Efficient Learning of Some Linear Simple Matrix Languages

Henning Fernau

WSI-2000-09

Henning Fernau

Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

E-Mail: fernau@informatik.uni-tuebingen.de
Telefon: (07071) 29-7565
Telefax: (07071) 29-5061

ISSN 0946-3852

Efficient learning of some linear simple matrix languages

Henning Fernau
Wilhelm-Schickard-Institut für Informatik;
Universität Tübingen
Sand 13;
D-72076 Tübingen;
Germany
fernau@informatik.uni-tuebingen.de

May 22, 2000

Abstract

We show that so-called deterministic even linear simple matrix grammars can be inferred in polynomial time using the query-based learner-teacher model MAT proposed by Angluin for learning deterministic regular languages in [4]. In this way, we extend the class of efficiently learnable languages beyond both the even linear languages and the even equal matrix languages proposed in [41, 48, 52, 53, 54, 57, 56].

1 Introduction

Learning languages using a teacher-learner-dialogue (also known as MAT –minimally adequate teacher– learning model) has become popular since Angluin published a polynomial-time learning algorithm for regular languages [4] (or more precisely deterministic finite automata). Alternative versions of algorithms for learning regular languages in the MAT model appeared in [9, 12, 28, 42, 63]. The limits of the model were explored in [5, 6]. One of the questions arising from Angluin's result is whether it can be extended beyond regular sets. For example, learning algorithms for deterministic one-counter automata [10] and systolic automata [62] were exhibited.

Radhakrishnan and Nagaraja [41] proposed even linear languages for learning theoretical purposes by giving a skeleton-based inference algorithm and showing possible applications in the area of inference of pictures. Both the work of Takada [52] and that of Sempere and García [48] showed how the learning problem of even linear languages (introduced in [1]) could be reduced to the learning of regular languages. Moreover, Takada and his colleagues proved the usefulness of the concept of control languages (originating from [22]) in the reduction of the learning problem of languages defined via controlled fixed grammars [52, 53, 54, 56, 57, 32, 34]. In particular, Takada used this concept to develop an efficient learning algorithm of what he called "even equal matrix languages" [53, 54, 57] which form a subclass of the equal matrix languages introduced by Siromoney [50]. Here, we follow the conventions of the monography [18] and refer to the equal matrix languages as "simple right-linear matrix languages". In [58] equal matrix languages and linear languages are compared.

As mentioned above, control languages are a natural tool for transferring Angluin's learnability result to, e.g., even equal matrix languages or even linear languages. Indeed, the learning problem for those classes is reduced to the learning problem of (regular) control languages by using universal grammar normal forms. Obviously, it is now possible to use, e.g., even linear languages as control languages for universal even linear grammars, hence obtaining a whole hierarchy (similar to those

of Khabbaz [30, 31]) of efficiently learnable language classes by iterating the argument sketched above. Takada's papers [57, 56] explore the learnability of levels of such hierarchies.

In this paper, we show how to learn what we call simple even linear matrix languages of arbitrary degree. In doing this, we extend Takada's previous results in two ways:

- Even equal matrix languages are a subset of even linear matrix languages.
- The Khabbaz/Takada hierarchy of even linear languages controlled by even linear languages and so forth is contained in the even linear matrix languages. More precisely, we show that even linear matrix languages controlled by even linear matrix languages yield even linear matrix languages, so that a further Khabbaz/Takada-like hierarchy extension of the efficiently learnable language classes is not possible.

Furthermore, we exhibit several hierarchical relations between the considered language families by making use of well-known properties of the control languages. In particular, we do not need to prove pumping lemmas for our language families, since the translation to the control language level allows us to apply pumping lemmas for regular languages directly.

For simple linear matrix languages, we refer to [39] and [18, p.68ff.]. Results on simple right-linear and context-free matrix languages can be found in [18, 27, 37, 38, 50, 51]. Equivalent formalizations can be found in [15] and [36]. In particular, all such languages are semilinear, a fact that may be of some importance for learning theoretic purposes [55].

Intriguingly, there may be another source of interest in the mentioned language families: Weir showed in [59] how the Khabbaz hierarchy [30] can be generalized in order to characterize tree adjoined languages, which play a prominent role in computer linguistics. A future possibility might be that those (formal) connections could lead to programs which are able to assist linguists who are designing grammars for natural languages, e.g., by creating proposals of such grammars automatically (in the sense of learning theory) or by certifying optimality conditions on these grammars, e.g., a minimal number of variables in the case of "deterministic grammars". Moreover, the class of even linear simple matrix languages we propose for learning-theoretic purposes contains both typical "pushdown languages" and typical "queue languages". This observation should be of particular interest when considering possible linguistic applications of learning theory, since natural languages typically consist of both parenthesis structures (as exemplified by relative clauses) and "copy-structures" (as can be found in Swiss German), cf. [49].

Such research might help close the "undesirable gap between the communities of linguists and computer scientists, more specifically the communities of computational linguists and formal language theoreticians" observed by Martín-Vide in [35].

This paper has the following structure: In Section 2, we present the basic definitions of simple matrix grammars necessary for our paper. In Section 3, we introduce the concept of universal grammar essential for Takada's approach to the learning problem. Section 4 contains various other characterizations of the language classes considered in this paper which are either valuable for learning theoretic purposes or for proving further results. In Section 5 we discuss how to apply our results to the inference of simple matrix grammars. Section 6 is devoted to exploring some relations between the different language families. We show especially how the work of Takada is extended –both concerning the learnability of even equal matrix languages and the learnability of the Khabbaz-like hierarchy induced by even linear control languages. Section 7 elaborates on several areas for possible research in the future.

2 Definitions and examples

In general, we use standard notions from formal language theory. So, $|x|$ denotes the length of a word x which is formally an element of the free monoid X^* generated by some finite alphabet X. $X^{<n}$ is the set of all words in X^* which are shorter than n. Let $X^n = X^{<n+1} \setminus X^{<n}$ and $X^{\leq n} = X^{<n+1}$. The neutral element in X^* –called the empty word– is denoted by λ.

Sometimes, we use Σ_m to denote an arbitrary but fixed m-element alphabet.

A *linear simple matrix grammar of degree* n, $n \geq 1$, is (cf. [39]) an $(n+3)$-tuple

$$G = (V_1, \ldots, V_n, \Sigma, M, S),$$

where $\{S\}, V_1, \ldots, V_n, \Sigma$ are pairwise disjoint alphabets ($V_N = \bigcup_{i=1}^{n} V_i \cup \{S\}$ contains the nonterminals and Σ the terminals), and M is a finite set of matrices of the form

1. $(S \to A_1 \ldots A_n)$, for $A_i \in V_i$, $1 \leq i \leq n$, or
2. $(A_1 \to x_1, \ldots, A_n \to x_n)$, for $A_i \in V_i$, $x_i \in \Sigma^*$, $1 \leq i \leq n$, or
3. $(A_1 \to x_1 B_1 y_1, \ldots, A_n \to x_n B_n y_n)$, for $A_i, B_i \in V_i$, $x_i, y_i \in \Sigma^*$, $1 \leq i \leq n$.

Matrices of the form 1.–2. are called *initial matrices*, *terminal matrices*, and *nonterminal matrices*, respectively.

We now define three restrictions on such grammars:

G is a *right-linear simple matrix grammar* if the nonterminal matrices satisfy

3′. $(A_1 \to x_1 B_1, \ldots, A_n \to x_n B_n)$, $A_i, B_i \in V_i$, $x_i \in \Sigma^*$, $1 \leq i \leq n$.

G is an *even linear simple matrix grammar* if the nonterminal matrices satisfy

3″. $(A_1 \to x_1 B_1 y_1, \ldots, A_n \to x_n B_n y_n)$, for $A_i, B_i \in V_i$, $x_i, y_i \in \Sigma^*$ such that $|x_i| = |y_j|$ for all $1 \leq i, j \leq n$.

G is an *even right-linear simple matrix grammar* (or *even equal matrix grammar* as introduced by Takada [54]) the nonterminal matrices satisfy

3‴. $(A_1 \to x_1 B_1, \ldots, A_n \to x_n B_n)$, $A_i, B_i \in V_i$, $x_i \in \Sigma^*$ such that $|x_1| = |x_i|$ for all $2 \leq i \leq n$.

Let $V_G = V_N \cup \Sigma$. For $x, y \in V_G^*$, we write $x \Rightarrow y$ iff either (i) $x = S$, $(S \to y) \in M$, or (ii) $x = u_1 A_1 v_1 \ldots u_n A_n v_n$, $y = u_1 w_1 v_1 \ldots u_n w_n v_n$, and $(A_1 \to w_1, \ldots, A_n \to w_n) \in M$. As usual, define $L(G) = \{x \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} x\}$, where $\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of relation $\Rightarrow$.

The families of linear simple matrix grammars of degree n, right-linear simple matrix grammars, even linear simple matrix grammars, and even right-linear simple matrix grammars, as well as the corresponding language families, are denoted by SL(n), SRL(n), ESL(n) and ESRL(n), respectively. Sometimes, we use notations like ESL(n,m) in order to specify the terminal alphabet Σ_m explicitly. Specifically, ESRL(1) = SRL(1) denotes the regular languages, ESL(1) the even linear languages, and SL(1) the linear languages.

In the following, we give three examples (which are the common languages used by linguists to prove that natural languages are not context-free) to show the power of the mechanisms.

Example 2.1 *Consider* $G_1 = (\{A_1\}, \{A_2\}, \{a, b\}, P_1, S)$, *where* P_1 *contains the following matrices:*

1. $(S \to A_1 A_2)$,
2. $(A_1 \to \lambda, A_2 \to \lambda)$,
3. $(A_1 \to aA_1, A_2 \to aA_2), (A_1 \to bA_1, A_2 \to bA_2)$.

G_1 *is an* ESRL*(2) grammar which generates* $L(G_1) = \{ww \mid w \in \{a, b\}^*\}$.

Example 2.2 *Consider* $G_2 = (\{A_1, B_1\}, \{A_2, B_2\}, \{a, b\}, P_2, S)$, *where* P_2 *contains the following matrices:*

1. $(S \to A_1 A_2)$,
2. $(A_1 \to \lambda, A_2 \to \lambda)$, $(B_1 \to \lambda, B_2 \to \lambda)$,

3. $(A_1 \to aA_1, A_2 \to aA_2), (A_1 \to bB_1, A_2 \to bB_2), (B_1 \to bB_1, B_2 \to bB_2)$.

$G_2 \in \mathrm{ESRL}(2)$ *generates* $L(G_2) = \{a^n b^m a^n b^m \mid n, m \geq 0\}$.

Example 2.3 *Consider* $G_3 = (\{A_1\}, \{A_2\}, \{A_3\}, \{a, b\}, P_3, S)$ *where* P_3 *contains the following matrices:*

1. $(S \to A_1 A_2 A_3)$,
2. $(A_1 \to \lambda, A_2 \to \lambda, A_3 \to \lambda)$,
3. $(A_1 \to aA_1, A_2 \to bA_2, A_3 \to aA_3)$.

G_3 *is an* ESRL*(3) grammar which generates* $L(G_3) = \{a^n b^n a^n \mid n \geq 0\}$.

3 Universal grammars and normal forms

Takada [52] proved the existence of a universal grammar for the class ESL(1) (in our notation). He used this notion in order to derive similar results for other language classes, as well [54, 56, 57]. The notion of universal grammar, which inspired some interest in formal language theory as early as 1980, see [24, 26, 29, 45], is based on the even older concept of control language, see [22] which we are going to deal with in the next subsection.

3.1 Control languages

Let $G = (V_1, \ldots, V_n, \Sigma, M, S) \in \mathrm{SL}(n)$ be chosen arbitrarily. To every valid derivation sequence, there corresponds (at least) one sequence of matrices $\pi \in M^*$[1] (which have been applied in order to get that derivation sequence), and every sequence of matrices $\pi \in M^*$ determines (at most) one valid derivation sequence. For short, we write $x \Rightarrow^\pi y$ if y is obtained from x by applying the matrices listed in the so-called *control word* π in that sequence. For $C \subseteq M^*$, define

$$L_C(G) = \{w \in \Sigma^* \mid \exists \pi \in C : S \Rightarrow^\pi w\}.$$

$L_C(G)$ is called the language generated by G with *control set* C.

In the following, we need two auxiliary notions. Let G be an SL grammar.

1. G is said to have *terminal matrix bound* $\ell_1 \geq 1$ if, for all terminal matrices $(A_1 \to x_1, \ldots, A_n \to x_n)$ in G, we have $|x_1| + \ldots + |x_n| \leq \ell_1$.
2. G is said to have *nonterminal matrix bound* $\ell_2 \geq 1$ if, for all nonterminal matrices $(A_1 \to x_1 B_1 y_1, \ldots, A_n \to x_n B_n y_n)$ in G, $\max(|x_1 y_1|, \ldots, |x_n y_n|) \leq \ell_2$ is valid.

Lemma 3.1 *Let* $n \geq 1$. *For every* SL(n) *grammar* G *and every regular control set* C *(given by a right-linear grammar for example), one can construct an* SL(n) *grammar* G' *generating* $L_C(G)$ *with the same terminal and nonterminal matrix bounds. The analogous statement is valid for* ESL(n), SRL(n) *and* ESRL(n) *grammars, as well.*

Proof: Let $G = (V_1, \ldots, V_n, \Sigma, M, S)$ be the given SL(n) grammar. Let the regular control language be given by a right-linear grammar $G_C = (V, M, P, \bar{S})$. Further assume that all rules in P have the form $A \to aB$ or $A \to a$ for $A, B \in V$, $a \in M_m^n$. Construct an ESL(n, m)-grammar

$$G' = (V_1 \times V, V_2, \ldots, V_n, \Sigma, M', S)$$

simulating C-controlled derivations in G by using the following matrices:

[1] Here and in the following, we consider the finite set of matrices as a new alphabet.

1. For every start matrix $m = (S \to A_1 \dots A_n)$ in M and every $\bar{S} \to mA \in P$, take $(S \to (A_1, A)A_2 \dots A_n)$ into M'.

2. For every terminal matrix $m = (A_1 \to x_1, \dots, A_n \to x_n)$ in M and every $A \to m \in P$, take $((A_1, A) \to x_1, A_2 \to x_2, \dots, A_n \to x_n)$ into M'.

3. For every nonterminal matrix $m = (A_1 \to x_1 B_1 y_1, \dots, A_n \to x_n B_n y_n)$ in M and every $A \to mB \in P$, put

$$((A_1, A) \to x_1(B_1, B)y_1, A_2 \to x_2 B_2 y_2, \dots, A_n \to x_n B_n y_n)$$

into M'.

It is rather obvious how the simulation of the grammars G and G_C work. Observe that the terminal and nonterminal matrix bounds are not influenced by the construction, and the even and right-linear properties are preserved. □

Therefore, we can deduce that regular control sets do not increase the descriptive power of simple linear matrix grammars. On the other hand, they might help simplify the notation of SL languages; to see this, we need another auxiliary notion. Consider $G_{\mathrm{SL}}(n, m, \ell_1, \ell_2) = (\{S_1\}, \dots, \{S_n\}, \Sigma_m, M(n, m, \ell_1, \ell_2), S)$ where $M(n, m, \ell_1, \ell_2)$ contains the following matrices:

1. $m_1 = (S \to S_1 \dots S_n)$.

2. $m_{x_1,\dots,x_n} = (S_1 \to x_1, \dots, S_n \to x_n)$ for all $x_i \in \Sigma_m^*$ such that $|x_1| + \dots + |x_n| \leq \ell_1$.

3. $m_{x_1,\dots,x_n;y_1,\dots,y_n} = (S_1 \to x_1 S_1 y_1, \dots, S_n \to x_n S_n y_n)$ for all $x_i, y_i \in \Sigma_m^*$ such that $\max\{|x_1 y_1|, \dots, |x_n y_n|\} \leq \ell_2$.

Let us term $G_{\mathrm{SL}}(n, m, \ell_1, \ell_2)$ *standard* SL(n) *grammar*. Imposing the appropriate restrictions, one can also define standard X-grammars for $X \in \{\mathrm{ESL}(n), \mathrm{SRL}(n), \mathrm{ESRL}(n)\}$, denoted by $G_X(n, m, \ell_1, \ell_2)$. Obviously, $L(G_X(n, m, \ell_1, \ell_2)) = \Sigma_m^*$.

Lemma 3.2 *Let $n \geq 1$ and $X \in \{\mathrm{SL}(n), \mathrm{ESL}(n), \mathrm{SRL}(n), \mathrm{ESRL}(n)\}$.*
For every X-grammar $G = (V_1, \dots, V_n, \Sigma_m, M, S)$ with terminal and nonterminal matrix bounds ℓ_1 and ℓ_2, a regular control set C can be constructed so that $L(G) = L_C(G_X(n, m, \ell_1, \ell_2))$.

Proof: The following construction works for $X = \mathrm{ESL}(n)$ and $X = \mathrm{SL}(n)$. The analogues for $X = \mathrm{SRL}(n)$ and $X = \mathrm{ESRL}(n)$ are easily obtained. Consider $G = (V_1, \dots, V_n, \Sigma_m, M, S)$. Define the right-linear grammar $G' = (V, M(n, m, \ell_1, \ell_2), P, S)$ by $V = V_1 \times \dots \times V_n \cup \{S\}$, and let P contain the following rules:

1. $S \to m_1(A_1, \dots, A_n)$ if $(S \to A_1 \dots A_n) \in M$.

2. $(A_1, \dots, A_n) \to m_{x_1,\dots,x_n}$ if $(A_1 \to x_1, \dots, A_n \to x_n) \in M$.

3. $(A_1, \dots, A_n) \to m_{x_1,\dots,x_n;y_1,\dots,y_n}(B_1, \dots B_n)$ if $(A_1 \to x_1 B_1 y_1, \dots, A_n \to x_n B_n y_n) \in M$.

The claim $L(G) = L_{L(G')}(G(n, m, \ell_1, \ell_2))$ can be shown by induction on the length of the control words. □

In conclusion, we can state that language class X (for $X \in \{\mathrm{SL}(n), \mathrm{ESL}(n), \mathrm{SRL}(n), \mathrm{ESRL}(n)\}$) can be characterized as containing those languages which can be generated by some standard grammar $G_X(n, m, \ell_1, \ell_2)$ with the help of a regular control set. We aim to narrow the set of necessary standard grammars for this construction in the following.

3.2 Decreasing nonterminal and terminal rule bounds

Lemma 3.3 *Let $n \geq 1$. Every* ESL(n) *language can be generated by some standard grammar $G_{\mathrm{ESL}}(n, m, \ell_1, 2)$ without unit productions (i.e., nonterminal matrices are of the form $m_{a_1,\ldots,a_n;b_1,\ldots,b_n}$ with $a_i, b_i \in \Sigma_m$ only) with the help of a regular control set.*

Proof: Due to Lemma 3.2, we can assume that $L \in \mathrm{ESL}(n)$, $L \subseteq \Sigma_m^*$, is generated as $L_C(G_{\mathrm{ESL}}(n, m, \ell_1, \ell_2))$, where $C \subseteq M(n, m, \ell_1, \ell_2)^*$ is a regular set.

Consider the following morphism $h : M(n, m, \ell_1, \ell_2)^* \to M(n, m, \ell_1, 2)^*$:

1. $h(m_1) = m_1$;
2. $h(m_{x_1,\ldots,x_n}) = m_{x_1,\ldots,x_n}$;
3. $h(m_{\lambda,\ldots,\lambda;\lambda,\ldots,\lambda}) = \lambda$, and furthermore,
   $h(m_{x_1,\ldots,x_n;y_1,\ldots,y_n}) = m_{b_{11},\ldots,b_{n1};c_{11},\ldots,c_{n1}} \ldots m_{b_{1r},\ldots,b_{nr};c_{1r},\ldots,c_{nr}}$,
   where $x_i = b_{i1} \ldots b_{ir}$ and $y_i = c_{ir} \ldots c_{i1}$, $b_i, c_i \in \Sigma_m$ for $1 \leq i \leq r \leq \ell_2/2$.

Obviously, $L = L_{h(C)}(G_{\mathrm{ESL}}(n, m, \ell_1, 2))$, and $h(C)$ is regular. □

In a similar way, analogous results can be proved for the other three classes considered here. Without a proof, we state the claim. Note that Takada proved a similar result for ESRL(n) without the help of control sets in his argument.

Lemma 3.4 *Let $n \geq 1$ and $X \in \{\mathrm{SL}(n), \mathrm{SRL}(n), \mathrm{ESRL}(n)\}$.*
Every X language can be generated by some standard grammar $G_X(n, m, \ell_1, 1)$ without unit productions with the help of a regular control set.□

In the following, we mainly focus on ESL(n) grammars. We say that an ESL(n)-grammar $G = (V_1, \ldots, V_n, \Sigma, M, S)$ is in *normal form* if the rules of G are of one of the following forms:

1. $(S \to A_1 \ldots A_n)$, for $A_i \in V_i$, $1 \leq i \leq n$, or
2. $(A_1 \to \lambda, \ldots, A_{n-1} \to \lambda, A_n \to x)$, for $A_i \in V_i$, $1 \leq i \leq n$, $x \in \Sigma^*$, $|x| < 2n$, or
3. $(A_1 \to x_1 B_1 y_1, \ldots, A_n \to x_n B_n y_n)$, for $A_i, B_i \in V_i$, $x_i, y_i \in \Sigma$, $1 \leq i \leq n$.

Theorem 3.5 *Every* ESL(n)*-grammar can be algorithmically transformed into an equivalent* ESL(n)*-grammar in normal form.*

Proof: Consider an arbitrary ESL(n)-grammar G with $L(G) \subseteq \Sigma_m^*$. According to Lemmas 3.2 and 3.3, a regular control set C can be constructed from G, so that $L(G) = L_C(G_{\mathrm{ESL}}(n, m, \ell_1, 2))$, where ℓ_1 is the terminal rule bound of G.

A typical control word from $M_{\mathrm{ESL}}(n, m, \ell_1, 2)^*$ (for a terminal derivation in G) looks like the following:

$$\pi = m_1 m_{b_{11},\ldots,b_{n1};c_{11},\ldots,c_{n1}} \ldots m_{b_{1q},\ldots,b_{nq};c_{1q},\ldots,c_{nq}} m_{x_1,\ldots,x_n} . \tag{1}$$

(Here, $b_{ij}, c_{ij} \in \Sigma_m$.) With the help of this control word, $G_{\mathrm{ESL}}(n, m, \ell_1, 2)$ generates the word

$$w = \prod_{\ell=1}^{n} \left(\prod_{j=1}^{q} b_{\ell,j} \cdot x_\ell \cdot \prod_{j=1}^{q} c_{\ell,q+1-j} \right) .$$

Assume now $x_i \neq \lambda$, i.e., $x_i = x_i' a$ for some $a \in \Sigma_m$. Consider the control word $\pi_{r,i}$ defined as

$$\begin{aligned}
m_1 \quad & m_{b_{11},\ldots,b_{i1},\boxed{c_{i1}},b_{i+2,1},\ldots,b_{n1};c_{11},\ldots,c_{i-1,1},\boxed{c_{i,2}},c_{i+1,1},\ldots,c_{n1}} \\
\cdot \quad & \textstyle\prod_{j=2}^{q-1} m_{b_{1j},\ldots,b_{ij},\boxed{b_{i+1,j-1}},b_{i+2,j},\ldots,b_{nj};c_{1j},\ldots,c_{i-1,j},\boxed{c_{i,j+1}},c_{i+1,j},\ldots,c_{nj}} \\
\cdot \quad & m_{b_{1q},\ldots,b_{iq},\boxed{b_{i+1,q-1}},b_{i+2,q},\ldots,b_{nq};c_{1q},\ldots,c_{i-1,q},\boxed{a},c_{i+1,q},\ldots,c_{nq}} \\
\cdot \quad & m_{x_1,\ldots,x_{i-1},\boxed{x_i'},\boxed{b_{i+1,q}x_{i+1}},x_{i+2},\ldots,x_n} .
\end{aligned}$$

(We indicate the important changes from π to $\pi_{r,i}$ by using box frames.) Obviously, w can also be generated using that control word, since

$$\begin{aligned} w = \; & \prod_{\ell=1}^{i-1} \left(\prod_{j=1}^{q} b_{\ell,j} \cdot x_\ell \cdot \prod_{j=1}^{q} c_{\ell,q+1-j} \right) \\ \cdot \; & \left(\prod_{j=1}^{q} b_{i,j} \right) \cdot x_i' \cdot \left(a \cdot \prod_{j=1}^{q-1} c_{i,q+1-j} \right) \\ \cdot \; & \left(c_{i,1} \cdot \prod_{j=1}^{q-1} b_{i+1,j} \right) \cdot \left(b_{i+1,q} x_{i+1} \right) \cdot \left(\prod_{j=1}^{q} c_{i+1,q+1-j} \right) \\ \cdot \; & \prod_{\ell=i+2}^{n} \left(\prod_{j=1}^{q} b_{\ell,j} \cdot x_\ell \cdot \prod_{j=1}^{q} c_{\ell,q+1-j} \right) . \end{aligned}$$

Passing from π to $\pi_{r,i}$ means a local "right shift" in the derivation tree structure.

For $1 \leq i < n$ and $d \geq 0$, consider now the rational transductions [2] $\tau_{i \to i+1,d}$ which map $M_{\mathrm{ESL}}(n,m,\ell_1,2)$ into itself. Such a mapping transforms π into $\pi_{r,i}$ as indicated above if $|x_i| > d$, and preserves π if $|x_i| \leq d$. (Since only control words of the form (1) are of interest, $\tau_{i \to i+1,d}$ may map other words in $M_{\mathrm{ESL}}(n,m,\ell_1,2)^*$ into the empty set.)

Analogously, one could define a "left shift" in the derivation tree structure. Consider again w generated using π as defined above under the condition that $x_i \neq \lambda$, here, $x_i = \bar{a} x_i''$. Define $\pi_{\ell,i}$ to be

$$\begin{aligned} m_1 \; & m_{b_{11},\ldots,b_{i-1,1},\boxed{b_{i2}},b_{i+1,1},\ldots,b_{n1};c_{11},\ldots,c_{i-2,1},\boxed{b_{i,1}},c_{i,1},\ldots,c_{n1}} \\ \cdot \; & \prod_{j=2}^{q-1} m_{b_{1j},\ldots,b_{i-1,j},\boxed{b_{i,j+1}},b_{i+1,j},\ldots,b_{nj};c_{1j},\ldots,c_{i-2,j},\boxed{c_{i-1,j-1}},c_{i,j},\ldots,c_{nj}} \\ \cdot \; & m_{b_{1q},\ldots,b_{i-1,q},\boxed{a},b_{i+1,q},\ldots,b_{nq};c_{1q},\ldots,c_{i-2,q},\boxed{c_{i-1,q-1}},c_{i,q},\ldots,c_{nq}} \\ \cdot \; & m_{x_1,\ldots,x_{i-2},\boxed{x_{i-1}c_{i-1,q}},\boxed{x_i''},x_{i+2},\ldots,x_n} . \end{aligned}$$

For $1 \leq i < n$ and $d \geq 0$, define the rational transductions $\tau_{i \to i-1,d}$ by $\pi \mapsto \pi_{\ell,i}$ if $|x_i| > d$, and $\pi \mapsto \pi$ if $|x_i| \leq d$.

First, transform a control word π into

$$\pi' := \tau_1(\pi) := \tau_{n-1 \to n,0}^{\ell_1}(\ldots(\tau_{1 \to 2,0}^{\ell_1}(\pi))\ldots) .$$

One easily verifies that

$$\pi' \in A(n,m) \cdot \{ m_{\lambda,\ldots,\lambda,x_n} \mid |x_n| \leq \ell_1 \}$$

with the prefix set

$$A(n,m) = \{m_1\}\{ m_{b_1,\ldots,b_n;c_1,\ldots,c_n} \mid b_i, c_i \in \Sigma_m \}^* .$$

Further, transform π' into

$$\pi'' := \tau_2(\pi') := \tau_{n \to 1,2s}^{2s}(\ldots(\tau_{n \to n-1,2s}^{2s}(\pi'))\ldots) ,$$

where s is the truncated result of dividing ℓ_1 by $2n$ and r is the remainder of such division, so that $\ell_1 = 2sn + r$, and

$$\tau_{n \to j,d}(\bar{\pi}) := \tau_{j+1 \to j,d}(\ldots(\tau_{n \to n-1,d}(\bar{\pi}))\ldots)$$

results in a "left-shift" in the derivation tree of $n - j$ units. Now,

$$\pi'' \in A(n,m) \cdot \{ m_{x_1,\ldots,x_n} \mid |x_1| = \ldots = |x_{n-1}| \text{ is even}, |x_n| < 2n + |x_1| \} . \tag{2}$$

Finally, consider the homomorphism g that acts as identity on all letters except for terminal rule matrices used in (2) which are transformed according to

$$\begin{aligned} & g(m_{a_{11}\ldots a_{1s}a'_{1s}\ldots a'_{11},\ldots a_{n-1,1}\ldots a_{n-1,s}a'_{n-1,s}\ldots a'_{n-1,1},a_{n1}\ldots a_{ns}a''_1\ldots a''_r a'_{ns}\ldots a'_{n1}}) \\ = \; & m_{a_{11},\ldots,a_{n1};a''_{11},\ldots,a''_{n1}} \cdots m_{a_{1s},\ldots,a_{ns};a''_{1s},\ldots,a''_{ns}} m_{\lambda,\ldots,\lambda,a''_1\ldots a''_r} , \end{aligned}$$

where $a_{ij}, a'_{ij}, a''_j \in \Sigma_m$ $r < 2n$.

So, we get a regular set $C' = g(\tau_2(\tau_1(C)))$ such that $L_C(G_{\mathrm{ESL}}(n,m,\ell_1,2)) = L_{C'}(G_{\mathrm{ESL}}(n,m,2n-1,2))$. Applying finally Lemma 3.1, we get the required normal form representation. □

[2] For further information on rational transductions, we refer to [11]; any *rational transduction* can be realized by a so-called *rational transducer*, i.e., a nondeterministic finite automaton with output. It is well-known that the regular languages are closed under rational transductions.

3.3 Universal grammars

Let $\mathcal{G}(n,m)$ be a family of linear simple matrix grammars, each having the same m-letter terminal alphabet Σ_m and the same degree n, defining a language class $\mathcal{L}(n,m)$. Now, $G_0 \in \mathcal{G}(n,m)$ is called *universal* for $\mathcal{L}(n,m)$ (with respect to regular control languages) if $\mathcal{L}(n,m) = \{L_C(G_0) \mid C \text{ is regular}\}$.

Takada has already shown in [54] that the ESRL(n,m)-grammar

$$G_R(n,m) = (\{S_1\},\ldots,\{S_n\},\Sigma,M_R(n,m),S), \tag{3}$$

is universal for ESRL(n,m), where $M_R(n,m)$ contains the matrices

1. $m_1 = (S \to S_1 \ldots S_n)$ and
2. $m_x = (S_1 \to \lambda,\ldots,S_{n-1} \to \lambda, S_n \to x)$ with $x \in \Sigma_m^{\leq n}$ and
3. $m_{b_1,\ldots,b_n} = (S_1 \to b_1S_1,\ldots,S_n \to b_nS_n)$ with $b_j \in \Sigma_m$ for $1 \leq j \leq n$.

We will demonstrate universality of the similar ESL(n,m)-grammar

$$G(n,m) = (\{S_1\},\ldots,\{S_n\},\Sigma,M(n,m),S), \tag{4}$$

where $M(n,m)$ contains the matrices

1. $m_1 = (S \to S_1 \ldots S_n)$ and
2. $m_x = (S_1 \to \lambda,\ldots,S_{n-1} \to \lambda, S_n \to x)$ with $x \in \Sigma_m^{\leq 2n}$ and
3. $m_{b_1\ldots b_n,c_1\ldots c_n} = (S_1 \to b_1S_1c_1,\ldots,S_n \to b_nS_nc_n)$ with $b_j,c_j \in \Sigma_m$ for $1 \leq j \leq n$.

Proposition 3.6 *$L(G(n,m)) = \Sigma_m^*$, and every word in Σ_m^* has a unique derivation in $G(n,m)$. Moreover, exists a linear time algorithm which transforms an input word $w \in \Sigma_m^*$ into its control word $\pi \in M(n,m)^*$.*

Proof: We only sketch the following short recursive transformation algorithm:

1. output(m_1).
2. if $|w| < 2n$ then output(m_w); stop.
3. if $|w| \geq 2n$ then if w is decomposed as $w = b_1w_1c_1\ldots b_nw_nc_n$, $b_i,c_i \in \Sigma_m$, $|w_1| = \ldots = |w_{n-1}|$, $|w_n| < |w_1| + 2n$ (cf. (2)), then
   (a) output$(m_{b_1\ldots b_n,c_1\ldots c_n})$;
   (b) set $w = w_1\ldots w_n$;
   (c) goto 2.

□

Theorem 3.7 *Grammar $G(n,m)$ defined as in (4) is universal for* ESL(n,m) *for each $n,m \geq 1$.*

Proof: $L_C(G(n,m)) \in$ ESL(n,m) according to Lemma 3.1. The proof of Theorem 3.5 reveals that for every $L \in$ ESL(n,m), there exists a regular control language C such that $L_C(G(n,m)) = L$. □

The last three results are essential for the transferrence of known MAT algorithms to the efficient learning of ESL(n) by using control languages. We return to this issue in Section 5.

Sometimes, we let φ_n denote the unique mapping of control words into derived words, i.e., we have $S \Rightarrow^{\pi} \varphi_n(\pi)$ using the universal grammar $G(n,m)$. On the other hand, to every word w derivable in a universal grammar controlled by language C, i.e., $w \in L_C(G(n,m))$, there corresponds a unique control word $\varphi_n^{-1}(w)$. If n is clear from the context, we will omit the index n.

3.4 Formal language issues

From the notion of universal grammar, several properties which are important from the formal language point of view (but which would also be interesting from the point of view of learning theory) can be easily deduced.

Corollary 3.8 *Each class* ESL(n,m), $n,m \geq 1$ *is closed under the Boolean operations union, intersection and complementation.*

Proof: Let $L_1, L_2 \in \mathrm{ESL}(n,m)$. Due to Theorem 3.7, there exist regular languages C_i such that $L_i = L_{C_i}(G(n,m))$ for $i = 1,2$. Now, for every binary logic operator δ we find

$$(w \in L_1 \,\delta\, w \in L_2) \iff (\varphi^{-1}(w) \in C_1 \,\delta\, \varphi^{-1}(w) \in C_2),$$

(which is true since derivations in $G(n,m)$ are unambiguous) so that the assertion concerning union and intersection is verified by taking $\delta \in \{\vee, \wedge\}$. Moreover, since $w \notin L_1$ iff $\varphi^{-1}(w) \notin C_1$, the closure of regular languages under complementation implies the corresponding closure property for the class ESL(n,m). □

Theorem 3.9 *Let $n \geq 1$ be fixed. The equivalence, inclusion, emptiness and finiteness problems are decidable for* ESL(n)*-grammars but undecidable for* SRL(n)*-grammars if $n > 1$.*

Proof: The undecidability results are proved in [18, Theorem 1.5.7] and [27]. For the decidability part, consider the representation of an ESL(n)-grammar G via a regularly (C-)controlled universal grammar $G(n,m)$ such that if $\pi \in C$, then $S \Rightarrow^{\pi} w$ in $G(n,m)$ for a terminal word w. If $w \in L(G)$, then there is trivially a $\pi \in C$ such that $S \Rightarrow^{\pi} w$ in $G(n,m)$. Therefore, all problems mentioned can be answered on the level of regular control languages with the aid of the known test algorithms for right-linear grammars. □

Remark 3.10 *Since the transformations φ_n and φ_n^{-1} can be computed in deterministic logarithmic space if n is fixed (even AC^0 circuits suffice), the computational complexities of the equivalence, inclusion, emptiness and finiteness problems for* ESL(n)*-grammars are the same as the complexities of the corresponding problems for right-linear grammars. In the case of deterministic* ESL(n)*-grammars (as defined below), the complexities of the mentioned decidability questions correspond to those for deterministic finite automata.*

In formal language theory, universal grammars have been used to derive Chomsky-Schützenberger-Stanley characterizations of language families. With this in mind, and similar to [26, Theorems 1 & 6], one can deduce:

Corollary 3.11 *For each $n,m \geq 1$, there exist an alphabet $\Sigma(n,m)$, a language $L(n,m) \subseteq \Sigma(n,m)^*$ and a letter-to-letter morphism $h : \Sigma(n,m)^* \to \Sigma_m^*$, so that, for each language $L \in$* ESL(n,m)*, there exists a regular set $R \subseteq \Sigma(n,m)^*$ such that*

$$L = h(L(n,m) \cap R).$$

On the other hand, every language represented as $h(L \cap R)$, where h is a letter-to-letter morphism mapping into Σ_m^* and R is a regular set, lies in ESL(n,m).

Proof: The idea of this construction is simple. Grammar $G(n,m)$ can be modified in such a way that it will print the protocol of its matrix-applications, codified in an extension of $G(n,m)$'s terminal alphabet: e.g., the leftmost nonterminal may use alphabet $M(n,m)$ instead of just Σ_m when printing the symbol to its left-hand side. More formally, a rule $m_{b_1 \ldots b_n, c_1 \ldots c_n}$ is replaced by

$$(S_1 \to m_{b_1 \ldots b_n, c_1 \ldots c_n} S_1 c_1, \ldots, S_n \to b_n S_n c_n).$$

If $C = \{m_1\}R'\{m_x \mid x \in \Sigma_m^*, |x| < 2n\}$ is the control set C as defined in the proof of Theorem 3.7, define $R = R'\Sigma_m^* \subset \Sigma(n,m)^* = (\Sigma_m \cup M(n,m))^*$. Finally, the homomorphism acts as identity on Σ_m and maps $m_{b_1 \ldots b_n, c_1 \ldots c_n}$ into b_1.

On the other hand, one can show using standard constructions that ESL(n) is closed under intersection with regular sets and letter-to-letter morphisms. □

Corollary 3.12 ESL(n) *is closed under (inverse) letter-to-letter morphisms and intersection with regular sets.*□

Remark 3.13 *The previous theorems and corollaries are also valid for the corresponding classes of* ESRL *languages, cf. [54]. We omit details here.*

We will now turn to simple matrix (right-)linear grammars without the "even"-restriction for a moment. The following normal form is easily obtainable for these grammars:

Lemma 3.14 *Let* $n \geq 1$.

1. *Every* SL(n)*-language can be generated by a* SL(n)*-grammar obeying*

   3^+ $(A_1 \to x_1 B_1 y_1, \ldots, A_n \to x_n B_n y_n)$,
   for $A_i, B_i \in V_i$, $x_i, y_i \in \Sigma^{\leq 1}$, $1 \leq i \leq n$.

   instead of condition 3 *in the definition of* SL(n)*-grammars.*

2. *Every* SRL(n)*-language can be generated by a* SRL(n)*-grammar obeying*

   3^{++} $(A_1 \to x_1 B_1, \ldots, A_n \to x_n B_n)$,
   for $A_i, B_i \in V_i$, $x_i \in \Sigma^{\leq 1}$, $1 \leq i \leq n$.

   instead of condition $3'$ *in the definition of* SRL(n)*-grammars.*

With this normal form, it is easy to prove an analogue of Theorem 3.7 for SL- and SRL-grammars, too. Instead of stating this formally, we only formulate the corresponding morphic characterization result for SL(n), which is similar to that of Chomsky-Schützenberger-Stanley.

Corollary 3.15 *For each* $n, m \geq 1$, *there exist a language* $L(n,m) \in$ ESL(n) *over some alphabet* $\Sigma(n,m)$ *and a morphism* $h : \Sigma(n,m)^* \to \Sigma_m^*$, *so that for each language* $L \in$ SL(n,m) *there exists a regular set* $R \subseteq \Sigma(n,m)^*$ *such that* $L = h(L(n,m) \cap R)$.

On the other hand, every language represented as $h(L(n,m) \cap R)$, *where* h *is a morphism mapping into* Σ_m^* *and* R *is a regular set, lies in* SL(n,m).

Proof: Using Lemma 3.14, it is easy to show via a padding construction that every SL(n)-language is a morphic image of some ESL(n)-language. So, from Cor. 3.11 we may deduce the first assertion in this corollary. Since SL(n) is closed under homomorphisms and intersection with regular sets by Theorem 1.5.6 of [18], the second claim follows. □

Remark 3.16 *Again, a statement similar to the previous corollary is valid for* SRL(n)*-languages, too.*

Corollary 3.17 *The full trio*[3] *closure of* ESRL(n) *equals* SRL(n).
The full trio closure of ESL(n) *equals* SL(n).

[3] A *full trio* is a family of languages closed under rational transductions, or equivalently, due to the theorem of Nivat [11], a family which is closed under arbitrary morphisms, inverse morphisms and intersection with regular sets. The *full trio closure* of a language family $\mathcal{L}$ is the smallest full trio containing $\mathcal{L}$.

4 Characterizations

We generalize some characterizations of even linear grammars (ESL(1), as we refer to them) of Sempere and García [48] to our case.

4.1 Deterministic grammars

Adapting Definition 2 of [48], we call an even simple (right-)linear matrix grammar in normal form *deterministic* if both $(A_1 \to a_1 B_1 b_1, \ldots, A_n \to a_n B_n b_n)$ and $(A_1 \to a_1 C_1 b_1, \ldots, A_n \to a_n C_n b_n)$ are rules of the grammar, then $B_1 = C_1, \ldots, B_n = C_n$.

Similarly to [48, Theorem 2], we can show:

Theorem 4.1 *Every even simple (right-)linear matrix language L of degree n can be generated by a deterministic even simple (right-)linear matrix grammar in normal form of degree n.*

Proof: L can be generated by a universal grammar G of degree n controlled by a regular set R. R can be given by a deterministic right-linear grammar G_R, where there are only productions of the form $A \to \lambda$ and $A \to aB$ satisfying the following condition: if $A \to aB$ and $A \to aC$ are rules of the grammar, then $B = C$. The conversion of G and G_R into an even simple (right-)linear matrix grammar shown in Lemma 3.1 yields a deterministic even simple (right-)linear matrix grammar in normal form of degree n. □

4.2 A Nerode-like equivalence

We now generalize [48, Theorem 3].[4] Given n, every word $w \in \Sigma^*$ can be partitioned uniquely in the form $w = u_1 v_1 \ldots u_{n-1} v_{n-1} u_n w' v_n$, where all u_i and v_i have the same length, and $|w'| < 2n$, cf. the algorithm given in the proof of Prop. 3.6. Hence, we can also write $u_i(w)$ for the subword u_i of w.

Given a ESL(n)-language $L \subseteq \Sigma^*$, two words w_1 and w_2, each of a length divisible by $2n$, are called *indistinguishable* under the language L, written $w_1 \equiv_L^n w_2$, iff for all $w \in \Sigma^*$,

$$\left(\prod_{j=1}^{n-1} u_j(w_1) u_j(w) v_j(w) v_j(w_1) \right) \cdot u_n(w_1) u_n(w) w'(w) v_n(w) v_n(w_1) \in L$$

if and only if

$$\left(\prod_{j=1}^{n-1} u_j(w_2) u_j(w) v_j(w) v_j(w_2) \right) \cdot u_n(w_2) u_n(w) w'(w) v_n(w) v_n(w_2) \in L\,.$$

Theorem 4.2 *$L \subseteq \Sigma^*$ is an* ESL(n)*-language iff equivalence relation $\equiv_L^n$ has a finite number of equivalence classes.*□

Since the proof is quite similar to [48, Theorem 3] (but much more tedious), we omit it here. Observe that there is another way of proving that result, namely by exploiting again the control language characterization and using the well-known Nerode equivalence on the control language level.

Remark 4.3 *Similarly to the well-known case of finite automata, the number of equivalence classes equals the number of states of the minimal DFA controlling $G(n,m)$. This number of minimal states is an important complexity measure for the learning algorithm we present in the next section 5.*

[4] A Nerode-like equivalence relation has already been considered by Amar and Putzolu in [2] as well as by Păun and Novotný in [40] for even linear languages. They use notions different from ours for the presentation of their results.

Remark 4.4 *Analogous statements can be made in case of* ESRL(n)*-languages. Here, we must take another equivalence relation of course.*

4.3 Protocol languages

Recall that L is *protocol language* for language family $\mathcal{L}$ iff for every language $L' \in \mathcal{L}$ there is a rational transducer τ such that $\tau(L') = L$.

Theorem 4.5 SRL(n) *is the class of languages with protocol language* $\{w^n \mid w \in \Sigma_2^*\}$.

Proof: Guess the control language and write the corresponding protocol with the aid of a rational transducer. □

Remark 4.6 *In [13, p. 116] another characterization of equal matrix languages is mentioned without a proof:* SRL *is the smallest class of languages containing the regular sets and closed under homomorphic duplication, i.e.,* $\langle h_1, \ldots, h_n \rangle(L) = \{h_1(w) \ldots h_n(w) \mid w \in L\}$. *Another characterization is possible via restricted checking stack automata, see [13, p. 116].*

Analogously, we obtain:

Theorem 4.7 SL(n) *is the class of languages with protocol language* $\{(ww^R)^n \mid w \in \Sigma_2^*\}$.

Interestingly, one can develop a corresponding characterization programme for "even" language classes as well, using a natural restriction on the notion of subsequential transducer as defined in [11]: Let us call a 7-tuple $\tau = (Q, X, Y, H, q_0, Q_f, q_f)$ *coding transducer with n-tail* if Q is a finite set of states, X and Y are finite (input and output) alphabets, $q_0 \in Q$ is the initial state, $Q_f \subseteq Q$ is the set of prefinal states, q_f is a final state and H is a finite subset of $(Q \setminus \{q_f\}) \times X \times Y \times (Q \setminus \{q_f\}) \cup Q_f \times X^{<n} \times \{\lambda\} \times \{q_f\}$. The interpretation is simple: If τ is in state $q \in Q \setminus \{q_f\}$ and reads symbol $a \in X$, then it may print symbol $a' \in Y$ and go into state q' iff $(q, a, a', q') \in H$; if it is in a prefinal state, it may also read a string of length at most n and go into the final state without producing any output symbol.

Without a formal proof, we present the next result, itself closely related to Cor. 3.12:

Theorem 4.8 *Let $n, m > 0$ be fixed.*

1. $L \in$ ESL(n, m) *iff there exists a coding transducer τ with $2n$-tail such that $x \in L$ iff* $\tau(x) \in \{(ww^R)^n \mid w \in \Sigma_{2nm}^*\}$.
2. $L \in$ ESRL(n, m) *iff there exists a coding transducer τ with n-tail such that $x \in L$ iff* $\tau(x) \in \{w^n \mid w \in \Sigma_{nm}^*\}$.□

Note that Cor. 3.17 can be deduced from our results on protocol languages as well.

5 An application to learning theory

We now turn to the learning of ESL(n) languages for a given $n \geq 1$. The model we use is the following: a learner L has the task of deducing an ESL(n) grammar for a certain ESL(n) language L known to its teacher T. At the beginning, L is informed only of the terminal alphabet Σ_m of L. L may query T about the following:

Membership query Is $w \in L$?

Equivalence query Does the ESL(n) grammar G generate L?

Teacher T reacts as follows to these questions:

1. To a membership-query, T answers either "yes" or "no".

2. To an equivalence-query, T answers either "yes" (here, the learning process may stop, since L has performed its task successfully) or "no, I will show you a counterexample w."

Following Theorem 3.9, teacher T can answer equivalence-queries algorithmically.

We now give a learning algorithm based on the ideas of Takada [52]. We assume there is a learner L′ which can learn deterministic finite automata, e.g., the one described by Angluin [4].

Learner L will transform a counterexample x received from T into a string $\varphi^{-1}(x)$ by using the algorithm given in the proof of Prop. 3.6.

On the other hand, a query w of L′ will be translated by L into the unique $\varphi(w)$ determined by $S \Rightarrow^w \varphi(w)$ using the universal grammar $G(n,m)$, see (4). That $\varphi(w)$ will be passed to the teacher T.

Finally, if learner L′ has a guess C of a possible regular (control) set, L transforms this guess into an ESL(n) grammar G_C for $L_C(G(n,m))$ as in the proof of Theorem 3.7 and passes G_C to teacher T.

So, L functions primarily as an interface between the ESL(n)-teacher T and the regular set learner L′.

Using Angluin's algorithm [4, pages 94–96] for learning regular languages, the above-sketched algorithm has a running time polynomial in both the number of states of a deterministic finite automaton for the control set C and the length of the longest counterexample and the parameter n (which influences the size of the alphabet of the control set to be learnt). So, in view of Theorem 4.1, we can only claim to have a polynomial learning algorithm if we want to learn *deterministic* even simple (right-)linear matrix grammar; else, we would get an exponential running time due to the state explosion of the subset construction hidden in the proof of Theorem 4.1. More precisely, the impossibility argument given in [6] applies here *a fortiori*.

Theorem 5.1 (Learnability) *For each $n \geq 1$, the class of deterministic* ESL(n) *grammars is inferrable in polynomial time using membership and equivalence queries within Angluin's MAT model.*□

Remark 5.2 *Observe that the functionality of the above-sketched relies only on the fact that there exists a one-to-one correspondence between words and "their" control words via the mappings φ, φ^{-1}. Since φ^{-1} can be seen (basically) as a family of permutations $\psi_{n,m} : \Sigma_m^n \to \Sigma_m^n$, any "reasonably computable" family of such permutations defines a "new" class of languages learnable in polynomial time. A source for such language families may be those with "rational structure generating functions" as considered in a series of papers by Kuich, cf. especially [33, 46]. The intuition experienced which hints at the fact that permutations are the backbone of all main constructions in this paper is further underlined by Theorem 4.8: In effect, the protocol language describes the family of permutations.*

In this context, it might be worth noting that ESL*-languages (and already even linear ones) contain all regular languages, since they come up simply by permuting the input of a finite automaton in a regular fashion. It would be of interest to know other language classes obtained by a family of permutations $\psi_{n,m}$ which extend the regular languages in this way.*

6 Hierarchy relations

6.1 Auxiliary results

In order to argue better about ESRL-languages, we need:

Proposition 6.1 *Let $j, n, m \geq 1$. Let*

$$G_R^{[j]}(n,m) = (\{S_1^{[1]}, \ldots, S_1^{[j]}\}, \ldots, \{S_n^{[j]}, \ldots, S_n^{[j]}\}, \Sigma, M_R^{[j]}(n,m), S) \tag{5}$$

contains the following matrices:

1. $m_1 = (S \to S_1^{[1]} \dots S_n^{[1]})$ *and*
2. $m_x = (S_1^{[1]} \to \lambda, \dots, S_{n-1}^{[1]} \to \lambda, S_n^{[1]} \to x)$ *with* $x \in \Sigma_m^{\leq jn}$ *and*
3. $m_{b_1,\dots,b_n}^{[i]} = (S_1 \to b_1 S_1^{[(i \bmod j)+1]}, \dots, S_n \to b_n S_n^{[(i \bmod j)+1]})$
   with $b_\nu \in \Sigma_m$ *for* $1 \leq \nu \leq n$, $1 \leq i \leq j$.

Then, every $L \in \mathrm{ESRL}(n,m)$ *can be generated by a universal grammar* $G_R^{[j]}(n,m)$ *controlled by a regular set* C, *i.e.,* $L = L_C(G_R^{[j]}(n,m))$.

The universal grammar $G_R^{[1]}(n,m)$ corresponds to that introduced by Takada, see (3), and for general j, the claimed normal form can be derived from Takada's with some formal effort. We will only sketch that proof in the following. **Proof:** For every regular C, $L_C(G_R^{[j]}(n,m)) \in \mathrm{ESRL}(n,m)$, since $G_R^{[j]}(n,m) \in \mathrm{ESRL}(n,m)$, see Lemma 3.1. On the other hand, let $L \in \mathrm{ESRL}(n,m)$ be given by a regular control language $C \subset (M_R(n,m))^*$, where C is generated by the right-linear grammar $G = (V, M_R(n,m), P, S)$. Now, let $j > 1$. We explain how to define a right-linear grammar $G^{[j]} = (V^{[j]}, M_R^{[j]}(n,m), P^{[j]}, S')$ with $L_{L(G^{[j]})}(G_R^{[j]}(n,m)) = L$. Namely, let

$$V^{[j]} = \{S'\} \cup V \times \{0, \dots, jn\} \cup V \times \{0, \dots jn\}^2 \times (\prod_{\nu=2}^{n} \Sigma_m^{\leq (j-1)\nu})^2 \times \Sigma_m^{\leq (j-1)n} \times \{1, \dots, j\}.$$

Instead of defining $P^{[j]}$ formally, we prefer to explain the semantics of the new nonterminal symbols, leaving details to the reader. S' is the new initial symbol. At the very start, the derivation of the control grammar $G^{[j]}$ nondeterministically selects between two options: Whether to derive a control word which mimics a κ-step derivation of G, where $\kappa < jn$ or whether to derive a longer control word. The simulation of the first option is done using the subalphabet $V \times \{0, \dots, jn\}$, where the second component of the alphabet realizes a step counter. The second simulation option is more involved and will be explained in the following, where the numbers refer to the components in the cartesian set product.

1. As in the simulation of the finite part of "short words" explained above, one needs to keep track of the nonterminal state the simulated grammar G assumes.
2. Similarly, one has to count up to jn in order to prevent premature termination, since short words have been dealt with before. Moreover, this counter helps to build up the information stored within the $(\prod_{\nu=2}^{n} \Sigma_m^{\leq (j-1)\nu})^2 \times \Sigma_m^{\leq (j-1)n}$ component as explained below.
3. The second counter is used to count the last jn derivation steps; the beginning of this counting process is guessed nondeterministically. The twofold use of this counter is explained below.
4. In order to explain the use of component $(\prod_{\nu=2}^{n} \Sigma_m^{\leq (j-1)\nu})^2 \times \Sigma_m^{\leq (j-1)n}$, consider the subdivision of a word $w = a_1 \dots a_{3k+1} \in L$ (this means, we restrict ourselves to the special case $n = 3$ and $j = 2$):

   $$a_1 \dots a_{k-1}, a_k; a_{k+1} \dots a_{2k-2}, a_{2k-1} a_{2k}; a_{2k+1} \dots a_{3k-3}, a_{3k-2} a_{3k-1} a_{3k}; a_{3k+1}.$$

   According to grammar $G_R(n,m)$, nonterminal S_1 would derive the part $a_1 \dots a_k$, nonterminal S_2 the part $a_{k+1} \dots a_{2k}$, and the third nonterminal S_3 would derived both $a_{2k+1} \dots a_{3k}$ and, by using a terminal matrix, the last letter a_{3k+1}. The described partition is indicated in the sample word by way of a semicolon. Similarly, the partition induced by the (intended!) derivation using $G_R^{[2]}(n,m)$ is indicated by way of a comma. Now, observe that symbol a_k is generated as the last terminal symbol from nonterminal S_1 in grammar $G_R(n,m)$, while it is the first terminal symbol generated from nonterminal $S^{[1]}$ in grammar $G_R^{[2]}(n,m)$. Therefore, in the simulation control of $G_R^{[2]}(n,m)$, symbol a_k is nondeterminally guessed and

stored in the initial part of the derivation, while the guess is verified in the final part of the derivation. Recall that there are step counters for the initial and final parts of the derivation as described above, which obviously help to verify this guess. Similarly, the strings $a_{2k-1}a_{2k}$ and $a_{3k-2}a_{3k-1}a_{3k}$ have to be guessed, stored and finally verified.

On the other hand, when simulating, e.g., the first application of nonterminal S_2, we "know" that symbol a_{k+1} has been generated. This knowledge has to be stored, because only the next derivation step of the simulating can actually make use of this knowledge, whilst then the "knowledge" concerning symbol a_{k+2} has to be stored. So, we need a "sliding window storage", whose size for the ν-th ($\nu > 1$) part of the word is (at most) $(j-1)\nu - 1$; this sliding window can therefore be formalized by a word from $\Sigma_m^{<(j-1)\nu}$.

5. Finally, a counter $\bmod j$ is needed to ensure that applications of termination matrices are only possible if the counter has the value of one. This mirrors the fact that in $G_R^{[j]}(n,m)$, terminal matrices can be applied only to the nonterminal sequence $(S_1^{[1]}, \ldots S_N^{[1]})$.

□

In order to avoid giving a long and tedious formal argument, we only state an analogous result for ESL languages.

Proposition 6.2 *Let $j, n, m \geq 1$. Every $L \in \mathrm{ESL}(n,m)$ can be generated by a universal grammar $G^{[j]}(n,m)$ controlled by a regular set C, i.e., $L = L_C(G^{[j]}(n,m))$, where*

$$G^{[j]}(n,m) = (\{S_1^{[1]}, \ldots, S_1^{[j]}\}, \ldots, \{S_n^{[j]}, \ldots, S_n^{[j]}\}, \Sigma, M^{[j]}(n,m), S) \tag{6}$$

contains the following matrices:

1. $m_1 = (S \to S_1^{[1]} \ldots S_n^{[1]})$ *and*
2. $m_x = (S_1^{[1]} \to \lambda, \ldots, S_{n-1}^{[1]} \to \lambda, S_n^{[1]} \to x)$ *with* $x \in \Sigma_m^{\leq 2jn}$ *and*
3. $m^{[i]}_{b_1 \ldots b_n; c_1 \ldots c_n} = (S_1 \to b_1 S_1^{[(i \bmod j)+1]} c_1, \ldots, S_n \to b_n S_n^{[(i \bmod j)+1]} c_n)$ *with* $b_\nu, c_\nu \in \Sigma_m$ *for* $1 \leq \nu \leq n$, $1 \leq i \leq j$.

We now show that non-membership of certain languages can be proved for language classes with universal grammar characterization without deriving a special pumping lemma as [57, Lemma 3.8].

Example 6.3 $L = \{(ww^R)^n \mid w \in \Sigma_2^*\} \in \mathrm{ESL}(n)$, *see Theorem 4.8. Consider now the mapping φ_R associating derivation words to control words via universal grammar G_R defined in Eq. (3). If $L \in \mathrm{ESRL}(n)$, then $\varphi_R^{-1}(L)$ is regular, but*

$$\varphi_R^{-1}(L) = \{m_1(ww^R)m_\lambda \mid w \in \{m_{a,\ldots,a}, m_{b,\ldots,b}\}^*\}.$$

Example 6.4 $L = \{a^{4k}(ab)^{4k} \mid k > 0\} \in \mathrm{SRL}(2)$. *The "control language" (assume the language would be in* ESRL(2)*) is*

$$\varphi_R^{-1}(L) = \{m_1(m_{a,a}m_{a,b})^{2k}(m_{a,a}m_{b,b})^k m_\lambda \mid k > 0\}$$

which is not regular, indeed, so that $L \notin \mathrm{ESRL}(2)$. We can show further to this that if L were in ESL(2)*, then*

$$\varphi^{-1}(L) = \{m_1(m_{aa;bb}m_{ab;aa})^k m_{aa;ab}^k m_\lambda \mid k > 0\}$$

would be regular, a contradiction.

6.2 Hierarchical relations

Theorem 6.5 *For all $n \geq 1$, we have:*

1. $\mathrm{ESRL}(n) \subset \mathrm{SRL}(n) \subset \mathrm{SL}(n)$;
2. $\mathrm{ESL}(n) \subset \mathrm{SL}(n)$;
3. $\mathrm{ESRL}(n) \subset \mathrm{ESL}(n)$.

Proof: The inclusion relations themselves follow by definition, aside from the last one. Since the inclusion proof itself is not completely apparent at first glance, we include it here. Consider $L \in \mathrm{ESRL}(n,m)$ with $L = L_C(H(n,m))$, where the regular control language C is given by a right-linear grammar $G = (V, M'(n,m), P, S)$. We define a right-linear grammar $\bar{G} = (\bar{V}, M(n,m), \bar{P}, \bar{S})$ which generates $\bar{C}$ and controls $G(n,m)$ so that $L_C(G_R^{[2]}(n,m)) = L_{\bar{C}}(G(n,m))$. Let $\bar{P} = P \times P \times \Sigma_m^{<2n}$.

initial matrices If $S \to m_{a_1,\ldots,a_n} A, B \to m_{b_1,\ldots,b_n} C, C \to m_\lambda \in P$,
we put $S \to m_{a_1,\ldots,a_n;b_1,\ldots,b_n}(A, B, \lambda)$ into $\bar{P}$.
If $S \to m_{a_1,\ldots,a_n} A, B \to m'_{b_1,\ldots,b_n} C, C \to m_{c_1\ldots c_k} \in P$, $0 < k < 2n$,
we put $S \to m_{a_1,\ldots,a_n;b_1,\ldots,b_{n-1},c_k}(A, B, b_n c_1 \ldots c_{k-1})$ into $\bar{P}$.

nonterminal matrices If $A \to m_{a_1,\ldots,a_n} A'$ and $B \to m'_{b_1,\ldots,b_n} B' \in P$, or $A \to m'_{a_1,\ldots,a_n} A'$ and $B \to m_{b_1,\ldots,b_n} B' \in P$, then put $(A, B', \lambda) \to m_{a_1,\ldots,a_n;b_1,\ldots,b_n}(A', B, \lambda)$ into $\bar{P}$ as well as $(A, B', c_1 \ldots c_k) \to m_{a_1,\ldots,a_n;b_1,\ldots,b_{n-1},c_k}(A', B, b_n c_1 \ldots c_{k-1})$ for all $0 < k < 2n$, $c_i \in \Sigma_m$.

terminal rules Put $(A, A, x) \to m_x$ into $\bar{P}$ for all $A \in P$, $x \in \Sigma_m^{<2n}$.

The derivation of the regular grammar is simulated in parallel from left to right (for the left-hand part of the control word) and from right to left (for its right-hand part). The only formal difficulty arises from the fact that the "remainder" of the input word decomposition is at a different location in linear and right-linear grammars. Instead of giving a formal induction argument, we supply an example of this construction (see Example 6.6 below).

Now, we show the strictness assertions:

Claim 1. (a) For $n = 2$, see Example 6.4. Similar counterexample languages for $n > 2$ can be easily defined.

(b) $L_n = \{a_1^k \ldots a_{2n}^k \mid k \geq 1\} \notin \mathrm{SRL}(n)$, see [18, Lemma 1.5.6 (iv)]. It is easy to give an $\mathrm{ESL}(n)$ grammar for this language.

Claim 2. For $n = 2$, see Example 6.4. Similar counterexample languages for $n > 2$ can be easily defined.

Claim 3. For the strictness of the inclusion, either consider Example 6.3 or the language given in part (b) of 1. □

Example 6.6 $G = (\{A_1, A_2, A_3\}, \{B_1, B_2\}, \{a, b\}, M, S)$ *with*

$$\begin{aligned} M &= \{(S \to A_1 B_1), (A_1 \to \lambda, B_1 \to \lambda)\} \\ &\cup \{(A_i \to a A_{(i \bmod 3)+1}, B_j \to b B_{3-j}) \mid 1 \leq i \leq 3, 1 \leq j \leq 2\} \end{aligned}$$

is a ESRL *grammar that generates*

$$L(G) = \{a^{6n} b^{6n} \mid n \geq 0\}.$$

We shall later consider the simulation of the following sample derivation in G:

$$\begin{aligned} S &\Rightarrow A_1 B_1 \Rightarrow a A_2 b B_2 \Rightarrow a a A_3 b b B_1 \\ &\Rightarrow a a a A_1 b b b B_2 \Rightarrow a a a a A_2 b b b b B_1 \Rightarrow a a a a a A_3 b b b b b B_2 \\ &\Rightarrow a a a a a a A_1 b b b b b b B_2 \Rightarrow a a a a a a b b b b b b. \end{aligned}$$

The construction of the preceding theorem yields the following equivalent ESL *grammar:*
$G' = (\{(A_1,A_1),\ (A_1,A_2),\ (A_1,A_3),\ (A_2,A_1),\ (A_2,A_2),\ (A_2,A_3),\ (A_3,A_1),\ (A_3,A_2),\ (A_3,A_3)\},$
$\{(B_1,B_1),\ (B_1,B_2),\ (B_2,B_1),\ (B_2,B_2)\},\ \{a,b\},\ M',\ S)$, *where*

$$\begin{aligned} M' &= \{(S \to (A_1,A_1)(B_1,B_1))\} \\ &\cup \{((A_i,A_i) \to \lambda, (B_j,B_j) \to \lambda) \mid 1 \le i \le 3, 1 \le j \le 2\} \\ &\cup \{((A_i, A_{(\ell \bmod 3)+1}) \to a(A_{(i \bmod 3)+1}, A_\ell)a, (B_j, B_{3-m}) \to b(B_{3-j}, B_m)b) \mid \\ &\qquad 1 \le i,\ell \le 3, 1 \le j,m \le 2\}. \end{aligned}$$

The sample derivation can be simulated as follows:

$$\begin{array}{rcccl} S & \Rightarrow & (A_1,A_1)(B_1,B_1) & \Rightarrow & a(A_2,A_3)ab(B_2,B_2)b \\ & \Rightarrow & aa(A_3,A_2)aabb(B_1,B_1)bb & \Rightarrow & aaa(A_1,A_1)aaabbb(B_2,B_2)bbb \\ & \Rightarrow & aaaaaabbbbbb & & \end{array}$$

Corollary 6.7 *Let* $n > 1$. $\mathrm{ESL}(n)$ *and* $\mathrm{ESRL}(n)$ *are not closed under*

- *homomorphism,*
- *inverse homomorphism,*
- *rational transductions,*
- *catenation, and*
- *Kleene star.*

Proof: Combining Cor. 3.12, 3.15 and Theorem 6.5, we obtain non-closure of the language families under homomorphism and hence under rational transductions.

Non-closure under catenation and Kleene star immediately follows from the corresponding results on $\mathrm{SL}(n)$ ($\mathrm{SRL}(n)$) languages, since the typical examples for proving such results are already $\mathrm{ESL}(n)$ ($\mathrm{ESRL}(n)$) languages. □

Following Greibach [25] and Khabbaz [30], let $\mathrm{CONTROL}(\mathcal{G},\mathcal{L})$ denote the family of languages which is defined by controlling type-$\mathcal{G}$-grammars using languages from $\mathcal{L}$. This concept is interesting for learning theory, since it can be iterated by defining $\mathrm{CONTROL}_0(\mathcal{G},\mathcal{L}) = \mathcal{L}$ and

$$\mathrm{CONTROL}_n(\mathcal{G},\mathcal{L}) = \mathrm{CONTROL}(\mathcal{G}, \mathrm{CONTROL}_{n-1}(\mathcal{G},\mathcal{L})),$$

and in case $\mathcal{G}$ contains an unambiguous grammar which, controlled by some language from $\mathcal{L}$, describes $\mathrm{CONTROL}(\mathcal{G},\mathcal{L})$, and if $\mathcal{L}$ can be learned efficiently, then the whole hierarchy $\mathrm{CONTROL}_n(\mathcal{G},\mathcal{L})$ can be learned efficiently as well. Takada explored in [56] the hierarchy $\mathrm{CONTROL}_n(\mathrm{ESL}(1),\mathrm{ESRL}(1))$. He posed the question what would happen if

- $\mathrm{CONTROL}(\mathrm{ESRL}(n),\mathrm{ESRL}(\ell))$ or
- the hierarchy $\mathrm{CONTROL}_n(\mathrm{ESRL}(2),\mathrm{ESRL}(2))$

were to be considered. The following theorem will answer that question (and others, too).

Theorem 6.8 *Let* $n,\ell \ge 1$. *Then*

1. $\mathit{CONTROL}(\mathrm{ESRL}(n),\mathrm{ESRL}(\ell)) = \mathrm{ESRL}(n\ell)$.
2. $\mathit{CONTROL}(\mathrm{ESL}(n),\mathrm{ESL}(\ell)) = \mathrm{ESL}(n\ell)$.
3. $\mathit{CONTROL}_n(\mathrm{ESRL}(2),\mathrm{ESRL}(2)) = \mathrm{ESRL}(2^{n+1})$.
4. $\mathit{CONTROL}_n(\mathrm{ESL}(1),\mathrm{ESL}(1)) = \mathrm{ESL}(2^n)$.

Proof: Claim 1. We will restrict the following argument to languages from $\bigcup_{k>0} \Sigma_m^{n\ell k}$. There is no loss of generality, since one can construct universal grammars both for ESRL(n) and for ESRL(ℓ) with terminating matrices of the form $(S_1 \to \lambda, \ldots, S_{r-1} \to \lambda, S_r \to x)$ with $r = n$ or $r = \ell$ and $x \in \Sigma^{<n\ell}$ as in Prop. 6.1.

Therefore, consider a word $w = a_1 a_2 \ldots a_{n\ell k}$ from an ESRL($n\ell$) language. This essentially means that the permuted sequence

$$\underbrace{\underbrace{(a_1 a_{k+1} \ldots a_{(n\ell-1)k+1})}_{n\ell \text{ elements}} (a_2 a_{k+2} \ldots a_{(n\ell-1)k+2}) \ldots (a_k a_{2k} \ldots a_{n\ell k})}_{k \text{ groups}} \tag{7}$$

lies in the canonical regular control language. Now, consider w as belonging to an ESRL(n) language, which means that the permuted sequence

$$\underbrace{(a_1 a_{\ell k+1} \ldots a_{(n-1)\ell k+1})(a_2 a_{\ell k+2} \ldots a_{(n-1)\ell k+2}) \ldots (a_{\ell k} a_{2\ell k} \ldots a_{n\ell k})}_{\ell k \text{ groups}}$$

lies in the canonical ESRL(ℓ) language, which in turn results in a regular control language containing

$$\underbrace{\underbrace{\underbrace{[\underbrace{(a_1 a_{\ell k+1} \ldots a_{(n-1)\ell k+1})}_{n \text{ elements}} (a_{k+1} \ldots) \ldots (a_{(\ell-1)k+1} \ldots a_{(n\ell-1)k+1})] \ldots .}_{\ell \text{ subgroups}}}_{k \text{ groups}}} \tag{8}$$

It is now easy to see that a finite automaton capable of recognizing sequences consisting of k groups each of $n\ell$ elements as in (7) (note that each "group" can be seen as a special letter in the control alphabet) can be easily modified to recognize sequences consisting of ℓk subgroups each of n elements as in (8) and vice versa, since each subgroup in (8) is merely a permuation of a group in (7) and because n and ℓ are considered constants.

Claim 2. By using Prop. 6.2, we can restrict our following discussion to words $w = a_1 a_2 \ldots a_{4n\ell k}$ of a length divisible by $4n\ell$. Assume that $w \in \text{ESL}(2n\ell)$. This essentially means that the permuted sequence

$$\underbrace{\underbrace{(a_1 a_{2k+1} \ldots a_{(2n\ell-1)2k+1}; a_{2k} a_{4k} \ldots a_{4n\ell k})}_{4n\ell \text{ elements}} \ldots (a_k a_{3k} \ldots a_{4n\ell k-k}; a_{k+1} \ldots a_{4n\ell k-k+1})}_{k \text{ groups}} \tag{9}$$

lies in the canonical regular control language. Now, regard w as an element of an ESL(n) language, meaning that the permuted sequence

$$\begin{array}{ll}
\underbrace{(a_1 a_{4\ell k+1} \ldots a_{(n-1)4\ell k+1}; a_{4\ell k} \ldots a_{4n\ell k})}_{\text{subgroup of } 2n \text{ elements}} & \ldots (a_{2k} a_{4\ell k+2k} \ldots a_{(n-1)4\ell k+2k}; \alpha) \\
(a_{2k+1} a_{4\ell k+2k+1} \ldots a_{(n-1)4\ell k+2k+1}; \beta) & \ldots\ldots \\
\ldots & \ldots\ldots \\
(a_{(\ell-1)2k+1} a_{4\ell k+(\ell-1)2k+1} \ldots a_{(n-1)4\ell k+(\ell-1)2k+1}; \gamma) & \ldots (a_{2\ell k} a_{6\ell k} \ldots a_{(4n\ell k-2\ell k)}; \delta)
\end{array}$$

[Remarks: The description has $2\ell k$ rows corresponding to groups of parenthesized element subgroups. We used the abbreviations

$$\begin{aligned}
\alpha &= a_{4\ell k-2k+1} \ldots a_{4n\ell k-2k+1}, \\
\beta &= a_{4\ell k-2k} \ldots a_{4n\ell k-2k}, \\
\gamma &= a_{4\ell k-(\ell-1)2k} \ldots a_{2n\ell k-(\ell-1)2k} \text{ and} \\
\delta &= a_{2\ell k+1} \ldots a_{(4n\ell k-2\ell k+1)} .]
\end{aligned}$$

lies in the canonical ESL(ℓ) language, which in turn results in a regular control language containing

$$
\begin{array}{l}
[(a_1 a_{4\ell k+1} \ldots a_{(n-1)4\ell k+1}; a_{4\ell k} \ldots a_{4n\ell k}) \\
(a_{2k+1} a_{4\ell k+2k+1} \ldots a_{(n-1)4\ell k+2k+1}; a_{4\ell k-2k} \ldots a_{4n\ell k-2k}) \\
\ldots \\
(a_{(\ell-1)2k+1} \ldots a_{(n-1)4\ell k+(\ell-1)2k+1}; a_{4\ell k-(\ell-1)2k} \ldots a_{4n\ell k-(\ell-1)2k}); \\
(a_{2k} a_{4\ell k+2k} \ldots a_{(n-1)4\ell k+2k}; a_{4\ell k-2k+1} \ldots a_{4n\ell k-2k+1}) \\
\ldots \\
(a_{2\ell k} a_{6\ell k} \ldots a_{(n-1)4\ell k+2\ell k}; a_{2\ell k+1} \ldots a_{(n-1)4\ell k+2\ell k+1})] \\
\ldots
\end{array}
$$

The whole word is decomposed into k groups the first one of which is listed above in detail. Such a group (indicated as a word within brackets) contains 2ℓ subgroups (indicated by parentheses) each having $2n$ elements. We want to consider this first group alone in order to verify that the set of indices collected therein is the same as the set of indices of the first group of the ESL($2n\ell$) decomposition listed in Eq. (9). The first part of the sequence $a_1 a_{2k+1} \ldots a_{(2n\ell-1)2k+1}; a_{2k} a_{4k} \ldots a_{4n\ell k}$ can be obtained by subsequently reading from that group:

- the first letter of the first part of the first subgroup of the first part;
- the first letter of the first part of the second subgroup of the first part; ...
- the first letter of the second part of the last subgroup of the second part;
- the first letter of the second part of the penultimate subgroup of the second part; ...
- the second letter of the first part of the first subgroup of the first part;
- the second letter of the first component of the second subgroup of the first part; ...

Similarly, the second part, i.e. $a_{2k} a_{4k} \ldots a_{4n\ell k}$ is obtainable via reading:

- the first letter of the first part of the first subgroup of the second part;
- the first letter of the first part of the second subgroup of the second part; ...
- the first letter of the second part of the last subgroup of the first part;
- the first letter of the second part of the penultimate subgroup of the first part; ...

Since a group spelled in this way comprises just one letter from the terminal alphabet of the control language, the assertion is verified, as the procedure described above results in re-naming the terminal alphabet.

Claims 3. and 4. These are straight-forward consequences from the first two assertions by induction. □

Theorem 6.9 *Let $n, n' \geq 1$. Then*

1. ESRL(n) $\subseteq$ ESRL(n') *iff n divides n';* ESRL(n) $\neq$ ESRL(n') *iff $n \neq n'$.*
2. ESL(n) $\subseteq$ ESL(n') *iff n divides n';* ESL(n) $\neq$ ESL(n') *iff $n \neq n'$.*

Proof: Let us first show the inclusions in case n divides n', i.e., $n' = \ell n$.
Claim 1. We can make use of Prop. 6.1 by assuming that we are to simulate the control language C of a universal $G_R^{[\ell]}(n,m)$ grammar by a control language C' of $G_R(\ell n, m)$. Since $L_{C'}(G_R(\ell n, m)) = L_{C''}(G_R(n,m))$ for some $C'' \in$ ESRL(ℓ) as shown in Theorem 6.8, the inclusion is proved by observing that regular languages are included in ESRL(ℓ).
Claim 2. This property can also be reduced to ESL(1) $\in$ ESL(ℓ) (which is not hard to comprehend, although the complete formal proof of this fact is quite tedious; basically, one can use ideas

from Theorem 6.5, taking some special care of the terminal matrices) by using Prop. 6.1 and Theorem 6.8.

As regards counterexamples, consider again the case of ESRL-grammars firstly. Assume that n does not divide n'. For a moment, assume further that n and n' are relative primes, i.e., the greatest common divisor of n and n' is one. Consider now

$$L_n = \{a_1^k \dots a_n^k \mid k \geq 1\} \in \mathrm{ESRL}(n).$$

We will show that the canonical control language C on the universal grammar $G_R(n', n)$ is not regular.[5] Assume the contrary, that C would be regular. This means that C is accepted by a minimal deterministic finite automaton (DFA) with p states, i.e., p is the "pumping constant" of the pumping lemma for regular languages. Consider the control word corresponding to

$$w = a_1^{2pn'} \dots a_n^{2pn'} \in L_n.$$

What does the corresponding control word (i.e., the induced permutation on w) look like? In order to avoid unnecessary complication, we restrict our argument to two concrete examples, covering the two cases $n > n'$ and $n < n'$. Firstly, suppose $n = 5$, $n' = 2$. The permutation of w is

$$w' = (a_1a_3)^{2p}(a_1a_4)^{2p}(a_2a_4)^{2p}(a_2a_5)^{2p}(a_3a_5)^{2p}.$$

If w' is accepted by some p-state DFA, by the pigeon-hole principle, some suffix

$$w'' = (a_1a_3)^{p'}(a_1a_4)^{2p}(a_2a_4)^{2p}(a_2a_5)^{2p}(a_3a_5)^{2p}$$

of w' must also be accepted for some $p' < 2p$, since some state of the DFA must be entered twice while scanning $(a_1a_3)^{2p}$. Since w' contains less a_1 symbols than a_2 symbols, the "reversed permutation" cannot lie in L_n, contradicting our assumption that C is regular. Secondly, consider $n = 2$, $n' = 5$. The permutation of w is $w' = (a_1^3a_2^2)^{2p}(a_1^2a_2^3)^{2p}$. The pigeon-hole principle now shows that some $w'' = (a_1^3a_2^2)^{p'}(a_1^2a_2^3)^{2p}$ must be accepted by the p-state DFA accepting C. This obviously destroys the balance between the number of occurrences of a_1 and that of a_2, so that the "reversed permutation" cannot lie in L_n, contradicting our assumption that C is regular. This argument can be easily generalized to any pair of relative primes n, n'. If n and n' are not relative primes, then $n = r\ell$, $n' = r\ell'$ for some $r > 1$. Now, $L_\ell^r \in \mathrm{ESRL}(n')$ iff $L_\ell \in \mathrm{ESRL}(\ell')$, but $L_\ell \notin \mathrm{ESRL}(\ell')$ according to the argument given above.

We consider now the case of ESL-grammars. The argument concerning a counterexample is similar to that given above for ESRL-grammars, but technically more involved. The following therefore contains only a brief outline of the proof. We now consider as a counterexample language

$$L'_n = \{a_1^k \dots a_{2n}^k \mid k \geq 1\} \in \mathrm{ESL}(n).$$

In the case of n being relatively prime to n', $L'_n \notin \mathrm{ESL}(n')$. This can be seen when taking into account

$$w = a_1^{2pn'} \dots a_{2n}^{2pn'} \in L'_n,$$

where we assume that there exists a p-state DFA for the control language of $G(n', 2n)$. In the case of $n = 5$, $n' = 2$ (representing the case $n > n'$), we must regard the permutation (assumed control word)

$$w' = (a_1a_5; a_6a_{10})^{2p}(a_2a_4; a_7a_9)^{2p}(a_3a_3; a_8a_8)^{p}$$

corresponding to w. In the case of $n = 2$, $n' = 5$ (representing the case $n < n'$), we obtain the permutation

$$w'' = (a_1a_1a_2a_3a_4; a_1a_2a_3a_4a_4)^{2p}(a_1a_2a_2a_3a_4; a_1a_2a_3a_3a_4)^{2p}.$$

Applying the pumping argument to w' and w'', one easily sees that the assumed p-state DFA cannot exist, contradicting the assumption $L'_n \in \mathrm{ESL}(n')$. The case when n and n' are not relatively prime can be dealt with as above. □

[5] In our counterexamples, we use a rather huge alphabet; by rather simple coding, one can find counterexamples of languages over the alphabet Σ_2.

7 Conclusions and prospects

We showed in this paper how techniques well-known in formal language theory can be used to generalize published learning algorithms for regular languages to infer non-regular languages. In actual fact, most of the technical parts of the paper are formal language results; but we also show how these results can be employed to design learning algorithms within the MAT model.

We suggest five ways to continue our work.

- It could be interesting to investigate sub-classes of even linear matrix languages: e.g., it is easy to define *even* n-parallel linear languages (compare [43, 60, 61]). Maybe, such classes have better learning properties. In general more restricted are the n-metalinear languages [21] (which lack the synchronization feature in n-parallel linear languages); however, an "even" version of this mechanism is conjectured to be uncomparable with its n-parallel linear even counterpart. Note that also for that case control language hierarchies have been investigated by Khabbaz [31].

- On the other hand, our results can be easily adapted to so-called k-linear (matrix) languages (compare [2, 47]). Unfortunately, we were unable to extend our arguments uniformly to cover all ratios k.

- Furthermore, there should be connections to learning of multi-tape finite automata as suggested by the constructions given in [44, 60, 52, 57]. Especially, Yokomori [64] uses a more general notion of "deterministic automaton" (more accurately referred to as "unambiguous automaton") based on works of Brauer and Lange [14] which are also learnable in polynomial time using an algorithm different from Angluin's.

- Instead of using a DFA learner in order to learn control languages of the universal grammar $G(n,m)$, one could use, e.g., deterministic one counter-learner, see [10], or deterministic binary systolic tree automata, confer [62], which easily extends the class of efficiently learnable languages.

- Other learning models should be taken into account. For example, which subclasses of $ESL(n)$ are learnable using only positive examples? According to the famous result of Gold [23], not all such languages can be learnable, but interesting subclasses can be easily identified using the framework of control languages, confer, e.g., [3, 20].

Finally, we should like to point to the fact that Balcázar, Díaz, Gavaldà and Watanabe developped in [8] a CRCW PRAM learning algorithm that uses polynomially many processors and learns DFA in time $O(n/\log n)$ exactly.[6] Since the machinery exhibited in this paper mainly uses uniform re-ordering of the letters of the input word in order to create an input of the DFA learner, we get a similarly efficient parallel learning algorithm for each of the classes considered here for learning as well. Moreover, the bound is tight, because it is shown in [8] (based on [7, 16, 17]) that no CRCW PRAM machine using a number of processors polynomial in n and m can identify DFA in $o(n/\log n)$ time.

Acknowledgments: We thank for discussions with E. Mäkinen, J. M. Sempere, F. Stephan, and Y. Takada. The talk at COCOON'99 [19] (based on this paper) was given by K. Reinhardt.

References

[1] V. Amar and G. Putzolu. On a family of linear grammars. *Information and Control*, 7:283–291, 1964.

[2] V. Amar and G. Putzolu. Generalizations of regular events. *Information and Control*, 8:56–63, 1965.

[6] Here, n is the number of states of the automaton and m is the length of the longest counterexample to an equivalence query.

[3] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.

[4] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[5] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[6] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proc. 23rd ACM Symp. Theory of Computing STOC*, pages 444–454, 1991.

[7] J. L. Balcázar, J. Díaz, R. Gavaldà, and O. Watanabe. The query complexity of learning DFA. *NEWGEN: New Generation Computing*, 12, 1994.

[8] J. L. Balcázar, J. Díaz, R. Gavaldà, and O. Watanabe. An optimal parallel algorithm for learning DFA. *Journal of Universal Computer Science*, 2(3):97–112, March 1996.

[9] F. Bergadano and S. Varricchio. Learning behaviors of automata from shortest counterexamples. In *Proc. 2nd Europ. Conf. Computational Learning Theory EUROCOLT'95*, volume 904 of *LNCS*, pages 380–391, 1995.

[10] P. Berman and R. Roos. Learning one-counter languages in polynomial time. In *Proc. 28th Symp. Foundations of Computer Science FOCS'87*, pages 61–67. IEEE, 1987.

[11] J. Berstel. *Transductions and Context-Free Languages*, volume 38 of *LAMM*. Stuttgart: Teubner, 1979.

[12] A. Birkendorf, A. Böker, and H.U. Simon. Learning deterministic finite automata from smallest counterexamples. In *Proc. 9th Ann. ACM/SIAM Symp. Discr. Alg. SODA'98*, pages 599–608, 1998.

[13] R. V. Book, S. A. Greibach, and C. Wrathall. Comparisons and reset machines. In *Automata, Languages and Programming ICALP'78*, volume 62 of *LNCS*, pages 113–124, 1978.

[14] W. Brauer and K.-J. Lange. Non-deterministic two-tape automata are more powerful than deterministic ones. In *Proc. 2nd Symp. Theoretical Aspects of Computer Science STACS'85*, volume 182 of *LNCS*, pages 71–79, 1985.

[15] J. A. Brzozowski. Regular-like expressions for some irregular languages. In *IEEE Conf. Record of 9th Ann. Symp. on Switching and Automata Theory SWAT*, pages 278–280, 1968.

[16] N.H. Bshouty and R. Cleve. On the exact learning of formulae in parallel. In *33rd Symp. Foundations of Computer Science FOCS'92*, pages 513–522, 1992.

[17] N.H. Bshouty, S.A. Goldman, T.R. Hancock, and S. Matar. Asking queries to minimize errors. In *Proc. 6th Annual ACM Conf. on Computational Learning Theory*, pages 41–50, 1993.

[18] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs in Theoretical Computer Science*. Berlin: Springer, 1989.

[19] H. Fernau. Efficient learning of some linear matrix languages. In *Computing and Combinatorics, 5th Ann. Intern. Conf. COCOON'99*, volume 1627 of *LNCS*, pages 221–230, 1999.

[20] P. García and E. Vidal. Inference of k-testable languages in the strict sense and applications to syntactic pattern recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:920–925, 1990.

[21] S. Ginsburg and E. H. Spanier. Finite-turn pushdown automata. *SIAM Journal of Control*, 4(3):429–453, 1966.

[22] S. Ginsburg and E. H. Spanier. Control sets on grammars. *Mathematical Systems Theory*, 2:159–177, 1968.

[23] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[24] S. Greibach. Comments on universal and left universal grammars, context-sensitive languages, and context-free grammar forms. *Information and Control*, 39:135–142, 1978.

[25] S. A. Greibach. One way finite visit automata. *Theoretical Computer Science*, 6(2):175–221, 1978.

[26] S. Hirose and M. Nasu. Left universal context-free grammars and homomorphic characterizations of languages. *Information and Control*, 50:110–118, 1981.

[27] O. Ibarra. Simple matrix languages. *Information and Control*, 17:359–394, 1970.

[28] O. H. Ibarra and T. Jiang. Learning regular languages from counterexamples. *Journal of Computer and System Sciences*, 43:299–316, 1991.

[29] T. Kasai. A universal context-free grammar. *Information and Control*, 28:30–34, 1975.

[30] N. A. Khabbaz. A geometric hierarchy of languages. *Journal of Computer and System Sciences*, 8:142–157, 1974.

[31] N. A. Khabbaz. Control sets on linear grammars. *Information and Control*, 25:206–221, 1974.

[32] T. Koshiba, E. Mäkinen, and Y. Takada. Learning deterministic even linear languages from positive examples. *Theoretical Computer Science*, 185(1):63–79, October 1997.

[33] W. Kuich. On the entropy of context-free languages. *Information and Control*, 16:173–200, 1970.

[34] E. Mäkinen. A note on the grammatical inference problem for even linear languages. *Fundamenta Informaticae*, 25:175–181, 1996.

[35] C. Martín-Vide. Natural language understanding: a new challenge for grammar systems. *Acta Cybernetica*, 12:461–472, 1996.

[36] H. Maurer and W. Kuich. Tuple languages. In *Proc. of the ACM International Computing Symposium*, pages 882–891, 1970.

[37] A. Pascu and Gh. Păun. On simple matrix grammars. *Bull. Math. de la Soc. Sci. Math. de Roumanie*, 20:333–340, 1976.

[38] A. Pascu and Gh. Păun. A homomorphic representation of simple matrix languages. *Information and Control*, 35:267–275, 1977.

[39] Gh. Păun. Linear simple matrix languages. *Elektronische Informationsverarbeitung und Kybernetik (EIK)*, 14:377–384, 1978.

[40] Gh. Păun and M. Novotný. On a family of linearly grammatizable languages. *Fundamenta Informaticae*, 10:143–148, 1987.

[41] V. Radhakrishnan and G. Nagaraja. Inference of even linear grammars and its application to picture description languages. *Pattern Recognition*, 21:55–62, 1988.

[42] R. L. Rivest and R. E. Schapire. Inference on finite automata using homing sequences. *Information and Computation*, 103:299–347, 1993.

[43] R. D. Rosebrugh and D. Wood. Image theorems for simple matrix languages and n-parallel languages. *Mathematical Systems Theory*, 8:150–155, 1974.

[44] A. L. Rosenberg. A machine realization of the linear context-free languages. *Information and Control*, 10:175–188, 1967.

[45] G. Rozenberg. A note on universal grammars. *Information and Control*, 34:172–175, 1977.

[46] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Volume I.* Berlin: Springer, 1997.

[47] A. L. Semenov. Regularity of languages k-linear for various k. *Soviet Math. Dokl.*, 15(2):492–496, 1974.

[48] J. M. Sempere and P. García. A characterization of even linear languages and its application to the learning problem. In *Proc. of the Second International Colloquium on Grammatical Inference (ICGI-94): Grammatical Inference and Applications*, volume 862 of *LNCS/LNAI*, pages 38–44, Berlin, 1994. Springer.

[49] S. M. Shieber. Evidence against the context-freeness of natural languages. *Linguistics and Philosophy*, 8:333–343, 1985.

[50] R. Siromoney. On equal matrix languages. *Information and Control*, 14:133–151, 1969.

[51] R. Siromoney. Unambiguous equal matrix languages. *Information and Control*, 20:1–8, 1972.

[52] Y. Takada. Grammatical inference of even linear languages based on control sets. *Information Processing Letters*, 28:193–199, 1988.

[53] Y. Takada. Algorithmic learning theory of formal languages and its applications. Technical Report IIAS-RR-93-6E, IIAS-SIS: International Institute for Advanced Study of Social Information Sciences, Fujitsu Laboratories, 1992. Also PhD Thesis.

[54] Y. Takada. Learning even equal matrix languages based on control sets. In *Parallel Image Analysis, ICPIA'92*, volume 652 of *LNCS*, pages 274–289, 1992.

[55] Y. Takada. Learning semilinear sets from examples and via queries. *Theoretical Computer Science*, 104:207–233, 1992.

[56] Y. Takada. A hierarchy of language families learnable by regular language learning. *Information and Computation*, 123:138–145, 1995.

[57] Y. Takada. Learning formal languages based on control sets. In *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *LNCS/LNAI*, pages 317–339, 1995.

[58] K. W. Wagner. On the intersection of the class of linear context-free languages and the class of single-reset languages. *Information Processing Letters*, 23:143–146, 1986.

[59] D. J. Weir. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, 104:235–261, 1992.

[60] D. Wood. n-linear simple matrix languages and n-parallel linear languages. Technical Report 72/3, McMaster University, Hamilton, Canada, 1972.

[61] D. Wood. n-linear simple matrix languages and n-parallel linear languages. *Rev. Roumaine Math. Pures Appl.*, 22:403–412, 1977.

[62] T. Yokomori. On learning systolic languages. In *Proc. of the 3rd Workshop on Algorithmic Learning Theory (ALT '92)*, volume 743 of *LNCS/LNAI*, pages 41–52, Tokyo, Japan, October 1992. Springer.

[63] T. Yokomori. Learning non-deterministic finite automata from queries and counterexamples. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 13: Machine Intelligence and Inductive Learning*, volume 13. Clarendon, 1994.

[64] T. Yokomori. Learning two-tape automata from queries and counterexamples. *Mathematical Systems Theory*, pages 259–270, 1996.