 \leq i \leq k \). For technical reasons that should become clear soon, we rather consider \( G_i' = (N_i', T, S_i', P_i') \), where \( N_i' = N_i \cup \{(S_i, A) \mid A \in N_0\} \) and \( P_i' \) contains, besides all rules from \( P_i \), rules of the form \( \langle S_i, A \rangle \rightarrow w \) whenever \( S_i \rightarrow w \in P_i \) for some \( w \in (N_i \cup T)^* \). This means, apart from \( L(G_i') = L(G_i) \) (as the new nonterminals will never be used in terminating derivations), that also \( L((N_i', T, (S_i, A), P_i')) = L(G_i) \) for any \( A \in N_0 \).

Since \( \text{LIN} \subseteq \text{MAT}(t; n, i', i''; m, j', j'') \), each \( G_i' \) can be simulated by a matrix ins-del system \( \Gamma_i = (V_i, T, \{(S_i, A) \mid A \in N_0\}, R_i) \) for \( 1 \leq i \leq k \), each of size \( (t; n, i', i''; m, j', j'') \). We assume, without loss of generality, that \( V_i \cap V_j = T \) if \( 1 \leq i \neq j \leq k \). Let us first consider the case \( i' \geq 1 \) and \( i'' = 0 \). We construct a matrix ins-del system \( \Gamma \) for \( G \) as follows:\(^4\): \( \Gamma = (V, T, \{(S_i, A) \mid A \in N_0\}, R \cup R') \), where

\[
V = \bigcup_{i=1}^k (V_i \cup \{(S_i, A) \mid A \in N_0\}) \cup N_0 \cup \{(A') \mid A \in N_0\};
\]

\[
R = \bigcup_{i=1}^k R_i; \quad \text{and for } t \geq 3, R' \text{ is the set } \{m_p \mid p \in P_0\}, \text{where:}
\]

- \( m_p = [(A', (S_i, B), \lambda)_{\text{ins}}, (\langle S_i, B \rangle, B', \lambda)_{\text{del}}, (\lambda, A', \lambda)_{\text{del}}] \text{ if } p = A \rightarrow B \in P_0 \);
- \( m_p = [(\lambda, A', \lambda)_{\text{del}}] \text{ if } p = A \rightarrow \lambda \in P_0 \) (terminating matrix).

For \( t = 2 \) and \( n \geq 2 \), we add the following matrix \( m_p \) instead of the above-defined matrix \( m_p \) into \( R' \): \( m_p = [(A', (S_i, B) B', \lambda)_{\text{ins}}, (\lambda, A', \lambda)_{\text{del}}] \text{ for } p = A \rightarrow B \in P_0 \).

Notice that \( A \neq B \), as we assume that \( G_0 \) is in normal form, which is important for both variants of \( m_p \).

The case when \( i' = 0 \) and \( i'' \geq 1 \) follows from Propositions 2 and 3. \( \square \)

Combining Theorem 5 with results from Fernau et al. (2018b), we have the following corollary. The strictness of the subset relation in the theorem below follows from Example 1.

**Corollary 1** The following assertions are true.

- \( \mathcal{L}_{\text{reg}}(\text{LIN}) \subseteq \text{MAT}(3; 1, 1, 0; 1, 0, 0) \cap \text{MAT}(3; 1, 0, 1, 0, 0) \).
- \( \mathcal{L}_{\text{reg}}(\text{LIN}) \subseteq \text{MAT}(2; 2, 1, 0; 1, 1, 0) \cap \text{MAT}(2; 1, 0, 1; 1, 0, 0) \). \( \square \)

5 Describing regular languages

Pete and Verlan (2012), matrices of maximum length 3 and size \((1, 1, 0; 1, 1, 0)\) were used to describe RE. By Krassovitskiy et al. (2011), ins-del systems of size \((1, 1, 0; 1, 1, 0)\) do not achieve computational completeness. However, it is strictly contained in the class in the context-free languages [see Krassovitskiy et al. (2011)]. This corresponds to matrix length one. In the following, we attempt to study the generative power of \((1, 1, 0; 1, 1, 0)\) with matrix length 2. With rules of size \((2; 1, 1, 0; 1, 1, 0)\), we simulate the right-linear rules \( p : X \rightarrow bY \) and \( X \rightarrow \lambda \). The non-context rules of tS González namely \( AB \rightarrow \lambda \) and \( CD \rightarrow \lambda \) can be simulated by rules of \((2; 1, 0, 0; 1, 1, 0)\) (see Fig. 2). So, if we are able to show that \( q : X \rightarrow Yb \) can also be simulated by some set of rules of size \((2; 1, 1, 0; 1, 1, 0)\), then \( \text{MAT}(2; 1, 1, 0; 1, 1, 0) \) will be computationally complete. However, unfortunately, the simulation of \( q : X \rightarrow Yb \) by rules of size \((2; 1, 1, 0; 1, 1, 0)\) is still open. This also explains the difficulty with our earlier claims in Fernau et al. (2019).

**Theorem**

\[ 6 \text{ REG} \subseteq \text{MAT}(2; 1, 1, 0; 1, 1, 0) \cap \text{MAT}(2; 1, 0, 1; 1, 0, 1). \]
Proof Consider a type-3 grammar $G = (N, T, P, S)$ in right-linear form, with the rules uniquely labelled with $p : X \to bY$ and $r : X \to \lambda$ where $X, Y \in N$ and $b \in T$. We can construct a matrix ins-del system $\Gamma = (V, T, S, M)$ with alphabet $V = N \cup T \cup \{p, p^{'}, p^{'\prime}, p^{'\prime\prime}\}$.

The set of matrices shown in Fig. 8a, b constitutes $M$.

Consider a sentential form $w = zX$ derivable in the grammar $G$, with $z \in T^*$. Assume we are about to apply a concrete rule $X \to bY \in P$, with $X, Y \in N$, yielding $w' = zbY$. Hence, the matrices listed in Fig. 8a should apply, one after the other, giving:

$$w \Rightarrow p_1 \alpha Xp'p \Rightarrow p_2 \alpha p''p \Rightarrow p_3 \alpha p''' \Rightarrow p_4 \alpha p'' \Rightarrow p_5 \alpha bp \Rightarrow p_6 \alpha bY \Rightarrow p_6 w'. $$

This shows that $L(G) \subseteq L(\Gamma)$. In the following, we are arguing why $L(G) \supseteq L(\Gamma)$.

Consider a string $w = zX$ that is derivable from $S$ in $\Gamma$. As no rule markers are present, only a matrix of type $p1$ or $p2$ can be applied.

Assume more concretely that matrix $p1$ or $p2$ belonging to $p : X \to bY$ is applied. If $p2$ is applied first before applying $p1$, then this means that $X$ is replaced by the marker $p''$ and no further matrices are possible to apply as they require either the markers $p \text{ or } p'$ or $X$ which are not present in the derived string. So, only $p1$ can be applied first and it can be applied any number of times, yielding $w_1(n) = zX(p'p)^n, n \geq 1$. On $w_1(n)$, matrices $p3$ and $p4$ are inapplicable due to the absence of $p''$. At first glance, it may appear that matrix $p5$ is applicable. However, if we closely look at the two rules in the matrix $p5$, the matrix is applicable if and only if $bp$ is a substring of our sentential form. We use this observation (calling it $ob1$) repeatedly. Note that every $p$ in $w_1(n)$ is preceded by $p'$ and hence $bp$ is not a substring in $w_1(n)$. This prohibits the application of $p5$ on $w_1(n)$. Finally, the absence of $p''$ renders $p6$ inapplicable. So, assuming that $p1$ is not applied again, the only matrix applicable on $w_1(n)$ is $p2$. On applying $p2$, the marker $p''$ is introduced in the place of $X$ which yields $w_2(n) = zX(p'p)^n$. The matrices $p1$ and $p2$ are hereafter not applicable due to the absence of $X$. Matrices $p4$ and $p6$ require the presence of $p''$ and are hence not applicable to $w_2(n)$. Observation $ob1$ prevents us from applying $p5$. The only matrix that is applicable to $w_2(n)$ is $p3$ which introduces yet another new marker $p''$ randomly and the second rule of $p3$ deletes the occurrence of $p''$ rightmost of $p''$. Thus we obtain a string $w_3(n)$ that can be described as $p'' \uplus \alpha p''p(p'p)^n - 1$. For convenience, we let $\tau_n = (p'p)^n - 1$. The absence of $X$ in $w_3(n)$ prevents applying matrices $p1$ and $p2$. We can now observe that the matrix $p3$ is applicable if and only $p''p'$ is a substring (we call this observation $ob2$). Since $w_3(n)$ contains neither $bp$ nor $p''p'$ as a substring, matrices $p5$ and $p3$ are inapplicable on $w_3(n)$ due to $ob1$, $ob2$. Hence, the only matrices applicable on $w_3(n)$ are either $p4$ or $p6$. As a last observation $ob3$, notice that matrices $p4$ and $p6$ are only applicable if $p'''$ is present.

Case 1: If we apply matrix $p6$ to $w_3(n)$, then the (randomly inserted) marker $p'''$ is deleted and we end up with $w_4(n) = xzbp$. Due to the absence of any of $X, p''p', p''$, $bp$, all rules become inapplicable due to $ob1$, $ob2$, $ob3$ and hence the derivation is stuck.

Case 2: If we apply matrix $p4$ to $w_3(n)$, then the marker $p''$ is deleted with the (randomly inserted) marker $p'''$ on its left. This enforces the marker $p'''$ that was introduced randomly should have been placed on the left of $p''$. Hence, we now know that $w_3(n) = xzbp\tau_n$. So, the first rule of $p4$ guarantees the presence of $p'''p''$ as a substring and also deletes $p''$. The second rule of $p4$ then introduces a $b$ (the intended $b$ of the simulation rule $X \to bY$) to the right of $p'''$, thus yielding $w_4(n) = zbp\tau_n$. The matrix $p4$ actually simulates the rewriting rule $p'' \to b$.

The absence of $X$ and $p''$ blocks the application of matrices $p1$ through $p4$. Now, either $p5$ or $p6$ is applicable. The order of application of the rules hereafter do not matter, as they are independent.

Case A: If we apply matrix $p5$ on $w_4(n)$, then we get $w_5(n) = xzbpY\tau_n$. The absence of $X, p''p'$, $bp$ enforces the application of matrix $p6$ which deletes the marker $p'''$, thus yielding $w_6(n) = xzbY\tau_n$. It is to be noted that matrix $p6$ need not be even applied after applying $p5$ and one can start the derivation based on some $Y$-rule, but of course $p6$ can be applied at any time. In particular, sentential forms that contain various $\tau$ for different rules $r$ are possible but do not invalidate our arguments, as all these triple-primed markers have to and can be deleted in a terminal derivation. Notice that if indeed several occurrences of $p'''$ are present upon applying matrix $p4$, there is clearly the danger of inserting $b$ to the right of the wrong occurrence of $p'''$ by the second rule of $p4$. However, a successful application of $p5$ enforces the correct occurrence of $p'''$ to be chosen by $p4$, again by observation $ob1$.

Case B: If we first apply matrix $p6$ on $w_4(n)$, then we get $w_5'(n) = xzb\tau_n$. The absence of $X, p''p'$ enforces the
deterministic application of matrix \( p_5 \) which yields \( w_6(n) = zbY_{\tau_n} \).

Notice that should we "forget" to apply matrix \( p_6 \), i.e., should \( p'' \) stay in the string, then we cannot make mischievous use of that left-over occurrence of \( p'' \) by applying the second rule of \( p_4 \) to it, because then there would not be any \( b \) left to \( Y \) as required by matrix \( p_5 \).

After applying both matrices, we get \( w_6(n) = zbY_{\tau_n} \). Since no markers \( p', p \) in \( \tau_n \) can be deleted hereafter, this enforces that the matrix \( p_1 \) was applied exactly once, i.e., \( n = 1 \), and hence \( \tau_n = \lambda \).

The singleton rule shown in Fig. 8b clearly simulate the rule \( r : X \rightarrow \lambda \). The above arguments show the first inclusion \( \text{REG} \subset \text{MAT}(2; 1, 1, 0; 1, 1, 0) \). The second inclusion result follows by Proposition 2. The strictness of the inclusions follow by Example 1. These considerations complete the proof.

6 Conclusions and further research directions

In this paper, using matrix ins-del systems, we have obtained some (improved) computational completeness results and described the regular closure of linear languages with small resource needs. It is interesting to note that if one could describe linear languages by a matrix insertion-deletion system of size \( s \), then with the same size \( s \), we could describe the regular closure of linear languages, as well. We have also given a complete picture of the state of the art of the generative power of the matrix ins-del systems with sum-norm 3 or 4 in Table 1. Finally, we believe that the normal form tsSGNF that we introduced offers some features that could be used in other computational completeness proofs. In particular, no substrings with nonterminals to the right of terminals are derivable in this normal form.

Further to some open problems mentioned in the introduction, We now present some further concrete research questions.

- It would be interesting to explore closure properties for matrix ins-del systems of small sizes. For instance, is the family \( \text{MAT}(2; 2, 1, 0; 1, 0, 0) \) closed under reversal? If this were true, then \( \text{MAT}(2; 2, 1, 0; 1, 0, 0) = \text{MAT}(2; 2, 0, 1; 1, 0, 0) \), which would also mean that the statement of Corollary 1 could be simplified.
- Do matrix ins-del systems of small sum-norm allow for efficient parsing? We are not aware of any research in this direction. Also this area seems to be largely neglected, although it is clear that this is of much importance if it comes to finally applying these generative devices in language processing.
- It has been argued in other places that ins-del systems could form the basis of biocomputing devices. Insertion and deletions would form the basic operations for such machines. Then the question arises how to program such devices. Following the paradigm of imperative programming, the most basic way of building programs would be to design program fragments consisting of basic operations that should be performed one after the other. This is exactly what matrix grammars can do. One future challenge would be to devise implementations that are based on these basic commands and their sequential execution.
- In a recent paper (Vu and Fernau 2021), Vu and Fernau studied matrix grammars with insertions, deletions and substitutions, a third operation whose relevance to biocomputing is explained in Beaver (1995) and Kari (1997), allowing for further restrictions on the resources studied in this paper. In this context, questions similar to the previous item arise.
Fernau H, Kuppusamy L, Raman I (2019) On matrix ins-del systems. In: Gopal TV, Jäger G, Steila S (eds) Theory and applications of models of computation, TAMC, volume 10185 of LNCS. Springer, pp 201–215

Fernau H, Kuppusamy L, Raman I (2017a) Computational completeness of path-structured graph-controlled insertion-deletion systems. In: Carayol A, Nicaud C (eds) Implementation and application of automata–22nd international conference, CIAA, volume 10329 of LNCS. Springer, pp 89–100

Fernau H, Kuppusamy L, Raman I (2017b) Graph-controlled insertion-deletion systems generating language classes beyond linearity. In: Pighizzini G, Câmpeanu C (eds) Descriptive complexity of formal systems—19th IFIP WG 1.02 international conference, DCFS, volume 10316 of LNCS. Springer, pp 128–139

Fernau H, Kuppusamy L, Raman I (2017c) On the computational completeness of graph-controlled insertion-deletion systems with binary sizes. Theor Comput Sci 682:100–121 Special Issue on Languages and Combinatorics in Theory and Nature

Fernau H, Kuppusamy L, Raman I (2017d) On the generative power of graph-controlled insertion-deletion systems with small sizes. J Autom, Lang Combin 22:61–92

Fernau H, Kuppusamy L, Verlan S (2017e) Universal matrix insertion grammars with small size. In: Patitz MJ, Stannett M (eds) Unconventional computation and natural computation—16th international conference, UCNC, volume 10240 of LNCS. Springer, pp 182–193

Fernau H, Kuppusamy L, Raman I (2018a) Computational completeness of simple semi-conditional insertion-deletion systems. In: Stepney S, Verlan S (eds) Unconventional computation and natural computation, UCNC, volume 10867 of LNCS. Springer, pp 86–100

Fernau H, Kuppusamy L, Raman I (2018b) Investigations on the power of matrix insertion-deletion systems with small sizes. Nat Comput 17(2):249–269

Fernau H, Kuppusamy L, Raman I (2018c) On describing the regular closure of the linear languages with graph-controlled insertion–deletion systems. RAIRO Inf théor Appl/Theor Informat Appl 52(1):1–21

Fernau H, Kuppusamy L, Raman I (2018d) Properties of language classes between linear and context-free. J Autom, Lang Combin 23(4):329–360

Fernau H, Kuppusamy L, Raman I (2019) On matrix ins-del systems of small sum-norm. In: Catania B, Královič R, Nawrocki J, Pighizzini G (eds) SOFSEM 2019: theory and practice of computer science - 45th international conference on current trends in theory and practice of computer science, volume 11376 of LNCS. Springer, pp 192–205

Freund R, Kogler M, Rogozhin Y, Verlan S (2010) Graph-controlled insertion-deletion systems. In: I. McQuillan and G. Pighizzini, editors, Proceedings 12th annual workshop on descriptive complexity of formal systems, DCFS, volume 31 of EPTCS, pp 88–98

Galiukschov BS (1981) Semicontextual grammars. Mat. logika i mat. ling., Kalinin University, pp 38–50 (in Russian)

Geffert V (1991) How to generate languages using only two pairs of parentheses. J Inf Process Cybern EIK 27(5/6):303–315

Haussler D (1983) Insertion languages. Inf Sci 31(1):77–89

Ivanov S, Verlan S (2015) Random context and semi-conditional insertion-deletion systems. Fundam Inform 138:127–144

Ivanov S, Verlan S (2017) Universality and computational completeness of controlled leftist insertion–deletion systems. Fundam Inform 155(1–2):163–185

Kari L (1991) On insertions and deletions in formal languages. PhD thesis, University of Turku, Finland

Kari L (1997) DNA computing: arrival of biological mathematics. Math Intell 19(2):9–22

Kari L, Thierrin G (1996) Contextual insertions/deletions and computability. Inf Comput 131(1):47–61

Krasovskiy A, Rogozhin Y, Verlan S (2011) Computational power of insertion-deletion (P) systems with rules of size two. Nat Comput 10:835–852

Kuppusamy L, Mahendran A (2016) Modelling DNA and RNA secondary structures using matrix insertion–deletion systems. Int J Appl Math Comput Sci 26(1):245–258

Kuppusamy L, Rama R (2003) On the power of tissue P systems with insertion and deletion rules. In: Pre-proc. of workshop on membrane computing, volume 28 of report RGML. University Tarragona, Spain, pp 304–318

Kari L, Mahendran A, Krishna SN (2011) Matrix insertion-deletion systems for bio-molecular structures. In: Natarajan R, Ojo AK (eds) Distributed computing and internet technology—7th international conference, ICDCIT, volume 6536 of LNCS. Springer, pp 301–312

Marcus M, Păun Gh (1990) Regulated Galiukschov semicontextual grammars. Kybernetika 26(4):316–326

Margenstern M, Păun Gh, Rogozhin Y, Verlan S (2005) Context-free insertion-deletion systems. Theor Comput Sci 330(2):339–348

Păun Gh, Rozenberg G, Salomaa A (1998) DNA computing: new computing paradigms. Springer

Petre I, Verlan S (2012) Matrix insertion–deletion systems. Theor Comput Sci 456:80–88

Stabler E (2004) Varieties of crossing dependencies: structure dependence and mild context sensitivity. Cogn Sci 28:699–720

Verlan S (2010) Recent developments on insertion–deletion systems. Comput Sci J Mold 18(2):210–245

Verlan S, Fernau H, Kuppusamy L (2020) Universal insertion grammars of size two. Theor Comput Sci 843:153–163

Vu M, Fernau H (2021) Adding matrix control: insertion–deletion systems with substitutions III. In: Bures T, Dondi R, Gamber J, Guerrini G, Jurzidinski T, Pahl C, Sikora F, Wong PWH (eds) SOFSEM 2021: theory and practice of computer science—47th international conference on current trends in theory and practice of computer science, SOFSEM, volume 12607 of LNCS. Springer, pp 577–592

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.