Descriptional Complexity of Matrix Simple Semi-conditional Grammars

Henning Fernau, Lakshmanan Kuppusamy, Indhumathi Raman

Descriptional Complexity of Matrix Simple Semi-Conditional Grammars

Henning Fernau[1][0000−0002−4444−3220],
Lakshmanan Kuppusamy[2][0000−0003−2358−905X], and
Indhumathi Raman[3][0000−0002−0981−9165]

[1] Fachbereich 4 – Abteilung Informatikwissenschaften, CIRT
Universität Trier, D-54286 Trier, Germany. fernau@uni-trier.de
[2] School of Computer Science and Engineering
VIT, Vellore-632 014, India. klakshma@vit.ac.in
[3] Department of Applied Mathematics and Computational Sciences
PSG College of Technology, Coimbatore-641 004, India. ind.amcs@psgtech.ac.in

Abstract. Matrix grammars are one of the first approaches ever proposed in regulated rewriting, prescribing that rules have to be applied in a certain order. Typical descriptional complexity measures incorporate the number of nonterminals or the length, i.e., the number of rules per matrix. In simple semi-conditional (SSC) grammars, the derivations are controlled by a permitting string or by a forbidden string associated to each rule. The maximum length i of permitting strings and the maximum length j of forbidden strings are called the degree of such grammars. Matrix SSC grammars (MSSC) put matrix grammar control on SSC rules. We consider the computational completeness of MSSC grammars with degrees $(2,1)$, $(2,0)$ and $(3,0)$. The results are important in the following aspects. (i) With permitting strings alone, it is unknown if SSC grammars are computational complete, while MSSC grammars describe RE even with severe further restrictions on their descriptional complexity. (ii) Matrix grammars with appearance checking with three nonterminals are computationally complete; however, the length is unbounded. With our constructions for MSSC grammars, we can even bound the length.

Keywords: simple semi-conditional grammars · matrix grammars · computational completeness · Geffert normal forms · descriptional complexity

1 Introduction

Matrix grammars (introduced by S. Ábrahám [1]) are regulated grammars in which rules are grouped into finite sequences called matrices. When a matrix is chosen to be applied, all rules in the sequence are applied in the given order. In semi-conditional (SC) grammars (introduced by Gh. Păun [15]), each rule is associated with two strings called the permitting string and the forbidden string. A rule can be applied to a sentential form w only if w contains the permitting string and does not contain the forbidden string as a subword. If both these control strings (permitting and forbidden) are absent in a rule, then the rule is

called *unconditional*; otherwise, the rule is termed *conditional*. The most interesting case is when the involved rewriting rules are context-free ; we will focus on this case in the following. Clearly, these are the rules which are responsible for a SC grammar to characterize the class of recursively enumerable languages, henceforth denoted RE [11,13,16]. With semi-conditional grammars, two variants are of special interest: (i) *simple semi-conditional grammars* (denoted as SSCG) in which at most either the permitting string or the forbidden string is present for each rule, see [11]; (ii) *permitting grammars* in which the forbidden string is absent for every rule, see [6]. Key observations in these domains include:

- Matrix grammars with appearance checking, having three nonterminals, are computationally complete, i.e., they characterize RE [3]. However, the lengths of the matrices are unbounded. It is not clear how to restrict the length while still bounding the number of nonterminals. It is known that matrix grammars without appearance checking are not computationally complete; see [9].
- The generative power of permitting grammars with no erasing rules is strictly included in the class of context-sensitive languages; refer to [6]. It is open if permitting grammars with erasing context-free rules describe RE.
- Results on the computational completeness of simple semi-conditional grammars are tabulated in Figure 1(a). It is unknown, for example, if five nonterminals or five conditional rules suffice to describe RE.
- Matrix simple semi-conditional (MSSC) grammars, matrix controlled grammars with SSC rules (introduced in [12]) have been known to characterize RE while limiting several descriptional complexity measures at the same time. These devices are in the focus of our paper.

The results of the paper are tabulated in Figure 1(b), including results from [12]. Our results clearly improve on the previously published ones. The many results we obtained can be viewed as trade-off results between different measures of descriptional complexity. We highlight the record-holders of single measures by putting them in bold-face in the table.

2 Preliminaries and Definitions

In this paper, it is assumed that the reader is familiar with the fundamentals of language theory and mathematics in general. Let Σ^* denote the free monoid generated by a finite set Σ called the alphabet under an operation termed concatenation. Any element of Σ^* is called a *word* or string (over Σ), while the *empty word* λ is the unit of Σ^*. Any subset of Σ^* is called a language. A word v is a *subword* (or substring) of $x \in \Sigma^*$ if there are words u, w such that $x = uvw$. Let $sub(x) \subseteq \Sigma^*$ denote the set of all subwords of $x \in \Sigma^*$. Clearly, $sub(x)$ is a finite language. Given a word $w \in \Sigma^*$, $|w|$ represents the length of w. Recall that a type-0 grammar can be specified by a quadruple $G = (N, T, S, P)$, where N is the *nonterminal* alphabet, T is the *terminal* alphabet, $S \in N$ is the *start symbol* and P is a finite set of re-write rules of the form $x \to y$, with $x, y \in (N \cup T)^*$. Type-0 grammars characterize the class RE of recursively enumerable languages. Rule $x \to y$ is *context-free* if $x \in N$.

Degree	#NT	#CR	Ref.
$(2,1)$	13	12	[13]
$(2,1)$	12	10	[16]
$(2,1)$	10	9	[10]
$(2,1)$	9	8	[4]
$(3,1)$	11	8	[16]
$(3,1)$	9	8	[14]
$(3,1)$	7	7	[4]
$(4,1)$	7	6	[4]
$(4,1)$	6	8	[4]

(a) Descriptional complexity measures of SSC grammars describing RE

Degree	#NT	#CR	#MAT	LEN	Ref.
$(2,1)$	6	6	2	4	[12]
$(3,1)$	7	4	**1**	6	[12]
$(2,1)$	6	4	2	**2**	Thm. 1
$(2,1)$	6	3	2	3	Thm. 2
$(2,1)$	6	3	**1**	4	Thm. 3
$(2,1)$	5	4	**1**	4	Thm. 5
$(2,1)$	5	3	3	**2**	Thm. 7
$(2,0)$	6	**2**	**1**	5	Thm. 4
$(2,0)$	5	3	**1**	5	Thm. 6
$(2,0)$	5	5	3	**2**	Thm. 8
$(2,0)$	5	**2**	2	4	Thm. 9
$(2,0)$	**4**	6	2	7	Thm. 13
$(3,0)$	**4**	5	3	3	Thm. 10
$(3,0)$	**4**	7	4	**2**	Thm. 11
$(3,0)$	**4**	3	2	5	Thm. 12

(b) Descriptional complexity measures of MSSC grammars describing RE

Fig. 1. Summary of computational completeness results for SSC and MSSC grammars; #NT, #CR and #MAT denote the number of nonterminals, the number of conditional rules, and the number of conditional matrices, respectively. LEN gives an upper bound on the lengths of matrices.

2.1 Matrix (and) Semi-conditional Grammars

Matrix and Semi-conditional grammars have been introduced within the area of regulated rewriting [2,15], mostly to enhance the power of context-free grammars beyond context-free languages. Matrix control allows to express that some rules have to be applied in a certain order, while semi-conditional control attaches permitting and forbidden strings to rules that confine the applicability of these rules. In the special case of simple semi-conditional grammars [11], only either permitting or forbidden strings may be present.

A matrix simple semi-conditional grammar (MSSC grammar for short) combines simple semi-conditional grammars and matrix grammars [12].

Definition 1. *An* MSSC grammar *is a quadruple $G = (N, T, S, M)$, where N, T and S have the same meaning as in a type-0 grammar and M is a finite set of sequences (called* matrices*) of the form*

$$m = [(A_1 \to x_1, P_1, F_1), \ldots, (A_\ell \to x_\ell, P_\ell, F_\ell)], \tag{1}$$

where $\ell \geq 1$, $A_k \in N$, $x_k \in (N \cup T)^*, P_k, F_k \in (N \cup T)^+ \cup \{0\}$ such that $P_k = 0$ or $F_k = 0$ for each $1 \leq k \leq \ell$. The strings P_k and F_k above are called the permitting *and* forbidding *conditions, respectively; 0 is a special symbol, $0 \notin N \cup T$. If $P_k = F_k = 0$, then rule $(A \to x_k, P_k, F_k)$ is called* unconditional*; otherwise, it is called* conditional*. Let c_m denote the number of conditional rules*

in matrix m from (1). *We call matrix m* conditional *if $c_m \geq 1$ and* unconditional *if $c_m = 0$. Moreover, m is called a* multi-production matrix *if $\ell \geq 2$ and is called a* single-production matrix *if $\ell = 1$. Number $\ell_m := \ell$ is termed the* length *of m.*

Reconsider m from (1). *Let $\alpha_0, \alpha_\ell \in (N \cup T)^*$. Then, $\alpha_0 \Rightarrow_m \alpha_\ell$ holds if there are strings $\alpha_1, \ldots, \alpha_{\ell-1} \in (N \cup T)^*$ such that, for all $k = 1, \ldots, \ell$, if $P_k \neq 0$, then $P_k \in sub(\alpha_{k-1})$, and if $F_k \neq 0$, then $F_k \notin sub(\alpha_{k-1})$, and furthermore, $\alpha_{k-1} = \alpha'_{k-1} A_k \alpha''_{k-1}$ and $\alpha_k = \alpha'_{k-1} x_k \alpha''_{k-1}$. Define $\Rightarrow_G = \bigcup_{m \in M} \Rightarrow_m$. Now, $L(G) = \{w \mid S \Rightarrow_G^* w \wedge w \in T^*\}$ is the* language *described by G.*

Observe that all conditions have to be met to successfully apply a matrix to a string; otherwise, the derivation does not succeed. More classical matrix grammars (without appearance checking) can be viewed as MSSC grammars where all matrices are unconditional. Likewise, simple semi-conditional grammars (abbreviated as SSC(G)) correspond to MSSC grammars where all matrices have length one. We now define six measures on the descriptional complexity of MSCC grammars that are crucial for our further studies.

Definition 2. *An MSSC grammar $G = (N, T, S, M)$ is of* size $(i, j; n; c, p, l)$, *if in every rule $(A \to x, \alpha, \beta)$ of a matrix $m \in M$, we have*

- $|\alpha| \leq i$ *and* $|\beta| \leq j$, *(the pair (i, j) is also called the* degree *of G);*[4]
- $|N| \leq n$ *(an upper bound on the number of **n**onterminals);*
- $\sum_{m \in M} c_m \leq c$ *(bounding the total number of **c**onditional rules in G);*
- $|\{m \in M \mid \ell_m > 1\}| \leq p$ *(upper-bounding the number of multi-**p**roduction matrices in G);*
- $\max\{\ell_m \mid m \in M\} \leq l$ *(bounding the matrix lengths in G).*

We denote by $\mathsf{MSSC}(i, j; n; c, p, l)$ the family of languages generated by MSSC grammars of size $(i, j; n; c, p, l)$. If the forbidding conditions are absent in every rule of M, then the degree of the system is $(i, 0)$ and is denoted as *matrix permitting grammar* (or *MP grammar* for short). For brevity, we simplify the notation $(A \to x, \alpha, 0)$ to $(A \to x, \alpha)$ and $(A \to x, 0, 0)$ to $(A \to x)$. For MP grammars, we abbreviate for the corresponding classes of languages $\mathsf{MP}(i; n; c, p, l) = \mathsf{MSSC}(i, 0; n; c, p, l)$. If $l = 1$, then $p = 0$, and (as noticed above), we rather face SSC grammars, so that we could write $\mathsf{MSSC}(i, j; n; c, 0, 1) = \mathsf{SSC}(i, j; n, c)$.

2.2 Geffert Normal Forms

In [8], quite a number of normal forms for type-0 grammars have been derived. They all differ by the number of nonterminals that are used and also by the number of non-context-free rules. We will hence speak of (n, r)-GNF to refer to a Geffert normal form with n nonterminals and r non-context-free rules. However, all these normal forms characterize the class of recursively enumerable languages, or RE languages for short. The best known normal form is the $(5, 2)$-GNF with nonterminals S (start symbol) and A, B, C, D that uses context-free rules with

[4] Here, the convention $|0| = 0$ applies.

S as its left-hand side in its first phase; after using the context-free rules, in a second phase non-context-free erasing rules $AB \to \lambda$ and $CD \to \lambda$ are applied to finally derive a string $t \in T^*$. Hence, the derivation in a grammar in $(5,2)$-GNF proceeds in two phases, where the first phase splits into two stages. In phase one, stage one, rules of the form $S \to uSa$ are used, with $u \in \{A,C\}^*$, $a \in T$. In stage two, rules of the form $S \to uSv$ are used, with $u \in \{A,C\}^*$ and $v \in \{B,D\}^*$. It is also shown in [7] that any attempt to mix the applications of rules of these two types cannot yield to a terminal string in view of the chosen encodings. Also, rules of the form $S \to uv$ are available that prepare the transition into phase two, where (i.e., in phase two) the erasing non-context-free rules are used exclusively. The normal form variations we discuss next are always derived from $(5,2)$-GNF by applying morphisms to all context-free rules, where in particular $S \mapsto S$ maintains the start symbol, so that in particular the rules involving the start symbol keep up the same form as with $(5,2)$-GNF, without further mentioning this below.

GNF with 4 nonterminals We now discuss another normal form of type-0 grammars due to Geffert [8]. We then list some important properties of this normal form that are discussed and proved in [4,5]. These follow from the constructions of the normal form and well-known properties of $(5,2)$-GNF.

The normal form $(4,1)$-GNF is obtained from $(5,2)$-GNF normal form (using nonterminals S, A, B, C, D) by applying the morphism $A \mapsto AB$, $B \mapsto C$, $C \mapsto A$ and $D \mapsto BC$ to all context-free rules. Moreover, we add one non-context-free erasing rule $ABC \to \lambda$. This means that the following properties hold:

Proposition 1. *[5] The following properties hold for $(4,1)$-GNF grammars:*

1. *If $S \Rightarrow^* w$, then $w \in \{A, AB\}^*\{S,\lambda\}\{BC,C\}^*(T(\{BC,C\} \cup T)^* \cup \{\lambda\})$.*
2. *If $S \Rightarrow^* w$, then $sub(w) \cap (\{BBB\} \cup \{C\}\{B\}^*\{A\}) = \emptyset$.*
3. *If $S \Rightarrow^* w$, with $w = w't$, where $w' \in \{A, AB\}^+\{BC,C\}^+$, $t \in (T(\{BC,C\} \cup T)^* \cup \{\lambda\})$, then w' contains exactly one occurrence from $\{ABC, AC, ABBC\}$ as a substring. We refer to this substring as the* central part *of w.*
   Only with ABC as central part, possibly $w' \Rightarrow^* \lambda$ as intended in a derivation that yields a terminal string. If AC or $ABBC$ occur as a central part of the string, then it will not derive to a terminal string.
4. *If $S \Rightarrow^* w$, with $w = w't$, where $w' \in \{A, AB\}^+ \cup \{BC,C\}^+$ and $t \in (T(\{BC,C\} \cup T)^* \cup \{\lambda\})$, then w does not derive any terminal string.*
5. *Again, the derivation proceeds in two phases, the first one is split into two stages. Only in phase two (where non-context-free erasing rules are applied), a central part will appear.*

The normal form $(4,2)$-GNF is obtained from $(5,2)$-GNF (using nonterminals S, A, B, C, D) by applying the morphism $A \mapsto CAA$, $B \mapsto BBC$, $C \mapsto CA$ and $D \mapsto BC$ to all context-free rules. Moreover, the two non-context-free erasing rules $AB \to \lambda$ and $CC \to \lambda$ are added. Then if $S \Rightarrow^* w$, then $w \in \{CA, CAA\}^*\{S,\lambda\}\{BC, BBC\}^*(T(\{BC, BBC\} \cup T)^* \cup \{\lambda\})$.

Remark 1. Though the *central part* is either AB or CC, unwanted strings like AC or CB can appear in the center; the derivation is stuck in these cases. Similar properties as stated in the previous proposition can be derived for $(4,2)$-GNF.

GNF with 3 nonterminals In [8], it was also proved that every RE language is generated by a type-0 grammar with three nonterminals only. The context-free rules of this $(3,1)$-GNF normal form (where $N = \{S, A, B\}$) are obtained from $(5,2)$-GNF by applying the morphism $A \mapsto AB$, $B \mapsto BBA$, $C \mapsto ABB$ and $D \mapsto BA$. The only non-context-free rule is $ABBBA \to \lambda$. It is sometimes more practical to work with the two non-context-free erasing rule $BBB \to \lambda$ and $AA \to \lambda$ instead, leading to a grammar in $(3,2)$-GNF.

Remark 2. The *central part* for $(3,1)$-GNF or $(3,2)$-GNF is either BBB or AA, but unwanted strings like $ABBA$ or $ABBBBA$ are possible in the center.

3 Computational completeness of MSSC grammars

In the proofs, we are making use of several variations of Geffert normal form as discussed above, starting with $(4,1)$-GNF.

Theorem 1. $\mathsf{MSSC}(2,1;6;4,2,2) = \mathsf{RE}$.

Proof. Let $L \in$ RE be generated by a grammar in $(4,1)$-GNF of the form $G = (N, T, P \cup \{ABC \to \lambda\}, S)$ such that P contains only context-free productions and $N = \{S, A, B, C\}$ (see Prop. 1). Next, we define the MSSC grammar $G' = (N', T, P' \cup P'', S)$, where $N' = N \cup \{A', C'\}$ (assuming that $\{A', C'\} \cap N = \emptyset$), P' contains (single-production) matrices of the form $[(S \to \alpha, 0, 0)]$ whenever $S \to \alpha \in P$ and P'' contains the following two (multi-production) matrices plus the (single-production) matrix $m3 = [(C' \to \lambda, 0, A')]$:

$$m1 = [(A \to A', 0, C'), (C \to C')]\,,$$
$$m2 = [(B \to \lambda, A'B), (A' \to \lambda, A'C')]\,.$$

We now show that $L(G') = L(G)$.

First, it is clear that context-free rules like $S \to \alpha \in P$ can be easily simulated by single-production matrices of the form $[(S \to \alpha, 0, 0)]$ and vice versa. In fact, it could be that $m1$ is applied in-between applying matrices $[(S \to \alpha, 0, 0)]$, but then only matrices $[(S \to \alpha, 0, 0)]$ can apply until switching to Phase two, where $m2$ might follow, which is discussed in detail next, assuming that $m1$ would have been applied last, not changing the resulting sentential form.

According to Prop. 1, then (after Phase one of the GNF grammar), ignoring boundary cases, we face a string of the form $\alpha\xi\beta t$, where $\alpha \in \{A, AB\}^*$, $\xi \in \{ABC, AC, ABBC\}$, $\beta \in \{B, BC\}^*$ and $t \in T(\{B, BC\} \cup T)^* \cup \{\lambda\}$. Assume that $\xi = ABC$, i.e., we are still in the situation that (in case $t \in T^*$) $\alpha\xi\beta t$ might derive a terminal string in G. Applying $ABC \to \lambda$ in G, we could arrive

at $\alpha\beta t$. This can be simulated by $\alpha\xi\beta t \Rightarrow_{m1} \alpha A'BC'\beta t \Rightarrow_{m2} \alpha C'\beta t \Rightarrow_{m3} \alpha\beta t$. By induction, this shows that $L(G) \subseteq L(G')$.

We are now finishing the proof of the converse direction $L(G) \supseteq L(G')$ by explaining the decisive induction step, starting out with some $\alpha\xi\beta t$ (as above) that is derivable both in G and in G'. If $\alpha\xi\beta t$ is the current sentential form in G' with $\xi = ABC$, then clearly only $m1$ applies (due to the absence of primed symbols). This means that any occurrence of A (which must be in $\alpha\xi$) and any occurrence of C (from $\xi\beta t$) is turned into the primed counterparts. Now, the presence of A' and C' prevents applying $m1$ or $m3$, forcing us into applying matrix $m2$ next. Now, the context checks become important. Both rules check for the presence of a certain symbol to the right of A'. As both checked symbols are different and there is only one occurrence of A' in the sentential form, the first rule in $m2$ must remove the occurrence of B to the right of A'. Moreover, to the right of that occurrence of B, C' must occur in the sentential form, which enforces that in fact the A occurring in ξ must have been turned into A' when applying $m1$, and likewise the C occurring in ξ must have been turned into C' before. In other words, we know that $\alpha ABC\beta t \Rightarrow_{m1} \alpha A'BC'\beta t \Rightarrow_{m2} \alpha C'\beta t$. Due to the presence of C' and the absence of A', $m3$ is the only applicable rule, leading to $\alpha\beta t$ as intended.

However, there are also situations to study where we know that within G, there will be no terminal string derivable at all, while this might happen within the simulated grammar G'.

Case 1. Consider the sentential form $\alpha AC\beta t$. Now, we could apply $m1$, turning one occurrence of A within αA into A' and one occurrence of C within $C\beta t$ into C'. Now, $m2$ is the only applicable rule. In order to apply the first rule in $m2$, B must have been sitting to the right of the A-occurrence that we have replaced when applying $m1$, but this means that now we find A' within the part of the word previously called α, where no C- or C'-occurrences are to be found, which would be necessary to apply the second rule of $m2$. In a sense, the A-occurrence in the central part AC serves as a barrier between the possible substrings AB within α and the C-ocurrences to match with. Hence, the derivation gets stuck.

Case 2. Consider the sentential form $\alpha ABBC\beta t$. Now, again $m1$ is the only applicable matrix. Turning one occurrence of A within αA into A' and one occurrence of C within $C\beta t$ into C'. With a similar argument as in the previous case, one sees that not both rules of the only applicable matrix $m2$ can be applied in sequence. Here, the second B-occurrence in the central part $ABBC$ serves as a barrier between the possible substrings AB within αAB and the C-ocurrences to match with.

Case 3. Boundary cases. This concerns situations where there is no central part available at all. In particular, this means that we are facing sentential forms like αt or βt with $\alpha \in \{A, AB\}^*$, $\beta \in \{B, BC\}^*$ and $t \in T(\{B, BC\} \cup T)^* \cup \{\lambda\}$. Now, either none of the matrices apply, or $m1$ might apply but then the derivation is stuck, because the permitting context $A'C'$ in $m2$ cannot be met.

In summary, we have shown that whenever starting with a string w that is derivable both in G' and in G, with G', we will be forced to produce a string that

is also derivable in G, which proves that $L(G) \supseteq L(G')$, so that together with our previous considerations, we have shown that $L(G) = L(G')$ as claimed. □

Notice that in the proof of the preceding theorem, we showed that after applying $m2$, we were forced to apply $m3$. So, the idea of simply appending $m3$ to $m2$ would work, giving a sort of trade-off result in terms of parameters. As the only purpose of the context check in the rule of $m3$ was to guarantee that $m3$ is not applied in another situation, we can furthermore omit context checks in the appended rule $C' \to \lambda$. Hence, otherwise following the same reasoning as before, the two (multi-production) matrices

$$m1 = [(A \to A', 0, A'), (C \to C')],$$
$$m2 = [(B \to \lambda, A'B), (A' \to \lambda, A'C'), (C' \to \lambda)]$$

provide the simulation of the non-context-free rule $ABC \to \lambda$ of a type-0 grammar G in $(4, 1)$-GNF. This shows the following result.

Theorem 2. $\mathsf{MSSC}(2, 1; 6; 3, 2, 3) = \mathsf{RE}$. □

With a similar intuition, one could observe that an application of matrix $m1$ should be always followed by an application of $m2$. By tuning a bit on the roles of A' and C', we propose the multi-production matrix

$$m = [(A \to A', 0, A'), (C \to C'), (B \to \lambda, A'B), (C' \to \lambda, A'C')]$$

to work together with the one-production matrix $[(A' \to \lambda, 0, 0)]$ in order to simulate the non-context-free rule $ABC \to \lambda$. This can be worked out to prove:

Theorem 3. $\mathsf{MSSC}(2, 1; 6; 3, 1, 4) = \mathsf{RE}$. □

Working out further the idea of merging matrices of Theorem 1, as $m1$, $m2$ and $m3$ must be applied in order, one can observe that the forbidden context checks become obsolete, leading us to use the following (multi-production) matrix

$$m = [(A \to A'), (C \to C'), (B \to \lambda, A'B), (A' \to \lambda, A'C'), (C' \to \lambda)]$$

to simulate the non-context-free rule $ABC \to \lambda$. This idea yields the following.

Theorem 4. $\mathsf{MSSC}(2, 0; 6; 2, 1, 5) = \mathsf{MP}(2; 6; 2, 1, 5) = \mathsf{RE}$. □

So far, we derived our results starting off from $(4, 1)$-GNF. We needed two more nonterminals in order to uniquely mark the boundaries of the central part of the sentential form. We are first trying to reduce the number of nonterminals at the expense of more conditional rules. To this end, we continue discussing our previous constructions, in particular, those based on the proof of Theorem 1. Observe that an application of matrix $m1$ should be always followed by an application of $m2$. By merging the roles of A' and C' within $\#$, we propose the multi-production matrix

$$m = [(A \to \#, 0, \#), (B \to \lambda, \#B, 0), (C \to \#, \#C, 0), (\# \to \lambda, \#\#, 0)]$$

to work together with the one-production matrix $[(\# \to \lambda, 0, 0)]$ in order to simulate the non-context-free rule $ABC \to \lambda$. This idea can be worked out to prove the following result.

Theorem 5. $\mathit{MSSC}(2,1;5;4,1,4) = \mathit{RE}$. □

Combining the ideas leading to the previous two theorems, we would again combine everything within one long matrix, now having the possibility to remove all forbidden context checks. This leads us to the following result, based on

$$m = [(A \to \#), (B \to \lambda, \#B), (C \to \#, \#C), (\# \to \lambda, \#\#), (\# \to \lambda)].$$

Theorem 6. $\mathit{MSSC}(2,0;5;3,1,5) = \mathit{MP}(2;5;3,1,5) = \mathit{RE}$.

In order to aim at the use of fewer nonterminals (but possibly with more small-length matrices), we turn to another GNF, namely, to $(4,2)$-GNF, which we are going to use in the following, trading off the number of conditional rules with avoiding forbidden strings.

Theorem 7. $\mathit{MSSC}(2,1;5;3,3,2) = \mathit{RE}$.

Proof. Let $L \in \mathsf{RE}$ be generated by a grammar in $(4,2)$-GNF of the form $G = (N, T, P \cup \{AB \to \lambda, CC \to \lambda\}, S)$ such that P contains only context-free productions and $N = \{S, A, B, C\}$ (see Rem. 1). Next, we define the MSSC grammar $G' = (V, T, P' \cup P'', S)$, where $N' = N \cup \{\#\}$ (assuming that $\# \notin N$), P' contains (single-production) matrices of the form $[(S \to \alpha, 0, 0)]$ whenever $S \to \alpha \in P$ and P'' contains the three (multi-production) matrices:

$$\begin{aligned} m1 &= [(A \to \#, 0, \#), (B \to \#)], \\ m2 &= [(C \to \#, 0, \#), (C \to \#)], \\ m3 &= [(\# \to \lambda, \#\#, 0), (\# \to \lambda)]. \end{aligned}$$

We can show $L(G') = L(G)$ by an inductive argument. □

Theorem 8. $\mathit{MSSC}(2,0;5;5,3,2) = \mathit{MP}(2;5;5,3,2) = \mathit{RE}$.

Proof. Let $L \in \mathsf{RE}$ be generated by a grammar in $(4,2)$-GNF of the form $G = (N, T, P \cup \{AB \to \lambda, CC \to \lambda\}, S)$ such that P contains only context-free productions and $N = \{S, A, B, C\}$ (see Rem. 1). Next, we define the MSSC grammar $G' = (N, T, P' \cup P'', S)$, where $N' = N \cup \{\#\}$ (assuming that $\# \notin N$), P' contains (single-production) matrices of the form $[(S \to \alpha, 0)]$ whenever $S \to \alpha \in P$ and P'' contains the three (multi-production) matrices:

$$\begin{aligned} m1 &= [(A \to \#, AB), (B \to \#, \#B)] \\ m2 &= [(C \to \#, CC), (C \to \#, \#C)] \\ m3 &= [(\# \to \lambda, \#\#), (\# \to \lambda)]. \end{aligned}$$

We can show $L(G') = L(G)$ by an inductive argument. □

Observe that in a correct simulation in Theorem 8, $m1$ must be followed by $m3$ and $m2$ must be followed by $m3$, as well. Hence, by appending $m3$ to $m1$ and to $m2$, we can obtain another computational completeness result with less but longer matrices, omitting the context conditions in the matrices $m1$ and $m2$ from the proof of Theorem 8.

Theorem 9. $\mathsf{MSSC}(2,0;5;2,2,4) = \mathsf{MP}(2;5;2,2,4) = \mathsf{RE}$. □

In order to further lower the number of nonterminals that we have to use, we are moving on to GNFs that are more parsimonious in this respect, i.e., to $(3,2)$-GNF or (equivalently) to $(3,1)$-GNF.

Theorem 10. $\mathsf{MSSC}(3,0;4;5,3,3) = \mathsf{MP}(3;4;5,3,3) = \mathsf{RE}$.

Proof. Let $L \in \mathsf{RE}$ be generated by a grammar in $(3,2)$-GNF of the form $G = (N,T,P \cup \{AA \to \lambda, BBB \to \lambda\}, S)$ such that P contains only context-free productions and $N = \{S,A,B\}$. We define the MSSC grammar $G = (N,T,P' \cup P'',S)$, where $N' = N \cup \{\#\}$ (assuming that $\# \notin N$), P' contains (single-production) matrices of the form $[(S \to \alpha, 0)]$ whenever $S \to \alpha \in P$ and P'' contains the following three matrices with five conditional rules:

$$m1 = [(B \to \#, BBB), (B \to \#, 0), (B \to \#, \#B\#)]$$
$$m2 = [(A \to \#\#, AA), (A \to \#\#, 0), (\# \to \lambda, \#\#\#)]$$
$$m3 = [(\# \to \lambda, \#\#\#), (\# \to \lambda, 0), (\# \to \lambda, 0)].$$

The intended simulation of $\alpha ABBBA\beta t \Rightarrow_G \alpha\beta t$ works as follows.

$$\alpha ABBBA\beta t \Rightarrow_{m1} \alpha A\#\#\#A\beta t \Rightarrow_{m3} \alpha AA\beta t \Rightarrow_{m2} \alpha\#\#\#\beta t \Rightarrow_{m3} \alpha\beta t\,.$$

Again, we have to prove that $L(G) = L(G')$ by an inductive argument based on some case analysis. □

Can we somehow shorten the matrices involved in the previous construction? What somehow comes to mind that the matrix dealing with A are stronger than those dealing with B in the sense of being more deterministic. As the non-determinism observed with the matrix $m1$ dealing with BBB is not a crucial drawback, one could use $m2' = [(A \to \#, AA), (A \to \#\#, \#A)]$ instead. Checking cases one sees that this does not create any additional problems indeed. We can also try to use $\#\#$ instead of $\#\#\#$ for marking purposes, which would allow us to shorten the length of $m3$ as well. Still, we have to split the matrix dealing with BBB, which is done as follows (at the cost of additional context checks):

$$m0 = [(B \to \#, BBB), (B \to \#, BB\#)]\,,$$
$$m1 = [(B \to \#, \#B\#), (\# \to \lambda, \#\#\#)]\,,$$
$$m2 = [(A \to \#, AA), (A \to \#, \#A)]\,,$$
$$m3 = [(\# \to \lambda, \#\#), (\# \to \lambda)]\,.$$

We explain the necessity of these context checks by one example: If we omitted the BBB check in the first rule of $m1$, we might enter the matrix not having BBB as the central part, but without the $BB\#$ context of the second rule, it might be that we had replaced two occurrences of B's that are both not within BBB, so that we could re-apply $m0$, or possibly also directly apply $m3$. Also observe that with the proposed version, the shortcut of applying $m3$ immediately after $m0$ would lead to no continuation, because the central part would have been successfully destroyed, so that from now on no other matrix is applicable. Based on this construction, one can show (analogously to the previous case):

Theorem 11. $\mathit{MSSC}(3,0;4;7,4,2) = \mathit{MP}(3;4;7,4,2) = \mathit{RE}$. □

We are now going to follow the strategy again to merge some of the matrices. Also, we could be more parsimonous with checking contexts in this case. This leads us to the following matrices simulating the non-context-free rules:

$$m1 = [(B \to \#, 0), (B \to \#, 0), (B \to \lambda, \#B\#), (\# \to \lambda, \#\#), (\# \to \lambda, 0)],$$
$$m2 = [(A \to \#, 0), (A \to \#, 0), (\# \to \lambda, \#\#), (\# \to \lambda, 0)].$$

These two matrices with three conditional rules form the basis of the following.

Theorem 12. $\mathit{MSSC}(3,0;4;3,2,5) = \mathit{MP}(3;4;3,2,5) = \mathit{RE}$. □

Alternatively, we could use more conditional rules and longer matrices (but smaller degrees) with the following matrix $m1'$ replacing $m1$ above: $m1' = [(B \to \#, 0), (B \to \#, \#B), (\# \to \#, B\#), (\# \to \#, A\#), (B \to \lambda, \#A), (\# \to \lambda, \#\#), (\# \to \lambda, 0)]$. Observe that if we have a string $w \in \{A, B, \#\}^*$ with two #-occurrences and moreover $\{\#A, \#B, A\#, B\#\} \subseteq \mathit{sub}(w)$, then either $A\#B$ and $B\#A$ or both $A\#A$ and $B\#B$ are substrings of w. Now if some B is deleted to produce w', so that afterwards $\#\#$ is a substring of w', then only $A\#B\#A \in \mathit{sub}(w)$ follows. This implies that $BBB \to \lambda$ is correctly simulated.

Theorem 13. $\mathit{MSSC}(2,0;4;6,2,7) = \mathit{MP}(2;4;6,2,7) = \mathit{RE}$. □

4 Conclusions and discussions

We have tried to describe the frontier of computational completeness for MSSC grammars, obtaining quite a lot of trade-off results between the six descriptional complexity measures that we studied. The natural question is if our bounds can be further improved or if there are more trade-off results than already presented. We are currently working on the idea of re-using the start nonterminal within matrices.

Conversely, it would be also good to know which descriptional complexity restrictions lead to language classes smaller than RE. For instance, we believe that, irrespectively of the size of the other parameters, one nonterminal is insufficient to describe all of RE. This can be seen by inspecting the proof of the corresponding result for graph-controlled grammars in [3, Theorem 15].

However, whether or not two or three nonterminals might suffice is open. Let us finally sketch the difficulties that we face when trying to use previous results on matrix languages in particular. In the proof of [3, Corollary 6], showing that three nonterminals $\{A, B, C\}$ suffice to generate any RE-language by using context-free matrix grammars with appearance checking, the appearance checks have been performed on rules $X \to C^g$ (for some large number g encoding failure). Picking some nonterminal $Y \neq X$, this check could be simulated by an SSC rule $(Y \to Y, 0, X)$. This bounds the degree to $(0, 1)$, as no permitting string checks are ever needed, and the nonterminals to three, yet, all other interesting parameters are unbounded. In particular, the matrix length is a classical parameter in matrix grammars, so that this observation might revive some

interest in the descriptional complexity of more classical rewriting systems (as matrix grammars), as there are still open problems in that area. For instance, it is also still unknown if context-free matrix grammars with two nonterminals only describe RE.

References

1. S. Ábrahám. Some questions of phrase-structure grammars, I. *Comput. Linguistics*, 4:61–70, 1965.
2. J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1989.
3. H. Fernau, R. Freund, M. Oswald, and K. Reinhardt. Refining the nonterminal complexity of graph-controlled, programmed, and matrix grammars. *Journal of Automata, Languages and Combinatorics*, 12(1/2):117–138, 2007.
4. H. Fernau, L. Kuppusamy, R. Oladele, and I. Raman. Improved descriptional complexity results for simple semi-conditional grammars. Submitted to Fundamenta Informaticae, 2019.
5. H. Fernau, L. Kuppusamy, and R. O. Oladele. New nonterminal complexity results for semi-conditional grammars. In F. Manea, R. G. Miller, and D. Nowotka, editors, *Sailing Routes in the World of Computation, 14th Conference on Computability in Europe, CiE*, volume 10936 of *LNCS*, pages 172–182. Springer, 2018.
6. Zs. Gazdag and K. Tichler. On the power of permitting semi-conditional grammars. In É. Charlier, J. Leroy, and M. Rigo, editors, *Developments in Language Theory - 21st International Conference, DLT*, volume 10396 of *LNCS*, pages 173–184. Springer, 2017.
7. V. Geffert. How to generate languages using only two pairs of parentheses. *Journal of Information Processing and Cybernetics EIK*, 27(5/6):303–315, 1991.
8. V. Geffert. Normal forms for phrase-structure grammars. *RAIRO Informatique théorique et Applications/Theoretical Informatics and Applications*, 25:473–498, 1991.
9. D. Hauschildt and M. Jantzen. Petri net algorithms in the theory of matrix grammars. *Acta Informatica*, 31:719–728, 1994.
10. T. Masopust. *Formal Models: Regulation and Reduction*. PhD thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic, 2007.
11. A. Meduna and M. Gopalaratnam. On semi-conditional grammars with productions having either forbidding or permitting conditions. *Acta Cybernetica*, 11:309–323, 1994.
12. A. Meduna and T. Kopeček. Simple semi-conditional versions of matrix grammars with a reduced regulating mechanism. *Computing and Informatics*, 23:287–302, 2004.
13. A. Meduna and M. Svec. Reduction of simple semi-conditional grammars with respect to the number of conditional productions. *Acta Cybernetica*, 15(3):353–360, 2002.
14. F. Okubo. A note on the descriptional complexity of semi-conditional grammars. *Information Processing Letters*, 110(1):36–40, 2009.
15. Gh. Păun. A variant of random context grammars: semi-conditional grammars. *Theoretical Computer Science*, 41:1–17, 1985.
16. Gy. Vaszil. On the descriptional complexity of some rewriting mechanisms regulated by context conditions. *Theoretical Computer Science*, 330:361–373, 2005.