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 any finitely generated Abelian automatic group is explanatorily learnable. Furthermore, patterns are considered in which variables occur at most n times. The classes of the languages generated by such patterns as well as the bounded unions of such languages are, for finitely generated automatic groups, explanatorily learnable by an automatic learner. On the other hand, the unions of up to two arbitrary pattern languages over the integers cannot be learnt by any automatic learner. Furthermore, there is an algorithm which provides for every automatic group G, given by the automata describing the group, a learning algorithm M_G such that either M_G explanatorily learns all the pattern languages over that group or there is no such learner at all, even not a non-recursive one. For some automatic groups, non-learnability results of natural classes of pattern languages are provided.

1 Introduction

Gold [9] 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, 7, 8, 23]. Inductive inference can be described as follows; see Section 2 for the formal details: The learner reads, one by one as input, 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 the language. While reading the data items from the text, the learner conjectures a sequence of hypotheses (grammars), one hypothesis at a time; subsequent data may lead to revision of earlier hypotheses. The learner is considered to have learnt

* S. Jain is supported in part by NUS grants R146-000-181-112, R252-000-534-112 and C252-000-087-001.
** F. Stephan is supported in part by NUS grants R146-000-181-112 and R252-000-534-112.

the target language L if the sequence of hypotheses converges syntactically to a grammar for L. The learner is said to learn the class $\mathcal{L}$ of languages if it learns each language in $\mathcal{L}$. The above model of learning is often referred to as explanatory learning (**Ex**-learning) or learning in the limit.

Angluin [2] introduced one important concept studied in learning theory, namely the concept of (non-erasing) pattern languages. Shinohara [27] 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 all the strings that could be obtained from the pattern using some substitution. Angluin showed that when substitutions allowed for variables are non-empty strings, then the class of pattern languages is **Ex**-learnable; Lange and Wiehagen [17] provided a polynomial time learner for non-erasing pattern languages. On the other hand, Reidenbach [25] showed that if arbitrary strings (including empty strings) are allowed for substitutions of variables, then the class of pattern languages is not **Ex**-learnable.

This paper explores learnability of pattern languages over groups; pattern languages and verbal languages (that are 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 [13, 14]. Miasnikov and Romankov [19] studied when verbal languages are regular (in certain representations of the group).

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

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

and $bab^{-1}a^{-3}b \in L(xax^{-1}yyab)$ by letting $x = b$ and $y = a^{-2}$. So the concatenation in the pattern is replaced by the group operation $\circ$ and in this way a pattern generates a subset of the group G. Note that the replacement of variables by the identity element of G is allowed.

This paper 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 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 one-one and onto mapping rep from G to A) and the relation $R = \{(rep(x), rep(y), rep(z)) : x, y, z \in G$

and $x \circ y = z\}$. Automatic groups in this paper follow the original approach by Hogdson [10, 11] and later by Khoussainov and Nerode [16] and Blumensath and Grädel [4, 5]; automatic groups have also been studied by Nies, Oliver, Thomas and Tsankov [20–22, 28].

In some cases, automatic groups allow to represent 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 $\{(e, x) : e \in E \wedge x \in L_e\}$ for some regular index set E. Automatic families allow to implement learners which are themselves automatic; such learners satisfy some additional complexity bound and results in a restriction though many complexity bounds in learning theory are not restrictive [6, 12, 24]. 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 [10, 16]; see the surveys of Khoussainov and Minnes [15] and Rubin [26] for more information.

Theorem 6 below strengthens Angluin's characterisation [1] result 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 criterion non-effectively (see Section 3 for definition of the tell-tale criterion). It follows from Angluin's work that this non-effective version of the tell-tale criterion is necessary. Note that for general indexed families, this non-effective version of tell-tale criterion is not sufficient and gives rise only to a behaviourally correct learner [3].

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 7 shows that $\mathbf{Pat}_1(G)$ is **Ex**-learnable for all automatic groups G, though $\mathbf{Pat}_2(G)$ is not **Ex**-learnable for some automatic group G. This group G has infinitely many generators. Theorem 10 shows that $\mathbf{Pat}_n(G)$ is **Ex**-learnable for all finitely generated automatic groups G (in fact, for any fixed m, even the class of unions of upto m such pattern languages is **Ex**-learnable).

Sections 5 and 6 consider learnability of the class of all pattern languages. Theorem 16 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 20 shows that for finitely generated Abelian groups G, $\mathbf{Pat}(G)$ as well as the class of unions of upto m pattern languages over G is **Ex**-learnable, for any fixed m. Theorem 14 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 (see Section 2 for the definition of automatic learners), though this hypothesis space cannot in general be an automatic family. However, the class of unions of upto two pattern languages over the integers with group operation + is not automatically **Ex**-learnable (see Theorem 18).

Some proofs are omitted due to space constraints.

2 Preliminaries

The symbol $\mathbb{N} = \{0, 1, 2, \ldots\}$ denotes the set of natural numbers and $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ denotes the set of integers.

For any alphabet set Σ, Σ^* denotes a set of strings over Σ. $\Sigma^{@}$ denotes the set of strings over Σ, where inverses of the members of Σ are also allowed in the string (this is useful when considering groups). The length of a string w is denoted by $|w|$. $w(i)$ denotes the i-th character of 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 $m = \max(\{|u|, |v|\})$. 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. The convolution is useful when considering relations with two or more inputs like the graph of a function.

A function f is said to be *automatic* if $\{conv(x, f(x)) : x \text{ in domain of } f\}$ is regular. An n-ary relation R is said to be *automatic* if $\{conv(x_1, x_2, \ldots, x_n) : (x_1, x_2, \ldots, x_n) \in R\}$ is regular. A structure $(A, R_1, R_2, \ldots, f_1, f_2, \ldots)$ is said to be automatic, if A is a regular domain of the structure, $f_1, f_2, \ldots$ are automatic functions from A^k to A for some k and $R_1, R_2, \ldots$ are automatic relations over the domain A^h for some h. A class $\mathcal{L}$ of languages over alphabet Σ is said to be an *automatic family* if there eixsts a regular index set I and there exist languages L_α, $\alpha \in I$, such that $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ and the set $\{conv(\alpha, x) : \alpha \in I, x \in \Sigma^*, x \in L_\alpha\}$ is regular. 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 is useful for showing that various relations or functions are automatic.

Fact 1 (Blumensath and Grädel [5], Hodgson [10, 11], Khoussainov and Nerode [16]). *Any relation or function that is first-order definable from existing automatic relations and functions is automatic.*

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 exists an identity element $\varepsilon \in G$ such that for all $a \in G$, $a \circ \varepsilon = \varepsilon \circ a = a$;
- (inverse) for all $a \in G$, there is 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 *commutative* or *Abelian* if for all $a, b \in G$, $a \circ b = b \circ a$. 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 $a \circ b^{-1} \circ c$.

Σ is a set of generators for a group $(G, \circ)$ if all members of G can be written as a finite string over members 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 $G = \{a_1^{m_1} \circ a_2^{m_2} \circ \ldots \circ a_n^{m_n} : m_1, m_2, \ldots, m_n \in \mathbb{Z}\}$ and for each group element 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 $L \subseteq G$, $rep(L) = \{rep(a) : a \in L\}$. Often a group member 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:

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

In this case $(A, \circ)$ is called an automatic presentation of the group $(G, \circ)$; in the following, for the ease of notation, the groups are identified with their automatic presentation. Note that the second clause above implies that the function mapping $\alpha \mapsto \alpha^{-1}$, computing 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 $\varepsilon \in \Sigma^*$ is the representative of $\varepsilon \in G$. 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). In the above group, the order given by $<$ is also automatic and so the entire automatic structure $(\mathbb{Z}, +, <)$ is often used.

Angluin [2] introduced the concept of pattern languages in the field of learning theory. This concept is adapted to the notion of pattern languages over groups as follows. Fix a group $(G, \circ)$. 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 members 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 using 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 $L(\pi) = \{sub(\pi) : sub \text{ is a substitution}\}$. Two patterns π_1 and π_2 are said to be *equivalent* (with respect to the group G) iff $L(\pi_1) = L(\pi_2)$. A *pattern language* (over a group G) is a language generated by some pattern π. In case the pattern π does not contain any constants, then the language $L(\pi)$ generated by π is called 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 upto 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 upto m pattern languages generated by patterns containing up to n occurrences of variables or inverted variables.

Proposition 2. *The classes* $\mathbf{Pat}_n(G)$ *and* $\mathbf{Pat}_n^m(G)$ *are automatic families for all automatic groups* $(G, \circ)$ *and* $m, n \in \mathbb{N}$.

Gold [9] introduced the model of learning in the limit which is described below. Fix a group $(G, \circ)$. A *text* T for a language L is a mapping from $\mathbb{N}$ to $rep(G) \cup \{\#\}$, where as mentioned earlier members of G are identified with $rep(G)$, and thus a text can be viewed as a sequence of elements of $G \cup \{\#\}$. A finite sequence is an initial segment of a text. Λ denotes the empty sequence. Let $|\sigma|$ denote the length of sequence σ. The content of a text T, denoted content(T), is the set of members of G in the range of T, that is, content$(T) = \{T(i) : T(i) \neq \#\}$. Similarly, the content of a finite sequence σ, denoted content(σ), is $\{\sigma(i) : i < |\sigma| \text{ and } \sigma(i) \neq \#\}$. Intuitively, #'s denote pauses in the presentation of data. T is a *text for* $L \subseteq G$ iff 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 conjecture. The learner has some initial memory and conjecture 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 $\mathcal{H} = \{H_\alpha : \alpha \in J\}$ for its conjectures. It is assumed for this paper 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 (d) to (f) of the definition give a basic learning criterion called explanatory learning.

Definition 3 (Based on Gold [9]). Fix a group $(G, \circ)$. Suppose I and J are some index sets (regular sets of strings over some finite alphabet).

Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is a class to be learnt and $\mathcal{H} = \{H_\beta : \beta \in J\}$ is a hypothesis space, with $L_\alpha, H_\beta \subseteq G$ for all $\alpha \in I$ and $\beta \in J$.

Suppose Δ is a finite alphabet used for storing memory by the learner and ? is a special symbol not in $J \cup \Delta^*$ used for null hypothesis as well as null memory.

(a) A *learner* is a recursive mapping from $(\Delta^* \cup \{?\}) \times G \cup \{\#\})$ to $(\Delta^* \cup \{?\}) \times (J \cup \{?\})$. A learner has initial memory $mem_0 \in \Delta^* \cup \{?\}$ and initial conjecture $hyp_0 \in J \cup \{?\}$.

(b) Suppose a learner $\mathbf{M}$ with initial memory mem_0 and initial hypothesis hyp_0 is given. 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 $(mem^T_{n+1}, hyp^T_{n+1}) = \mathbf{M}(mem^T_n, T(n))$.

Intuitively, mem^T_{n+1} and hyp^T_{n+1} denote the memory and conjecture of the learner $\mathbf{M}$ after receiving input $T[n+1]$.

(c) M converges on T to a hypothesis hyp iff for all but finitely many n, $hyp^T_n = hyp$.

(d) $\mathbf{M}$ is said to **Ex**-learn a language 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 $H_{hyp} = L$.

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

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

The notation **Ex** in the above definition stands for "explanatory learning" and was first introduced by Gold [9]. 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 $\mathbf{M}$ is said to make a mind change [7, 8] at $T[n]$, if $? \neq hyp^T_n \neq hyp^T_{n+1}$ as defined in the above definition. A learner makes at most m mind changes on a text T iff $\{n : ? \neq hyp^T_n \neq hyp^T_{n+1}\} \leq m$.

A learner is said to be *automatic* if its updating function is automatic, that is, if the relation {$conv$(old mem, datum, new mem, hypothesis) : $\mathbf{M}$(old mem, datum) = (new mem, hypothesis)} is automatic.

For ease of presentation of proofs, sometimes it is informally described how the learner updates its memory and hypothesis when a new datum is presented. Furthermore, sometimes a language is directly used as hypothesis rather than an 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 4 (Gold [9]). *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

A *tell-tale set* [1] for a language L with respect to a class $\mathcal{L}$, is a finite subset D of L such that, for every $L' \in \mathcal{L}$, $D \subseteq L' \subseteq L$ implies $L' = L$. A class $\mathcal{L}$ satisfies Angluin's tell-tale condition non-effectively iff every language L in $\mathcal{L}$ has a tell-tale set with respect to $\mathcal{L}$. A class $\mathcal{L} = \{L_i : i \in I\}$ 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 $\mathcal{L} = \{L_i : i \in I\}$, where I is a recursive set, such that $\{\langle i, x\rangle : x \in L_i\}$ is recursive.

Proposition 5 (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 non-effectively. Angluin's result is now used to obtain a characterisation for learnability of $\mathbf{Pat}^m(G)$, for any automatic group G.

Theorem 6. *Given an automatic group G, if $\mathbf{Pat}^m(G)$ satisfies Angluin's tell-tale condition non-effectively, then $\mathbf{Pat}^m(G)$ is $\mathbf{Ex}$-learnable.*

Thus, $\mathbf{Pat}^m(G)$ is $\mathbf{Ex}$-learnable iff it satisfies Angluin's tell-tale condition non-effectively. The above result also holds when considering $\mathbf{Pat}^m_n(G)$ instead of $\mathbf{Pat}^m(G)$. Note that the above result implies that the $\mathbf{Ex}$-learnability of $\mathbf{Pat}^m(G)$ or $\mathbf{Pat}^m_n(G)$ 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 $\mathbf{Pat}^m_n(G)$ is $\mathbf{Ex}$-learnable for all finitely generated automatic groups G. For non-finitely generated automatic groups, $\mathbf{Pat}_1(G)$ is $\mathbf{Ex}$-learnable, but $\mathbf{Pat}_2(G)$ is not $\mathbf{Ex}$-learnable.

Theorem 7. *For every automatic group $(G, \circ)$, the class $\mathbf{Pat}_1(G)$ can be $\mathbf{Ex}$-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 $\mathbf{Ex}$-learnt by any learner.*

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:

Oliver and Thomas [22] 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 [22, 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 $j_{a,1}, \ldots, j_{a,m}$ with

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

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 an element of H', a corresponding coordinate tuple from $\mathbb{Z}^h$ with respect to fixed h group elements generating 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 members 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 $U \circ \beta \cap U \circ \beta' = \emptyset$. Now the following family is automatic for each constant n:

$$\mathcal{Q}_n(\mathcal{R}, H') = \{(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h) : h \leq n \text{ and } U_1, U_2, \ldots, U_h \in \mathcal{R} \text{ and } \beta_1, \beta_2, \ldots, \beta_h \in H'\}.$$

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

Proposition 8. *For every n, the class $\mathcal{Q}_n(\mathcal{R}, H')$ has an automatic* **Ex***-learner using $\mathcal{Q}_n(\mathcal{R}, H')$ as the hypothesis space.*

Proof. The learner in its memory maintains a set of candidate conjectures, where each candidate conjecture 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 memorised using a convolution of the representation of each of the candidates.

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

Initially the learner has only one candidate which is the empty union. At any stage of the learning process, when a new input w is processed, the following is done for each candidate in the current set. If a candidate $(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h)$ with $h < n$ does not contain w, 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 a member of $\mathcal{R}$ and β_{h+1} exists and is the length-lexicographically least member of H' such that $w \in U_{h+1} \circ \beta_{h+1}$. If no such U_{h+1}, β_{h+1} exist, 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 all the above operations are automatic. Furthermore, note that for each replaced candidate, at most $|\mathcal{R}|$ new candidates are added in. As none of these unions has more than n terms, the overall number of candidates considered through 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 the language L to be learnt, the learner will eventually see a counter example 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 maintained by the learner.

Suppose $L = (U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h)$, where $h \leq n$ is minimal. Without loss of generality assume that β_i are length-lexicographically least member β_i' of H' such that $U_i \circ \beta_i = U_i \circ \beta_i'$. Then eventually, $(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h)$ is added to the candidate set by the learner. This can be shown by induction as follows. When the first datum different from # is observed by the learner, it adds $(U_{i_1} \circ \beta_{i_1})$ in the candidate set, for some $i_1 \in \{1, 2, \ldots, h\}$. Now, suppose the learner has placed $(U_{i_1} \circ \beta_{i_1}) \cup (U_{i_2} \circ \beta_{i_2}) \cup \ldots \cup (U_{i_k} \circ \beta_{i_k})$ 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 $(U_{i_1} \circ \beta_{i_1}) \cup (U_{i_2} \circ \beta_{i_2}) \ldots \cup (U_{i_k} \circ \beta_{i_k}) \cup (U_{i_{k+1}} \circ \beta_{i_{k+1}})$ to the candidate set where $\{i_1, i_2, \ldots, i_k\} \subset \{i_1, i_2, \ldots, i_k, i_{k+1}\} \subseteq \{1, 2, \ldots, h\}$. This happens at the stage when the learner is first presented an element $w \in L - (U_{i_1}, \beta_{i_1}) \cup (U_{i_2}, \beta_{i_2}) \cup \ldots \cup (U_{i_k}, \beta_{i_k})$. Thus, by induction the learner will eventually put $(U_1 \circ \beta_1) \cup (U_2 \circ \beta_2) \cup \ldots \cup (U_h \circ \beta_h)$, in the candidate set. As $(U_1, \beta_1) \cup (U_2, \beta_2) \cup \ldots \cup (U_h, \beta_h)$ is never dropped from the candidate set, eventually the learner only outputs the length-lexicographically least correct conjecture (among the conjectures that it has in its candidate set). Thus, the learner **Ex**-learns L. As L was an arbitrary member of $\mathcal{Q}_n(\mathcal{R}, H')$, it follows that the learner **Ex**-learns $\mathcal{Q}_n(\mathcal{R}, H')$. □

Proposition 9. *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 $\mathbf{Pat}_n(G) \subseteq \mathcal{Q}_{n'}(\mathcal{R}, H')$.*

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 [12]); furthermore the domain of this function is regular, as a finite automaton can determine whether an index of a member of $\mathcal{L}$ is for a language in $\mathcal{L}'$. Thus Proposition 8 and Proposition 9 give the following theorem.

Theorem 10. *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.*

5 Automatic Learning of all Patterns

If $(G, \circ)$ is Abelian automatic group, then, in some cases, the class of all pattern languages is learnable by an automatic learner. For this, sometimes a non-automatic family must be considered as the hypothesis space, as the class of pattern languages may not always form an automatic family.

Example 11. *The group $(\mathbb{Z}, +)$ does not have any automatic presentation $(A, +)$ in which there is an automatic family $\{A_i : i \in I\}$ of $\mathbf{Pat}(G)$.*

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

Proposition 12. *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, α can be chosen as any member of L.*

Proposition 13. *Suppose L is a pattern language over a finitely generated free commutative group and β_1, β_2 are two distinct members of L. Then, effectively from β_1 and β_2, a finite set of patterns can be found, one of which generates L.*

In the case that a group is finite, the class of its pattern languages is obviously **Ex**-learnable simply by maintaining a list of all elements observed. So for the following result, only the case where the group is infinite is interesting.

Theorem 14. *Let $(G, \circ)$ be a finitely generated Abelian automatic group. Then the class $\mathbf{Pat}(G)$ of its pattern languages has, in a suitable representation of the group, an automatic $\mathbf{Ex}$-learner which uses a suitable automatic hypothesis space.*

Proof. If the group $(G, \circ)$ is finite then there is clearly an automatic **Ex**-learner for $\mathbf{Pat}(G)$. 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 in reverse binary and also has an automatic ordering $<$ for its elements; the second coordinate (from A) is represented in any automatic way. The group element is represented as a convolution of the representation of the respective coordinates. The learner keeps in memory:

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

Note that all data (z'', a') with $a' \neq a$ can be ignored. 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, it is easy to verify that the above learner **Ex**-learns $(Z,+)\times(A,\bullet)$. 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.

Remark 15. Note that the learning algorithm of Proposition 14 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 is not of this form. Furthermore, for some Abelian groups like $(\{\frac{m}{n} : n>0$ and n does not contain any prime factor twice$\},+)$ the class of pattern languages is not learnable. However, it is unknown whether this group is automatic; most likely, this group is not automatic.

Theorem 16. *There exists an automatic group G generated by two elements such that* $\mathbf{Pat}(G)$ *is not* **Ex***-learnable.*

Proof. Consider the group with two generators a,b and the following equations for the 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

- $L(x_1 a x_1^{-1} a) = \{b^{2i} : i\in\mathbb{Z}\}$;
- $L(bx_1 b^{-1} x_1^{-1}) = \{\varepsilon, b^2\}$;
- $L_i = L(b^{-2i}\circ(bx_1b^{-1}x_1^{-1})\circ\ldots\circ(bx_{2i}b^{-1}x_{2i}^{-1})) = \{b^{-2i}, b^{-2i+2},\ldots,b^{-2},\varepsilon, b^2,\ldots,b^{2i-2},b^{2i}\}$.

It is easy to see that $L_1\subset L_2\subset L_3\subset\ldots$ and $\bigcup_{i\in\mathbb{N}} L_i = L(x_1ax_1^{-1}a)$. Thus, $\mathbf{Pat}(G)$ is not **Ex**-learnable by Lemma 4. □

The class of verbal languages over the group used in the above theorem is **Ex**-learnable, however for some group G, the class of verbal languages is not **Ex**-learnable.

Proposition 17. *There is a finitely generated automatic group where the class of verbal languages is not* **Ex***-learnable.*

6 Learning Bounded Unions of Patterns

Recall that the Prüfer group is the group of all rationals of the form $m/2^n$ with $0\le m<2^n$ and the rule that when $x+y\ge 1$ for group elements, one identifies this number with $x+y-1$. In the Prüfer group, all patterns are either singletons or the full group; thus the bounded unions of the 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.

Theorem 18. *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* **Ex***-learner for any automatic representation of the group.*

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

Theorem 19. *Suppose G is a finitely generated free commutative group. Then $\mathbf{Pat}^k(G)$ is* **Ex***-learnable.*

Proof. Let Σ be the finite set of generators for G. Suppose T is a text for $L \in \mathbf{Pat}^k(G)$. Let P_G be a set of at most k patterns such that $L = \bigcup_{\pi \in P_G} L(\pi)$. Without loss of generality assume that the number of patterns in P_G generating at least two elements is minimised. The learner $\mathbf{M}$ keeps the following memory:

(a) the full input sequence $T[n]$ seen so far and
(b) a finite labeled tree; 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 k.

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 in the simple path from the root to z (excluding the root, but including z if it is not the root).

Initially, the tree in the memory of the learner $\mathbf{M}$ has only one node, the root. On input $T(n)$, the learner 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 $r < k$, where the root is at depth 0, and $\text{card}(X_z) \geq k - r + 1$, then using Proposition 13 find the finite number of possible patterns which contain a pattern equivalent to any 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 a finite branching degree and only leaf nodes can be expanded, the tree stabilizes as n goes to infinity. The learner's conjecture 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 $\text{card}(X_z) \leq k - r$. Otherwise, $Y_z = \Sigma^*$ (in this later case, Y_z can be considered as spoiled).
(C) Conjecture minimal Y_z where z is a leaf of the tree (minimal is taken subset wise, as observed for strings in $\Sigma^0 \cup \Sigma^1 \cup \ldots \cup \Sigma^n$). Here minimal language Y_z is not computable, so the learner just considers each language Y_z on strings (over Σ) of length at most n, and then chooses the minimal based on this. In case of multiple such minimal Y_z, choose the z which is leftmost in the tree. Note that the set of at most k patterns generating the language Y_z can be correspondingly computed.

To see that **M Ex**-learns $\mathbf{Pat}^k(G)$, note that, by induction, for each n, after seeing input $T[n+1]$, for each leaf z in the tree, for r, Y_z, X_z as defined in (B):

(i) content$(T[n+1]) \subseteq Y_z$;
(ii) if card$(X_z) > k - r$, then the depth of z is k (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' (excluding the root) is labeled only using patterns equivalent to some patterns in P_G.

Thus, for large enough n, for z' as given by item (iv) above, $Y_{z'} \subseteq L \subseteq Y_{z'}$. It follows that, for large enough n, the conjecture of **M** is L. □

As every finitely generated Abelian group has a normal divisor of the form $(\mathbb{Z}^k, +)$ of finite index, the following theorem can be shown.

Theorem 20. *Consider any finitely generated commutative group (not necessarily free commutative group) G. Then* $\mathbf{Pat}^k(G)$ *is* **Ex***-learnable using some representation of the elements of the group.*

7 Conclusions

In this paper the learnability of pattern languages over groups was studied. 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 occurrence 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 (and their bounded unions) is **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.

Wiehagen [29] called a learner *iterative* if its memory is identical to the most recent hypothesis. Proposition 8, Theorem 10, Theorem 14, Theorem 19 and Theorem 20 can be shown using iterative learners which use class-preserving hypothesis spaces [18]. It is an open question whether all **Ex**-learnable classes of pattern languages can be learnt iteratively.

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. Achim Blumensath. *Automatic structures.* Diploma thesis, RWTH Aachen, 1999.
5. 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.
6. 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.
7. 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.
8. John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
9. E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
10. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.
11. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
12. 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 Chitat on his sixtieth birthday, pages 94–113, World Scientific, 2012.
13. Olga Kharlampovich and Alexei Myasnikov. Elementary theory of free non-Abelian groups. *Journal of Algebra*, 302(2):451–552, 2006.
14. Olga Kharlampovich and Alexei Myasnikov. Definable subsets in a hyperbolic group. *International Journal of Algebra and Computation*, 23(1):91–110, 2013.
15. Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. *Proceedings of Logic Colloquium 2007. Lecture Notes in Logic*, 35:132–176, 2010.
16. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity* (International Workshop LCC 1994). Springer LNCS 960:367–392, 1995.
17. Steffen Lange and Rolf Wiehagen. Polynomial time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.
18. Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53:88–103, 1996.
19. Alexei Myasnikov and Vitaly Romankov. On rationality of verbal subsets in a group. *Theory of Computing Systems*, 52(4):587–598, 2013.
20. André Nies. Describing groups. *Bulletin of Symbolic Logic*, 13:305–339, 2007.
21. André Nies and Richard M. Thomas. FA-presentable groups and rings. *Journal of Algebra*, 320:569–585, 2008.
22. 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.
23. 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.
24. 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.
25. Daniel Reidenbach. A non-learnable class of E-pattern languages. *Theoretical Computer Science*, 350:91–102, 2006.

26. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.
27. 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.
28. Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.
29. Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)*, 12(1–2):93–99, 1976.