Learning Pattern Languages Over Groups*

Rupert Hölzl[1], Sanjay Jain[2] and Frank Stephan[2 3]

[1] Institute 1, Faculty of Computer Science, Universität der Bundeswehr München, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
Email: r@hoelzl.fr

[2] School of Computing, National University of Singapore, Singapore 117417.
Email: sanjay@comp.nus.edu.sg

[3] Department of Mathematics, National University of Singapore, Singapore 119076.
Email: fstephan@comp.nus.edu.sg

Abstract. This article studies the learnability of classes of pattern languages over automatic groups. It is shown that the class of bounded unions of pattern languages over finitely generated Abelian automatic groups is explanatorily learnable. For patterns in which variables occur at most n times, it is shown that the classes of languages generated by such patterns as well as their bounded unions are, for finitely generated automatic groups, explanatorily learnable by an automatic learner. In contrast, automatic learners cannot learn the unions of up to two arbitrary pattern languages over the integers. Furthermore, there is an algorithm which, given an automaton describing a group G, generates a learning algorithm M_G such that either M_G explanatorily learns all pattern languages over G or there is no learner for this set of languages at all, not even a non-recursive one. For some automatic groups, non-learnability results of natural classes of pattern languages are provided.

1 Introduction

Gold [13] introduced inductive inference, a model for learning classes of languages $\mathcal{L}$ (a language is a subset of Σ^* for some alphabet Σ) from positive data; this model was studied extensively in the subsequent years [1, 2, 10, 11, 32]. Inductive inference can be described as follows: The learner reads as input, one by one, elements of a language L from a class $\mathcal{L}$ of languages; these elements are provided in an arbitrary order and with arbitrarily many repetitions and pauses; such a presentation of data is called a *text* for L. While progressively reading it, the learner forms hypotheses about which grammar may have produced the language it is being shown; subsequent data may cause it to revise earlier hypotheses. One says the learner learns L if the sequence of hypotheses converges to a grammar that indeed produces L. The learner is said to learn the entire class $\mathcal{L}$ of languages if it learns each $L \in \mathcal{L}$. This model of learning is often referred to as explanatory learning (**Ex**-learning) or learning in the limit (the reader is referred to Section 2 for the formal details of this and other concepts used informally in this introduction).

* This research has been supported by Ministry of Education Academic Research Fund Tier 1 grants R146-000-181-112 and R252-000-534-112 to Frank Stephan (PI) and Sanjay Jain (Co-PI); furthermore, S. Jain is supported in part by NUS grant C252-000-087-001.

An important concept in learning theory introduced by Angluin [2] is that of (non-erasing) pattern languages, that is, languages that can be generated by substituting variables in patterns with non-empty strings (in this article only finite strings are considered). Shinohara [37] generalised it to the concept of *erasing pattern languages* in which the variables are allowed to be substituted by empty strings. Suppose Σ is an alphabet set (usually finite) and X is an infinite set of variables. A pattern is a string over $\Sigma \cup X$. A substitution is a mapping from X to Σ^*. Using different substitutions for variables, different strings can be generated from a pattern. The language generated by a pattern is the set of strings that can be obtained from the pattern using some substitution. Angluin showed that when variables can be substituted only by non-empty strings then the class of pattern languages is **Ex**-learnable; Lange and Wiehagen [26] provided a polynomial-time learner for non-erasing pattern languages. On the other hand, Reidenbach [35] showed that if arbitrary strings (including empty strings) are allowed as substitutions, then the class of pattern languages is not **Ex**-learnable if the alphabet size is 2, 3 or 4. Shinohara and Arimura [38] considered the learning of unbounded unions of pattern languages from positive data.

This article explores learnability of pattern languages over groups; pattern languages and verbal languages (that is, languages generated by patterns without constants) have been used in group theory extensively, for example in the work showing the decidability of the theory of the free group [22, 23]. Miasnikov and Romankov [28] studied under which conditions verbal languages are regular (in certain representations of the group).

In the following, consider a group $(G, \circ)$. The elements of G are then used as constants in patterns and the substitutions map variables to group elements; for example, if

$$L(xax^{-1}yyab) = \{x \circ a \circ x^{-1} \circ y \circ y \circ a \circ b\colon x, y \in G\}$$

then letting $x = b$ and $y = a^{-2}$ produces $bab^{-1}a^{-3}b \in L(xax^{-1}yyab)$. That is, concatenation in the pattern is replaced by the group operation $\circ$, producing a subset of G. Note that variables may be replaced by G's identity element. Note that this mechanism allows to have group-pattern languages which are not pattern languages. For example, consider the language generated by the pattern $bxbx^{-1}$ over the group with two generators a, b and satisfying $a \circ b = b^{-1} \circ a$ and $a \circ a = \varepsilon$. It satisfies $L(bxbx^{-1}) = \{\varepsilon, b^2\}$, which cannot be a pattern language as defined by Angluin as any such language is either a singleton or infinite. Furthermore, the language generated by the pattern $x_0^4 x_1^4$ over the group considered in Proposition 20 cannot even be generated as a finite union of pattern languages as defined by Angluin. Geilke and Zilles [12] considered relational pattern languages in which the symbols that can be substituted for variables can be required to respect additional restrictions given via relations; for example, it might be required that the symbol a substituted for variable x and the symbol b substituted for variable y satisfy $R(a, b)$ for some binary relation R. Note that since the present article studies group pattern languages only over automatic groups, these group pattern languages are regular. Thus, they can be generated by the trivial pattern x together with the regular language itself as the unary relation. In contrast, note that there exist relational pattern languages using regular relations that are not regular [12].

This article considers when pattern languages over groups and their bounded unions are **Ex**-learnable, where the focus is on automatic groups. Informally, an automatic group or more generally an automatic structure can be defined as follows — see Section 2 for formal details. Consider a structure $(A, R_1, R_2, \ldots)$, where $A \subseteq \Sigma^*$ and $R_1, R_2, \ldots$ are relations over Σ^* (an n-ary function can be considered as a relation over $n+1$ arguments — n inputs and one output). The structure is said to be automatic if the set A is regular and each of the relations is automatic (that is, regular), where multiple arguments are given to the automata in parallel with shorter inputs being padded by some special symbol to make the lengths of all inputs the same. An automatic group is an automatic structure (A, R), where A is a representation of G (that is, there exists a bijective mapping *rep* from G to A) and $R = \{(rep(x), rep(y), rep(z)) \colon x, y, z \in G$ and $x \circ y = z\}$. The definition of automatic groups in this article follows the original approach by Hodgson [15, 16] and later by Khoussainov and Nerode [25] and Blumensath and Grädel [5, 6]; they have also been studied by Nies, Oliver, Thomas and Tsankov [29–31, 39].

In some cases, automatic groups allow representing the class of all pattern languages over the group or some natural subclass of it as an automatic family, $(L_e)_{e \in E}$, which is given by an automatic relation

$$\{(e, x) \colon e \in E \wedge x \in L_e\}$$

for some regular set E which is called the *index set* of the family. Automatic families sometimes allow implementing learners which are themselves automatic, meaning that the graph of the function describing the learner, considered as a relation, is automatic. Though many complexity bounds in learning theory are not restrictive [8, 9, 21, 34], requiring the learners to be automatic is a restriction [19]. The use of automatic structures and families has the further advantage that the first-order theory of these structures is decidable and that first-order definable functions and relations are automatic [15, 25]; see the surveys of Khoussainov and Minnes [24] and Rubin [36] for more information.

Theorem 8 below strengthens Angluin's characterisation result [1] on the learnability of indexed families of languages by showing that for the class of pattern languages over an automatic group to be learnable, it is already sufficient that they satisfy Angluin's tell-tale condition noneffectively – see Section 3 for the definition of the tell-tale condition. It follows from Angluin's work that this noneffective version of the tell-tale condition is necessary. Note that for general indexed families, this noneffective version of the tell-tale condition is not sufficient and gives rise only to a behaviourally correct learner [3]. Note that behaviourally correct learning [4, 10, 33] is a strict generalisation of explanatory learning.

Section 4 explores the learnability of the class of pattern languages when the number of occurrences of variables in the pattern is bounded by some constant. Let $\mathbf{Pat}_n(G)$ denote the class of pattern languages over group G where the number of occurrences of the variables in the pattern is bounded by n. Then, Theorem 9 shows that $\mathbf{Pat}_1(G)$ is $\mathbf{Ex}$-learnable for all automatic groups G, though $\mathbf{Pat}_2(G)$ is not $\mathbf{Ex}$-learnable for some automatic group G. This group G has infinitely many generators. Theorem 12 shows that $\mathbf{Pat}_n(G)$ is $\mathbf{Ex}$-learnable for all finitely generated automatic groups G – in fact, for any fixed m, even the class of unions of up to m such pattern languages is $\mathbf{Ex}$-learnable.

Sections 5 and 6 consider learnability of the class of all pattern languages. Theorem 18 shows that for some automatic group G generated by two elements, $\mathbf{Pat}(G)$, the class of all pattern languages over G, is not **Ex**-learnable. On the other hand, Theorem 24 shows that for finitely generated Abelian groups G, $\mathbf{Pat}(G)$ as well as the class of unions of up to m pattern languages over G is **Ex**-learnable, for any fixed m. Theorem 16 shows that for the class of pattern languages over finitely generated Abelian groups the learners can even be made automatic using a suitable representation of the group and hypothesis space, though this hypothesis space cannot in general be an automatic family. However, the class of unions of up to two pattern languages over the integers with group operation $+$ is not automatically **Ex**-learnable, see Theorem 22.

2 Preliminaries

The symbol $\mathbb{N}$ denotes the set of natural numbers $\{0, 1, 2, \ldots\}$ and the symbol $\mathbb{Z}$ denotes the set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$. For any alphabet Σ, Σ^* denotes the set of finite strings over Σ; in this article all strings are assumed to be finite. For the purpose of the study of algebraic groups, let Σ^{-1} denote the set of inverses of the elements of Σ. Furthermore, let $\Sigma^{@}$ be the set $(\Sigma \cup \Sigma^{-1})^*$, that is, the set of all strings composed of elements of Σ and their inverses. The notation @ can be carried over to regular expressions similarly. The length of a string w is denoted by $|w|$ and $w(i)$ denotes the $(i+1)$-th symbol in the string, that is $w = w(0)w(1)w(2)\ldots w(|w|-1)$, where each $w(i)$ is in Σ (or $\Sigma \cup \Sigma^{-1}$, depending on the context).

The convolution of two strings u and v is defined as follows: Let $m = \max(\{|u|, |v|\})$ and let $\Diamond \notin \Sigma$ be a special symbol used for padding words. If $i < |u|$ then let $u'(i) = u(i)$ else let $u'(i) = \Diamond$; similarly, if $i < |v|$ then let $v'(i) = v(i)$ else let $v'(i) = \Diamond$. Now, $conv(u, v) = w$ is the string of length m such that, for $i < m$, $w(i) = (u'(i), v'(i))$. The convolution over n-tuples of strings is defined similarly. For example, $conv(abb, aa, aaab) = (a, a, a)(b, a, a)(b, \Diamond, a)(\Diamond, \Diamond, b)$. In their convoluted form, multiple inputs can be supplied to a standard finite automaton in a synchronous fashion, which enables for example the study of the regularity of relations with two or more inputs.

A function f is *automatic* if $\{conv(x, f(x)) \colon x \in \mathrm{dom}(f)\}$ is regular. An n-ary relation R is *automatic* if $\{conv(x_1, x_2, \ldots, x_n) \colon (x_1, x_2, \ldots, x_n) \in R\}$ is regular. A structure $(A, R_1, R_2, \ldots, f_1, f_2, \ldots)$ is said to be automatic if A is regular and, for all i, f_i is an automatic function from A^{k_i} to A for some k_i and R_i is an automatic relation over A^{h_i} for some h_i. A class $\mathcal{L}$ of languages over alphabet Σ is said to be an *automatic family* if there exists a regular *index set* I and there exist languages L_α, $\alpha \in I$, such that $\mathcal{L} = \{L_\alpha \colon \alpha \in I\}$ and the set $\{conv(\alpha, x) \colon \alpha \in I, x \in \Sigma^*, x \in L_\alpha\}$ is regular. The following are examples of automatic families:

- The class of sets with up to k elements for a constant k;
- The class of all finite and cofinite subsets of $\{0\}^*$;
- The class of all intervals of an automatic linear order on a regular set;
- Given an automatic presentation of $(\mathbb{Z}, +, <)$ and a first-order formula $\Phi(x, a_1, \ldots, a_k)$ with parameters $a_1, \ldots, a_k \in \mathbb{Z}$, the class consisting of all sets $\{x \in \mathbb{Z} \colon \Phi(x, a_1, \ldots, a_k)\}$ with $a_1, \ldots, a_k \in \mathbb{Z}$.

For $x, y \in \Sigma^*$ for a finite alphabet Σ, let $x <_{ll} y$ iff $|x| < |y|$ or $|x| = |y|$ and x is lexicographically before y, where some fixed ordering among symbols in Σ is assumed. Let $\leq_{ll}$, $>_{ll}$ and $\geq_{ll}$ be defined analogously. The relation $<_{ll}$ is called the *length-lexicographical order* on Σ.

The following fact will be useful for showing that relations and functions are automatic.

Fact 1 (Blumensath and Grädel [6], Hodgson [15, 16], Khoussainov and Nerode [25]). *Any relation or function that is first-order definable from existing automatic relations and functions is automatic.*

Definition 2. A group is a set of elements G along with an operation $\circ$ such that the following conditions hold:

Closure:	for all $a, b \in G$, $a \circ b \in G$;
Associativity:	for all $a, b, c \in G$, $(a \circ b) \circ c = a \circ (b \circ c)$;
Identity:	there is an $\varepsilon \in G$ such that for all $a \in G$, $a \circ \varepsilon = \varepsilon \circ a = a$;
Inverse:	for all $a \in G$ there exists an $a^{-1} \in G$ such that $a \circ a^{-1} = a^{-1} \circ a = \varepsilon$.

Often when referring to the group G, the group operation $\circ$ is implicit. A group $(G, \circ)$ is said to be *Abelian* if for all $a, b \in G$, $a \circ b = b \circ a$. A *subgroup* of $(G, \circ)$ is a subset H of G such that $(H, \circ)$ is a group. A *normal subgroup* of $(G, \circ)$ is a subgroup $(H, \circ)$ of $(G, \circ)$ such that for all $g \in G$, $\{g \circ h \colon h \in H\} = \{h \circ g \colon h \in H\}$. The index of a subgroup $(H, \circ)$ in $(G, \circ)$ is $\mathrm{card}(\{\{g \circ h \colon h \in H\} \colon g \in G\})$.

Example 3. *Examples for groups are the integers with addition, the rationals with addition or finite groups such as the group of all sequences of moves (modulo equivalence) on Rubik's cube.*

The Prüfer groups [14] are frequently used examples. The Prüfer group of base 2 *is given by the set of rationals* $\{\frac{m}{2^n} \colon m, n \in \mathbb{N}$ *and* $0 \leq m < 2^n\}$ *together with the group operation* $\circ$ *defined as follows: if* $x + y < 1$*, then* $x \circ y = x + y$ *else* $x \circ y = x + y - 1$*. Analogously, one can define Prüfer groups for every base* $p \geq 2$*. Then the Prüfer groups of base* p *and base* q *are isomorphic iff* p *and* q *contain the same prime factors, ignoring differences in multiplicity of those factors. Typically only Prüfer groups of prime base are studied.*

Another example of a group is that of shifts and negated shifts on the integers, that is, of all mappings of the form $x \mapsto x + c$ *and* $x \mapsto -x + c$ *where* $c \in \mathbb{Z}$*; the group operation is the concatenation of such mappings. This group will be defined abstractly and used in the proof of Theorem 18.*

When considering groups, a string over G is identified with the element of G obtained by replacing concatenation with $\circ$: thus for $a, b, c \in G$, $ab^{-1}c$ represents the group element $a \circ b^{-1} \circ c$.

$\Sigma \subseteq G$ is a set of generators for a group $(G, \circ)$ if all elements of G can be written as a finite string over elements of Σ and their inverses, where the concatenation operation is replaced by $\circ$. Note that, in general, the set of generators may be finite or infinite. An Abelian group $(G, \circ)$ is said to be a *free Abelian group* generated by a finite set $\{a_1, a_2, \ldots, a_n\}$ of generators iff

$$G = \{a_1^{m_1} \circ a_2^{m_2} \circ \ldots \circ a_n^{m_n} \colon m_1, m_2, \ldots, m_n \in \mathbb{Z}\}$$

and for each element in G the choice of $m_1, m_2, \ldots, m_n$ is unique.

Usually a group $(G, \circ)$ is represented using a set of representatives over a finite alphabet Σ via a one-one function *rep* from G to Σ^*. Then, $rep(\alpha)$ is said to be the representative of $\alpha \in G$. For $L \subseteq G$, $rep(L) = \{rep(a) \colon a \in L\}$. Often a group element is identified with its representative, and thus $L \subseteq G$ is also identified with $rep(L)$.

A group $(G, \circ)$ is said to be automatic if there exists a one-one function *rep* from G to Σ^*, where $rep(\alpha)$ is the representative of $\alpha \in G$, such that the following conditions hold:

- $A = \{rep(\alpha) \colon \alpha \in G\}$ is a regular subset of Σ^*;
- The function $f(rep(\alpha), rep(\beta)) = rep(\alpha \circ \beta)$ is automatic.

In this case (A, f) is called an automatic presentation of the group $(G, \circ)$; in the following, for ease of notation, the groups are identified with their automatic presentation. Note that the second clause above implies that the function $\alpha \mapsto \alpha^{-1}$, producing inverses, is also automatic, as it can be defined using a first-order formula over automatic functions. Without loss of generality it is assumed that the empty word over Σ is the representative of G's identity element; accordingly both will be denoted by ε.

An example of an automatic group is $(\mathbb{Z}, +)$, where $+$ denotes addition. The representation used for this automatic group is the reverse binary representation of numbers where the leftmost bit is the least significant bit (the sign of the number can be represented using a special symbol). It is easy to see that the order $<$ is automatic in this representation as well, so $(\mathbb{Z}, +, <)$ is an automatic structure, too.

Angluin [2] introduced the concept of pattern languages to the subject of learning theory; the corresponding notions for pattern languages over a group $(G, \circ)$ are defined as follows. A *pattern* π is a string over $G \cup \{x_1, x_2, \ldots\} \cup \{x_1^{-1}, x_2^{-1}, \ldots\}$, where $X = \{x_1, x_2, \ldots\}$ is a set of variables. Sometimes the symbols x, y, z are also used for variables. The elements of G appearing in a pattern are called constants. A substitution is a mapping from X to G. Note that the substitution of variables by ε is allowed. Let $sub(\pi)$ denote the string formed from π by applying the substitution *sub*, that is, by replacing every variable x by $sub(x)$ and x^{-1} by $(sub(x))^{-1}$ in the pattern π. The language generated by π, denoted $L(\pi)$, is the set $L(\pi) = \{sub(\pi) \colon sub$ is a substitution$\}$.

As an example, if G is the group of the integers with addition, the pattern $\pi' = x + x + 1$ generates the language $L(\pi')$ of odd numbers; for example, 17 is obtained by substituting 8 for x. Similarly, the patterns $\pi'' = x + x + x + x + x + x + y + y + y + y + y + y + y + y + y$ and $\pi''' = (-x) + (-x) + z + x + x + z + z$ both generate all multiples of 3: π''' directly maps z to $3z$, and from π'' one can obtain $3z$ by letting $sub(x) = -z$ and $sub(y) = z$. Note that for better readability of the above examples, the patterns were written in an additive form, where the group operation $+$ is explicitly mentioned and inverses are denoted by negative numbers. It is common in group theory, however, to denote groups in a multiplicative form where the inverse of an arbitrary element x is denoted by x^{-1} and where the group operation symbol is omitted; these conventions apply in particular to non-Abelian groups. Since such groups appear in the present work, this notation will be used in many places. This also allows shortening expressions containing multiple consecutive repetitions of the same group element using exponents. With these conventions the above patterns become $\pi' = x^2 1$, $\pi'' = x^6 y^9$ and $\pi''' = x^{-2} z x^2 z^2$.

Two patterns π_1 and π_2 are said to be *equivalent* (with respect to the group G) iff $L(\pi_1) = L(\pi_2)$. In the above example, $L(\pi'') = L(\pi''')$. A *pattern language* (over a group G) is a language generated by some pattern π. As the example of π'' and π''' shows, a given pattern language is typically generated by many different patterns. If the pattern π does not contain any constants then $L(\pi)$ is called a *verbal* language. So π'' is verbal and $L(\pi''')$ is a verbal language while $L(\pi')$ is not. Over an Abelian group, a pattern language is verbal iff it contains the neutral element; however, if G is not Abelian and $a \in G$ is a fixed element, then $L(a \circ x \circ a^{-1} \circ x^{-1})$ need not be a verbal language.

Let $\mathbf{Pat}(G)$ denote the class of all pattern languages over the group $(G, \circ)$ and $\mathbf{Pat}^m(G)$ denote the class of all unions of up to m pattern languages over the group G. Let $\mathbf{Pat}_n(G)$ denote the class of pattern languages generated by patterns containing up to n occurrences of variables or inverted variables and correspondingly $\mathbf{Pat}_n^m(G)$ denote the class of all unions of up to m pattern languages generated by patterns containing up to n occurrences of variables or inverted variables.

Proposition 4. *The classes $\mathbf{Pat}_n(G)$ and $\mathbf{Pat}_n^m(G)$ are automatic families for all automatic groups $(G, \circ)$ and all $m, n \in \mathbb{N}$.*

Proof. Let a pattern $a_0 x_{s_1} a_1 x_{s_2} a_2 \ldots x_{s_n} a_n$ with $a_0, a_1, \ldots, a_n \in G$ and $s_1, \ldots, s_n \in \{-n, \ldots, n\}$ be given; here, for ease of notation, x_{-k} is written for the string x_k^{-1} and x_0 for the empty string ε. This pattern is then represented by the index $e = conv(a_0, s_1, a_1, \ldots, s_n, a_n)$. It is easy to see that the set E of such indices is regular, so one can use E as index set for $\mathbf{Pat}_n(G)$.

To define L_e, let $g \in L_e$ iff there are $g_1, \ldots, g_n \in G$ with $g = a_0 \circ g_{s_1} \circ a_1 \circ g_{s_2} \circ a_2 \circ \ldots \circ g_{s_n} \circ a_n$; here, for ease of notation, let g_{-k} denote the inverse element of g_k and let g_0 denote the identity element ε of G. It is easy to see that the languages L_e are first-order defined using automatic parameters from their indices e. Thus $\mathbf{Pat}_n(G)$ is an automatic family.

Similarly, $\mathbf{Pat}_n^m(G)$ is an automatic family by letting indices of the form $conv(m, e_1, \ldots, e_m)$, where $e_i \in E$ for $i = 1, \ldots, m$, represent the set $L_{e_1} \cup \ldots \cup L_{e_m} \in \mathbf{Pat}_n^m(G)$. $\square$

Note that Example 13 shows that for any automatic representation A of the group $(\mathbb{Z}, +)$, $\mathbf{Pat}(A)$ is not an automatic family. Thus the above result does not generalise to $\mathbf{Pat}^m(G)$.

Gold [13] introduced the model of learning in the limit which is described below. Fix a group $(G, \circ)$. A *text* T is a mapping from $\mathbb{N}$ to $rep(G) \cup \{\#\}$, but since the elements of G are identified with the elements of $rep(G)$ as described above, one can think of a text as a sequence of elements of $G \cup \{\#\}$. A finite sequence is an initial segment of a text. Let $|\sigma|$ denote the length of sequence σ. The content of a text T, denoted $\text{content}(T)$, is the set of elements of G in the range of T, that is, $\text{content}(T) = \{T(i) \colon T(i) \neq \#\}$. Similarly, the content of a finite sequence σ is $\{\sigma(i) \colon i < |\sigma| \text{ and } \sigma(i) \neq \#\}$ and denoted by $\text{content}(\sigma)$. Intuitively, #'s denote pauses in the presentation of data. T is a *text for* $L \subseteq G$ iff $\text{content}(T) = L$. Let $T[n]$ denote the initial segment of T of length n.

Intuitively, a learner reads from the input, one element at a time, some text T for a target language L. Based on this new element, the learner updates its memory and hypothesis. The learner has some initial memory and hypothesis before it has received any data. Note that a text denotes only positive data being presented to the learner; the learner is never given information about what is *not* in the target language L. The learner uses some hypothesis space $\mathcal{H} = \{H_\alpha : \alpha \in J\}$ from which it draws its hypotheses. It is assumed for this article that the hypothesis space is uniformly recursive, that is, $\{(\alpha, x) : \alpha \in J, x \in H_\alpha\}$ is recursive.

More formally, a learner is defined as follows. Parts (c) to (e) of the definition give a basic learning criterion called explanatory learning.

Definition 5 (Based on Gold [13]). Fix a group $(G, \circ)$. Suppose I and J are regular sets of strings over some finite alphabet and let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ and $\mathcal{H} = \{H_\beta : \beta \in J\}$ with $L_\alpha, H_\beta \subseteq G$ for all $\alpha \in I$ and $\beta \in J$. Let Δ be a finite alphabet and let ? be a special symbol not in $J \cup \Delta^*$ that is used as null hypothesis as well as for null memory.

(a) A *learner* $\mathbf{M}$ *using hypothesis space* $\mathcal{H}$ is a recursive function mapping the set $(\Delta^* \cup \{?\}) \times (G \cup \{\#\})$ to the set $(\Delta^* \cup \{?\}) \times (J \cup \{?\})$ together with an *initial memory state* $mem_0 \in \Delta^* \cup \{?\}$ and an *initial hypothesis* $hyp_0 \in J \cup \{?\}$.

(b) Suppose a text T for a language $L \subseteq G$ is given.
   - Let $mem_0^T = mem_0$ and $hyp_0^T = hyp_0$.
   - Let $(mem_{n+1}^T, hyp_{n+1}^T) = \mathbf{M}(mem_n^T, T(n))$.

   Intuitively, mem_{n+1}^T and hyp_{n+1}^T denote the memory and hypothesis of the learner $\mathbf{M}$ after receiving input $T[n+1]$.

   M *converges* on T to a hypothesis hyp iff, for all but finitely many n, $hyp_n^T = hyp$.

(c) $\mathbf{M}$ $\mathbf{Ex}$-*learns* L *with respect to hypothesis space* $\mathcal{H}$ iff for all texts T for L, $\mathbf{M}$ converges on T to a hypothesis hyp such that $H_{hyp} = L$. Note that $\lim_{n\to\infty} mem_n^T$ need not exist.

(d) $\mathbf{M}$ $\mathbf{Ex}$-*learns* $\mathcal{L}$ *with respect to hypothesis space* $\mathcal{H}$ iff $\mathbf{M}$ $\mathbf{Ex}$-learns each $L \in \mathcal{L}$ with respect to hypothesis space $\mathcal{H}$. In this case $\mathbf{M}$ is said to be an $\mathbf{Ex}$-*learner for* $\mathcal{L}$.

(e) $\mathcal{L}$ is $\mathbf{Ex}$-*learnable* iff there exists a recursive learner $\mathbf{M}$ and a hypothesis space $\mathcal{H}$ such that $\mathbf{M}$ $\mathbf{Ex}$-learns $\mathcal{L}$ with respect to hypothesis space $\mathcal{H}$.

The letters "**Ex**" in the above definition stand for "explanatory learning"; this notion was first introduced by Gold [13]. Often, reference to the hypothesis space $\mathcal{H}$ is dropped and is implicit. Furthermore, in some cases when learning automatic families, some automatic family is used as the hypothesis space.

A learner $\mathbf{M}$ *makes a mind change* [10, 11] at $T[n+1]$, if $? \neq hyp_n^T \neq hyp_{n+1}^T$ as defined in the above definition. A learner makes at most m mind changes on a text T iff $|\{n : ? \neq hyp_n^T \neq hyp_{n+1}^T\}| \leq m$.

A learner is said to be *automatic* if its updating function is automatic, that is, if the relation

$$\{conv(mem, x, mem', hyp) : \mathbf{M}(mem, x) = (mem', hyp)\}$$

is automatic.

For ease of presentation of proofs, sometimes only informal descriptions are given of how a learner updates its memory and hypothesis when a new datum is presented to it. Similarly, sometimes it is said only informally that a learner conjectures a language, rather than its index; the index conjectured is implicit in such a case. The following lemma by Gold is useful to show some of the results below.

Lemma 6 (Gold [13]). *Suppose $L_0 \subset L_1 \subset \ldots$ and $L = \bigcup_{i \in \mathbb{N}} L_i$. Then the class $\{L\} \cup \{L_0, L_1, \ldots\}$ is not* **Ex***-learnable.*

3 A characterisation

In this section, for automatic groups G, **Ex**-learnability of $\mathbf{Pat}^m(G)$ and $\mathbf{Pat}^m_n(G)$ is characterised in terms of the non-effective version of Angluin's tell-tale condition.

Recall that a *tell-tale set* [1] for a language L with respect to $\mathcal{L}$, is a finite subset D of L such that, for every $L' \in \mathcal{L}$ it holds that $D \subseteq L' \subseteq L$ implies $L' = L$. When $\mathcal{L}$ is clear from context, the phrase "with respect to $\mathcal{L}$" is often dropped. A class $\mathcal{L}$ of languages satisfies Angluin's tell-tale condition noneffectively iff every language $L \in \mathcal{L}$ has a tell-tale set with respect to $\mathcal{L}$. A family $\mathcal{L} = (L_i)_{i \in I}$ of languages satisfies Angluin's tell-tale condition effectively iff for each $i \in I$, a tell-tale set D for L_i with respect to $\mathcal{L}$ can be enumerated effectively in i. Furthermore, $\mathcal{L}$ is called an indexed family iff there exists an indexing $\mathcal{L} = \{L_i \colon i \in I\}$, where I is a recursive set, such that $\{conv(i, x) \colon i \in I, x \in L_i\}$ is recursive. Note that every automatic family is an indexed family. It is well known that for any indexed family and any two indexings, $\{L_i \colon i \in I\}$ and $\{L_j \colon j \in J\}$ of $\mathcal{L}$ (where I and J are recursive sets), $(L_i)_{i \in I}$ satisfies Angluin's tell-tale condition effectively iff $(L_j)_{j \in I}$ satisfies Angluin's tell-tale condition effectively. Thus, it is well-defined to say that an indexed family $\mathcal{L}$ satisfies Angluin's tell-tale condition effectively iff $(L_i)_{i \in I}$ does so, for some indexing $\mathcal{L} = \{L_i \colon i \in I\}$.

Proposition 7 (Angluin [1]). *An indexed family $\mathcal{L}$ is* **Ex***-learnable iff it satisfies Angluin's tell-tale condition effectively.*

Baliga, Case and Jain [3] gave a similar characterisation for behaviourally correct learning (using an acceptable numbering as hypothesis space) for indexed families satisfying Angluin's tell-tale condition noneffectively. Behaviourally correct learning [4, 10, 33] is a weaker learning notion where a successful learner for L on text T for L is required to eventually only output grammars that produce L; the learner is not required to converge to a single such grammar.

Now Proposition 7 allows to characterise learnability of $\mathbf{Pat}^m(G)$, for any automatic group G.

Theorem 8. *Given an automatic group G, if $\mathbf{Pat}^m(G)$ satisfies Angluin's tell-tale condition noneffectively, then $\mathbf{Pat}^m(G)$ is* **Ex***-learnable.*

Proof. Assume that $\mathbf{Pat}^m(G)$ satisfies Angluin's tell-tale condition noneffectively. Now it is shown that $\mathbf{Pat}^m(G)$ is **Ex**-learnable. Let $S_0, S_1, \ldots$ be a recursive enumeration of all sets of at most m patterns as canonical indices. Let $L_i = \bigcup_{\pi \in S_i} L(\pi)$. As the group G is automatic, for each pattern π, one can effectively obtain a finite automaton accepting $L(\pi)$. Thus, it is decidable, given i and j, whether $L_i \subseteq L_j$. Furthermore, the membership problem for L_i is uniformly decidable in i. Without loss of generality assume that $L_0 = G$.

Now consider the following learner on input text T. The learner uses hypothesis space $\{L_e \colon e \in \mathbb{N}\}$. After having seen input $T[n]$, it stores content$(T[n])$ in its memory. Let

(a) $A(n) = \{j \colon j \leq n$ and content$(T[n]) \subseteq L_j\}$ and
(b) $B(n) = \{j \in A(n) \colon \forall j' < j\,[\text{if } j' \in A(n) \text{ then } L_j \subset L_{j'}]\}$.

Then, after having seen $T[n]$, the learner conjectures $L_{\max(B(n))}$. Intuitively, $A(n)$ above denotes the set of all indices $\leq n$ of languages which contain the input sample $T[n]$. $B(n)$ denotes the set of all indices in $A(n)$ of languages which are strictly included in all languages with smaller indices in $A(n)$.

Now it is shown that this learner **Ex**-learns $\mathbf{Pat}^m(G)$. Suppose the input text T is for L_e, where e is minimal with this property. Let $s > e$ be large enough such that

(c) $T[s]$ is a tell-tale set for L_e with respect to $\mathbf{Pat}^m(G)$ and
(d) for all $e' < e$, if $L_e \not\subseteq L_{e'}$, then content$(T[s]) \not\subseteq L_{e'}$.

Then, for all $s' > s$,

(e) $B(s)$ contains e as by (d) any $e' < e$ such that $L_e \not\subseteq L_{e'}$ is not an element of $A(s)$;
(f) $B(s)$ does not contain any $e'' > e$ as for any $e'' > e$, either content$(T[s]) \not\subseteq L_{e''}$ or $L_{e''}$ is not a proper subset of L_e by (c).

Thus, the learner conjectures L_e on input $T[s']$ for all $s' > s$ and it follows that it **Ex**-learns $\mathbf{Pat}^m(G)$. ☐

The above result also holds when considering $\mathbf{Pat}_n^m(G)$ instead of $\mathbf{Pat}^m(G)$. Note that the above result implies that the **Ex**-learnability of $\mathbf{Pat}^m(G)$ or $\mathbf{Pat}_n^m(G)$ does not depend on the automatic representation chosen for the group; however, the learnability by an automatic learner might still depend on it.

4 Learning patterns with up to n variable occurrences

In this section it is shown that the class $\mathbf{Pat}_n^m(G)$ is **Ex**-learnable for all finitely generated automatic groups G. For other automatic groups, while $\mathbf{Pat}_1(G)$ is always **Ex**-learnable, in general $\mathbf{Pat}_2(G)$ is not.

Theorem 9. *For every automatic group $(G, \circ)$, $\mathbf{Pat}_1(G)$ can be* **Ex***-learnt by an automatic learner which makes at most one mind change. There is, however, an automatic group $(G, \circ)$ such that $\mathbf{Pat}_2(G)$ cannot be* **Ex***-learnt by any learner.*

Proof. The class $\mathbf{Pat}_0(G)$ contains only languages generated by patterns without variables. Thus, $\mathbf{Pat}_0(G) = \{\{\alpha\} \colon \alpha \in G\}$. $\mathbf{Pat}_1(G)$ contains, in addition to the languages in $\mathbf{Pat}_0(G)$, all languages generated by patterns of the form $\alpha x \beta$, where $\alpha, \beta \in G$; thus these languages are of the form $\alpha \circ G \circ \beta$. Each of these languages is equal to G, as any group element γ can be generated by the pattern $\alpha x \beta$ by choosing the substitution $sub(x) = \alpha^{-1} \circ \gamma \circ \beta^{-1}$. A learning algorithm for $\mathbf{Pat}_1(G)$ waits for the first datum α and then conjectures $\{\alpha\}$ until a second and different datum β arrives in the input. At that point, the learner makes a mind change to the pattern x denoting the language $L(x) = G$; thereafter, the learner never changes its mind again.

For the nonlearnability result, a specific group G is constructed as follows. The group is generated by an infinite set of generators $a_0, b_0, a_1, b_1, a_2, b_2, \ldots$ and the following rules governing the generators:

- $a_k^{-1} = a_k$, $b_k^{-1} = b_k^2$;
- $b_k \circ a_k = a_k \circ b_k^{-1}$;
- all a_i commute among each other;
- all b_j commute among each other;
- if $i \neq j$ then a_i and b_j commute (which implies that a_i^{-1} and b_j^{-1} commute as well).

Intuitively, for each fixed k, one can think of a_k, b_k as generating the symmetric group S_3. Geometrically this can be interpreted as operations on the equilateral triangle with vertices ε, b_k, and b_k^{-1}. Then b_k would be a 120° rotation around the centre of the triangle and a_k would be a flip operation along the line that runs through the top vertex and the point at the centre of the opposite side of the triangle.

Thus, each group element formed using composition of finitely many of the above generators can be equivalently formed by composing a finite sequence of components, where the k-th component is formed by only using composition among the generators $a_k, b_k, a_k^{-1}, b_k^{-1}$ and ε. Using the rules listed above, this gives that a group element can be considered as a finite sequence of components, where the k-th component is one of $\varepsilon, a_k, b_k, a_k \circ b_k, b_k^{-1}, a_k \circ b_k^{-1}$. Therefore, each group element can be represented by a finite string w over an alphabet with six symbols; for ease of notation, these six symbols are denoted by $\varepsilon, a_k, b_k, a_k \circ b_k, b_k^{-1}, a_k \circ b_k^{-1}$ when they appear in the k-th component. In case that $k > |w|$, the k-th component is ε.

Due to the commutative properties stated above, to represent this group automatically it is enough to implement the group operation $\circ$ component-wise. For this one may need to pad strings using ε in some places. If the output string of the automatic operation has trailing ε's, these are removed. It is well-known that a so constructed group is automatic.

For $A \subseteq \mathbb{N}$, consider the set $H(A) \subseteq G$ and group element $w_A \in G$ defined as follows:

- $H(A) = \{w \in \Sigma^* : \forall k\, [\text{if } k < |w| \text{ and } k \in A \text{ then } w(k) \in \{\varepsilon, b_k, b_k^{-1}\} \text{ else } w(k) = \varepsilon]\}$;
- For finite A, let w_A be such that if $k \in A$ then $w_A(k) = a_k$ else $w_A(k) = \varepsilon$.

Now consider the following patterns and the corresponding languages:

- The pattern xx satisfies $L(xx) = H(\mathbb{N})$. To see this first note that, for $\ell \in \{0, 1, -1\}$, $(a_k \circ b_k^\ell)^2 = \varepsilon$ and $(b_k^\ell)^2 = b_k^{-\ell}$. Thus, for every substitution $w = sub(x)$, $(w^2)(k) = (w(k))^2$ is one of $\varepsilon, b_k, b_k^{-1}$. Hence, $L(xx) \subseteq H(\mathbb{N})$. Furthermore, each $w \in H(\mathbb{N})$ belongs to $L(xx)$ by using the substitution $sub(x) = w'$, where $w'(k) = w(k)^{-1}$. Thus, $H(\mathbb{N}) \subseteq L(xx)$. It follows that $L(xx) = H(\mathbb{N})$.

– For any finite set A, the pattern $w_A x w_A x^{-1}$ satisfies $L(w_A x w_A x^{-1}) = H(A)$. To see this note that, for $h \in \{0,1\}$ and $\ell \in \{0,1,-1\}$, $a_k \circ a_k^h \circ b_k^\ell \circ a_k \circ b_k^{-\ell} \circ a_k^{-h} \in \{b_k^\ell, b_k^{-\ell}\}$ and $a_k^h \circ b_k^\ell \circ b_k^{-\ell} \circ a_k^{-h} = \varepsilon$. Thus, $L(w_A x w_A x^{-1}) \subseteq H(A)$. Furthermore, $H(A) \subseteq L(w_A x w_A x^{-1})$, as any $w \in H(A)$ can be generated by the pattern $w_A x w_A x^{-1}$ using the substitution $sub(x) = w$ (note that for $k \notin A$, $w(k) = \varepsilon$).

As $H(\{0\}) \subset H(\{0,1\}) \subset H(\{0,1,2\}) \subset \ldots$ and $\bigcup_{n \in \mathbb{N}} H(\{0,1,\ldots,n\}) = H(\mathbb{N})$, the class $\mathbf{Pat}_2(G)$ is not $\mathbf{Ex}$-learnable by Lemma 6. □

In the following, it will be shown that for finitely generated automatic groups, for all $n \in \mathbb{N}$, the class $\mathbf{Pat}_n(G)$ has an automatic learner. For this result, it is necessary to recall some facts from group theory. Intuitively, for automatic finitely generated groups $(G, \circ)$, the elements of G can be seen as the products of an element from a finite subset H of G and some powers of finitely many generators of a normal Abelian subgroup H' of G. This allows to define classes $\mathcal{Q}_n(\mathcal{R}, H')$ of subsets of G, for some finite class $\mathcal{R}$ of regular subsets of G with some special properties. These $\mathcal{Q}_n(\mathcal{R}, H')$ are learnable, and for one such $\mathcal{R}$ and some n', $\mathcal{Q}_{n'}(\mathcal{R}, H')$ includes $\mathbf{Pat}_n(G)$.

Oliver and Thomas [31] showed that a finitely generated group is automatic iff it is Abelian by finite, that is, it has an Abelian subgroup of finite index. They furthermore noted [31, Remark 3] that if any group has an Abelian subgroup of finite index, then it also has an Abelian normal subgroup of finite index. Also note that if a finitely generated group has a normal subgroup of finite index, then this normal subgroup is finitely generated.

Thus, given an automatic finitely generated group $(G, \circ)$, it can be assumed without loss of generality that there is a finite subset H of G and generators $b_1, \ldots, b_m$ of a normal Abelian subgroup H' of G, such that every group element of G is of the form $a \circ b_1^{\ell_1} \circ b_2^{\ell_2} \circ \ldots \circ b_m^{\ell_m}$ where $a \in H$ and $\ell_1, \ell_2, \ldots, \ell_m \in \mathbb{Z}$. Furthermore, for each $a \in H$ and generator b_i, there are $j_{a,i,1}, \ldots, j_{a,i,m}$ with

$$b_i \circ a = a \circ b_1^{j_{a,i,1}} \circ b_2^{j_{a,i,2}} \circ \ldots \circ b_m^{j_{a,i,m}}.$$

Therefore it can be assumed without loss of generality that the group is represented as a convolution of $a \in H$ and $\ell_1, \ldots, \ell_m \in \mathbb{Z}$ in some automatic presentation of $(\mathbb{Z}, +, <)$. Then this allows to automatically carry out various group operations such as $\circ$, testing membership in H' and finding, for any element c of H' and any fixed group elements $d_1, \ldots, d_h$ that generate H', a tuple $(k_1, \ldots, k_h) \in \mathbb{Z}^h$ such that $c = d_1^{k_1} \circ \ldots \circ d_h^{k_h}$.

For G, H, H' as above, let S be a finite set of generators of H'; note that for each $S' \subseteq S$, the set generated by the elements of S' is a regular subset of H'. Furthermore, let $\mathcal{R}$ be a finite set of regular subsets of G with the property that for every $U \in \mathcal{R}$ and $\beta, \beta' \in H'$, either $U \circ \beta = U \circ \beta'$ or $U \circ \beta \cap U \circ \beta' = \emptyset$. Note that there exist such $\mathcal{R}$, as one can take $\mathcal{R}$ to be $\{\emptyset, G\}$. Another such $\mathcal{R}$ is constructed in Proposition 11, for which $\mathcal{Q}_{n'}(\mathcal{R}, H')$ defined below includes $\mathbf{Pat}_n(G)$, for some appropriate n'.

Now the following family is automatic for each constant n:

$$\mathcal{Q}_n(\mathcal{R}, H') = \left\{ (U_1 \circ \beta_1) \cup \ldots \cup (U_h \circ \beta_h) \colon \begin{array}{c} h \leq n \text{ and } U_1, \ldots, U_h \in \mathcal{R} \\ \text{and } \beta_1, \ldots, \beta_h \in H' \end{array} \right\}.$$

Here, an element of $\mathcal{Q}_n(\mathcal{R}, H')$ can be represented as follows: As $\mathcal{R}$ is finite, its elements can be represented using finitely many symbols. Each pair (U, β) can then be represented using $conv(u, b)$, where u is a single symbol representing the element U of $\mathcal{R}$ and b is the usual representation of β as group element. Each element of $\mathcal{Q}_n(\mathcal{R}, H')$ can now be represented using a convolution of the representations of up to n pairs (U, β).

Proposition 10. *For every n, the class $\mathcal{Q}_n(\mathcal{R}, H')$ has an automatic* **Ex***-learner using the hypothesis space $\mathcal{Q}_n(\mathcal{R}, H')$.*

Proof. The learner maintains in its memory a set of candidate hypotheses, where each candidate hypothesis is of the form

$$(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h),$$

with $h \leq n$, each $U_i \in \mathcal{R}$ and each $\beta_i \in H'$. It will be the case that the number of candidates in the set is bounded by some constant c. The set of candidates can be represented in memory as a convolution of the representations of the individual candidates.

The hypothesis of the learner at any stage is the minimal candidate (subset-wise) among all the candidates. In case of several minimal candidates the one with the length-lexicographically smallest representation is used. Note that choosing such a minimal candidate is an automatic operation.

Initially the learner has only one candidate which is the empty union. At any stage, when a new input w is processed, the following is done for each candidate in the current set: If a candidate

$$(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h)$$

does not contain w, then

(i) if $h < n$ then the candidate is replaced by a set of candidates of the form

$$(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h) \cup (U_{h+1} \circ \beta_{h+1}),$$

which satisfy that U_{h+1} is an element of $\mathcal{R}$ and that β_{h+1} exists and is the length-lexicographically least element of H' such that $w \in U_{h+1} \circ \beta_{h+1}$;

(ii) if $h = n$ or no such U_{h+1}, β_{h+1} exist as in (a) above, then $(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h)$ is simply dropped from the candidate set (note that, for $h < n$ and for an input text from the class $\mathcal{Q}_n(\mathcal{R}, H')$, there always exist such a U_{h+1}, β_{h+1}).

Note that all the above operations are automatic. Furthermore, note that for each replaced candidate, at most $|\mathcal{R}|$ new candidates are added. As none of these unions has more than n terms, the overall number of candidates considered during the runtime of the algorithm is at most $1 + |\mathcal{R}| + \ldots + |\mathcal{R}|^n$; if c is chosen equal to this number then never more than c candidates need to be memorised.

Now, suppose T is a text for $L \in \mathcal{Q}_n(\mathcal{R}, H')$. For all candidates which are not supersets of L, the learner will eventually see a counterexample and remove the candidate from the candidate set. Thus only candidates containing all elements of L will survive in the limit in the candidate set.

Suppose $L = (U_1 \circ \beta_1) \cup \ldots \cup (U_h \circ \beta_h)$, where $h \leq n$ is minimal. Without loss of generality assume that, for each $i \in \{1, 2, \ldots, h\}$, β_i is the length-lexicographically least element β_i' of H' such that $U_i \circ \beta_i = U_i \circ \beta_i'$. Then eventually, $(U_1 \circ \beta_1) \cup \ldots \cup (U_h \circ \beta_h)$ is added to the candidate set by the learner, as can be shown by induction as follows: When the first datum different from # is observed by the learner, it adds $(U_{i_1} \circ \beta_{i_1})$ to the candidate set, for some $i_1 \in \{1, 2, \ldots, h\}$. Now, suppose the learner has placed $(U_{i_1} \circ \beta_{i_1}) \cup \ldots \cup (U_{i_k} \circ \beta_{i_k})$ in the candidate set with $k < h$ and $\{i_1, i_2, \ldots, i_k\} \subset \{1, 2, \ldots, h\}$. Then, eventually, it will add $(U_{i_1} \circ \beta_{i_1}) \cup \ldots \cup (U_{i_{k+1}} \circ \beta_{i_{k+1}})$ to the candidate set where $\{i_1, \ldots, i_k\} \subset \{i_1, \ldots, i_k, i_{k+1}\} \subseteq \{1, 2, \ldots, h\}$. This happens as soon as the learner is first presented some $w \in L - [(U_{i_1}, \beta_{i_1}) \cup \ldots \cup (U_{i_k}, \beta_{i_k})]$. Thus, by induction the learner will eventually put $(U_1 \circ \beta_1) \cup \ldots \cup (U_h \circ \beta_h)$, in the candidate set. As $(U_1, \beta_1) \cup \ldots \cup (U_h, \beta_h)$ is never dropped from the candidate set, eventually the learner only outputs the length-lexicographically least correct hypothesis among the hypotheses in its candidate set. Thus, it **Ex**-learns L. As L was arbitrary in the class $\mathcal{Q}_n(\mathcal{R}, H')$, it follows that the learner **Ex**-learns $\mathcal{Q}_n(\mathcal{R}, H')$. □

Wiehagen [40] called a learner *iterative* if its memory is identical to its most recent hypothesis; for the discussion of iterative learners here and later in the article, class-preserving hypothesis spaces [27] are considered. A hypothesis space $\mathcal{H} = \{H_\beta \colon \beta \in J\}$ is called class-preserving for learning $\mathcal{L} = \{L_\alpha \colon \alpha \in I\}$, iff $\mathcal{H}$ and $\mathcal{L}$ contain exactly the same languages (though there may be different numbers of copies of the same language in $\mathcal{L}$ and $\mathcal{H}$). The learner for Proposition 10 can be easily made iterative as follows. The new learner outputs hypothesis $\mathrm{pad}(h, m)$, whenever the learner from Proposition 10 outputs hypothesis h and has memory m. Here, $\mathrm{pad}(h, m)$ is a one-one automatic function, and in the new hypothesis space $\mathrm{pad}(h, m)$ represents the same language as h in the hypothesis space used by the learner in Proposition 10. Note that as an automatic function $\mathrm{pad}(h, m)$ is automatically invertible in the sense that both $\mathrm{pad}(h, m) \mapsto h$ and $\mathrm{pad}(h, m) \mapsto m$ are automatic. As both the hypothesis and the memory of the learner in Proposition 10 converge on texts for languages in $\mathcal{Q}_n(\mathcal{R}, H')$, the modified learner just described iteratively learns $\mathcal{Q}_n(\mathcal{R}, H')$.

For each $a \in H$ and generator b_i of H' there is a $\gamma_{a,i}$ in H' with $b_i \circ a = a \circ \gamma_{a,i}$. Note that, for all $\beta \in H'$, $b_i \circ (a \circ \beta) = (b_i \circ a) \circ \beta = (a \circ \gamma_{a,i}) \circ \beta = a \circ (\gamma_{a,i} \circ \beta) = a \circ (\beta \circ \gamma_{a,i}) = (a \circ \beta) \circ \gamma_{a,i}$. That allows defining $\gamma_{a \circ \beta, i} = \gamma_{a,i}$ for all $\beta \in H'$ and to extend the mapping $\alpha \mapsto \gamma_{\alpha,i}$ to all $\alpha \in G$. This will be used in the proof of the following result.

Proposition 11. *Every automatic finitely generated group $(G, \circ)$ has a normal Abelian subgroup H' of finite index such that, for each n, there is a set $\mathcal{R}$ of finitely many regular languages and an n' such that $\mathbf{Pat}_n(G) \subseteq \mathcal{Q}_{n'}(\mathcal{R}, H')$.*

Proof. Suppose $\{b_1, b_2, \ldots, b_m\}$ are the generators of the normal Abelian subgroup H' of G and H is a finite subset of G such that every element of G can be expressed as $a \circ b$ for some $a \in H$ and $b \in H'$. For $a \in H$ and $i \in \{1, \ldots, m\}$, suppose $\gamma_{a,i}$ is such that $b_i \circ a = a \circ \gamma_{a,i}$.

Before proceeding with the full proof, consider an example as follows. Consider the pattern $a_1 \circ b_1^5 \circ x \circ a_2 \circ x^{-1} \circ y^2$ and substitutions $sub(x) = a_3 \circ b_1^{h_1} \circ b_2^{h_2} \circ \ldots \circ b_m^{h_m}$ and $sub(y) = a_4 \circ b_1^{k_1} \circ b_2^{k_2} \circ \ldots \circ b_m^{k_m}$, where a_3 and a_4 are some fixed elements of H and $h_1, h_2, \ldots, h_m, k_1, k_2, \ldots, k_m$ range over $\mathbb{Z}$. Then the elements of G generated using the above substitutions are of the form

$$a_1 \circ a_3 \circ a_2 \circ a_3^{-1} \circ a_4^2 \circ \gamma^5_{a_3 \circ a_2 \circ a_3^{-1} \circ a_4^2,1} \circ (\gamma_{a_2 \circ a_3^{-1} \circ a_4^2,1} \circ \gamma^{-1}_{a_4^2,1})^{h_1} \circ (\gamma_{a_2 \circ a_3^{-1} \circ a_4^2,2} \circ \gamma^{-1}_{a_4^2,2})^{h_2} \circ \ldots \circ (\gamma_{a_2 \circ a_3^{-1} \circ a_4^2,m} \circ \gamma^{-1}_{a_4^2,m})^{h_m} \circ (\gamma_{a_4,1} \circ b_1)^{k_1} \circ (\gamma_{a_4,2} \circ b_2)^{k_2} \circ \ldots \circ (\gamma_{a_4,m} \circ b_m)^{k_m}.$$

Thus, for fixed a_3, a_4, the set of strings generated by the pattern can be brought into the form

$$\{a \circ z_1^{h_1} \circ \ldots \circ z_m^{h_m} \circ z_{m+1}^{k_1} \circ \ldots \circ z_{2m}^{k_m} \colon h_1, \ldots, h_m, k_1, \ldots, k_m \in \mathbb{Z}\} \circ \beta$$

for some $a \in H$ and $\beta, z_1, \ldots, z_{2m} \in H'$; the full pattern language is the union of up to $|H|^2$ many such sets, as the pattern generating the language has 2 distinct variables, namely x and y and thus there are $|H|^2$ many ways to choose (a_3, a_4).

In the general case, when there are up to n occurrences of variables, the pattern language is a union of up to $|H|^n$ languages of the form

$$\{a \circ \prod_{z \in Z} z^{h_z} \colon h_z \in \mathbb{Z} \text{ for } z \in Z\} \circ \beta$$

where $a \in H$, $\beta \in H'$ and Z is some, possibly empty, set of specific elements of H', where $|Z| \leq mn$ and each element of Z is a product of up to n elements of the form $\gamma_{a,i}$, $a \in H$. Note that $\gamma_{\varepsilon,i} = b_i$ and thus the case of the b_i being at the end of the pattern is covered. Let $\mathcal{R}$ be the collection of all the sets of the form

$$\{a \circ \prod_{z \in Z} z^{h_z} \colon h_z \in \mathbb{Z} \text{ for } z \in Z\}.$$

It follows that $\mathcal{R}$ is finite, as each Z is a subset of the set Z' of products of up to n elements from some set $\{\gamma_{a,i} \colon a \in H\}$ with $i \in \{1, 2, \ldots, m\}$; therefore Z' has at most

$$c' = m \cdot (1 + |H| + |H|^2 + \ldots + |H|^n)$$

elements and $\mathcal{R}$ has at most $|H| \cdot 2^{c'}$ many elements.

Assume that $U = \{a \circ \prod_{z \in Z} z^{h_z} \colon h_z \in \mathbb{Z} \text{ for } z \in Z\} \in \mathcal{R}$ and there are $\alpha \in G$ and $\beta', \beta'' \in H'$ with $\alpha \in U \circ \beta' \cap U \circ \beta''$. This means that for each $z \in Z$ there are h'_z, h''_z such that

$$\alpha = a \circ \prod_{z \in Z} z^{h'_z} \circ \beta' = a \circ \prod_{z \in Z} z^{h''_z} \circ \beta'' \text{ and thus } \beta'' \circ (\beta')^{-1} = \prod_{z \in Z} z^{h'_z - h''_z}$$

which implies that $U \circ \beta' = U \circ \beta''$. Thus the languages in $\mathcal{R}$ satisfy the requirements in the definition of $\mathcal{Q}_{n'}(\mathcal{R}, H')$, where $n' = |H|^n$, as each pattern language in $\mathbf{Pat}_n(G)$ is the union of up to $|H|^n$ many languages of the form $U \circ \beta$ with $U \in \mathcal{R}$ and $\beta \in H'$. $\square$

Note that if an automatic learner can learn an automatic family $\mathcal{L}$ using $\mathcal{L}$ as the hypothesis space and $\mathcal{L}' \subseteq \mathcal{L}$ is an automatic subfamily, then there is another automatic learner which learns $\mathcal{L}'$ using $\mathcal{L}'$ as the hypothesis space. This holds as there is an automatic function which translates any index in $\mathcal{L}$ for a language $L \in \mathcal{L}'$ into an index for L in $\mathcal{L}'$, see Jain, Ong, Pu and Stephan [21]; furthermore the domain of this function is regular, as a finite automaton can determine whether an index of an element of $\mathcal{L}$ is for a language in $\mathcal{L}'$. Thus Proposition 10 and Proposition 11 give the following theorem.

Theorem 12. *Let $(G, \circ)$ be a finitely generated automatic group. Then, for each n, there is an automatic* **Ex***-learner for the class $\mathbf{Pat}_n(G)$. Furthermore, for each $m, n \in \mathbb{N}$, there is an automatic* **Ex***-learner for the class $\mathbf{Pat}_n^m(G)$ using $\mathbf{Pat}_n^m(G)$ itself as the hypothesis space.*

As the learner in Proposition 10 can be made iterative, the learner for $\mathbf{Pat}_n^m(G)$ above can also be made iterative for a suitable class-preserving hypothesis space.

5 Automatic learning of all patterns

If $(G, \circ)$ is an Abelian automatic group, then, in some cases, the class of all pattern languages is learnable by an automatic learner. However, even in these cases, one sometimes needs to use a nonautomatic family as hypothesis space. The reason is that the class of pattern languages over $(G, \circ)$ is not guaranteed to form an automatic family.

Example 13. *$(\mathbb{Z}, +)$ does not have any presentation $(A, +)$ for which $\mathbf{Pat}(A)$ is an automatic family.*

Proof. Suppose that an automatic presentation $(A, +)$ of $(\mathbb{Z}, +)$ exists such that $\mathbf{Pat}(A) = \{A_i \colon i \in I\}$ is an automatic family with index set I. Let

$$J = \{j \in I \colon 0 \in A_j \wedge \forall i \in I \text{ with } i <_{ll} j \; [A_i \neq A_j]\}.$$

Then $\{A_j \colon j \in J\}$ is an automatic family containing the languages $L \in \mathbf{Pat}(A)$ with $0 \in L$; furthermore, for every such L there is a unique $j \in J$ with $L = A_j$.

By Proposition 14 below, each pattern language (over A) containing 0 is of the form $L(x^c) = \{c \cdot m \colon m \in \mathbb{Z}\}$, for some $c \in \mathbb{N}$. Note also that for all $c \in \mathbb{N}$, $\{c \cdot m \colon m \in \mathbb{Z}\}$ belongs to $\{A_j \colon j \in J\}$. For each $c \in \mathbb{N}$, let $rep(c) = j$ such that $A_j = \{c \cdot m \colon m \in \mathbb{Z}\}$.

Note that, for $c, d \in \mathbb{N}$, $\{c \cdot m \colon m \in \mathbb{Z}\} \subseteq \{d \cdot m \colon m \in \mathbb{Z}\}$ iff c is a multiple of d. Furthermore, this relation is first-order definable from automatic parameters on J via

$$c \text{ is a multiple of } d \iff \forall x \in A \, [x \in A_{rep(c)} \Rightarrow x \in A_{rep(d)}].$$

This implies that the structure $(\mathbb{N}, |)$, where $a|b$ denotes that a divides b, is automatic; however, Blumensath [5, Proposition 5.19] showed that $(\mathbb{N}, |)$ does not have an automatic presentation, a contradiction. $\square$

The following two propositions will be crucial in this and the subsequent section.

Proposition 14. *Suppose L is a pattern language over an Abelian group $(G, \circ)$. Then L is generated by a pattern of the form αx^n for some $\alpha \in G$ and $n \in \mathbb{N}$. Furthermore, $\alpha \in L$ can be chosen arbitrarily.*

Proof. The proposition holds trivially when L has only one element; so assume that L has at least two elements. By commutativity of $(G, \circ)$ one can assume without loss of generality that L is generated by a pattern $\alpha x_1^{n_1} x_2^{n_2} \ldots x_k^{n_k}$ with $\alpha \in G$. Let n be the greatest common divisor of $n_1, n_2, \ldots, n_k$. Then the following two statements hold:

(a) $\alpha x_1^{n_1} x_2^{n_2} \ldots x_k^{n_k}$ can be generated by the pattern αx^n by using the substitution $x_1^{n_1/n} x_2^{n_2/n} \ldots x_k^{n_k/n}$ for x;

(b) There are integers m_i satisfying $n = m_1 \cdot n_1 + m_2 \cdot n_2 + \ldots + m_k \cdot n_k$ and αx^n can be generated by the pattern $\alpha x_1^{n_1} x_2^{n_2} \ldots x_k^{n_k}$ by using the substitution x^{m_i} for x_i.

It follows from the above that $L(\alpha x_1^{n_1} x_2^{n_2} \ldots x_k^{n_k}) = L(\alpha x^n)$. Finally note that if $\beta \in L(\alpha x^n)$, then, for some $\delta \in G$, $\beta = \alpha\delta^n$ and $\alpha = \beta(\delta^{-1})^n$. It follows that $L(\alpha x^n) = L(\beta x^n)$. □

Proposition 15. *Suppose L is a pattern language over a finitely generated free Abelian group and β_1, β_2 are two distinct elements of L. Then, effectively from β_1 and β_2, a finite set of patterns can be found, one of which generates L.*

Proof. By Proposition 14, suppose L is generated by αx^n. Suppose $\gamma = \beta_1 \beta_2^{-1}$, where γ, β_1, β_2 are assumed to be in reduced form (that is, they do not contain a generator and its inverse). Then, $\gamma = (\gamma')^n$ for some γ', where γ' is in reduced form. Then for every generator b appearing in γ, say as b^m, it holds that n divides m. As there are only finitely many such n, it immediately follows that L is produced by one of finitely many patterns of the form $\beta_1 x^n$. □

Recall from Example 13 that there is no automatic family for the pattern languages over the group of integers with addition. However, if the underlying group $(\mathbb{Z}, +)$ is represented in an automatic way, it turns out that by using the hypothesis space consisting of all $L_{conv(a,b)} = a \cdot \mathbb{Z} + b$ with $a, b \in \mathbb{Z}$ and either $0 \leq b < a$ or $a = 0$ one can still ensure some good properties: Firstly, for each fixed index $conv(a, b)$, the set $L_{conv(a,b)}$ is regular; secondly, for each fixed x, the set of all $conv(a, b)$ with $x \in L_{conv(a,b)}$ is regular as well, as for almost all valid indices $conv(a, b)$, $x \in L_{conv(a,b)} \Leftrightarrow x \in \{-a + b, b, a + b\}$; the only exception being the finitely many pairs $conv(a, b)$ with $a \neq 0$ and $|a| \leq |x|$. As a result, if g denotes the function $(conv(a, b), x) \mapsto L_{conv(a,b)}(x)$, the structure

$$(\mathbb{Z}, \{conv(a, b) \colon a, b \in \mathbb{Z} \text{ and } (0 \leq b < a \text{ or } a = 0)\}, +; g)$$

is semiautomatic as defined by Jain, Khoussainov, Stephan, Teng and Zou [18]; that is, all the sets, functions and relations appearing before the semicolon are automatic, while the functions appearing after the semicolon become automatic when one fixes all but one of their inputs.

Now an automatic learner can keep track of the first number b' observed and conjecture $L_{conv(0,b')}$ until it sees further data. If the learner has seen at least two distinct nonnegative integers, then it keeps track of the least two distinct nonnegative integers in the input, say b and

$b+a$ respectively, where $0 \le b < b+a$. If $0 \le b < a$ then the learner conjectures $L_{conv(a,b)}$, else it repeats its previous conjecture.

This learner is automatic and uses a hypothesis space with decidable equality. An easier version of the algorithm can be found in the proof of the next result; however, the equality of hypotheses is not automatic there. On the other hand, that version works for a larger class of groups.

Theorem 16. *Let $(G, \circ)$ be a finitely generated Abelian automatic group. Then, for a suitable representation of $(G, \circ)$, there exists an automatic* **Ex***-learner for* **Pat**(G) *using a suitable hypothesis space.*

The class of pattern languages of a finite group is obviously **Ex**-learnable simply by maintaining a list of all elements observed. So only the case of infinite groups is interesting.

Proof. If the group $(G, \circ)$ is finite then there clearly is an automatic **Ex**-learner for **Pat**(G). So assume that the group is infinite, Abelian and finitely generated; all such groups are automatic. Furthermore, such a group is isomorphic to $(\mathbb{Z}, +) \times (A, \bullet)$ for some automatic group $(A, \bullet)$. Without loss of generality the first coordinate (from $\mathbb{Z}$) is represented using a reverse binary representation so that the ordering $<$ on it is automatic; the second coordinate (from A) can be represented in any automatic way. Then any element of $(\mathbb{Z}, +) \times (A, \bullet)$ can be represented as a convolution of the representations for the two coordinates.

Now a learner is built which keeps the following data in its memory:

- The first datum (z, a) different from # observed in the input;
- (z', a), if $S = \{z'' \colon (z'', a)$ has appeared in the input text so far and $z'' > z\} \neq \emptyset$ and $z' = \min(S)$.

The learner ignores all data (z'', a') with $a' \neq a$. Note that this does not hurt learnability as, by Proposition 14, the learner only needs to find the parameters α, n of a pattern αx^n to learn the input language. If only one element (z, a) is in the memory, then the learner conjectures the constant pattern which generates $\{(z, a)\}$. If two elements (z, a) and (z', a) are in the memory, where $z' > z$, then the learner conjectures $(z, a) \circ x^{z'-z}$. Note that the elements $(z, a), (z', a)$ can be generated by any pattern of the form $(z, a) \circ x^n$ iff $z' - z$ is a multiple of n. Now, using Proposition 14, it is easy to verify that the above learner **Ex**-learns $(\mathbb{Z}, +) \times (A, \bullet)$. This completes the proof. □

In the above theorem, if another hypothesis space or representation of the group G is chosen, then the learner might fail to be automatic. Note though that the learner from the above proof can easily be made iterative.

Remark 17. Note that the learning algorithm of Theorem 16 works for every automatic Abelian group of the form $(\mathbb{Z}, +) \times (A, \bullet)$, independently of what the second part of the direct product is. This gives a more general learning algorithm which covers many automatic groups, but not all of them; for example, the Prüfer group [14] is not of this form. However, it covers all infinite finitely generated Abelian automatic groups. It is clear that the pattern languages over the finite groups can be learnt by an automatic learner as well.

Furthermore, for some Abelian groups the class of pattern languages is not **Ex**-learnable, an example being

$$\left(\left\{\tfrac{m}{n}\colon m \in \mathbb{Z} \text{ and } n > 0 \text{ and no prime factor of } n \text{ occurs twice in the factorisation of } n\right\}, +\right).$$

However, it is not known and in fact seems unlikely that this group is automatic. To see that the above class is not **Ex**-learnable suppose otherwise. Then let D be a tell-tale set for $L(x)$ with respect to the class of pattern languages over the above group. Let m be a prime number such that none of the denominators of elements in D has m as a factor. Then, $D \subseteq L(y^m) \subset L(x)$, contradicting that D is a tell-tale for $L(x)$.

Theorem 18. *There exists an automatic group G generated by two elements such that* $\mathbf{Pat}(G)$ *is not* **Ex***-learnable.*

Proof. Consider the group G with two generators a, b and the following group operation $\circ$:

- $a \circ b = b^{-1} \circ a$;
- $a \circ a = \varepsilon$.

Thus every group element is either of the form b^i or ab^i for some $i \in \mathbb{Z}$. Furthermore, consider the pattern languages

$$\begin{aligned} L(x_1 a x_1^{-1} a) &= \{b^{2i}\colon i \in \mathbb{Z}\}; \\ L(b x_1 b^{-1} x_1^{-1}) &= \{\varepsilon, b^2\}; \\ L_i &= L(b^{-2i} \circ (b x_1 b^{-1} x_1^{-1}) \circ \ldots \circ (b x_{2i} b^{-1} x_{2i}^{-1})) \\ &= \{b^{-2i}, b^{-2i+2}, \ldots, b^{-2}, \varepsilon, b^2, \ldots, b^{2i-2}, b^{2i}\}. \end{aligned}$$

It is easy to see that $L_1 \subset L_2 \subset L_3 \subset \ldots$ and $\bigcup_{i \in \mathbb{N}} L_i = L(x_1 a x_1^{-1} a)$. Thus, $\mathbf{Pat}(G)$ is not **Ex**-learnable by Lemma 6. ☐

This result contrasts with the following result about the same group G, where $\mathbf{Verb}(G)$ is the class of verbal languages over G and $\mathbf{Verb}^*(G)$ denotes the class of finite unions of elements of $\mathbf{Verb}(G)$.

Proposition 19. *Let $(G, \circ)$ be as in the proof of Theorem 18. Then* $\mathbf{Verb}^*(G)$ *is* **Ex***-learnable.*

Proof. It is first shown that every verbal language over G is $\{\varepsilon\}$ or of the form $ab^@ \cup \bigcup_{k \in F} (b^k)^@$ or of the form $\bigcup_{k \in F} (b^k)^@$ for some finite set F. The idea to show this is similar to the one used in the proof of Proposition 11. Consider the sublanguages (subsets of the verbal language) generated by replacing each variable x_k in the pattern by either $\widetilde{x}_k$ or $a \circ \widetilde{x}_k$, with $\widetilde{x}_k$ only taking values from $b^@$. Each such sublanguage is of the form $(b^{\ell_1})^@ \circ (b^{\ell_2})^@ \circ \ldots \circ (b^{\ell_k})^@$ or of the form $a \circ (b^{\ell_1})^@ \circ (b^{\ell_2})^@ \circ \ldots \circ (b^{\ell_k})^@$, where $\ell_1, \ell_2, \ldots, \ell_k \in \mathbb{Z}$. In the first case, if $\ell > 0$ is the greatest common divisor of all ℓ_h, the sublanguage is either $\{\varepsilon\}$ or $(b^\ell)^@$. In the second case, at least one variable must occur in the pattern an odd number of times; setting this variable to $a \circ b^\ell$ and all other variables to ε results in the value $a \circ b^\ell$ (since $a \circ b^\ell \circ a \circ b^\ell = \varepsilon$ in G). Thus, the sublanguage in the second case is $a \circ b^@$. Note that the verbal language is a finite union of sublanguages of the above forms. Thus, every verbal language over G is $\{\varepsilon\}$ or of the form $ab^@ \cup \bigcup_{k\in F}(b^k)^@$ or of the form $\bigcup_{k\in F}(b^k)^@$ for some finite set F.

The preceding discussion allows building a learner as follows: It starts with the initial hypothesis $\emptyset$ and if it sees only the datum ε, it updates its hypothesis to $\{\varepsilon\}$. If at some point the learner sees for the first time a datum of the form ab^k, then it updates the current hypothesis H to $H \cup \{a \circ b^h \colon h \in \mathbb{Z}\}$. Whenever it observes a datum b^k which is not yet in the hypothesis, then it updates the current hypothesis H to $H \cup \{b^{kh} \colon h \in \mathbb{Z}\}$. It is clear that the learner converges to a correct hypothesis once it has seen all b^k with $k \in F$ and — in case that $a \circ b^@$ belongs to the language — some datum of the form ab^k. Thus, the learner **Ex**-learns $\mathbf{Verb}^*(G)$. ☐

There also exist automatic groups whose verbal languages are not learnable.

Proposition 20. *There is a finitely generated automatic group G such that* $\mathbf{Verb}(G)$ *is not* **Ex**-*learnable.*

Proof. Consider the group $(G, \circ)$ given by Jain, Miasnikov and Stephan [20, Section 6]. The group has the generators a, b, c, d with the special rules $a \circ c = c^{-1} \circ a$, $a \circ d = d \circ a$, $b \circ c = d \circ b$, $b \circ d = c \circ b$, $c \circ d = d \circ c$, $a \circ a = \varepsilon$, $b \circ b = \varepsilon$ and $a \circ b \circ a \circ b = b \circ a \circ b \circ a$. It was shown in [20] that every element of G can be expressed as $\alpha \circ c^m d^n$, where $\alpha \in \{\varepsilon, a, b, ab, ba, aba, bab, abab\}$. The subgroup generated by c, d is an Abelian subgroup isomorphic to $(\mathbb{Z} \times \mathbb{Z}, +)$. Therefore the group G is finitely generated and Abelian by finite, hence it has an automatic presentation by Oliver and Thomas [31].

It was shown in [20] that the pattern $\sigma_0 = x_0^4 = x_0x_0x_0x_0$ generates the language $\{c^{2i}d^{2i} \colon i \in \mathbb{Z}\} \cup \{c^{2i}d^{-2i} \colon i \in \mathbb{Z}\} \cup \{c^{4i'}d^{4i''} \colon i', i'' \in \mathbb{Z}\}$. Hence, the pattern $\pi = x_0^4x_1^4$ generates the language

$$\{c^{2i}d^{2i'} \colon i, i' \in \mathbb{Z} \text{ and } i \equiv i' (\text{mod } 2)\}.$$

Now it is shown that this equals the ascending union of the pattern languages generated by the patterns $(\sigma_h)_{h\in\mathbb{N}}$, defined inductively via $\sigma_{h+1} = \sigma_h^{h+1}x_{h+1}\sigma_h^h x_{h+1}^{-1}$. Note that

- $(c^nd^m)^{h+1} \circ c^id^j \circ (c^nc^m)^h \circ d^{-j}c^{-i} = c^{(2h+1)n}d^{(2h+1)m}$,
- $(c^nd^m)^{h+1} \circ ac^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}a = c^nd^{(2h+1)m}$,
- $(c^nd^m)^{h+1} \circ bc^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}b = c^{(h+1)n+hm}d^{(h+1)m+hn}$,
- $(c^nd^m)^{h+1} \circ abc^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}ba = c^{(h+1)n-hm}d^{(h+1)m+hn}$,
- $(c^nd^m)^{h+1} \circ bac^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}ab = c^{(h+1)n+hm}d^{(h+1)m-hn}$,
- $(c^nd^m)^{h+1} \circ abac^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}aba = c^{(h+1)n-hm}d^{(h+1)m-hn}$,
- $(c^nd^m)^{h+1} \circ babc^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}bab = c^{(2h+1)n}d^m$ and
- $(c^nd^m)^{h+1} \circ ababc^id^j \circ (c^nd^m)^h \circ d^{-j}c^{-i}baba = c^nd^m$.

The last equation shows that any element c^nd^m generated by the pattern σ_h can also be generated by the pattern σ_{h+1}. To see this use the same substitution for the variables $x_0, x_1, \ldots, x_h$, as used by σ_h for generating $c^n d^m$ and in addition use $sub(x_{h+1}) = ababc^i d^j$. Consequently, for all h, $L(\sigma_h) \subseteq L(\sigma_{h+1})$.

Also, by induction over h, it can be seen that, for all h, $L(\sigma_h) \subseteq L(\pi)$. Furthermore, for all $c^n d^m \in L(\sigma_h)$, $c^{(2h+1)n} d^m$, $c^n d^{(2h+1)m}$ and $c^{(2h+1)n} d^{(2h+1)m}$ belong to $L(\sigma_{h+1})$. Hence, $\{c^{2n} d^{2m} \colon n, m \in 2\mathbb{Z}+1\}$ is contained in $\bigcup_{h\in\mathbb{N}} L(\sigma_h)$. Note that $\{c^{2n} d^{2m} \colon n, m \in 2\mathbb{Z}\}$ is already contained in $L(\sigma_0)$. Therefore $\bigcup_{h\in\mathbb{N}} L(\sigma_h) = L(\pi)$.

Let $F_0 = \{-1, 1\}$ and $F_{h+1} = \{(h+1) \cdot i + h \cdot j \colon i, j \in F_h\}$. Note that F_h is a finite subset of $\mathbb{Z}$ and is closed under negation. Furthermore, $F_h \subseteq F_{h+1}$ as can be seen by noting that $(h+1) \cdot i + h \cdot (-i) = i$ for each $i \in F_h$. Now, by induction it is easy to verify that, for all h, $L(\sigma_h) \subseteq \{c^{4i} d^{4j} \colon i, j \in \mathbb{Z}\} \cup \{c^{s\cdot(4i+2)} d^{t\cdot(4i+2)} \colon i \in \mathbb{N} \text{ and } s, t \in F_h\}$. Thus, for all h and all but finitely many odd m, $c^2 d^{2m}$ does not belong to $L(\sigma_h)$. It follows that there are infinitely many h such that $L(\sigma_h) \subset L(\sigma_{h+1})$. Hence, there is an (infinitely often strictly) ascending chain of languages in $\mathbf{Verb}(G)$ whose union is in $\mathbf{Verb}(G)$ as well. Thus, $\mathbf{Verb}(G)$ is not $\mathbf{Ex}$-learnable by Lemma 6. $\square$

6 Learning bounded unions of patterns

In this section, the conditions are studied under which the class of bounded unions of pattern languages (over some group) is learnable, and when such learners can be automatic.

Recall the definition of the Prüfer groups from Example 3. In a Prüfer group, all pattern languages are either singletons or the full group; thus the bounded unions of these pattern languages have an automatic learner. In contrast to this, the automatic learnability of unions of two pattern languages fails already for the group of the integers with addition. The following proposition is useful for the next result.

Proposition 21 (Jain, Luo and Stephan [19]). *Suppose* $\mathbf{M}$ *is an automatic learner. Then, for some constant* c*, the length of the memory of* $\mathbf{M}$ *after input* $T[n]$ *is bounded by* $cn + \max(\{|T(i)| \colon i < n\})$.

Theorem 22. *Consider the group* $(\mathbb{Z}, +)$*. The class* $\mathbf{Pat}^2(\mathbb{Z})$ *of unions of up to two pattern languages over the integers does not have an automatic* $\mathbf{Ex}$*-learner for any automatic representation of the group.*

Proof. The proof follows the approach of analysing the memory constraints of automatic learners [8, 9, 19] and deriving that an automatic learner cannot memorise the required amount of data.

From now on, let $\log(a)$ be defined as $1 + \min(\{n \in \mathbb{N} \colon -2^n < a < 2^n\})$. For any automatic representation of $\mathbb{Z}$, starting with one of $-1, 0, 1$, the value $a \in \mathbb{Z}$ can be obtained by at most $\log(a)$ applications of the functions $b \mapsto b + b$, $b \mapsto b + b + 1$ or $b \mapsto b + b - 1$; each of these automatic functions increases the length of the representation by at most a constant. Thus, in any automatic representation of $\mathbb{Z}$, the representation of a number $a \in Z$ is bounded in length by $O(\log(a))$.

Fix an arbitrary $n \in \mathbb{N}$ and $r_1, r_2, \ldots, r_n \in \{1, 2, 3, \ldots, 2^n\}$. Let $p_1, p_2, \ldots, p_n$ be the first n prime numbers $2, 3, 5, 7, \ldots$ and for any $m \in \{1, 2, \ldots, n\}$, let

$$q_m = \left(\prod_{k \in \{1,\ldots,n\} - \{m\}} p_k \right) \cdot (1 + p_m \cdot r_m).$$

Now, suppose by way of contradiction that some automatic learner $\mathbf{M}$ $\mathbf{Ex}$-learns $\mathbf{Pat}^2(\mathbb{Z})$. Since for some constant c, $p_n \leq cn\log(n)$ and $q_n \leq 2^{n+1} \cdot (cn\log(n)+1)^n$, each of the numbers $q_1, \ldots, q_n$ can be represented in the underlying automatic representation of $\mathbb{Z}$ by using at most $c'n\log(n)$ many symbols, where c' is a suitable constant.

Let σ be a finite sequence such that $\mathrm{content}(\sigma) = \{q_1, q_2, \ldots, q_n\}$ and let u denote the memory of $\mathbf{M}$ after receiving input σ. Then, by Proposition 21, the length of u is $O(n\log(n))$.

Let $L_m = L(x^{p_m}) \cup \{q_m\}$ and T_m be a text for $L(x^{p_m})$. Note that, for $i \neq j$, $q_i \in L(x^{p_j})$, as q_i is divisible by p_j. Thus, for each $m \in \{1, 2, \ldots, n\}$, $\{q_1, q_2, \ldots, q_n\} \subseteq L_m$ and σT_m is a text for L_m. By assumption, $\mathbf{M}$ converges on the input text σT_m to a hypothesis for L_m. Now $\{q_m\} = L_m - L(x^{p_m})$ and thus first p_m and then q_m can be reconstructed from u in the limit (note that p_m is the unique prime which divides all but one member of L_m). Thus using different T_i, the whole of $(q_1, q_2, \ldots, q_n)$ and thus $(r_1, r_2, \ldots, r_n)$ can be reconstructed from u in the limit. As $r_1, r_2, \ldots, r_n \in \{1, 2, 3, \ldots, 2^n\}$ can be chosen arbitrarily, the memory u must be able to code 2^{n^2} many different choices of the $(r_1, r_2, \ldots, r_n)$ as otherwise these numbers cannot be reconstructed in the limit. However, there are only $2^{O(n\log(n))}$ many choices for u, as the alphabet used to code u is finite. This gives a contradiction for large enough n.

Therefore, there is no automatic $\mathbf{Ex}$-learner for $\mathbf{Pat}^2(\mathbb{Z})$. $\square$

Though there is no automatic $\mathbf{Ex}$-learner for unions of two pattern languages over $\mathbb{Z}$, the next results will show that for finitely generated Abelian groups, there is a recursive (nonautomatic) $\mathbf{Ex}$-learner for unions of pattern languages.

Theorem 23. *Suppose G is a finitely generated free Abelian group. Then the class $\mathbf{Pat}^m(G)$ is $\mathbf{Ex}$-learnable.*

Proof. Let Σ be the finite set of generators for G. Suppose T is a text for $L \in \mathbf{Pat}^m(G)$. Let P_G be a set of at most m patterns such that $L = \bigcup_{\pi \in P_G} L(\pi)$. Without loss of generality assume that P_G is chosen such that the number of patterns in P_G that generate at least two distinct elements of G is minimal.

The learner $\mathbf{M}$ stores the full input sequence $T[n]$ seen so far and a finite labeled tree in its memory. The labels on the nodes of the tree are patterns that generate at least two elements of G, except for the root which has an empty label. The tree is finitely branching and has depth of at most m. For any leaf z in the tree, the language S_z associated with the leaf is the union of the languages generated by the labels on the nodes (including the node z) in the path from the root to z.

Initially, the tree in the memory of $\mathbf{M}$ has only one node, the root. On input $T(n)$, $\mathbf{M}$ updates the tree as described in (A).

(A) For any leaf z in the tree, let $X_z = \text{content}(T[n+1]) - S_z$. If z is at depth $r < m$, where the root is at depth 0, and $\text{card}(X_z) > m - r$, then apply Proposition 15 to find a finite list of patterns which is guaranteed to contain equivalent patterns for every pattern that generates at least two elements of X_z. Each of these finitely many patterns is added as a child of z in the tree.

As the tree in the memory of the learner is of bounded depth, has finite branching degree and only leaf nodes can be expanded, the tree stabilises as n goes to infinity. The learner's hypothesis is computed as follows:

(B) For the tree as above after receiving $T[n+1]$, define the language Y_z for any leaf z at depth r of the tree as follows. Let $X_z = \text{content}(T[n+1]) - S_z$. Let $Y_z = S_z \cup X_z$, if $\text{card}(X_z) \leq m - r$. Otherwise, $Y_z = \Sigma^*$ (in this case, Y_z can be considered spoiled).

(C) One calls a $Y_{z'}$ n-minimal if $Y_{z'} \cap (\Sigma^0 \cup \ldots \cup \Sigma^n)$ is subset-wise minimal among all $Y_z \cap (\Sigma^0 \cup \ldots \cup \Sigma^n)$, where z is a leaf. Then $\mathbf{M}$ conjectures the n-minimal Y_z that is leftmost in the tree. Note that the set of at most m patterns generating the language Y_z can be correspondingly computed.

To see that $\mathbf{M}$ $\mathbf{Ex}$-learns $\mathbf{Pat}^m(G)$, note by induction that, for each n, after $\mathbf{M}$ has seen and processed $T[n+1]$ as above, the following statements hold for each leaf z in the tree and for r, Y_z and X_z as defined in (B):

(i) $\text{content}(T[n+1]) \subseteq Y_z$;
(ii) if $\text{card}(X_z) > m - r$, then the depth of z is m as otherwise children would have been added to the leaf z in (A);
(iii) the pattern labels on the simple path from the root to z generate pairwise distinct languages;
(iv) for some leaf z', the simple path from the root to z' is labeled only using patterns equivalent to patterns in P_G.

Thus, for large enough n and for z' as given by item (iv) above, it holds that $Y_{z'} \subseteq L \subseteq Y_{z'}$. It follows that, for large enough n, $\mathbf{M}$ conjectures L. □

The learner used in the proof of Theorem 23 can also be made iterative using a suitable class-preserving hypothesis space. To see this, first note that instead of storing "the full input sequence $T[n]$ seen so far", the memory of the learner can be modified so that for each leaf z of the tree, the information X_z (as defined in (A)) is also kept in memory as long as the size of X_z is bounded by $m - r$. The convergence of the memory of the learner then holds as argued in the proof. Hence, the learner can be made iterative by just padding the memory of the learner to its hypothesis as described in the remark after Proposition 10.

Similarly, the learner for the following theorem can be made iterative, using a suitable hypothesis space.

Theorem 24. *Any finitely generated (not necessarily free) Abelian group $(G, \circ)$ has an automatic presentation such that, for all m, $\mathbf{Pat}^m(G)$ is $\mathbf{Ex}$-learnable using some suitable hypothesis space.*

Proof. Any finitely generated Abelian group $(G, \circ)$ is equivalent to $A \circ B$, where $A \cap B = \{\varepsilon\}$, A is a finite subgroup of G and B is a finitely generated free Abelian subgroup of G. So consider such A and B and let $s = |A|$. Note that for all $a \in A$, $a^s = \varepsilon$. Now, for any pattern $\alpha\beta x^n$ over G, such that $\alpha \in A$ and $\beta \in B$, $L(\alpha \circ \beta x^n) = \bigcup_{a \in A} L(\alpha \circ a^n \circ \beta \widetilde{x}^n) = \bigcup_{a \in A} \alpha \circ a^{(n \bmod s)} L(\beta \widetilde{x}^n)$, where $\widetilde{x}^n$ is only allowed substitutions from B. Thus, each pattern language $L(\alpha\beta x^n)$ over G is a union of at most $|A|$ many pattern languages of the form $\alpha' \circ L(\beta \widetilde{x}^n)$, where $\alpha' \in A$ and $L(\beta \widetilde{x}^n)$ is a pattern language over B. The statement now follows from Theorem 23. $\square$

7 Conclusion and Future Work

Class	Automatic	**Ex**-learnable	**Ex**-learnable for finitely generated G	**Ex**-learnable for finitely generated Abelian G
$\mathbf{Pat}_n(G)$	Yes	For $n = 1$	Automatic learner	Automatic learner
$\mathbf{Pat}_n^m(G)$	Yes	For $n = 1$	Automatic learner	Automatic learner
$\mathbf{Pat}(G)$	No	No	No	Automatic learner
$\mathbf{Pat}^m(G)$	No	No	No	Recursive learner

Table 1. Summary of when the pattern language class over G is learnable. For learnability, the best possible type of learner is given, if it exists.

This article studied the learnability of pattern languages over groups. It was shown that for every finitely generated automatic group G, the class of pattern languages over G generated by patterns having a bounded number of variable occurrences is **Ex**-learnable. The same holds for bounded unions of such languages. Furthermore, for finitely generated Abelian groups G, the class of all pattern languages over G, as well as for each m the class of unions of m pattern languages, are **Ex**-learnable. However, for some non-Abelian automatic group G generated by two elements, the class of pattern languages over G is not **Ex**-learnable. Similarly, for some infinitely generated group G, even the class of pattern languages over G generated by patterns having at most two occurrences of variables is not **Ex**-learnable. Table 1 gives a summary of these results.

It was shown that in some cases the learners can be made iterative using a suitable hypothesis space. It is an open question whether, for every automatic group, it holds that if the class of all pattern languages over this group is **Ex**-learnable then it is also iteratively learnable.

In the standard setting of pattern languages (over the semigroup of strings and concatenation), Geilke and Zilles [12] investigated how learnability changes when the elements substituted for variables are required to be related to each other in some given ways. Compare this with the group setting of the present article where every variable x is required to satisfy $x \circ x^{-1} = \varepsilon$. This can be seen as an implicit relational requirement in the sense of Geilke and Zilles by replacing x^{-1} with a new variable symbol x' and then adding a new side condition $x \circ x' = \varepsilon$.

More generally, one could ask what happens if side conditions are added that consist of equations involving additional patterns, that is, if one looks at languages $L(\pi\colon \pi' = \pi'')$, where $L(\pi\colon \pi' = \pi'')$ is defined as the language containing all words of the form $\pi(x_1, x_2, \ldots, x_n)$ such that $\pi'(x_1, x_2, \ldots, x_n) = \pi''(x_1, x_2, \ldots, x_n)$. Multiple such side conditions may be allowed. Note that due to the closure of automatic structures under first-order definitions with automatic parameters, each such pattern language is, for an automatic semigroup, a regular set; though, in general, the class of all such sets does not form an automatic family.

Using this approach, one could, on one hand, generalise the notion of patterns to semigroups and monoids, as the above framework provides a method to deal with the fact that in a monoid, for fixed a and y, there is not always an x with $x \circ a = y$ or with $x \circ y = a$. On the other hand, the extra power of the definition leads to unlearnability in many cases, so that one has to be very restrictive in the choice of relations permitted. For example, when applying the approach to the monoid $(\mathbb{N}, +, 0)$, the language $L(x_1\colon x_1 + x_2 = c)$ would just be $\{x_1 \in \mathbb{N} : x_1 \leq c\}$ and so the class of all pattern languages containing up to three occurrences of variables and up to one side condition, which also contains $\mathbb{N} = L(x_1)$, is unlearnable due to Lemma 6. Similarly, for the Prüfer group $(G, +)$ of base 2, the class of all pattern languages is learnable without side conditions; but permitting a single side condition renders the class unlearnable, as the set $L(x\colon 2^n \cdot x = 0)$ is just $\{m \cdot 2^{-n}\colon 0 \leq m < 2^n\}$ and the ascending union of these sets is the pattern language $L(x)$ which again implies non-learnability of the class by Lemma 6; note that here $2^n \cdot x$ stands for the sum of 2^n terms x in the pattern.

The authors would like to point out that this is merely a sketch of the implications of the approach of Geilke and Zilles when it is applied to learning patterns over groups and monoids. To gain a more detailed understanding, further work is needed.

Acknowledgements. A preliminary version of this article appeared at the conference on Algorithmic Learning Theory (ALT 2016) [17]. The authors would like to thank the referees of that conference as well as the referees of this journal for detailed comments that helped to improve the presentation of this article.

References

1. Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control* 45:117–135, 1980.
2. Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
3. Ganesh Baliga, John Case and Sanjay Jain. The synthesis of language learners. *Information and Computation*, 152:16–43, 1999.
4. Janis Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.
5. Achim Blumensath. *Automatic structures.* Diploma thesis, RWTH Aachen, 1999.
6. Achim Blumensath and Erich Grädel. Automatic structures. *Fifteenth Annual IEEE Symposium on Logic in Computer Science, Santa Barbara*, LICS 2000, pages 51-62. IEEE Computer Society Press, Los Alamitos, CA, 2000.

7. Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
8. John Case, Sanjay Jain, Yuh Shin Ong, Pavel Semukhin and Frank Stephan. Automatic learners with feedback queries. *Journal of Computer and System Sciences*, 80:806–820, 2014.
9. John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *Logical Methods in Computer Science*, 9(3), 2013.
10. John Case and Chris Lynes. Machine inductive inference and language identification. *Proceedings of the Nineth International Colloquium on Automata, Languages and Programming*, ICALP 1982, *Springer LNCS* 140:107–115, 1982.
11. John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
12. Michael Geilke and Sandra Zilles. Learning Relational Patterns. In *Algorithmic Learning Theory: 22nd International Workshop (ALT' 2011)*, volume 6925 of *Lecture Notes in Artificial Intelligence*, pages 84–98. Springer-Verlag, 2011.
13. E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
14. Pierre Antoine Grillet. Abstract Algebra. Springer, 2007.
15. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.
16. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
17. Rupert Hölzl, Sanjay Jain and Frank Stephan. Learning Pattern Languages over Groups. In *Algorithmic Learning Theory: 27th International Workshop (ALT' 2016)*, volume 9925 of *Lecture Notes in Artificial Intelligence*, pages 174–188. Springer-Verlag, 2016.
18. Sanjay Jain, Bakhadyr Khoussainov, Frank Stephan, Dan Teng and Siyuan Zou. Semiautomatic structures. Computer Science – Theory and Applications – Ninth International *Computer Science Symposium in Russia*, CSR 2014, Moscow, Russia, June 7–11, 2014. Proceedings. Springer LNCS 8476:204–217, 2014.
19. Sanjay Jain, Qinglong Luo and Frank Stephan. Learnability of automatic classes. *Journal of Computer and System Sciences*, 78:1910–1927, 2012.
20. Sanjay Jain, Alexei Miasnikov and Frank Stephan. The complexity of Verbal Languages over Groups. *27th Annual ACM/IEEE Symposium on Logic in Computer Science*, (LICS) 2012, pages 405–414. IEEE Computer Society, 2012.
21. Sanjay Jain, Yuh Shin Ong, Shi Pu and Frank Stephan. On automatic families. *Proceedings of the eleventh Asian Logic Conference* in honour of Professor Chong Chi Tat on his sixtieth birthday, pages 94–113, World Scientific, 2012.
22. Olga Kharlampovich and Alexei Myasnikov. Elementary theory of free non-Abelian groups. *Journal of Algebra*, 302(2):451–552, 2006.
23. Olga Kharlampovich and Alexei Myasnikov. Definable subsets in a hyperbolic group. *International Journal of Algebra and Computation*, 23(1):91–110, 2013.

24. Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. *Proceedings of Logic Colloquium 2007. Lecture Notes in Logic*, 35:132–176, 2010.
25. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity* (International Workshop LCC 1994). Springer LNCS 960:367–392, 1995.
26. Steffen Lange and Rolf Wiehagen. Polynomial time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.
27. Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53:88–103, 1996.
28. Alexei Myasnikov and Vitaly Romankov. On rationality of verbal subsets in a group. *Theory of Computing Systems*, 52(4):587–598, 2013.
29. André Nies. Describing groups. *Bulletin of Symbolic Logic*, 13:305–339, 2007.
30. André Nies and Richard M. Thomas. FA-presentable groups and rings. *Journal of Algebra*, 320:569–585, 2008.
31. Graham Oliver and Richard M. Thomas. Automatic presentations for finitely generated groups. *Twentysecond Annual Symposium on Theoretical Aspects of Computer Science* (STACS 2005), Stuttgart, Germany, Proceedings. *Springer LNCS*, 3404:693–704, 2005.
32. Daniel Osherson, Michael Stob and Scott Weinstein. *Systems That Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists.* Bradford — The MIT Press, Cambridge, Massachusetts, 1986.
33. Daniel Osherson and Scott Weinstein. Criteria for language learning. *Information and Control*, 52:123–138, 1982.
34. Lenny Pitt. Inductive inference, DFAs, and computational complexity. *Analogical and Inductive Inference, Proceedings of the Second International Workshop*, AII 1989. Springer LNAI 397:18–44, 1989.
35. Daniel Reidenbach. A non-learnable class of E-pattern languages. *Theoretical Computer Science*, 350:91–102, 2006.
36. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.
37. Takeshi Shinohara. Polynomial time inference of extended regular pattern languages. *RIMS Symposia on Software Science and Engineering, Kyoto, Japan, Proceedings*. Springer LNCS 147:115–127, 1982.
38. Takeshi Shinohara and Hiroki Arimura. Inductive Inference of Unbounded Unions of Pattern Languages from Positive Data. In *Algorithmic Learning Theory: 7th International Workshop (ALT' 1996)*, volume 1160 of *Lecture Notes in Artificial Intelligence*, pages 256–271. Springer-Verlag, 1996.
39. Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.
40. Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)*, 12(1–2):93–99, 1976.