Natural Computing (2021) 20:671–689
https://doi.org/10.1007/s11047-021-09866-y

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

Henning Fernau[1] · Lakshmanan Kuppusamy[2] · Indhumathi Raman[3]

Accepted: 21 June 2021 / Published online: 8 September 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 $s = (k; n, i', i''; m, j', j'')$, 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 $n + i' + i'' + m + j' + j''$ 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), (2; 2, 1, 0; 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

A preliminary version of this paper appeared in Proceedings of SOFSEM 2019, LNCS 11376, pp. 192–205, 2019.

✉ Henning Fernau
fernau@uni-trier.de

✉ Lakshmanan Kuppusamy
klakshma@vit.ac.in

✉ Indhumathi Raman
ind.amcs@psgtech.ac.in

[1] Fachbereich 4-Abteilung Informatikwissenschaften, CIRT, Universität Trier, 54286 Trier, Germany

[2] School of Computer Science and Engineering, VIT University, Vellore 632 014, India

[3] Department of Applied Mathematics and Computational Sciences, PSG College of Technology, Coimbatore 641 004, India

follows: if a string η 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 δ 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 Păun 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 Păun 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* (Kuppusamy et al. 2011; Petre and Verlan 2012; Kuppusamy and Mahendran 2016). Viewing insertions and deletions as elementary operations for biocomputing (Păun et al. 1998), matrices can be seen as a very simple control mechanism.

In a matrix ins-del system, the insertion-deletion rules are given in matrix form. If a matrix is chosen for derivation, then all the rules in that matrix are applied in order and no rule of the matrix is exempted. In the size $s = (k; n, i', i''; m, j', j'')$ of a matrix ins-del system, the parameters (from left to right) denote the maximum number of rules (*length*) in any matrix, the maximal length of the inserted string, the maximal length of the left context for insertion, the maximal length of the right context for insertion, the maximal length of the deleted string, the maximal length of the left context for deletion, maximal length of the right context for deletion. We denote the language classes generated by matrix ins-del systems of size s by $\mathrm{MAT}(s)$. The tuple formed by the last six parameters, namely $(n, i', i''; m, j', j'')$, is called the *ins-del size*. We call the sum of its parameters the *sum-norm* of the (matrix) ins-del system.[1]

It is known that ins-del systems are computationally complete, i.e., they characterize the family RE of recursively enumerable languages, which readily transfers to the mentioned variants. Descriptional complexity then aims at investigating which of the resources are really needed to obtain computational completeness. For instance, is it really necessary to permit insertion operations that check out contexts of arbitrary length? For resource restrictions that do not (or are not known to) suffice to achieve computational completeness, one is interested in seeing which known families of languages can be still generated. As in our case, for several families of matrix ins-del systems, it is even unknown if all of CF (the context-free languages) can be generated, we then look at the rather large sub-family $\mathcal{L}_{reg}(\mathrm{LIN})$, the *regular closure* of LIN. 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, $\mathrm{MAT}(*; 1, 0, 0; 1, 1, 0) = \mathrm{RE}$ iff $\mathrm{MAT}(*; 1, 0, 0; 1, 0, 1) = \mathrm{RE}$(here the left and right contexts size within insertion/deletion are exchanged), but the questions whether $\mathrm{MAT}(*; 1, 0, 0; 2, 1, 0) = \mathrm{RE}$ or whether $\mathrm{MAT}(*; 2, 1, 0; 1, 0, 0) = \mathrm{RE}$ (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 earlier and in Fernau et al. (2019), the preliminary version of the paper, an attempt was made to reduce the matrix length from 3 to 2. However, a flaw in the proof was later pointed out by one of the referees and therefore this case remains open. Besides this size, matrix ins-del systems of sizes (3; 1, 1, 0; 1, 0, 0), (3; 1, 0, 1; 1, 0, 0), (2; 2, 1, 0; 1, 0, 0), (2; 2, 0, 1; 1, 0, 0) are not known to describe RE, not even all of CF. However, these systems have been shown to describe at least the class MLIN of metalinear languages (Fernau et al. 2018b). It is known that $\mathrm{MAT}(1; 1, 1, 0; 1, 0, 0) \cup \mathrm{MAT}(3; 1, 0, 0; 1, 0, 0) \neq \mathrm{RE}$ (Fernau and Kuppusamy 2017). In particular, it has been shown that the Parikh images of languages from $\mathrm{MAT}(3; 1, 0, 0; 1, 0, 0)$ coincide with the (commutative) Petri net languages and that even allowing for longer matrices would not increase the generative power when keeping the ins-del size of (1, 0, 0; 1, 0, 0). We consider a family of languages strictly between MLIN and CF — namely, the *regular closure* of LIN, which is the smallest language class containing LIN but being closed under the three regular operations union, concatenations and Kleene star. We show that if $\mathrm{LIN} \subseteq \mathrm{MAT}(s)$, then $\mathcal{L}_{reg}(\mathrm{LIN}) \subseteq \mathrm{MAT}(s)$. As a consequence, it follows that matrix ins-del systems of the sizes mentioned above contain $\mathcal{L}_{reg}(\mathrm{LIN})$. 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 and some non-

[1] In papers like (Krassovitskiy et al. 2011), our *sum-norm* is termed as *total size*. We avoid the latter terminology just to avoid its confusion with size s of $\mathrm{MAT}(s)$.

Table 1 Generative power of 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)	≥ 1	$\neq$ RE	Inferred from Krassovitskiy et al. (2011)
3	(1, 0, 0; 1, 1, 0), (1, 0, 0; 1, 0, 1)	1	$\neq$ RE	Inferred from Krassovitskiy et al. (2011)
		≥ 2	?	OPEN
3	(1, 1, 0; 1, 0, 0), (1, 0, 1; 1, 0, 0)	3	$\supset$ MLIN	Fernau et al. (2018b)
		3	$\supset \mathcal{L}_{reg}$(LIN)	Fernau et al. (2019), Corollary 1
4	(1, 0, 0; 1, 1, 1)	3	RE	Fernau et al. (2018b)
4	(1, 1, 1; 1, 0, 0)	3	RE	Fernau et al. (2018b)
		2	RE	Theorem 4
4	(1, 0, 0; 2, 1, 0), (1, 0, 0; 2, 0, 1)	≥ 1	?	OPEN
4	(1, 0, 0; 1, 2, 0), (1, 0, 0; 1, 0, 2)	3	RE	Fernau et al. (2019), Theorem 2
4	(1, 2, 0; 1, 0, 0), (1, 0, 2; 1, 0, 0)	3	RE	Fernau et al. (2019)
		2	RE	Theorem 3
4	(1, 1, 0; 1, 1, 0), (1, 0, 1; 1, 0, 1)	3	RE	Petre and Verlan (2012) and Fernau et al. (2018b)
		2	$\supset$ REG	Theorem 6
		1	$\subset$ CF	Krassovitskiy et al. (2011)
4	(1, 1, 0; 1, 0, 1), (1, 0, 1; 1, 1, 0)	3	RE	Petre and Verlan (2012) and Fernau et al. (2018b)
		1	$\neq$ RE	Krassovitskiy et al. (2011)
4	(1, 1, 0; 2, 0, 0), (1, 0, 1; 2, 0, 0)	2	RE	Petre and Verlan (2012) and Fernau et al. (2018b)
		1	$\neq$ RE	Krassovitskiy et al. (2011)
4	(2, 0, 0; 2, 0, 0)	≥ 1	$\neq$ RE	Krassovitskiy et al. (2011)
4	(2, 0, 0; 1, 1, 0), (2, 0, 0; 1, 0, 1)	2	RE	Petre and Verlan (2012) and Fernau et al. (2018b)
		1	$\neq$ RE	Krassovitskiy et al. (2011)
4	(2, 1, 0; 1, 0, 0), (2, 0, 1; 1, 0, 0)	2	$\supset$ MLIN	Fernau et al. (2018b)
		2	$\supset \mathcal{L}_{reg}$(LIN)	Fernau et al. (2019), Corollary 1

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 arguments, because in particular there is no way to have mixtures of terminals and nonterminals at the right end of a 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 LIN $\subseteq$ GCID$(k; n, i', i''; m, j', j'')$, then $\mathcal{L}_{reg}$(LIN) $\subseteq$ 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. MAT$(3; 1, 0, 0; 1, 2, 0)$ = MAT$(3; 1, 0, 0; 1, 0, 2)$ = RE.
2. MAT$(2; 1, 1, 0; 1, 1, 0)$ = MAT$(2; 1, 0, 1; 1, 0, 1)$ = RE.
3. MAT$(3; 1, 2, 0; 1, 0, 0)$ = MAT$(3; 1, 0, 2; 1, 0, 0)$ = RE.
4. $\mathcal{L}_{reg}$(LIN)$\subsetneq$MAT$(3; 1, 1, 0; 1, 0, 0) \cap$ MAT $(3; 1, 0, 1;\ 1, 0, 0)$.

[2] One should also keep in mind the slightly different meanings of the parameter lists.

5. $\mathcal{L}_{reg}(\mathrm{LIN}) \subsetneq \mathrm{MAT}(2;2,1,0;1,0,0) \cap MAT(2;2,0,1;1,0,0)$.

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 $\mathrm{MAT}(2;1,1,1;1,0,0) = \mathrm{RE}$, 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 Σ^* denote the free monoid generated by the *alphabet* (finite set) Σ. The elements of Σ^* are called *strings* or *words*; λ denotes the empty string, L^R and $\mathcal{L}^R$ denote the *reversal* of language L and language family $\mathcal{L}$, respectively. RE and LIN denote the families of recursively enumerable languages and linear languages, respectively. Occasionally, we use the shuffle operator, written as ⧢.

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*, tsSGNF 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 $AB \to \lambda$ and $CD \to \lambda$,
- the context-free rules are of one of the following forms:
  (a) $X \to Yb$ or $X \to b'Y$ where $X \in N^{(0)}$, $Y \in N^{(0)} \cup N'$, $b \in T$,
  (b) $X \to Yb''$ or $X \to b'Y$ where $X, Y \in N'$, or $S' \to \lambda$, where $b' \in \{A, C\}$, $b'' \in \{B, D\}$ and $X \neq Y$ in (a) and (b);
  (c) possibly, there is also the rule $S \to \lambda$.

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

Table 2 How the results of this section/paper improve/complement previous results

Result	Theorem	Improves/relates to	References
$\mathrm{MAT}(3;1,0,0;1,2,0) = \mathrm{RE}$	Theorem 2	$\mathrm{MAT}(3;1,0,0;1,1,1) = \mathrm{RE}$	Fernau et al. (2018b)
		$\mathrm{MAT}(3;1,0,0;1,1,0) = ???$	OPEN
$\mathrm{MAT}(2;1,2,0;1,0,0) = \mathrm{RE}$	Theorem 3	$\mathrm{MAT}(3;1,2,0;1,0,0) = \mathrm{RE}$	Fernau et al. (2019)
		$\mathrm{MAT}(3;1,1,1;1,0,0) = \mathrm{RE}$	Fernau et al. (2018b)
$\mathrm{MAT}(2;1,1,0;1,1,0) = ???$	OPEN	$\mathrm{MAT}(2;1,1,0;1,1,1) = \mathrm{RE}$	Fernau et al. (2018b)
		$\mathrm{MAT}(2;1,1,1;1,1,0) = \mathrm{RE}$	Fernau et al. (2018b)
		$\mathrm{MAT}(3;1,1,0;1,1,0) = \mathrm{RE}$	Fernau et al. (2018b)
$\mathrm{MAT}(2;1,1,1;1,0,0) = \mathrm{RE}$	Theorem 4	$\mathrm{MAT}(3;1,1,1;1,0,0) = \mathrm{RE}$	Fernau et al. (2018b)

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 $AB \to \lambda$ and $CD \to \lambda$ 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 $S' \to \lambda$ in the derivation. In Phase II, only the non-context-free erasing rules are applied repeatedly and the derivation ends. By induction, it is clear that sentential forms derivable by tsSGNF grammars belong to $\{A, C\}^* N^{(0)} T^* \cup \{A, C\}^* (N' \cup \{\lambda\}) \{B, D\}^* T^*$.

Finally, notice that [similar to Fernau et al. (2017e)], we can also assume that $X \neq Y$ in the context-free rules, as we can 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., $L \in$ RE, there exists a type-0 grammar G in 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 $r_i = [(u_1, \alpha_1, v_1)_{t_1}, (u_2, \alpha_2, v_2)_{t_2}, \ldots, (u_k, \alpha_k, v_k)_{t_k}]$. For $1 \leq j \leq k$, $u_j, v_j \in V^*$, $\alpha_j \in V^+$ and $t_j \in \{ins, del\}$.

The triplet $(u_j, \alpha_j, v_j)_{t_j}$ is called an ins-del rule and the pair (u_j, v_j) is termed the *context* with u_i as the left and v_i as the right context for α_j in t_j; α_j is called *insertion string* if $t_j = ins$ and *deletion string* if $t_j = del$. The elements of A are called *axioms*. For all contexts of t where $t \in \{ins, del\}$, 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 $(u, \eta, v)_{ins}$, which means that the string η is inserted between u and v. A deletion rule is of the form $(u, \delta, v)_{del}$, which means that the string δ is deleted between u and v. Applying $(u, \eta, v)_{ins}$ corresponds to applying the rewrite rule $uv \to u\eta v$, and applying $(u, \delta, v)_{del}$ corresponds to applying the rewriting rule $u\delta v \to uv$.

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 $x \Rightarrow_{r_i} y$, 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 Γ is defined as $L(\Gamma) = \{w \in T^* \mid x \Rightarrow_* w, \text{for some} x \in A\}$, where $\Rightarrow_*$ (as usual with matrix ins-del systems) denotes the reflexive and transitive closure of $\Rightarrow := \bigcup_{r \in R} \Rightarrow_r$.

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 $L_1 = \{a^n b^m c^n d^m \mid m, n \geq 1\}$ 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: $m1 = [(a, a, \lambda)_{ins}, (c, c, \lambda)_{ins}]$, $m2 = [(b, b, \lambda)_{ins}, (d, d, \lambda)_{ins}]$. We note that the matrices

$$m1' = [(\lambda, a, a)_{ins}, (\lambda, c, c)_{ins}], \quad m2' = [(\lambda, b, b)_{ins}, (\lambda, d, d)_{ins}]$$

also generate L_1. Hence,

$$L_1 \in \text{MAT}(2; 1, 1, 0; 0, 0, 0) \cap \text{MAT}(2; 1, 0, 1; 0, 0, 0)\,.$$

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 $A \to x$, 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 $A \to x$ either obeys $x \in T \cup \{\lambda\}$ or $x \in TN$ or $x \in NT$. It is well known that LIN is not closed under concatenation and Kleene star. This motivates to consider the class $\mathcal{L}_{reg}(\text{LIN})$ 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 $A \to aB$ or $A \to \lambda$, with $A \in N$, $B \in N \backslash \{A\}$ and $a \in T$. The

following grammatical characterization for $\mathcal{L}_{reg}(\mathrm{LIN})$ was shown in Fernau et al. (2018d).

Proposition 1 *Fernau et al. (2018d) Let $L \subseteq T^*$ be some language. Then, $L \in \mathcal{L}_{reg}(\mathrm{LIN})$ 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_i that can be obtained from G by using rules P_i 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 $\mathcal{L}$ 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*

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

Proposition 3 *Fernau et al. (2018c) $\mathcal{L}_{reg}(\mathrm{LIN})$ 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_{ll} the labels of the context-free rules of the form $X \to Yb$ (which resemble left-linear rules) and in P_{rl} the labels of the context-free rules of the form $X \to bY$ (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 $S \to \lambda$ 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 $\mathrm{MAT}(3;1,0,0;1,2,0) = \mathrm{MAT}(3;1,0,0;1,0,2) = \mathrm{RE}$.

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_{ll} and P_{rl} 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 system $\Gamma = (V, T, \{S\}, M)$, where the alphabet of Γ is

$$V = N \cup T \cup \{p, p', p'', p''' \mid p \in P_{rl}\} \cup \{q, q', q'' \mid q \in P_{ll}\} \cup \{Z\}.$$

The set of matrices M of Γ consists of the matrices described in the following.

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

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

Recall that applying the rule $h : S' \to \lambda$ 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 Γ 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 Γ.

Simulation of $p : X \to bY$: consider the string $\alpha X\beta$ derivable from S in G, with $X \in N^{(0)} \cup N'$ and $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^* T^*$ according to Remark 1. We now show that on applying the matrices introduced for simulating rules from P_{rl}, we can derive $\alpha bY\beta$ within Γ, starting from $\alpha X\beta$. 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 $p \sqcup p' \sqcup \alpha X \beta$. However, the third rule of $p1$ is applicable only when $p'p$ is inserted before the nonterminal X. This shows that $\alpha X\beta \Rightarrow_{p1} \gamma_1$ is possible if and only if $\gamma_1 = \alpha p' p \beta$.

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

$$\alpha X\beta \Rightarrow_{p1} \alpha p' p\beta \Rightarrow_{p2} \alpha p'''p''p\beta \Rightarrow_{p3} \alpha p'''bp\beta \Rightarrow_{p4} \alpha bY\beta.$$

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

Simulation of $f : AB \to \lambda$ *or* $g : CD \to \lambda$: consider the sentential form $\alpha AB\beta$ 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 Γ. Any string from $Z \sqcup \alpha AB\beta$ can be transformed into $\alpha ZAB\beta$ 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 $w \in V^*$ such that $S \Rightarrow_* w$ in Γ:

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 $w = \alpha Xu$, where $\alpha \in \{A, C\}^*$ and $u \in T^*$: w is derivable in G;
3. If one symbol X from N' occurs in w, then $w = \alpha X\beta u$, where $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^*$ 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 $w = \alpha r' r \beta u$, where $\alpha \in \{A, C\}^*$, r is some context-free rule from G with left-hand side X, $\beta \in \{B, D\}^*$ and $u \in T^*$: $\alpha X\beta u$ is derivable in G;
   (b) or $w = \alpha p'''p''p\beta u$, where $\alpha \in \{A, C\}^*$, $p \in P_{rl}$ with $p : X \to bY$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $\alpha X\beta u$ is derivable in G;

$$\begin{array}{lcl} p1 & = & [(\lambda, p, \lambda)_{ins},\ (\lambda, p', \lambda)_{ins}, (p'p, X, \lambda)_{del}] \\ p2 & = & [(\lambda, p'', \lambda)_{ins}, (\lambda, p''', \lambda)_{ins}, (p'''p'', p', \lambda)_{del}] \\ p3 & = & [(\lambda, b, \lambda)_{ins},\ (p'''b, p'', \lambda)_{del}] \\ p4 & = & [(\lambda, Y, \lambda)_{ins},\ (bY, p, \lambda)_{del},\ (\lambda, p''', \lambda)_{del}] \end{array}$$

(a) Simulating $X \to Yb$

$$\begin{array}{lcl} q1 & = & [(\lambda, q, \lambda)_{ins},\ (\lambda, q', \lambda)_{ins}, (q'q, X, \lambda)_{del}] \\ q2 & = & [(\lambda, q'', \lambda)_{ins}, (\lambda, q''', \lambda)_{ins}, (q'''q'', q', \lambda)_{del}] \\ q3 & = & [(\lambda, b, \lambda)_{ins},\ (q'''b, q'', \lambda)_{del}] \\ q4 & = & [(\lambda, Y, \lambda)_{ins},\ (Yb, q, \lambda)_{del},\ (\lambda, q''', \lambda)_{del}] \end{array}$$

(b) Simulating $X \to Yb$

Fig. 1 Matrices of size (3; 1, 0, 0; 1, 2, 0) for simulating the context-free rules of tsSGNF

$$
\begin{array}{lcllcll}
h1 & = & [(\lambda, S', \lambda)_{del}, & (\lambda, Z, \lambda)_{ins}] & \text{move-}Z & = & [(\lambda, Z, \lambda)_{del}, \quad (\lambda, Z, \lambda)_{ins}] \\
f1 & = & [(Z, A, \lambda)_{del}, & (Z, B, \lambda)_{del}] & \text{del-}Z & = & [(\lambda, Z, \lambda)_{del}] \\
g1 & = & [(Z, C, \lambda)_{del}, & (Z, D, \lambda)_{del}] & & & \\
\end{array}
$$

Fig. 2 Matrices of size (2; 1, 0, 0; 1, 1, 0) for simulating the erasing rules of tsSGNF

(c) or $w = \alpha p''' bp\beta u$, where $\alpha \in \{A, C\}^*$, $p \in P_{rl}$ with $p : X \to bY$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $\alpha X \beta u$ is derivable in G;

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

6. If Z occurs in w, then $w \in Z \sqcup\!\sqcup \alpha\beta u$, where $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $\alpha\beta u$ 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, $w = \alpha X \beta u$, where $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^*$ and $u \in T^*$. In fact, the conditions are more specific about some details regarding X and β. All rules but $r1$ (for some context-free rule r) or $h1$ require the presence of some rule marker (or of Z) and are hence not applicable.

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

Condition 5(a) Consider any sentential form w that satisfies 5(a). Hence, $w = \alpha r' r \beta u$, where $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^*$ and $u \in T^*$. Moreover, $\alpha X \beta u$ 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 $r = p \in P_{rl}$ and $r = q \in P_{ll}$. In the first case, we inevitably derive $w' = \alpha p''' p'' p \beta u$ [see 5(b)], and in the second case, we arrive at $w' = \alpha q' q'' b \beta u$ [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, $w = \alpha p'' bp\beta u$, where $\alpha \in \{A, C\}^*$, $p \in P_{rl}$ with $p : X \to bY$, $\beta \in \{B, D\}^*$ and $u \in T^*$: $\alpha X \beta u$ 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, $w \Rightarrow_{p3} w' = \alpha bY \beta u$, bringing us into Case 2 or 3, as G can derive this string from $\alpha X \beta u$ (induction hypothesis) by applying p.

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

Condition 5(d) As the reader may check, no matrix from Γ 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, $f1$, $g1$. 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 $w \in Z \sqcup\!\sqcup T^*$, as now a terminal string was derived in Γ that is also generated by G. Matrix $f1$ can only be applied if Z sits immediately to the left of the substring AB. The effect of applying $f1$ 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 $\mathrm{MAT}(3; 1, 2, 0; 1, 0, 0) = \mathrm{MAT}(3; 1, 0, 2; 1, 0, 0) = \mathrm{RE}$ 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 $\mathrm{MAT}(2; 1, 2, 0; 1, 0, 0) = \mathrm{MAT}(2; 1, 0, 2; 1, 0, 0) = \mathrm{RE}$.

Before we begin our proof, we highlight the key feature of the markers first. In order to simulate, say, $AB \to \lambda$, we have to use deletion rules $(\lambda, A, \lambda)_{del}$ and $(\lambda, B, \lambda)_{del}$, 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 \$, which is present in the axiom itself, serves as a

Fig. 3 Simulating context-free rules of tsSGNF by matrix rules of size (2; 1, 2, 0; 1, 0, 0)

$$\begin{aligned} p1 &= [(X,p,\lambda)_{ins},(\lambda,\$,\lambda)_{del}] \\ p2 &= [(Xp,p',\lambda)_{ins},(\lambda,X,\lambda)_{del}] \\ p3 &= [(pp',p'',\lambda)_{ins},(p,p''',\lambda)_{ins}] \\ p4 &= [(p'''p',b,\lambda)_{ins},(\lambda,p,\lambda)_{del}] \\ p5 &= [(bp'',Y,\lambda)_{ins},(\lambda,p'',\lambda)_{del}] \\ p6 &= [(\lambda,p''',\lambda)_{del},(bY,\$,\lambda)_{ins}] \\ p7 &= [(\lambda,p',\lambda)_{del}] \end{aligned}$$

(a) Simulating $X \to bY$

$$\begin{aligned} q1 &= [(X,q,\lambda)_{ins},(\lambda,\$,\lambda)_{del}] \\ q2 &= [(Xq,q',\lambda)_{ins},(\lambda,X,\lambda)_{del}] \\ q3 &= [(qq',q'',\lambda)_{ins},(q,q''',\lambda)_{ins}] \\ q4 &= [(q'''q',Y,\lambda)_{ins},(\lambda,q,\lambda)_{del}] \\ q5 &= [(Yq'',b,\lambda)_{ins},(\lambda,q'',\lambda)_{del}] \\ q6 &= [(\lambda,q''',\lambda)_{del},(Yb,\$,\lambda)_{ins}] \\ q7 &= [(\lambda,q',\lambda)_{del}] \end{aligned}$$

(b) Simulating $X \to Yb$

semaphore flag as known in concurrent programming, preventing simulation cycles from being interrupted. We will use this trick also in other simulations below.

Proof Consider a type-0 grammar $G = (N, T, P, S)$ in tsSGNF, with the rules uniquely labelled with $P_{ll} \cup P_{rl}$. Recall the decomposition $N = N^{(0)} \cup N' \cup N''$ by tsSGNF. We can construct a matrix ins-del system $\Gamma = (V, T, \{\$S\}, M)$ with alphabet $V = N \cup T \cup P_{rl} \cup P_{ll} \cup K$ where K is the following set of markers:

$$\begin{aligned} &\{p, p', p'', p''' \mid p \in P_{rl}\} \cup \{q, q', q'', q''' \mid q \in P_{ll}\} \\ &\quad \cup \{f, f', f'', f^3, f^4, f^5, f^6, f^7, f^8, f^9, f^{10}, \\ &\quad g, g', g'', g^3, g^4, g^5, g^6, g^7, g^8, g^9, g^{10}, \$\}. \end{aligned}$$

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

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

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

$$\begin{aligned} \$ \sqcup\!\sqcup \alpha X\beta \ni w^{\$} &\Rightarrow_{p1} \alpha Xp\beta \Rightarrow_{p2} \alpha pp'\beta \Rightarrow_{p3} \alpha pp'''p'p''\beta \Rightarrow_{p4} \alpha p'''p'bp''\beta \\ &\Rightarrow_{p5} \alpha p'''p'bY\beta \Rightarrow_{p6} \alpha p'bY\$\beta \Rightarrow_{p7} \alpha bY\$\beta = w'^{\$}. \end{aligned}$$

Quite similarly, applying the matrices of Fig. 3b (introduced for simulating rules from P_{ll} of the type $X \to Yb$), we can derive some string $w'^{\$} \in \$ \sqcup\!\sqcup \alpha Yb\beta$ within Γ, starting from some $w^{\$} \in \$ \sqcup\!\sqcup w$, simulating $w = \alpha X\beta \Rightarrow \alpha Yb\beta = w'$ in G.

$$\begin{aligned} \$ \sqcup\!\sqcup \alpha X\beta \ni w^{\$} &\Rightarrow_{q1} \alpha Xq\beta \Rightarrow_{q2} \alpha qq'\beta \Rightarrow_{q3} \alpha qq'''q'q''\beta \Rightarrow_{q4} \alpha q'''q'Yq''\beta \\ &\Rightarrow_{q5} \alpha q'''q'Yb\beta \Rightarrow_{q6} \alpha q'Yb\$\beta \Rightarrow_{q7} \alpha Yb\$\beta = w'^{\$}. \end{aligned}$$

Every time a context-free rule is simulated by either p or q rules, the marker \$ is first deleted and finally re-inserted at a somewhat random position at the end of every simulation. In other words, the \$ gets shuffled around the string during Phase I. The phase transition rule $h : S' \to \lambda$ is simulated by applying $h1$, 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, Γ has to apply τ at the very end.

We now discuss Phase II in detail, focussing on $f : AB \to \lambda$. Let $w = \alpha AB\beta$ 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 $w^\$ \in \$ \sqcup\!\sqcup w$ is derivable in Γ (by induction). We can now see that

$$
\begin{aligned}
w^\$ &\Rightarrow_{f1} \alpha f AB\beta \Rightarrow_{f2} \alpha f f' Af'' B\beta \Rightarrow_{f3} \alpha f f' Af'' f^3 Bf^5\beta \\
&\Rightarrow_{f4} \alpha f f' f'' f^6 f^3 Bf^5\beta \Rightarrow_{f5} \alpha f f' f^3 Bf^5\beta \Rightarrow_{f6} \alpha f f' f^7 Bf^5\beta \\
&\Rightarrow_{f7} \alpha f' f^7 Bf^4 f^5\beta \Rightarrow_{f8} \alpha f' f^7 f^4 f^8 f^5\beta \Rightarrow_{f9} \alpha f' f^7 f^5\beta \Rightarrow_{f10} \alpha f' f^7 f^9\beta \\
&\Rightarrow_{f11} \alpha f' f^9 f^{10}\beta \Rightarrow_{12} \alpha f^{10}\beta \Rightarrow_{f13} w'^\$ \in \$ \sqcup\!\sqcup \alpha\beta\,.
\end{aligned}
$$

The purpose of introducing a \$ in $f13$ is to enable another simulation of $AB \to \lambda$ or of $CD \to \lambda$. 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 τ, 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 $w = \alpha X\beta$ derivable from S in G, with $X \in N^{(0)} \cup N'$ and $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^* T^*$ according to Remark 1. We know by our previous arguments that the variation $w^\$ \in \$ \sqcup\!\sqcup w$ can be derived in Γ. 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 $S' \to \lambda$ 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 $\$ \sqcup\!\sqcup \{A, C\}^* (N^{(0)} \cup N') \{B, D\}^* T^*$, 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 $K \backslash \{\$\}$, 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. 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 \$ 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 applicable matrix is now $f2$, which also makes clear that the f was inserted left to some A-occurrence (and hence within α) in w. To highlight the chosen insertion place, let $\alpha = \alpha' A \alpha''$ such that $w' = \alpha' f A \alpha'' X\beta$. Now, if $w' \Rightarrow_{f2} w''$, we can conclude that $w'' = \alpha' f f' A f'' \alpha'' X\beta$. The only applicable matrix is now $f4$, leading to $w'' \Rightarrow_{f4} \alpha' f f' f'' f^6 \alpha'' X\beta \Rightarrow_{f5} \alpha' f f' \alpha'' X\beta$. Notice that the application of $f5$ that we displayed is in fact the only possibility. But now, the derivation is stuck. Therefore, there is no use of applying $f1$ on $w^\$$.

We can only simulate p rules on $w^\$$ that have X as their left-hand side, because the correctness of this (unique) symbol is tested as left context in $p1$. This enforces $w^\$ \Rightarrow_{p1} \alpha X p\beta$ as intended. As \$ is deleted and no (variation of) f is present as a marker, in particular no fi matrix is applicable for $i = 1, \ldots, 13$. As p is the only marker in $\alpha X p\beta$, only $p2$ is applicable indeed, which enforces $\alpha X p\beta \Rightarrow_{p2} \alpha p p'\beta$ as intended. The presence of p and p' makes three matrices interesting to apply: $p2$, $p3$ and $p7$. However, the absence of X renders applying $p2$ impossible. Yet, $p7$ could be applied, leading to $\alpha p\beta$. But how to continue from here? Any rule dealing with p either requires some symbol like X

Fig. 4 Simulating non-context-free rules by matrix rules of size (2; 1, 2, 0; 1, 0, 0)

$$
\begin{array}{lcl}
f1 &=& [(\lambda, \$, \lambda)_{del}, (\lambda, f, \lambda)_{ins}] \\
f2 &=& [(fA, f'', \lambda)_{ins}, (f, f', \lambda)_{ins}] \\
f3 &=& [(f''B, f^5, \lambda)_{ins}, (f'', f^3, \lambda)_{ins}] \\
f4 &=& [(\lambda, A, \lambda)_{del}, (f'f'', f^6, \lambda)_{ins}] \\
f5 &=& [(\lambda, f^6, \lambda)_{del}, (\lambda, f'', \lambda)_{del}] \\
f6 &=& [(f'f^3, f^7, \lambda)_{ins}, (\lambda, f^3, \lambda)_{del}] \\
f7 &=& [(f^7B, f^4, \lambda)_{ins}, (\lambda, f, \lambda)_{del}] \\
f8 &=& [(\lambda, B, \lambda)_{del}, (f^7 f^4, f^8, \lambda)_{ins}] \\
f9 &=& [(\lambda, f^8, \lambda)_{del}, (\lambda, f^4, \lambda)_{del}] \\
f10 &=& [(f^7 f^5, f^9, \lambda)_{ins}, (\lambda, f^5, \lambda)_{del}] \\
f11 &=& [(f^7 f^9, f^{10}, \lambda)_{ins} (\lambda, f^7, \lambda)_{del}] \\
f12 &=& [(\lambda, f', \lambda)_{del}, (\lambda, f^9, \lambda)_{del}] \\
f13 &=& [(\lambda, f^{10}, \lambda)_{del}, (\lambda, \$, \lambda)_{ins}]
\end{array}
\qquad
\begin{array}{lcl}
g1 &=& [(\lambda, \$, \lambda)_{del}, (\lambda, g, \lambda)_{ins}] \\
g2 &=& [(gC, g'', \lambda)_{ins}, (g, g', \lambda)_{ins}] \\
g3 &=& [(g''D, g^5, \lambda)_{ins}, (g'', g^3, \lambda)_{ins}] \\
g4 &=& [(\lambda, C, \lambda)_{del}, (g'g'', g^6, \lambda)_{ins}] \\
g5 &=& [(\lambda, g^6, \lambda)_{del}, (\lambda, g'', \lambda)_{del}] \\
g6 &=& [(g'g^3, g^7, \lambda)_{ins}, (\lambda, g^3, \lambda)_{del}] \\
g7 &=& [(g^7B, g^4, \lambda)_{ins}, (\lambda, g, \lambda)_{del}] \\
g8 &=& [(\lambda, D, \lambda)_{del}, (g^7 g^4, g^8, \lambda)_{ins}] \\
g9 &=& [(\lambda, g^8, \lambda)_{del}, (\lambda, g^4, \lambda)_{del}] \\
g10 &=& [(g^7 g^5, g^9, \lambda)_{ins}, (\lambda, g^5, \lambda)_{del}] \\
g11 &=& [(g^7 g^9, g^{10}, \lambda)_{ins} (\lambda, g^7, \lambda)_{del}] \\
g12 &=& [(\lambda, g', \lambda)_{del}, (\lambda, g^9, \lambda)_{del}] \\
g13 &=& [(\lambda, g^{10}, \lambda)_{del}, (\lambda, \$, \lambda)_{ins}]
\end{array}
$$

to the left of p (in matrix $p2$) or some p' to the right of p (in matrix $p3$) or the presence of $p''p'$ (in matrix $p4$). The absence of \$ also prohibits starting another rule simulation. In other words, the derivation is stuck. This shows that we have to continue with $\alpha pp'\beta \Rightarrow_{p3} \alpha pp'''p'p''\beta$, as was our intention. Observe that neither \$ nor X nor pp' nor bp'' nor Y is present in the current sentential form, which means that only $p4$ or $p7$ might apply. After applying $p7$, we obtain $\alpha pp'''p''\beta$, which again has no substring like \$, X, pp', bp'' or Y. Additionally, the substring $p'''p'$ is now lacking, which means that none of the rules is applicable and hence the derivation is blocked. Therefore, we have to apply matrix $p4$ as intended, enforcing $\alpha pp'''p'p''\beta \Rightarrow_{p4} \alpha p'''p'bp''\beta$. The absence of \$, X (or Y) and p blocks pi for $i = 1, 2, 3, 4, 6$, as well as $f1$. If we apply $p7$, we arrive at $\alpha p'''bp''\beta$. Still, symbols X, Y and p are missing, so that $p5$ would be the only applicable rule, leading to $\alpha p'''bp''\beta \Rightarrow_{p5} \alpha p'''bY\beta$. Due to the absence of a \$-marker, the only matrix that can deal with the substring $p'''b$ is $p6$, which leads to $\alpha bY\$\beta$ as intended, although by following a strategy slightly different from the one presented before. Finally, we have to discuss what happens if we apply $p5$ on $\alpha p'''p'bp''\beta$ (as presented above). We arrive at $\alpha p'''p'bY\beta$. As the markers p and \$ are missing, only $p6$ and $p7$ are applicable. If $p7$ is applied first, then arguments similar to the previous ones show that now only $p6$ is applicable, leading to $\alpha bY\$\beta$ as intended, although by again following a slightly different strategy. Alternatively, we consider applying $p6$ on $\alpha p'''p'bY\beta$ as intended, leading to $\alpha p'bY\$\beta$. When we apply $p7$ now, we have arrived at $\alpha bY\$\beta$ with the derivation strategy presented above.

However, there is one last catch in this simulation. Nobody forces us to apply $p7$ on $\alpha p'bY\$\beta$; we could also keep p' within the string and continue with simulations of context-free rules or (failed) simulations of non-context-free rules (as discussed above) or, after applying $h1$, we might even start (successfully) simulating non-context-free rules (as discussed below). A similar analysis is valid for the q rules, leading to a string like $\alpha q'Yb\$\beta$. Now, observe that any matrix (apart from $p7$ or $q7$ that erase p' or q') that makes use of p' (or q') expects p or p''' (q or q''', respectively) to the left of p' (or q'), which cannot happen, as the (unique) symbols from $N^{(0)} \cup N'$ of the tsSGNF grammar stay to the right of p' (or q'). Therefore, the presence of single-primed rule markers is harmless, they are to be deleted at any time later using $p7$ or $q7$. Hence, we ignore them in our discussions. In particular, assuming that we only have to discuss a string $w = \alpha X\beta$ derivable from S in G (as we did so far) is not devaluating our argumentation.

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 $w' \in \$ \sqcup\!\sqcup w$ is by Γ. As we are using tsSGNF, this means that $w \in \{A, C\}^*\{B, D\}^*T^*$ by Remark 1. Notice that by the discussions performed so far, this observation translates to the simulating grammar Γ.

First, assume that $w = \alpha A\zeta\beta$ for some $\alpha \in \{A, C\}^*$, $\zeta \in \{B, D\}$ and $\beta \in \{B, D\}^*T^*$, or $\zeta \in T$ and $\beta \in T^*$, or $w = \alpha A$ is a sentential form derivable in G, corresponding to some string $w^{\$}$ derivable in Γ. 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''). 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 p' as discussed above. Also, by the very structure of the matrices simulating an f rule, no progress on a sentential form β with $\beta \in \{B, D\}^*T^*$ is observed beyond the possible application of $f1$. One application of $f1$ 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 $f \sqcup\!\sqcup w$. All matrices simulating the f rule require the presence of (multiply) primed versions of the marker f, apart from $f2$ which has to be applied next. Matrix $f2$ checks that an A must be immediately to the right of f, which means that f has been previously inserted within the prefix αA of w. Hence, the resulting string w_2 can be (equivalently) obtained from w by first picking one occurrence of A within the prefix αA 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 $f2$ cannot be applied again. This leads to the string $\alpha' ff'Af''\alpha''\zeta\beta$, with $\alpha A = \alpha' A\alpha''$. All matrices fi but $f3$ require at least one of the substrings \$, fA, $f'f''$ or f^k ($3 \leq k \leq 10$) to be present, which renders them inapplicable. However, matrix $f3$ 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 $w = \alpha A\zeta\beta$ and $\alpha = \alpha'\alpha''$ (all other strings are stuck at this point). Therefore, after applying matrix $f3$, we arrive at the string $w_3 = \alpha ff'Af''f^3Bf^5\beta$. Now, none of the matrices $f5 - f13$ are applicable due to the absence of one of the markers $f^4, f^6, f^7, f^9, f^{10}$. The matrices $f1, f2, f3$ are also inapplicable due to the absence of \$, fA, and $f''B$, respectively. Hence, the only applicable rule is $f4$. The first rule in $f4$ 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 $f'Af''$) is deleted, then we will have the substring $f'f''$ in order to apply the second rule of $f4$. Hence, we get $w_4 = \alpha ff'f''f^6f^3Bf^5\beta$. Again, matrices $f1 - f3, f6 - f13$ still remain inapplicable due to the same

reason as above. The only applicable matrices on w_4 are $f4$ (again) or $f5$.

Suppose $f4$ is applied again (repeatedly for $k \geq 0$ times) on w_4, then some k number of As in α are deleted freely and $(f^6)^k$ is inserted after $f'f''$, leading to a string $w'_4 = \alpha' ff'f''(f^6)^k f^6 f^3 B f^5 \beta$, where α' is obtained from α by deleting k occurrences of A. For further continuation of the derivation, the only other applicable matrix is $f5$. Since there is a unique occurrence of f'', the matrix $f5$ can be applied only once, which will delete only one f^6 leading to the string $w''_4 = \alpha' ff'(f^6)^k f^3 B f^5 \beta$. Assuming $f4$ is applied no more, no other matrix (including $f6$), is applicable due to the absence of one of $\$, fA, f''B, f'', f^4, f^7, f^9, f^{10}$. Hence, the derivation is stuck here if $k > 0$. We hence apply matrix $f5$ on $w_4 = \alpha ff'f''f^6 f^3 B f^5 \beta$ to get $w_5 = \alpha ff'f^3 B f^5 \beta$.

Due to the absence of one of the markers or substrings $\$, fA, f'', f^4, f^7, f^9, f^{10}$, no matrix other than $f6$ is applicable. Applying $f6$ on w_5 yields $w_6 = \alpha ff'f^7 B f^5 \beta$. The absence of $\$, fA, f'', f^3, f^4, f^7 f^5, f^8, f^9, f^{10}$ blocks the applicability of all fi matrices except $f7$. Applying $f7$ on w_6 leads to the string $w_7 = \alpha f'f^7 B f^4 f^5 \beta$ as intended. Matrices $f1 - f7$ are not applicable, since the markers $\$, f, f'', f^3$ are not present in w_7. Similarly, matrices $f9$, $f11$, $f12$ and $f13$ are not applicable on w_7, since the markers f^8, f^9 and f^{10} are not present. Additionally, $f10$ is inapplicable due to the absence of $f^7 f^5$. Thus, the only applicable matrix is $f8$. The first rule in $f8$ 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 $f8$ can be applied. Hence, we get $w_8 = \alpha f^3 f^7 f^4 f^8 f^5 \beta$. The matrices $f1 - f7$, $f11 - f13$ are inapplicable due to the absence of one of the markers $\$, f, f^3, f'', f^9, f^{10}$. The matrix $f10$ is inapplicable, since we do not the have the context $f^7 f^5$ in w_8. The only applicable matrices on w_8 are $f8$ (again) or $f9$.

Suppose $f8$ is applied again (repeatedly for $k \geq 0$ times) on w_8, then some k number of Bs in β are deleted freely and $(f^8)^k$ is inserted after $f^7 f^4$, leading to a string $w'_8 = \alpha f'f^7 f^4 (f^8)^k f^8 f^5 \beta'$, where $\beta' = \beta$ without some k occurrences of B. For further continuation of the derivation, the only other applicable matrix is $f9$. Since there is a unique occurrence of f^4, the matrix $f9$ can be applied only once, which will delete only one f^8, leading to the string $w'_9 = \alpha f'f^7 (f^8)^k f^5 \beta$. Due to the absence of any of $\$, f, f'', f^3, f^4, f^9, f^{10}$, no matrix is applicable on w'_9 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 $w_9 = w'_9 = \alpha f'f^7 f^5 \beta$.

The matrices $f1 - f9, f11 - f13$ are clearly inapplicable on w_9 due to the absence of the markers $\$, f, f', f'', f^4, f^9, f^{10}$. So we could now apply $f10$ on w_9, then we will have $w_{10} = \alpha f'f^7 f^9 \beta$. Due to absence of essential markers, all matrices except $f11$ and $f12$ are inapplicable. If the latter is applied, then we get $w'_{10} = \alpha f^7 \beta$ and the derivation is stuck as no continuation is possible. In particular, we cannot apply matrix $f7$ due to the absence of f, a marker that we cannot introduce by applying $f1$ due to the absence of the marker $\$. So, $f11$ is applied onto w_{10} and we get $w_{11} = \alpha f'f^9 f^{10} \beta$. Now, the only applicable matrices are $f12$ and $f13$. Both the rules can be applied in any order (independently) to get $w' = \$ \sqcup\!\sqcup \alpha\beta$. Observe that there are also scenarios where only $f13$ is applied, so that we stick with $w'' = \$ \sqcup\!\sqcup \alpha f' f^9 \beta$. Now, we can indeed start a new cycle of simulation with $f1$ and $f2$, but then at latest, in order to find the substring $f''B$ as required by $f3$, we should apply $f12$ to connect α and β.

Let us finally return to the situation when we try to apply $g1$ on $w^\$$, with $w = \alpha A \zeta \beta$ for some $\alpha \in \{A, C\}^*$, $\zeta \in \{B, D\}$ and $\beta \in \{B, D\}^* T^*$, 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 $w^\$$ derivable in Γ. If $w \Rightarrow_{g1} w'$, then $w' \in g \sqcup\!\sqcup w$. If we now apply $g2$, this means that we have split α as $\alpha' C \alpha''$, so that for $w' \Rightarrow_{g2} w''$, $w'' = \alpha' g g' C g'' \alpha'' A \zeta \beta$, with ζ and β 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 $g3$.

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 $\$ \sqcup\!\sqcup T^*$.

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 $\mathrm{MAT}(2; 1, 1, 1; 1, 0, 0) = \mathrm{RE}$.

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_{ll} and P_{rl} 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; 1, 0, 0) for simulating context-free rules of tsSGNF

$$p1 = [(\lambda,\$,\lambda)_{del},(X,p,\lambda)_{ins}]$$
$$p2 = [(X,p',p)_{ins},(\lambda,X,\lambda)_{del}]$$
$$p3 = [(p',b,p)_{ins},(\lambda,p',\lambda)_{del}]$$
$$p4 = [(b,Y,p,)_{ins}]$$
$$p5 = [(b,\$,Y)_{ins},(\lambda,p,\lambda)_{del}]$$

(a) Simulating $p : X \to bY$

$$q1 = [(\lambda,\$,\lambda)_{del},(X,q,\lambda)_{ins}]$$
$$q2 = [(X,q',q)_{ins},(\lambda,X,\lambda)_{del}]$$
$$q3 = [(q',Y,q)_{ins},(\lambda,q',\lambda)_{del}]$$
$$q4 = [(Y,b,q)_{ins}]$$
$$q5 = [(Y,\$,b)_{ins},(\lambda,q,\lambda)_{del}]$$

(b) Simulating $q : X \to Yb$

Fig. 6 Matrices with context-free deletion simulating $f : AB \to \lambda$ and $g : CD \to \lambda$

$$f1 = [(\lambda,\$,\lambda)_{del},(\lambda,f,\lambda)_{ins}]$$
$$f2 = [(f,f',A)_{ins},(B,f^3,\lambda)_{ins}]$$
$$f3 = [(A,f'',B)_{ins},(\lambda,f,\lambda)_{del}]$$
$$f4 = [(\lambda,A,\lambda)_{del},(f',f^4,f'')_{ins}]$$
$$f5 = [(\lambda,B,\lambda)_{del},(f'',f^5,f^3)_{ins}]$$
$$f6 = [(\lambda,f'',\lambda)_{del},(f^4,f^6,f^5)_{ins})]$$
$$f7 = [(f^6,f^7,f^5)_{ins},(\lambda,f^4,\lambda)_{del}]$$
$$f8 = [(\lambda,f',\lambda)_{del},(\lambda,f^3,\lambda)_{del}]$$
$$f9 = [(\lambda,f^5,\lambda)_{del},(\lambda,f^6,\lambda)_{del}]$$
$$f10 = [(\lambda,f^7,\lambda)_{del},(\lambda,\$,\lambda)_{ins}]$$

$$g1 = [(\lambda,\$,\lambda)_{del},(\lambda,g,\lambda)_{ins}]$$
$$g2 = [(g,g',C)_{ins},(D,g^3,\lambda)_{ins}]$$
$$g3 = [(C,g'',D)_{ins},(\lambda,g,\lambda)_{del}]$$
$$g4 = [(\lambda,C,\lambda)_{del},(g',g^4,g'')_{ins}]$$
$$g5 = [(\lambda,D,\lambda)_{del},(g'',g^5,g^3)_{ins}]$$
$$g6 = [(\lambda,g'',\lambda)_{del},(g^4,g^6,g^5)_{ins})]$$
$$g7 = [(g^6,g^7,g^5)_{ins},(\lambda,g^4,\lambda)_{del}]$$
$$g8 = [(\lambda,g',\lambda)_{del},(\lambda,g^3,\lambda)_{del}]$$
$$g9 = [(\lambda,g^5,\lambda)_{del},(\lambda,g^6,\lambda)_{del}]$$
$$g10 = [(\lambda,g^7,\lambda)_{del},(\lambda,\$,\lambda)_{ins}]$$

system $\Gamma = (V,T,\{\$S\},M)$ with alphabet set $V = N \cup T \cup K$, where

$$K = \{p,p' \mid p \in P_{rl}\} \cup \{q,q' \mid q \in P_{ll}\} \cup \{\$\} \cup \{f,f',f'',f^3,\ldots,f^7,g,g',g'',g^3,\ldots,g^7\}$$

The set of matrices M of Γ consists of the matrices described in Fig. 5a, b and in Fig. 6.

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

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

nonterminal X, we have $w_2 = \alpha p' p \beta$. Applying matrices $p3$, $p4$, $p5$ in the specified order, we have the following:

$$w^{\$} \Rightarrow_{p1} \alpha X p \beta \Rightarrow_{p2} \alpha p' p \beta \Rightarrow_{p3} \alpha b p \beta \Rightarrow_{p4} \alpha b Y p \beta \Rightarrow_{p5} \alpha b \$ Y \beta.$$

We note that at every cycle of context-free rule simulation, the $\$$ is deleted at the beginning of the simulation and introduced at the end of the simulation so as to enable the start of next simulation cycle.

The phase transition rule $h : S' \to \lambda$ is simulated by applying $h1$, 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, Γ has to apply τ at the very end.

We now discuss Phase II in detail, focusing on $f : AB \to \lambda$. Let $w = \alpha AB\beta$ 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 $w^{\$} \in \$ \sqcup\!\sqcup w$ is derivable in Γ (by induction). We can now see that

$$\begin{aligned} w^{\$} &\Rightarrow_{f1} \alpha f AB\beta \Rightarrow_{f2} \alpha f f' AB f^3 \beta \Rightarrow_{f3} \alpha f' A f'' B f^3 \beta \\ &\Rightarrow_{f4} f' f^4 f'' B f^3 \beta \Rightarrow_{f5} \alpha f' f^4 f'' f^5 f^3 \beta \Rightarrow_{f6} \alpha f' f^4 f^6 f^5 f^3 \beta \\ &\Rightarrow_{f7} \alpha f' f^6 f^7 f^5 f^3 \beta \Rightarrow_{f8} \alpha f^6 f^7 f^5 \beta \Rightarrow_{f9} \alpha f^7 \beta \\ &\Rightarrow_{f10} \alpha w'^{\$} \in \$ \sqcup\!\sqcup \alpha\beta \end{aligned}$$

We now proceed to prove that $L(\Gamma) = L(G)$, starting with the inclusion $L(G) \subseteq L(\Gamma)$. Consider the string $w^{\$} \in \$ \sqcup\!\sqcup \alpha X \beta$ derivable from $\$S$ 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 $w_1 = \alpha X p \beta$. On applying matrix $p2$, which inserts a p' between X and p and then deletes the

The purpose of introducing a $\$$ in the last rule $f10$ or $g10$ is to enable another simulation of $AB \to \lambda$ or of $CD \to \lambda$. 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 τ, thereby terminating Phase II of 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 $\alpha \in \{A, C\}^*$, $\beta \in \{B, D\}^* T^*$ according to Remark 1. We know by our previous arguments that the variation $w^\$ \in \$ \sqcup\!\sqcup w$ can be derived in Γ. 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 $S' \to \lambda$ 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 $K \backslash \{\$\}$, we can focus on applying $p1$, $q1$ or $f1$. Here, we ignore the matrices $g1$, ..., $g10$ due to their similarity to the fi matrices. Also, the simulations of the p rules and q rules are very similar, so that we only discuss the first ones.

We now start to apply $p1$ on $w^\$$ which deletes the marker \$ and then inserts a p to the right of X. Hence, $w^\$ \Rightarrow_{p1} \alpha X p \beta$. The absence of $\$, p', Y$ 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 $w_2 = \alpha p' p \beta$. 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 bY 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 $p' \to b$ to w_2, yielding $w_3 = \alpha b p \beta$. The absence of $\$, X, p', Y$ forces us to apply the only applicable matrix $p4$ on w_3 that inserts a Y before p, yielding $w_4 = \alpha b Y p \beta$. The only applicable matrix is now $p5$ which inserts a \$ between the recently introduced bY 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 $w^\$ \in \$ \sqcup\!\sqcup w$ is by Γ. As we are using tsSGNF, this means that $w \in \{A, C\}^* \{B, D\}^* T^*$ by Remark 1.

First, assume that $w = \alpha A \zeta \beta$ for some $\alpha \in \{A, C\}^*$ and $\beta \in \{B, D\}^* T^*$, or $\zeta \in T$ and $\beta \in T^*$, or $w = \alpha A$ is a sentential form derivable in G, corresponding to some string $w^\$$ derivable in Γ. 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''). Also, no rule from the simulation of Phase I is applicable anymore due to the absence of symbols from $N^{(0)} \cup N'$. Also, by the very structure of the matrices simulating an f rule, no progress on a sentential form β 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 $f \sqcup\!\sqcup w$. 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^4. However, in order to introduce f^4, 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 αA of w_1, thus verifying the decomposition $w_1 = \alpha' f A \alpha'' \zeta \beta$, with $\alpha A = \alpha' A \alpha''$. 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^3 after any B, thus obtaining $w_2 = \alpha' f f' A \alpha'' \beta' B f^3 \beta''$, with $\zeta \beta = \beta' B \beta''$. The same matrix cannot be applied again, as the substring fA 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^3 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, α'' must end with A and β' 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 AD.

Hence, we have $w_3 = \alpha' f' A \alpha'' f'' \beta' B f^3 \beta''$. The absence of the contexts $\$, fA, AB$ 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 $\alpha' f' A \alpha'' f'' = \alpha' f' A f''$ in w_3, i.e., if $\alpha'' = \lambda$. Similarly, the matrix $f5$ demands that if a B is deleted randomly, then we should get the substring $f''f^3$ in the sentential form in order to apply the second rule of $f5$ and this is possible only if $f'' \beta' B f^3 \beta'' = f'' B f^3 \beta''$ in w_3, i.e., if $\beta' = \lambda$. In summary, we find that necessarily $w_3 = \alpha' f' A f'' B f^3 \beta''$. Now, matrices $f4$ and $f5$ are applied in any order. They basically simulate the rewriting rules $f'A \to f'f^4$ and $f''B \to f''f^5$, so that we get $w_4 = \alpha' f' f^4 f'' f^5 f^3 \beta''$. 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 $f4$ and $f5$ before proceeding.

Note that from now on the matrices $f4$ or $f5$ can be re-applied if and only if $f'f''$ or $f''f^3$ 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 $f4$ or $f5$, 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 $f'f''$ and $f''f^3$ is never a substring in the resulting sentential form, so as to avoid an unintended (re)application of rules $f4$, $f5$ which will delete random occurrences of A or B (see *ob**).

This danger is handled by replacing f'' with f^6 in matrix $f6$. One may wonder what if the matrix $f6$ was never applied to the sentential form $w_4 = \alpha' f' f^4 f'' f^5 f^3 \beta''$. Due to *ob**, the only matrices that are applicable to w_4 are $f6$ and $f8$. If $f6$ was avoided and $f8$ was applied, then the markers f', f^3 are deleted, leaving behind $w_5 = \alpha' f^4 f'' f^5 \beta''$. At this point, no matrix is applicable except $f6$. Hence the matrix $f6$ is somehow enforced to be applied on w_4 or w_5 which replaces the marker f'' as f^6. Due to absence of f'' and *ob**, matrices $f4$ and $f5$ can never be re-applied. If f'' has to be introduced again (using matrix $f3$), 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 $f7$ is to make sure that f^4 and f^5 are deleted in different matrices. The markers are deleted using matrices $f8$ to $f10$ in any desired (applicable) order yielding $w_9^{\$} = \$ \sqcup\!\sqcup \alpha' \beta''$. Three such derivations are shown below.

- $w_4 = \alpha' f' f^4 f'' f^5 f^3 \beta'' \Rightarrow_{f6} \alpha' f' f^4 f^6 f^5 f^3 \beta'' \Rightarrow_{f7} \alpha' f' f^6 f^7 f^5 f^3 \beta'' \Rightarrow_{f8}$
  $\alpha' f^6 f^7 f^5 \beta'' \Rightarrow_{f9} \alpha' f^7 \beta'' \Rightarrow_{f10} w_9^{\$}$.
- $w_4 = \alpha' f' f^4 f'' f^5 f^3 \beta'' \Rightarrow_{f8} w_5 = \alpha' f^4 f'' f^5 \beta'' \Rightarrow_{f6} \alpha' f^4 f^6 f^5 \beta'' \Rightarrow_{f7}$
  $\alpha' f^6 f^7 f^5 \beta'' \Rightarrow_{f9} \alpha' f^7 \beta'' \Rightarrow_{f10} w_9^{\$}$.
- $w_4 = \alpha' f' f^4 f'' f^5 f^3 \beta'' \Rightarrow_{f6} \alpha' f' f^4 f^6 f^5 f^3 \beta'' \Rightarrow_{f7} \alpha' f' f^6 f^7 f^5 f^3 \beta'' \Rightarrow_{f9} \alpha' f' f^7 f^3 \beta'' \Rightarrow_{f8} \alpha' f^7 \beta'' \Rightarrow_{f10} w_9^{\$}$.

Let us finally return to the situation when we try to apply $g1$ on $w^{\$}$, with $w = \alpha A \zeta \beta$ for some $\alpha \in \{A, C\}^*$, $\zeta \in \{B, D\}$ and $\beta \in \{B, D\}^* T^*$, 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 $w^{\$}$ derivable in Γ. If $w \Rightarrow_{g1} w'$, then $w' \in g \sqcup\!\sqcup w$. If we now apply $g2$, this means that we have split α as $\alpha' C \alpha''$ and $\zeta\beta$ like $\beta' D \beta''$, with ζ and β satisfying the conditions stated above, so that for $w' \Rightarrow_{g2} w''$, $w'' = \alpha' g g' C \alpha'' A \beta' D g^3 \beta''$. However, now the derivation is stuck, as w'' contains no substring CD as required by matrix $g3$.

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^6, f^7, f^8 and their relevant insertion and deletion rules are not necessary. One idea would be to construct matrices like $f7' = [(\lambda, f', \lambda)_{del}, (\lambda, f'', \lambda)_{del})]$ and $f8' = [(\lambda, f^3, \lambda)_{del}, (\lambda, \$, \lambda)_{ins}]$. The problem is that however, such ideas might not work as one can start to apply these matrices $f7'$ and $f8'$ soon after applying $f3$, leading to unintended situations. This would make a correctness proof more tedious if not impossible at all. With the introduction of the markers f^6, 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 (2; 1, 1, 1; 1, 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, 1, 0; 1, 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.*

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

For quick reference, we present the matrix ins-del rules of $\mathrm{MAT}(3; 1, 1, 0; 1, 0, 0)$ that simulates the linear rules $p : X \to aY$, $q : X \to Ya$, $f : X \to \lambda$ in Fig. 7a and the matrix ins-del rules of $\mathrm{MAT}(2; 2, 1, 0; 1, 0, 0)$ 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 $(k; n, i', i''; m, j' j'')$ describing LIN, then the same system will describe $\mathcal{L}_{reg}(\mathrm{LIN})$.

Theorem 5 *For all integers* $n, m \geq 1$, $t \geq 2$ *and* $i', i'', j', j'' \geq 0$ *with* $t + n \geq 4$ *and* $i' + i'' \geq 1$, *if every* $L \in \mathrm{LIN}$ *can be generated by a* $\mathrm{MAT}(t; n, i', i''; m, j', j'')$ *system*

Fig. 7 Matrix ins-del system describing LIN (from Fernau et al. (2018b))

$$\begin{array}{lcl} p1 & = & [(X,p,\lambda)_{ins},\ (\lambda,X,\lambda)_{del}] \\ p2 & = & [(p,Y,\lambda)_{ins},\ (p,a,\lambda)_{ins},\ (\lambda,p,\lambda)_{del}] \\ q1 & = & [(X,q,\lambda)_{ins},\ (\lambda,X,\lambda)_{del}] \\ q2 & = & [(q,a,\lambda)_{ins},\ (q,Y,\lambda)_{ins},\ (\lambda,q,\lambda)_{del}] \\ f & = & [(\lambda,X,\lambda)_{del}] \end{array}$$

(a) $\mathrm{MAT}(3;1,1,0;1,0,0)$

$$\begin{array}{lcl} p1 & = & [(X,p,\lambda)_{ins},\ (\lambda,X,\lambda)_{del}] \\ p2 & = & [(p,aY,\lambda)_{ins},\ (\lambda,p,\lambda)_{del}] \\ q1 & = & [(X,q,\lambda)_{ins},\ (\lambda,X,\lambda)_{del}] \\ q2 & = & [(q,Ya,\lambda)_{ins},\ (\lambda,q,\lambda)_{del}] \\ f & = & [(\lambda,X,\lambda)_{del}] \end{array}$$

(b) $\mathrm{MAT}(2;2,1,0;1,0,0)$

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

Proof Let $L \in \mathcal{L}_{reg}(\mathrm{LIN})$ 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 $G_R = (N_0, N'', S, P_0)$ and linear grammars $G_i = (N_i, T, S_i, P_i)$ for $1 \le i \le k$. For technical reasons that should become clear soon, we rather consider $G'_i = (N'_i, T, S_i, P'_i)$, where $N'_i = N_i \cup \{\langle S_i, A\rangle \mid A \in N_0\}$ and P'_i contains, besides all rules from P_i, rules of the form $\langle S_i, A\rangle \to w$ whenever $S_i \to w \in P_i$ 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 $L((N'_i, T, \langle S_i, A\rangle, P'_i)) = L(G_i)$ for any $A \in N_0$.

Since $\mathrm{LIN} \subseteq \mathrm{MAT}(t;n,i',i'';m,j',j'')$, each G'_i can be simulated by a matrix ins-del system $\Gamma_i = (V_i, T, \{S_i\}, R_i)$ for $1 \le i \le k$, each of size $(t;n,i',i'';m,j',j'')$. We assume, without loss of generality, that $V_i \cap V_j = T$ if $1 \le i < j \le k$. Let us first consider the case $i' \ge 1$ and $i'' = 0$. We construct a matrix ins-del system Γ for G as follows[4]: $\Gamma = (V, T, \{\langle S_i, A\rangle A' \mid S \to S_i A \in P\}, R \cup R')$, where

$$V = \left(\bigcup_{i=1}^{k}(V_i \cup \{\langle S_i, A\rangle \mid A \in N_0\})\right) \cup N_0 \cup \{A' \mid A \in N_0\};$$

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

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

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

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 $i'' \ge 1$ follows from Propositions 2 and 3. □

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.*

- $\mathcal{L}_{reg}(\mathrm{LIN}) \subsetneq \mathrm{MAT}(3;1,1,0;1,0,0) \cap \mathrm{MAT}(3;1,0,1;1,0,0)$,
- $\mathcal{L}_{reg}(\mathrm{LIN}) \subsetneq \mathrm{MAT}(2;2,1,0;1,0,0) \cap \mathrm{MAT}(2;2,0,1;1,0,0)$. □

5 Describing regular languages

Petre 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 $p : X \to bY$ and $X \to \lambda$. The non-context rules of tsSGNF namely $AB \to \lambda$ and $CD \to \lambda$ can be simulated by rules of (2; 1, 0, 0; 1, 1, 0) (see Fig. 2). So, if we are able to show that $q : X \to Yb$ can also be simulated by some set of rules of size (2; 1, 1, 0; 1, 1, 0), then $\mathrm{MAT}(2;1,1,0;1,1,0)$ will be computationally complete. However, unfortunately, the simulation of $q : X \to Yb$ 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 $\mathrm{REG} \subsetneq \mathrm{MAT}(2;1,1,0;1,1,0) \cap \mathrm{MAT}(2;1,0,1;1,0,1)$.

[3] The technical condition on MAT ins-del systems is not that severe, as we can always take a new start symbol and first generate any finite set with the resources at hand.

[4] There is one subtlety with the case when $\lambda \in L(G)$: in that case, λ should be added as an axiom of Γ.

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

$$V = N \cup T \cup \{p, p', p'', p'''\}.$$

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

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

$$w \Rightarrow_{p1} \alpha X p' p \Rightarrow_{p2} \alpha p'' p' p \Rightarrow_{p3} p''' \sqcup \alpha p'' p \Rightarrow_{p4} \alpha p''' bp \Rightarrow_{p5} \alpha p''' bY \Rightarrow_{p6} w'.$$

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

Consider a string $w = \alpha X$ that is derivable from S in Γ. 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 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 $w_1(n) = \alpha X(p'p)^n, n \geq 1$. 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 $w_2(n) = \alpha p''(p'p)^n$. 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 $p''' \sqcup \alpha p'' p(p'p)^{n-1}$. 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 $w'_3(n) = \alpha p'' p \tau_n$. 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 $w_3(n) = \alpha p''' p'' p \tau_n$. 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 $w_4(n) = \alpha p''' bp \tau_n$. 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 $w_5(n) = \alpha p''' bY \tau_n$. The absence of X, p'', bp enforces the application of matrix $p6$ which deletes the marker p''', thus yielding $w_6(n) = \alpha bY \tau_n$. 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 r''' 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 $w'_5(n) = \alpha bp \tau_n$. The absence of X, p'', p''' enforces the

$$
\begin{array}{lcl}
p1 & = & [(X,p,\lambda)_{ins},(X,p',\lambda)_{ins}] \\
p2 & = & [(X,p'',\lambda)_{ins},(\lambda,X,\lambda)_{del}] \\
p3 & = & [(\lambda,p''',\lambda)_{ins},(p'',p',\lambda)_{del}] \\
p4 & = & [(p''',p'',\lambda)_{del},(p''',b,\lambda)_{ins}] \\
p5 & = & [(b,Y,\lambda)_{ins},(Y,p,\lambda)_{del}] \\
p6 & = & [(\lambda,p''',\lambda)_{del}]
\end{array}
$$

(a) Simulating $p : X \to bY$

$$r1 \quad = \quad [(\lambda,X,\lambda)_{del}]$$

(b) Simulating $r : X \to \lambda$

Fig. 8 Matrices of size (2; 1, 1, 0; 1, 1, 0) for simulating right-linear rules $X \to$ and $X \to \lambda$

deterministic application of matrix $p5$ which yields $w_6(n) = \alpha bY\tau_n$.

Notice that should we "forget" to apply matrix $p6$, 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 $p4$ to it, because then there would not be any b left to Y as required by matrix $p5$.

After applying both matrices, we get $w_6(n) = \alpha bY\tau_n$. Since no markers p', p in τ_n can be deleted hereafter, this enforces that the matrix $p1$ was applied exactly once, i.e., $n = 1$, and hence $\tau_n = \lambda$.

The singleton rule shown in Fig. 8b clearly simulate the rule $r : X \to \lambda$. The above arguments show the first inclusion $\mathrm{REG} \subseteq \mathrm{MAT}(2;1,1,0;1,1,0)$. 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 $\mathrm{MAT}(2;2,1,0;1,0,0)$ closed under reversal? If this were true, then $\mathrm{MAT}(2;2,1,0;1,0,0) = \mathrm{MAT}(2;2,0,1;1,0,0)$, 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.

Acknowledgements We are grateful to the referees of this paper for spotting some errors in one of the results of the previous version of the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Alhazov A, Krassovitskiy A, Rogozhin Y, Verlan S (2011) P systems with minimal insertion and deletion. Theor Comput Sci 412(1–2):136–144

Beaver D (1995) Computing with DNA. J Comput Biol 2(1):1–7

Dassow J, Păun Gh (1989) Regulated rewriting in formal language theory, volume 18 of EATCS monographs in theoretical computer science. Springer

Fernau H, Kuppusamy L (2017) Parikh images of 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) Descriptional 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 descriptional complexity of formal systems, DCFS, volume 31 of EPTCS, pp 88–98

Galiukschov BS (1981) Semicontextual grammars. Mat. logica 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

Krassovitskiy 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

Kuppusamy 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, Gamper J, Guerrini G, Jurdzinski 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