Automatic Learning of Subclasses of Pattern Languages

John Case[1], Sanjay Jain[⋆,2], Trong Dao Le[2],
Yuh Shin Ong[2], Pavel Semukhin[⋆⋆,3] and Frank Stephan[⋆⋆⋆,2,4]

[1] Department of Computer and Information Sciences,
University of Delaware, Newark, DE 19716-2586, USA.
case@cis.udel.edu
[2] Department of Computer Science, National University of Singapore,
Singapore 117417, Republic of Singapore.
sanjay@comp.nus.edu.sg, daole@nus.edu.sg and yuhshin@gmail.com
[3] Department of Computer Science, University of Regina, Canada.
pavel@semukhin.name
[4] Department of Mathematics, National University of Singapore,
Singapore 119076, Republic of Singapore.
fstephan@comp.nus.edu.sg

Abstract. Automatic classes are classes of languages for which a finite automaton can decide the membership problem for the languages in the class, in a uniform way, given an index for the language. For alphabet size of at least 4, every automatic class of erasing pattern languages is contained, for some constant n, in the class of all languages generated by patterns which contain (1) every variable only once and (2) at most n symbols after the first occurrence of a variable. It is shown that such a class is automatically learnable using a learner with the length of the long-term memory being bounded by the length of the first example seen. The study is extended to show the learnability of related classes such as the class of unions of two pattern languages of the above type.

1 Introduction

The present work carries on investigations of learnability properties in connection with automatic structures. The underlying model of learnability is inductive inference [1, 12, 20, 29]. Additionally, (1) the target class of languages for learning is an automatic family [14–16, 18, 19, 22], that is, membership problem for the class to be learnt can be recognised by a finite automaton in a uniform way, and (2) the learner itself is automatic [17]. These learners are given by a function, where in each step/round, the learner outputs a hypothesis and updates its long term memory based on its previous memory and a current input. This function is required to be regular, that is, it must be recognised by a finite automaton. Such learners may be considered to be more realistic than learners which have access to all past data. Another motivation for the work goes

⋆ Supported in part by NUS grants C252-000-087-001 and R252-000-420-112.
⋆⋆ Supported by NUS grant R146-000-114-112.
⋆⋆⋆ Supported in part by NUS grants R146-000-114-112 and R252-000-420-112.

back to the programme of Khoussainov and Nerode [22] to establish an automata theoretic counterpart of recursive model theory; so one might view the current line of research also as an automata theoretic counterpart of standard inductive inference.

Learners with explicit bounds on the long term memory have already been studied previously in the general setting of algorithmic learners, see [11, 23]. Such learners are often modeled by a device having a long term memory which is updated in each round. In each round, the computation of the learner depends only on the previous value of the long term memory and the current datum from the input. The update function is required to be recursive. In the current paper, we consider learners for which the update function of the learner is automatic [17]. Such learners can learn, for example, the class of all closed intervals $\{x \in \Sigma^* : \alpha \sqsubseteq x \sqsubseteq \beta\}$ with respect to an automatic linear order $\sqsubset$ and the class $\{\alpha\Sigma^* : \alpha \in \Sigma^*\}$ of all languages which are the set of extensions of a fixed string. On the other hand, automatic learners are severely memory restricted (due to the mechanism involved). For an alphabet Σ with at least two symbols, automatic learners fail to learn classes like $\{\Sigma^* - \{\alpha\} : \alpha \in \Sigma^*\}$, as they cannot keep track of all the data they have seen so far. All classes given in these examples are represented by an automatic family [18], that is, a class where the membership relation is uniformly decided by an automatic function taking an index and a word as an input.

We will mainly be concentrating on learning subclasses of pattern languages [2] and related classes which are automatic. Angluin initiated the study of *pattern languages* [1, 2] in learning theory; here a pattern π is a string of variables and constants; the language generated by π is the set of all words which can be obtained by replacing variables in the pattern π by non-empty strings. As an example, consider

$$\pi = 01xy200zx1;$$

the variables (for substitutions) are x, y, z and the constants/terminals are $0, 1, 2$. The word 01220020011221 is generated by the pattern π by letting $x = 22$, $y = 00$ and $z = 11$.

Shinohara [34] introduced the concept of *erasing pattern languages* in which the variables are allowed to be substituted by empty strings; we follow this approach and consider all pattern languages as erasing in our paper. Shinohara [34] also introduced the concept of regular patterns, in which each variable occurring, occurs only once. The *regular pattern languages* are those languages which are generated by regular patterns. These language classes have been well-studied and found various applications. In the present work, we mainly focus on automatic classes of pattern languages like $\mathcal{P}_n$ which consists of all languages generated by a regular pattern whose variables occur only among the last n symbols of the pattern. Furthermore, we study natural variants like classes containing the unions of two members of a fixed $\mathcal{P}_n$ or patterns which permit not only variables for strings but also variables for single symbols. The classes $\mathcal{P}_n$ are quite representative for the automatic learning of patterns as every automatic family of regular pattern languages is contained in one such class $\mathcal{P}_n$ (see [18]); so the main focus of the current paper is to continue the study of the learning power of automatic learners [16–18] for the automata-theoretic counterparts of the well-studied and natural classes of pattern languages as well as classes which consist of the unions of two pattern languages.

We summarise the organisation of our paper. Section 2 below gives the preliminaries related to the model (for both learning and automatic classes) used in this paper. Section 3 deals with the learnability properties of certain concrete classes, namely various interesting automatic classes of pattern languages. In particular, we show that each class $\mathcal{P}_n$ is learnable by an automatic learner where the long term memory is bounded in length by the length of the longest word seen in the input. In Section 4, we investigate the learnability of related classes which contain the unions of two members of $\mathcal{P}_n$. We show that such a class is learnable if either all unions are disjoint or the alphabet size is at least three. Section 5 deals with automatic learnability of character pattern languages, where the variables are allowed to be replaced only by one character.

2 The Model

The set of natural numbers is denoted by $\mathbb{N}$. The symbol Σ denotes a finite alphabet. The empty string is denoted by ε. We let Σ^i denote the set of all strings of length i over the alphabet Σ. We let $u \cdot v$, or simply uv denote the concatenation of the strings u and v. The length of a string x is denoted by $|x|$. A string of length n over Σ will be treated as a function from the set $\{0, 1, 2, \ldots, n-1\}$ to Σ. Thus, a string x of length n is the same as $x(0)x(1)x(2)\ldots x(n-1)$. For $m \leq |x|$, $x[m]$ denotes the prefix of x of length m, that is $x[m] = x(0)x(1)\cdots x(m-1)$. We let $x <_{ll} y$ denote that x is length-lexicographically before y, that is, $|x| < |y|$ or $|x| = |y|$ and x is lexicographically before y. We let $x \leq_{ll} y$ denote that $x <_{ll} y$ or $x = y$. Let $\mathit{lleast}(S)$ denote the length-lexicographically least string in the set S. We use $w \preceq w'$ to denote that w is a prefix of w', and $w \prec w'$ to denote that w is a proper prefix of w'. We say that the strings w, w' are left-consistent iff either $w \preceq w'$ or $w' \preceq w$. We say that the strings w, w' are right-consistent iff either w is a suffix of w' or w' is a suffix of w.

Cardinality of a set A is denoted by $\mathrm{card}(A)$. The symbols $\subseteq, \subset, \supseteq, \supset$ respectively denote subset, proper subset, superset and proper superset. We use $A \subseteq^* B$ to denote that $A - B$ is finite.

The convolution of two strings $x = x(0)x(1)\ldots x(m-1)$ and $y = y(0)y(1)\ldots y(n-1)$ is defined as follows. Let $r = \max(\{m, n\})$, $x' = x'(0)x'(1)\ldots x'(r-1)$ and $y' = y'(0)y'(1)\ldots y'(r-1)$, where (i) $x'(i) = x(i)$, if $i < m$, $x'(i) = \diamond$ otherwise, and (ii) $y'(i) = y(i)$, if $i < n$, $y'_i = \diamond$ otherwise. Intuitively, $\diamond$ is appended to the shorter string to make both the strings to be of the same length. Now, $\mathrm{conv}(x, y) = (x'(0), y'(0))(x'(1), y'(1))\ldots(x'(r-1), y'(r-1))$. Note that $\mathrm{conv}(x, y)$ is a string over the alphabet $((\Sigma \cup \{\diamond\}) \times (\Sigma \cup \{\diamond\}))^*$. Similarly, one can define conv on multiple arguments. A relation R or a function f is called *automatic* if the sets $\{\mathrm{conv}(x_1, x_2, \ldots, x_n) : R(x_1, x_2, \ldots, x_n)\}$ and $\{\mathrm{conv}(x_1, x_2, \ldots, x_m, y) : f(x_1, x_2, \ldots, x_m) = y\}$, respectively, are regular.

Intuitively, giving convolution of two strings represents giving the two strings in parallel to the automaton, one character of each string at a time. Note that giving the two inputs in parallel rather than serially is crucial as, for example, $\{\mathrm{conv}(0^n, 1^n) : n \in \mathbb{N}\}$ is regular, but $\{0^n 1^n : n \in \mathbb{N}\}$ is not. Thus, the function $f(0^n) = 1^n$ will be automatic while functions like $f(x) = 0^{2|x|}$ and $f(x) = xx$ are not automatic, as their graphs $\{\mathrm{conv}(x, 0^{2|x|}) : x \in \Sigma^*\}$ and

$\{\text{conv}(x, xx) : x \in \Sigma^*\}$ are not regular. Also the concatenation is not automatic, but it is possible to move a constant number of symbols around or to move all symbols by a constant distance. Therefore $f(x) = 0x$ is automatic; another example of an automatic function is the function exchanging the first and last symbol (of nonempty inputs). Forming the convolution is also an automatic function. A further example of an automatic function is to extract some symbol from a well-specified position; so $f(\text{conv}(x, y)) = y(|x|)$ is automatic, where $f(x, y)$ is taken to be some special symbol in the case that $|y| \leq |x|$. Some examples of automatic predicates from the prior literature include predicates to compare the length of strings, the lexicographic order and the length-lexicographic order. More information on automatic functions can be found in survey articles on automatic structures [21, 33].

A family of languages $\{L_\alpha : \alpha \in I\}$, where each $L_\alpha \subseteq D$, is said to be automatic iff D and I are regular sets (over some finite alphabet Σ and Γ respectively) and the set $\{\text{conv}(\alpha, x) : \alpha \in I \wedge x \in L_\alpha\}$ is regular. The sets D and I above are respectively called the domain and index domain for the automatic family. Usually, we will assume that $D = \Sigma^*$, for some finite alphabet Σ. An example of an automatic family is that of the closed intervals: $L_{\text{conv}(\alpha,\beta)} = \{x : \alpha \leq_{lex} x \leq_{lex} \beta\}$; however, the class of all regular languages is not contained in an automatic family [18].

An automatic structure is a structure (usually of a finite signature) whose domain, functions and relation are automatic. In a more general sense, a structure that is isomorphic to an automatic structure is also called automatic.

It can be shown that any family, function or relation which is first-order definable using automatic families and other automatic parameters, is again automatic [8, 22].

Fact 1 (Blumensath, Grädel [8], Khoussainov, Nerode [22]). *Any relation that is first-order definable from existing automatic relations is automatic.*

We will implicitly use the above fact in defining automatic learners. Properties such as decidability of first order theory make automatic structures a useful tool not only in learning theory but also in other areas such as model checking and Boolean algebras [7, 8, 22, 32, 33]. Moreover, though the class of all regular languages is learnable using queries [4], it is not learnable under the usual inductive inference criteria from positive data [1, 12]. Therefore, it is interesting to investigate which subclasses of regular languages are learnable from positive data and which are not. For example, Angluin [3] considered learnability of the class of k-reversible languages. These studies were later extended [10, 13]. In this context, it is useful to consider which automatic families are learnable and which not.[1]

The present work considers learning in the setting of automatic structures. The learning task (also called target class) is a class of languages, $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ over a domain $D \subseteq \Sigma^*$, where I is the index domain. The learner uses a hypothesis space $\mathcal{H} = \{H_\beta : \beta \in J\}$ to express its conjectures/hypotheses (here J is the index domain for the hypothesis space). For this paper both the target class as well as the hypothesis space are automatic families.

[1] As noted by Jain, Luo and Stephan [17], even the class of 0-reversible languages is not automatic; however, as mentioned in the abstract and as will be seen below, some very nice classes of regular languages are automatic classes and learnable automatically, that is, by learners which are given using finite automata.

A *text* T is a mapping from $\{0,1,2,\ldots\}$ to $D \cup \{\#\}$. Here the symbol $\# \notin \Sigma$ denotes pauses in the presentation of data. The content of a text T, denoted $content(T)$, is $range(T) - \{\#\}$. A text T is for a language L iff $content(T) = L$. We let σ range over initial segments of texts, and let $content(\sigma) = range(\sigma) - \{\#\}$.

Definition 2. (Based on Gold [12]) Suppose D is a regular domain (over some finite alphabet Σ) and I, J are regular index sets (over some finite alphabet).

Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is a target class and $\mathcal{H} = \{H_\beta : \beta \in J\}$ is a hypothesis space, which are both automatic families of languages with $L_\alpha, H_\beta \subseteq D$.

Suppose Δ is a finite alphabet (used for storing memory by learners) and ? is a special symbol not in $\Delta^* \cup J$.

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

(b) Suppose a learner M with initial memory mem_0 and initial hypothesis hyp_0 and a text T for a language L is given.
 (i) Let $mem_0^T = mem_0, hyp_0^T = hyp_0$.
 (ii) For $k > 0$, let $(mem_k^T, hyp_k^T) = M(mem_{k-1}^T, T(k-1))$.
 Note that the memory mem_k^T and hypothesis hyp_k^T of the learner depend only on the portion $T[k]$ of the input. We refer to mem_k^T, hyp_k^T as the memory and hypothesis of the learner M after having seen the input $T[k]$.
 (iii) We say that M converges on text T to a hypothesis β iff there exists a t such that $hyp_t^T = \beta$ and, for all $t' \geq t$, $hyp_{t'}^T \in \{\beta, ?\}$.
 (iv) We say that M learns the language L (using hypothesis space $\mathcal{H}$) from the text T iff M converges on text T to a hypothesis β such that $H_\beta = L$.

(c) We say that M learns a language L (using hypothesis space $\mathcal{H}$) iff M learns L from all texts for the language L (using hypothesis space $\mathcal{H}$).

(d) We say that M learns $\mathcal{L}$ (using hypothesis space $\mathcal{H}$) iff M learns all languages in $\mathcal{L}$ (using hypothesis space $\mathcal{H}$).

(e) A class $\mathcal{L}$ is said to be learnable iff some learner M learns $\mathcal{L}$ using some hypothesis space $\mathcal{H}'$.

Intuitively, in part (b) of the definition above, the learner is receiving, over time, one by one, the elements of the text $T(0), T(1), T(2), \ldots$ for the input language L. As it is receiving these inputs, it possibly updates its memory after each datum, and outputs a hypothesis/conjecture. The learner learns the input language from the text, if the sequence of its hypotheses converges to an index for the language L. A special symbol ? is used as a conjecture or memory by the learner. As a memory, ? denotes empty memory (which is different from memory being ε). As a conjecture, ? denotes that either the learner does not change its previous hypothesis (this is useful for some memory limited models of learner) or the learner has not yet seen enough data for its initial conjecture (this is useful in models of learning where the number of hypotheses output is relevant and false conjectures are penalised).

Sometimes, for ease of presentation, we just define learner M as acting over time on inputs $T(0), T(1), \ldots$, and updating its memory/conjecture as it receives more and more inputs. In such cases, we just define the initial memory/conjecture of the learner and say how it updates its memory (and outputs its conjecture) when it receives a new input. Furthermore, when the hypothesis space is clear from context, then we drop the reference to "(using hypothesis space $\mathcal{H}$)" in learnability.

Note that, for learning a class $\mathcal{L}$, the hypothesis space must contain the family $\mathcal{L}$ to be learnt. When we do not restrict the memory length or computational power of the learner, the above learning model is equivalent to Gold's model of inductive inference [12] (called explanatory learning or learning in the limit). Based on a result of Angluin [1] characterizing algorithmic learnability of *general* indexed classes, Proposition 3 below characterises the general *algorithmic* learnability of automatic classes.[2] Note that the version of Angluin's condition for automatic classes, as used in Proposition 3, can be checked explicitly for automatic families. Hence it is decidable whether an automatic family is learnable by an *algorithmic* learner or not. In what follows, for simplicity, the tell-tale condition will be referred to as Angluin's, although the simplifications stemming from the decidability of the first order theory of automatic classes are added in.[3]

Proposition 3 (Based on Angluin [1]). *An automatic family $\{L_\alpha : \alpha \in I\}$ is learnable by a recursive learner iff, for every $\alpha \in I$, there is a bound b_α such that, for all $\beta \in I$, the implication*

$$[\{x \in L_\alpha : |x| \leq b_\alpha\} \subseteq L_\beta \subseteq L_\alpha] \Rightarrow [L_\beta = L_\alpha]$$

holds.

One calls the set $\{x \in L_\alpha : |x| \leq b_\alpha\}$ above a tell-tale set for L_α, and the condition *Angluin's tell-tale condition.* Note that one can take $b_\alpha = |\alpha| + c$ for a suitable constant c (see [18]). This constant c depends on the family $\{L_\alpha : \alpha \in I\}$ but is independent of α.

Angluin's tell-tale condition solves the question of algorithmic learnability of automatic classes. Therefore, for learning automatic families, it is more interesting to consider automatic learners which have a superior run-time behaviour than usual learners as hypothesis and updated memory of automatic learners can be computed in time *linear in the length of the previous memory and current datum*; this is explained in the following remark.

Remark 4. Any automatic function f can be computed in linear time.

To see this, suppose f is an automatic function from Σ_1^* to Σ_2^*. Suppose the automaton which accepts $\{\mathrm{conv}(x, f(x)) : x \text{ in the domain of } f\}$ has Q as its set of states, q_0 as its starting state, δ as its transition function and F as its set of final states. As f is a function, we have that, for all x in the domain of f, $|f(x)| \leq |x| + |Q|$.

[2] Note that herein the focus will, nonetheless, remain primarily on the *automatic* learnability of automatic classes and not on their general algorithmic learnability.

[3] In the setting of general indexed classes, Angluin needed and employed a slightly more complicated condition where she required that the still finite tell-tale sets can be uniformly recursively enumerated; this condition would also work in Proposition 3.

On input x, below we describe how to compute $f(x)$ in time linear in $|x|$. Consider a directed graph G defined as follows. The vertex set $V(G)$ of G is $\{(q,i) : q \in Q, i \leq |x| + |Q|\}$. For $i < |x| + |Q|$, there is an edge from (q,i) to $(q', i+1)$, iff there exists a $b \in \Sigma_2 \cup \{\diamond\}$ such that $\delta(q, (x(i), b)) = q'$ (where we take $x(i)$ to be $\diamond$, for $i \geq |x|$). Let $S \subseteq V(G)$ be the set of nodes which are reachable from $(q_0, 0)$. Let, (q,j) be the unique node, if any, such that $(q,j) \in S$, $j \geq |x|$ and $q \in F$. Such a node, if any, is unique as f is a function and the automaton accepts the graph of f. Furthermore, as the automaton accepts $\text{conv}(x,y)$ iff $y = f(x)$, the path from $(q_0, 0)$ to (q,j) in the graph G is unique, and for $(q', k), (q'', k+1)$ in this path, there exists a unique b_k such that $\delta(q', (x(k), b_k)) = q''$. Now, $f(x) = b_0 b_1 \ldots b_{j-1}$.

One can compute the above $b_0 b_1 \ldots b_{j-1}$ in time linear in $|x|$ as follows. First note that one can define the graph $V(G)$, and find S and (q,j) as above in linear time. Let $s_j = q$. Now, inductively define s_k, b_k, for $k = j-1$ to $k = 0$ as follows. Let s_k be the unique state in Q such that, $(s_k, k) \in S$ and there is an edge from (s_k, k) to $(s_{k+1}, k+1)$, and b_k is the unique member of Σ_2 such that $\delta(s_k, (x(k), b_k)) = s_{k+1}$. The above computation can be done in constant time for each $k < j$, and thus one can compute $f(x)$ in time linear in $|x|$.

Definition 5. Suppose a learner M with initial memory mem_0 and initial hypothesis hyp_0 is given. Suppose $\mathcal{H} = \{H_\beta : \beta \in J\}$ is a hypothesis space, which is an automatic family.

For a text T: let $mem_0^T = mem_0, hyp_0^T = hyp_0$, and for $k > 0$, let $(mem_k^T, hyp_k^T) = M(mem_{k-1}^T, T(k-1))$.

(a) [17] A learner M is called an *automatic learner* iff its graph is automatic. That is, M is automatic iff $\{\text{conv}(mem, x, mem', hyp') : M(mem, x) = (mem', hyp')\}$ is regular.
(b) [36] M is said to be *iterative* iff for all texts T, for all t, $mem_t^T = hyp_t^T$.
(c) [5] M is said to be *consistent* iff for all texts T, for all t, $content(T[t]) \subseteq H_{hyp_t^T}$.
(d) [29] M is said to be *confident* iff for all texts T, either all of $hyp_0^T, hyp_1^T, \ldots$ are ? or the sequence $hyp_0^T, hyp_1^T, \ldots$ converges to some hypothesis β (in the sense of Definition 2 (b) (iii)).

Note that the above constraints are required even for input texts for a language outside the class to be learnt.

Automatic learners cannot memorise all data they observe; hence the learner can no longer access the full past history of the data seen so far. Thus, in general, the requirement of a learner to be automatic is a real restriction and learners cannot be made automatic by just applying Pitt's delaying technique [30].

Long term memory limitations were first introduced by Freivalds, Kinber and Smith [11]. The variations of long term memory in the context of automatic learners were considered by Jain, Luo and Stephan [17].

Suppose T is the input text, and the memory and hypothesis of the learner after having seen the input $T[t]$ are respectively, mem_t and hyp_t. The length-restriction for memory we often consider is: length of the memory is bounded by the length of the longest datum seen so far plus a constant, that is for some constant c, for all text T and $t \in \mathbb{N}$, $|mem_t^T| \leq max(\{|T(s)| : s < t\}) + c$; we often refer to such memory bounded learners as word length memory bounded. For the ease of notation, the "plus a constant" is omitted in the notations below. Note that the learner is not constrained regarding which alphabet it uses for its memory. Therefore, the learner might, for example, store the convolution of up to some constant number of examples (in case the memory does not exceed the allowed bound). Note that, in the case that memory is unbounded or the bound allows storage of the hypothesis, then the learner can memorise the most recent hypothesis output, and, thus, abstain from outputting ?.

For many learning paradigms of automatic learning, one can choose the hypothesis space $\mathcal{H}$ to be the same as $\mathcal{L}$. However, when the amount of the memory allowed to the learner depends on the length of the hypothesis or when the long term memory of the learner has to be the most recent hypothesis, as in the case of iterative learning, this requirement may be a restriction. The main reason for hypothesis space not to be critical in many cases is that one can automatically convert the indices from one automatic family to another for the languages which are common to both automatic families. Only in the case of iterative learning and bounds given by the length of the hypothesis, it is often important to have the ability to store some additional information into the hypothesis — which is impossible in the case of a one-one hypothesis space. For example, Theorem 9 requires a special class preserving hypothesis space, if one considers iterative learning. Here a hypothesis space $\{H_\beta : \beta \in J\}$ is called class preserving (class comprising) [26] for the target class $\{L_\alpha : \alpha \in I\}$, if $\{L_\alpha : \alpha \in I\} = \{H_\beta : \beta \in J\}$ (respectively, $\{L_\alpha : \alpha \in I\} \subseteq \{H_\beta : \beta \in J\}$).

Note that, in contrast, hypothesis spaces do matter for learning general indexed families by recursive learners (satisfying various properties) [26, 27].

3 Automatic Classes of Pattern Languages

Learning theorists have studied the learnability of the class of pattern languages extensively [2, 9, 25, 31, 34]. Although the full generality of pattern languages cannot be brought over into an automatic setting, there are still rich automatic classes of pattern languages which deserve to be investigated [18].

Definition 6. [2, 34] Let Σ be a finite alphabet and V be a set of variables, disjoint from Σ.

(a) A *pattern* is any string over $(\Sigma \cup V)^*$.
(b) A *substitution* θ is a homomorphism from the set of patterns to the set of patterns that maps each $a \in \Sigma$ to a. The image of pattern π under the substitution θ is denoted by $\pi\theta$.
(c) The language associated with a pattern π, denoted by $Lang(\pi)$, is the set $\{\pi\theta : \theta \text{ is a substitution and } \pi\theta \in \Sigma^*\}$.
(d) A pattern π is called a *regular pattern* iff each variable appearing in π appears exactly once. If π is a regular pattern, then $Lang(\pi)$ is called a *regular pattern language*.

Regarding part (c), there are two cases which have been considered in the literature: In the case of an erasing pattern language [34] one permits substitutions that map variables to the empty string ε; in the case of a non-erasing pattern language [2], the substitutions must map each variable to a non-empty string. In the present work, a "pattern language" is by default an "erasing pattern language", that is the substitutions of variables are allowed to be ε.

Example 7. (a) Suppose Σ is a finite alphabet. The class of all sets $L_\alpha = \alpha\Sigma^*$, where $\alpha \in \Sigma^*$, consists of all regular erasing pattern languages generated by patterns of the form αx; this class is iteratively learnable by an automatic learner. The hypothesis space used by the learner is $\{H_\alpha : \alpha \in \Sigma^* \cup \{\text{emp}\}\}$, where emp $\notin \Sigma^*$, $H_{\text{emp}} = \emptyset$, and $H_\alpha = L_\alpha$, for $\alpha \in \Sigma^*$.

The initial memory of the learner is ?, and initial conjecture of the learner is emp. For $\alpha, w \in \Sigma^*$, $M(?, \#) = (?, \text{emp})$; $M(?, w) = (w, w)$; $M(\alpha, \#) = (\alpha, \alpha)$, and $M(\alpha, w) = (w', w')$, where w' is the longest common prefix of α and w.

It is easy to verify that M is automatic.

Now consider any text T for a language L. Let the memory and the hypothesis of the learner M after having seen the input $T[k]$ be mem_k^T and hyp_k^T respectively. Now, for any k, the following two conditions hold:

(i) if $content(T[k]) = \emptyset$, then $mem_k^T = ?$ and $hyp_k^T = \text{emp}$;
(ii) if $content(T[k]) \neq \emptyset$, then mem_k^T ($= hyp_k^T$) is the longest common prefix of all strings in $content(T[k])$.

Thus, M is a consistent learner and converges on any text T for a non-empty language to the hypothesis α such that α is the longest common prefix of all strings in $content(T)$. Note that M converges on the text $\#^\infty$ to emp. Thus, M is a confident learner and it learns each language in $\mathcal{L}$.

(b) Let $L_\alpha = \alpha\Sigma^*01\Sigma^*$, where $\alpha \in \Sigma^*$, $0 \in \Sigma$, $1 \in \Sigma$ and $I = \Sigma^*$. Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$. Then, $\mathcal{L}$ is an automatic class of erasing pattern languages, where $L_\alpha = Lang(\alpha x 01 y)$. $\mathcal{L}$ is learnable by the learner M described in part (a) using hypothesis space $\{H_\alpha : \alpha \in \Sigma^* \cup \{\text{emp}\}\}$, where $H_\alpha = L_\alpha$, for $\alpha \in \Sigma^*$, and $H_{\text{emp}} = \emptyset$. This learner is confident, though not consistent: for example, if there are inputs such as 111, then the conjecture of M would not contain it. One can however modify the learner to make it consistent (using a different hypothesis space).

Shinohara [34] considered the class of languages which are generated by regular patterns, that is, patterns in which the variables do not repeat. In this paper we consider some subclasses of regular pattern languages. For ease of notation, when considering regular pattern languages, we use only one variable symbol @. It is to be understood that each occurrence of @ in the pattern represents a different variable.

Definition 8. Fix $n \in \{1, 2, 3, \ldots\}$ and an alphabet Σ.

(a) Let G_n be the union of $\{\varepsilon\}$ and all $@ \cdot (\Sigma \cup \{@\})^m$ with $m < n$. That is G_n represents the set of all regular patterns which are either ε or start with a variable and are of length at most n. Note that G_n is finite and $\Sigma^* \in G_n$.

(b) $\mathcal{P}_n$ denotes the class of pattern languages which can be generated by a regular pattern where variables, if any, in the pattern only appear within the last n symbols of the pattern. That is, $\mathcal{P}_n = \{Lang(u \cdot v) : u \in \Sigma^*, v \in G_n\}$.

For example, the pattern 010232012012@12@1@ generates a language in $\mathcal{P}_6$. Jain, Ong, Pu and Stephan [18] showed that every class $\mathcal{P}_n$ can be given as an automatic family. Furthermore, every automatic class of languages generated by regular patterns is a subclass of some $\mathcal{P}_n$.

Theorem 9. *For all $n > 0$, $\mathcal{P}_n$ has an automatic learner which is consistent, confident and word length memory bounded; in fact the length of the memory of the learner is even bounded by the length of the first datum seen plus one. This learner can also be made iterative.*

Proof. The memory of the learner is either ? or of the form $\mathrm{conv}(x, \alpha)$, where $x \in \Sigma^*$ and $|\alpha| = |x| + 1$. Here the alphabet set used for α is $\{X : X \subseteq G_n\}$. Note that G_n is finite and thus the alphabet set used for α is also finite.

The hypothesis space used by the learner is $\mathcal{H} = \{H_\beta : \beta \in \{\mathrm{emp}\} \cup \{\mathrm{conv}(x, \alpha) : x \in \Sigma^*, \alpha \in (\{X : X \subseteq G_n\})^*, |\alpha| = |x| + 1\}\}$, where $H_{\mathrm{emp}} = \emptyset$ and $H_{\mathrm{conv}(x,\alpha)} = \mathit{Lang}(\pi')$ for the length-lexicographically least element π' of $A = \{y \cdot \pi : y \text{ is a prefix of } x \text{ and } \pi \in \alpha(|y|)\}$ such that there is no $\pi'' \in A$ with $\mathit{Lang}(\pi'') \subset \mathit{Lang}(\pi')$. Note that π' can be defined using first order formula over automatic relations as follows:

$$\begin{aligned}
&(\exists 0^\ell \leq_{ll} x)\ (\exists \tau \in \alpha(\ell))\ [[\pi' = x[\ell]\tau] \text{ and} \\
&\quad (\forall 0^{\ell'} \leq_{ll} x)\ (\forall \tau' \in \alpha(\ell'))\ [\mathit{Lang}(x[\ell']\tau') \not\subset \mathit{Lang}(x[\ell]\tau)] \text{ and} \\
&\quad (\forall 0^{\ell'} \leq_{ll} x)\ (\forall \tau' \in \alpha(\ell'))\ [x[\ell]\tau \leq_{ll} x[\ell']\tau' \text{ or} \\
&\qquad (\exists 0^{\ell''} \leq_{ll} x)\ (\exists \tau'' \in \alpha[\ell''])\ [\mathit{Lang}(x[\ell'']\tau'') \subset \mathit{Lang}(x[\ell']\tau')]]]].
\end{aligned}$$

Here note that $x[\ell]\tau$ is the string obtained by the concatenation of the prefix of length ℓ of x with τ. Furthermore, a relation such as $\mathit{Lang}(\pi_1) \subseteq \mathit{Lang}(\pi_2)$ is automatic, as it can be given by the first order formula $(\forall w)\,[w \in \mathit{Lang}(\pi_1) \Rightarrow w \in \mathit{Lang}(\pi_2)]$. Thus, $\mathcal{H}$ is an automatic family.

The learner M is defined as follows, where $x, w \in \Sigma^*$ and $\alpha \in \{X : X \subseteq G_n\}^*$.

- The learner M has initial memory ? and initial conjecture emp. The hypothesis of the learner will always be linked to its memory, that is, if *mem* is the memory of the learner after seeing input $T[k]$, then its hypothesis after seeing $T[k]$ will be $\mathit{conj}(\mathit{mem})$, where $\mathit{conj}(?) = \mathrm{emp}$ and $\mathit{conj}(\mathit{mem}) = \mathit{mem}$, for $\mathit{mem} \neq ?$.
- The learner M does not change its memory/conjecture on input #. That is, $M(\mathit{mem}, \#) = (\mathit{mem}, \mathit{conj}(\mathit{mem}))$.
- $M(?, x) = (\mathrm{conv}(x, \alpha), \mathit{conj}(\mathrm{conv}(x, \alpha)))$, where, for each prefix y of x, $\alpha(|y|)$ is the set of $\pi \in G_n$ such that $x \in \mathit{Lang}(y \cdot \pi)$. Note that this computation is automatic, as it is given by the formula: $(\forall 0^\ell \leq_{ll} x)\,(\forall \pi \in G_n)\,[\pi \in \alpha(\ell) \text{ iff } x \in \mathit{Lang}(x[\ell]\pi)]$.
- $M(\mathrm{conv}(x, \alpha), w) = (\mathrm{conv}(x, \alpha'), \mathit{conj}(\mathrm{conv}(x, \alpha')))$, where, for each prefix y of x, $\alpha'(|y|)$ is the set of all π in $\alpha(|y|)$ such that $w \in \mathit{Lang}(y \cdot \pi)$. Note that this computation is automatic, as it is given by the formula: $(\forall 0^\ell \leq_{ll} x)\,(\forall \pi \in G_n)\,[\pi \in \alpha'(\ell) \text{ iff } [\pi \in \alpha(\ell) \text{ and } w \in \mathit{Lang}(x[\ell]\pi)]]$.

As M above is first order definable using automatic functions/relations, M is automatic. Suppose T is the input text, and let mem_k^T and hyp_k^T denote the memory and hypothesis of M after having seen the input $T[k]$. If $\mathit{mem}_k^T \neq ?$, then let x_k^T, α_k^T be such that $\mathit{mem}_k^T = (x_k^T, \alpha_k^T)$. Note the following properties:

(P1) The length of the memory of the learner is bounded by the length of the first datum seen (plus 1).

(P2) If $\mathit{content}(T[k]) \neq \emptyset$, then $\mathit{mem}_k^T \neq ?$ and x_k^T is the first datum different from # in $T[k]$ and, for each prefix y of x_k^T, $\alpha(|y|)$ consists of all $\pi \in G_n$ such that $\mathit{content}(T[k]) \subseteq \mathit{Lang}(y \cdot \pi)$.

(P3) From (P2) it follows that if $mem_t^T \neq ?$, then for all prefixes y of x_k^T, $\alpha_{t+1}^T(|y|) \subseteq \alpha_t^T(|y|)$. Thus the memory sequence (and hence hypothesis sequence) of M on text T converges. Thus, M is confident.

(P4) From (P2) it also follows that $Lang(conj(mem_k^T))$ is a minimal language (generated by the length lexicographically smallest pattern, in case of several such minimal languages) in $\mathcal{P}_n$ which contains $content(T[k])$. Hence, the learner M is consistent and learns $\mathcal{P}_n$.

Note that the above learner can be easily made iterative, as the hypothesis space chosen for this learner is such that the memory of the learner can be obtained from the hypothesis used in this algorithm. □

From now on, for ease of presentation, we will not explicitly give the first order formulas as in the above theorem.

4 Automatic Classes of the Unions of Two Pattern Languages

We now generalise the techniques from Section 3 in order to learn the unions of pattern languages. Our main results are that the automatic class of disjoint unions of two members from $\mathcal{P}_n$ is automatically learnable (Theorem 15) and that also, for an alphabet size of at least three, the class of arbitrary unions of two members from $\mathcal{P}_n$ is automatically learnable (Theorem 21).

Proposition 10. *Let $n > 0$ and an automatic hypothesis space $\mathcal{H} = \{H_\beta : \beta \in J\}$ be given. Suppose that the automatic learners $M_1, M_2, \ldots, M_n$ are consistent and confident. Then, there exists another automatic learner N which is (1) consistent, (2) confident and (3) converges on a text T for a language L to an index for L whenever at least one of the learners $M_1, M_2, \ldots, M_n$ converges on T to an index for L. Furthermore, if the learners $M_1, M_2, \ldots, M_n$ are word length memory bounded, then so is N.*

Proof. The new learner N maintains as long term memory the convolution of the memories of $M_1, M_2, \ldots, M_n$. If $M_1, M_2, \ldots, M_n$ conjecture hypothesis $\beta_1, \beta_2, \ldots, \beta_n$, then N conjectures β_i for the least i such that there is no j with $H_{\beta_j} \subset H_{\beta_i}$. As $M_1, M_2, \ldots, M_n$ are automatic, consistent and confident, so is N.

Now consider any given text T for some language L. The learners $M_1, M_2, \ldots, M_n$ converge on a text T to hypotheses $\beta_1, \beta_2, \ldots, \beta_n$, such that $H_{\beta_j} \supseteq L$ for all $j \in \{1, 2, \ldots, n\}$. Now if $i = \min(\{j \in \{1, 2, \ldots, n\} : H_{\beta_j} = L\})$ exists, then N converges on T to β_i, as $H_{\beta_i} = L \subseteq H_{\beta_j}$ for all $j \in \{1, 2, \ldots, n\}$ and $L \subset H_{\beta_j}$ for all $j < i$. □

Proposition 11. *For all $n > 0$ and $\pi \in G_n$, $\mathcal{L} = \{Lang(u\cdot\pi) \cup \{z\} : u, z \in \Sigma^*, z \notin Lang(u\cdot\pi)\}$ is consistently and confidently learnable by an automatic learner which is word length memory bounded.*

Proof. As for $\pi = \varepsilon$, the proposition clearly holds, assume $\pi \neq \varepsilon$. We will define two learners. Both of them are confident and consistent and at least one of them will succeed on any given text for a language $Lang(u \cdot \pi) \cup \{z\}$, where $u, z \in \Sigma^*$, $\pi \in G_n - \{\varepsilon\}$, $z \notin Lang(u \cdot \pi)$. The proposition then follows using Proposition 10. Fix these parameters u and z from now onwards.

The first learner M_1 works in the case that z is the first datum different from $\#$ in the input text. The hypothesis space used by M_1 is $\mathcal{H} = \{H_{\mathrm{conv}(w,x,v)} : w, x, v \in \Sigma^* \cup \{\#\}\}$, where, for $w, x, v \in \Sigma^*$, $H_{\mathrm{conv}(\#,\#,\#)} = H_{\mathrm{conv}(w,x,\#)} = \Sigma^*$, $H_{\mathrm{conv}(w,\#,\#)} = \{w\}$, $H_{\mathrm{conv}(w,x,v)} = Lang(v \cdot \pi) \cup \{w\}$; hypotheses different from the above are not used and can be assumed to represent Σ^*.

The memory of M_1 is of the form $\mathrm{conv}(w, x, v)$, where $w, x, v \in \Sigma^* \cup \{\#\}$. If mem is the memory of M_1 after having seen input $T[k]$, then its hypothesis after having seen input $T[k]$ is also mem. We thus just describe below the memory update of M_1 (ignoring its hypothesis).

Intuitively, w is the first datum different from $\#$ that M_1 receives, $x \neq w$ is the second such datum and v is the longest prefix of x such that all data received belong to $Lang(v \cdot \pi) \cup \{w\}$. The values of $\#$ for w, x are used to denote unknown values of w, x. When w, x are not $\#$, $v = \#$ is used to denote that there is no v such that all input data belong to $Lang(v \cdot \pi) \cup \{w\}$. Formally, initial memory of M_1 is $\mathrm{conv}(\#, \#, \#)$. M_1 does not change its memory on input $\#$. Below, let $w, x, v, y \in \Sigma^*$.

- $M_1(\mathrm{conv}(\#, \#, \#), w) = M_1(\mathrm{conv}(w, \#, \#), w) = \mathrm{conv}(w, \#, \#)$.
- For $x \neq w$, $M_1(\mathrm{conv}(w, \#, \#), x) = \mathrm{conv}(w, x, v)$, for v being the longest prefix of x such that $x \in Lang(v \cdot \pi)$; where if no such v exists, then we let $v = \#$.
- For $y \neq w$, $M_1(\mathrm{conv}(w, x, v), y) = \mathrm{conv}(w, x, v')$, for v' being the longest prefix of v such that $y \in Lang(v' \cdot \pi)$; where if no such v' exists, then we let $v' = \#$.
- $M_1(\mathrm{conv}(w, x, v), w) = \mathrm{conv}(w, x, v)$.
- $M_1(\mathrm{conv}(w, x, \#), y) = \mathrm{conv}(w, x, \#)$.

Note that the length of v (after w and x get their values different from $\#$) is monotonically non-increasing until it gets the value $\#$, if ever. It is now easy to verify that M_1 is consistent, confident and learns $L = Lang(u \cdot \pi) \cup \{z\}$ from text T for L if the first datum different from $\#$ in T is z.

The second learner M_2 works in the case that the first datum x in T is different from z. The hypothesis space used by the learner is $\mathcal{H} = \{H_{\mathrm{conv}(w,x,v)} : w, x, v \in \Sigma^* \cup \{\#\}\}$, where for $w, x, v \in \Sigma^*$, $H_{\mathrm{conv}(\#,\#,\#)} = H_{\mathrm{conv}(w,x,\#)} = H_{\mathrm{conv}(\#,x,\#)} = \Sigma^*$, $H_{\mathrm{conv}(\#,x,v)} = Lang(v \cdot \pi)$, $H_{conv(w,x,v)} = Lang(v \cdot \pi) \cup \{w\}$ (other undefined hypotheses are not used and can be assumed to represent Σ^*).

The memory of M_2 is $\mathrm{conv}(w, x, v)$, where $w, x, v \in \Sigma^* \cup \{\#\}$. If mem is the memory of M_2 after having seen input $T[k]$, then its hypothesis after having seen input $T[k]$ is also mem. We thus just describe below the memory update of M_2 (ignoring its hypothesis).

Intuitively, x is the first datum different from $\#$ received by the learner, v is the longest prefix of x such that all data in the input, except maybe for one datum (with w denoting this datum, if any), belong to $Lang(v \cdot \pi)$. If and when such a v does not exist, v is taken as $\#$. Formally, initial memory of the learner is $\mathrm{conv}(\#, \#, \#)$. M_2 does not change its memory on input $\#$. Below, let $w, x, v, y \in \Sigma^*$.

- $M_2(\mathrm{conv}(\#, \#, \#), x) = \mathrm{conv}(\#, x, v)$, where v is the longest prefix of x such that $x \in Lang(v \cdot \pi)$; if no such v exists, then we let $v = \#$.
- $M_2(\mathrm{conv}(\#, x, v), y) = \mathrm{conv}(\#, x, v)$, if $y \in Lang(v \cdot \pi)$; otherwise $M_2(\mathrm{conv}(\#, x, v), y) = (y, x, v)$.
- $M_2(\mathrm{conv}(w, x, v), w) = \mathrm{conv}(w, x, v)$.
- For $y \neq w$, $M_2(\mathrm{conv}(w, x, v), y) = \mathrm{conv}(w', x, v')$, where, v' is the longest prefix of v such that at least one of y, w is in $Lang(v' \cdot \pi)$; where w' is $\#$, if both w, y belong to $Lang(v' \cdot \pi)$, otherwise, w' is the one of w, y which does not belong to $Lang(v' \cdot \pi)$.
  If there is no v' as above, then $M_2(\mathrm{conv}(w, x, v), y) = (w, x, \#)$.
- $M_2(\mathrm{conv}(w, x, \#)) = \mathrm{conv}(w, x, \#)$.
- $M_2(\mathrm{conv}(\#, x, \#)) = \mathrm{conv}(\#, x, \#)$.

It is easy to verify that M_2 is consistent and learns $Lang(u \cdot \pi) \cup \{z\}$, if z is not the first datum different from # in the input text. Furthermore, note that length of the third component v in the memory of M_2, after the first input datum different from # is received, is monotonically nonincreasing, until, if ever, it becomes equal to #. Also, for each value of v, the first component w of the memory changes at most once, from # to some element in Σ^*. Thus, M_2 is confident.

Hence, for each text T for $L = Lang(u \cdot \pi) \cup \{z\}$, either M_1 or M_2 converges on T to a correct hypothesis for L. This, along with Proposition 10 implies that $\mathcal{L}$ is learnable by a consistent and confident automatic learner which is word length memory bounded. □

Theorem 12. *For all $n > 0$, $\{L \cup \{z\} : L \in \mathcal{P}_n, z \in \Sigma^* - L\}$ is consistently and confidently learnable by an automatic learner. Furthermore, the learner is word length memory bounded.*

Proof. The theorem follows from Proposition 10, Proposition 11, the finiteness of G_n, and the fact that each language in $\mathcal{P}_n$ is of the form $Lang(u \cdot \pi)$ for some $\pi \in G_n$, $u \in \Sigma^*$. □

Proposition 13. *Suppose $n > 0$. Suppose $\pi, \pi' \in G_n - \{\varepsilon\}$ are such that the constant suffixes of π and π' are not right-consistent. Then, $\mathcal{L} = \{Lang(u \cdot \pi) \cup Lang(v \cdot \pi') : u, v \in \Sigma^*\}$ is learnable by a consistent and confident automatic learner which is word length memory bounded.*

Proof. Note that the requirements on π and π' imply that $Lang(\pi) \cap Lang(\pi') = \emptyset$. The learner essentially tries to learn $Lang(u \cdot \pi)$ and $Lang(v \cdot \pi')$ separately. The hypothesis space used by the learner is

$$\mathcal{H} = \{H_\beta : \beta \in \{\mathrm{emp}\} \cup \{\mathrm{conv}(u', v', c, c_u, c_v) : u', v' \in \Sigma^*, c, c_u, c_v \in \{0, 1\}\}\},$$

where, for $u', v' \in \Sigma^*, c_u, c_v \in \{0, 1\}$,

- $H_{\mathrm{emp}} = \emptyset$,
- $H_{\mathrm{conv}(u', v', 0, c_u, c_v)} = \Sigma^*$,
- $H_{\mathrm{conv}(u', v', 1, 1, 1)} = Lang(u' \cdot \pi) \cup Lang(v' \cdot \pi')$,
- $H_{\mathrm{conv}(u', v', 1, 0, 1)} = Lang(v' \cdot \pi')$,
- $H_{\mathrm{conv}(u', v', 1, 1, 0)} = Lang(u' \cdot \pi)$.

Other hypotheses are not used, and thus can be assumed to represent Σ^*.

If $content(T[k]) = \emptyset$, then after having seen the input $T[k]$, the memory and hypothesis of the learner are ? and emp, respectively. Otherwise, the memory and the hypothesis of the learner after having seen input $T[k]$ are $\mathrm{conv}(u', v', c, c_u, c_v)$ satisfying the following conditions, where $u', v' \in \Sigma^*$, $c, c_u, c_v \in \{0, 1\}$:

(P1) c_u is 1 if $content(T[k]) \cap Lang(\pi) \neq \emptyset$ and 0 otherwise. If $c_u = 1$, then u' is the longest string such that all strings in $content(T[k]) \cap Lang(\pi)$ belong to $Lang(u'\pi)$; otherwise $u' = \varepsilon$.

(P2) c_v is 1 if $content(T[k]) \cap Lang(\pi') \neq \emptyset$ and 0 otherwise. If $c_v = 1$, then v' is the longest string such that all strings in $content(T[k]) \cap Lang(\pi')$ belong to $Lang(v'\pi)$; otherwise $v' = \varepsilon$.

(P3) If all strings in $content(T[k])$ are in $Lang(\pi) \cup Lang(\pi')$, then c is 1. Otherwise, c is 0.

Note that one can easily automatically update the memory to satisfy the above properties. It is easy to verify that the learner consistently learns $\mathcal{L}$.

Furthermore, the strings u' and v' above are prefixes of the first datum different from # in the input which belongs to $Lang(\pi)$ and $Lang(\pi')$, respectively. Furthermore, u', v' are also monotonically non-increasing in length (except for the initial change from ε to a prefix of the first datum different from # which belongs to $Lang(\pi)$ and $Lang(\pi')$, respectively). Also, once the value of c is 0, it never changes its value. Similarly, once the value of c_u (respectively c_v) is 1, it never changes its value. Thus, the learner is confident. Hence the proposition follows. $\square$

Proposition 14. *Suppose $n > 0$. Suppose $\pi, \pi' \in G_n - \{\varepsilon\}$ and $a, b \in \Sigma$ are such that $a \neq b$. Then, $\{Lang(uav\pi) \cup Lang(ubw\pi') : u, v, w \in \Sigma^*\}$ is learnable by a consistent and confident automatic learner which is word length memory bounded.*

Proof. The hypothesis space used by the learner is $\mathcal{H} = \{H_{\mathrm{conv}(u,uav,ubw,c,\alpha,\beta,c',c'')} : u, v, w, \alpha, \beta \in \Sigma^*, c, c', c'' \in \{0, 1\}\} \cup \{H_{\mathrm{emp}}\}$, where $H_{\mathrm{emp}} = \emptyset$ and

$$H_{\mathrm{conv}(u,uav,ubw,c,\alpha,\beta,c',c'')} = \begin{cases} Lang(uav\pi) \cup Lang(ubw\pi'), & \text{if } c = 1; \\ \Sigma^*, & \text{otherwise.} \end{cases}$$

The initial memory of the learner is ?. After having seen input text $T[t]$, the memory of the learner is either ? (if $content(T[t]) = \emptyset$) or of the form $\mathrm{conv}(u, uav, ubw, c, \alpha, \beta, c', c'')$, where $u, v, w, \alpha, \beta \in \Sigma^*$ and $c, c', c'' \in \{0, 1\}$. Intuitively,

(P1) u is the longest common prefix of all strings in $content(T[t])$.

(P2) If $c' = 1$, then α is the longest string such that $content(T[t]) \subseteq Lang(\alpha a\pi)$; $c' = 0$ denotes that such an α does not exist (in this case the actual value of α is irrelevant).

(P3) If $c'' = 1$, then β is the longest string such that $content(T[t]) \subseteq Lang(\beta b\pi')$; $c'' = 0$ denotes that such a β does not exist (in this case the actual value of β is irrelevant).

(P4) If $c = 0$, then $content(T[t]) \cap ua\Sigma^* = \emptyset$ or $content(T[t]) \cap ub\Sigma^* = \emptyset$ or there exist no v, w such that $content(T[t]) \subseteq Lang(uav\pi) \cup Lang(ubw\pi')$.

(P5) If $c = 1$, then v, w are the longest strings such that $content(T[t]) \subseteq Lang(uav\pi) \cup Lang(ubw\pi')$. Furthermore, $content(T[t]) \cap ua\Sigma^* \neq \emptyset$ and $content(T[t]) \cap ub\Sigma^* \neq \emptyset$.

Formally, the memory and the hypothesis of the learner are defined as follows. Suppose T is the input text, and mem_t^T and hyp_t^T are the memory and the hypothesis of the learner after having seen input $T[t]$. If $mem_t^T = ?$, then $hyp_t^T = \text{emp}$, else $hyp_t^T = mem_t^T$. Thus, we just describe how mem_{t+1}^T is obtained from mem_t^T.

Initially memory of the learner is $mem_0^T = ?$. The learner will not change its memory on $\#$, thus $mem_t^T = ?$ if $content(T[t]) = \emptyset$. If $content(T[t]) \neq \emptyset$, then suppose the memory of the learner after having seen input $T[t]$ is $mem_t^T = \text{conv}(u_t, u_t a v_t, u_t a w_t, c_t, \alpha_t, \beta_t, c_t', c_t'')$. Now, mem_{t+1}^T is defined from mem_t^T and $T(t)$ via the following automatic updating function.

(1) If $T(t) = \#$, then $mem_{t+1}^T = mem_t^T$.

(2) If $mem_t = ?$ (that is $content(T[t]) = \emptyset$), and the new input $T(t) = x \neq \#$, then $mem_{t+1} = (u_{t+1}, u_{t+1} a v_{t+1}, u_{t+1} a w_{t+1}, c_{t+1}, \alpha_{t+1}, \beta_{t+1}, c_{t+1}', c_{t+1}'')$, where
$u_{t+1} = x$, $c_{t+1} = 0$ (v_{t+1} and w_{t+1} are irrelevant in this case and we can take them to be ε).
α_{t+1} is the longest prefix of x, if any, such that $x \in Lang(\alpha_{t+1} a \pi)$. If such α_{t+1} exists, then $c_{t+1}' = 1$; otherwise $c_{t+1}' = 0$ (in case $c_{t+1}' = 0$, then the value of α_{t+1} is irrelevant and we can take it to be ε).
β_{t+1} is the longest prefix of x, if any, such that $x \in Lang(\beta_{t+1} b \pi')$. If such β_{t+1} exists, then $c_{t+1}'' = 1$; otherwise $c_{t+1}'' = 0$ (in case $c_{t+1}'' = 0$, then the value of β_{t+1} is irrelevant and we can take it to be ε).

(3) Suppose $T(t) = x' \neq \#$ and $mem_t = \text{conv}(u_t, u_t a v_t, u_t b w_t, c_t, \alpha_t, \beta_t, c_t', c_t'')$. Then $mem_{t+1} = \text{conv}(u_{t+1}, u_{t+1} a v_{t+1}, u_{t+1} b w_{t+1}, c_{t+1}, \alpha_{t+1}, \beta_{t+1}, c_{t+1}', c_{t+1}'')$, where

(i) Update of u_{t+1}: u_{t+1} is the longest common prefix of u_t and x'.

(ii) Update of c_t' and α_{t+1}:
If $c_t' = 0$, then $c_{t+1}' = 0$, $\alpha_{t+1} = \alpha_t$.
If $c_t' = 1$, then α_{t+1} is the longest prefix of α_t such that $\alpha_{t+1} a$ is a prefix of $\alpha_t a$ and $x' \in Lang(\alpha_{t+1} a \pi)$; if such a α_{t+1} exists, then $c_t' = 1$, else $c_{t+1}' = 0$ and the value of α_{t+1} is irrelevant.

(iii) Update of c_t'' and β_{t+1}:
If $c_t'' = 0$, then $c_{t+1}'' = 0$, $\beta_{t+1} = \beta_t$.
If $c_t'' = 1$, then β_{t+1} is the longest prefix of β_t such that $\beta_{t+1} b$ is a prefix of $\beta_t b$ and $x' \in Lang(\beta_{t+1} b \pi')$; if such a β_{t+1} exists, then $c_t'' = 1$, else $c_{t+1}'' = 0$ and the value of β_{t+1} is irrelevant.

(iv) Update of $v_{t+1}, w_{t+1}, c_{t+1}$:
Case 1: u_{t+1} is a proper prefix of u_t.
(* Note that in this case, there are $a', b' \in \Sigma$, $a' \neq b'$, such that all strings in $content(T[t])$ as well as u_t extend $u_{t+1} a'$ and x' extends $u_{t+1} b'$. *)
If $u_t \in u_{t+1} a \Sigma^*$, and $x' \in u_{t+1} b \Sigma^*$, $c_t' = 1$, $u_{t+1} a$ is a prefix of $\alpha_t a$ and there exists a longest string w such that $x' \in Lang(u_{t+1} b w \pi')$, then, v_{t+1} is such that $u_{t+1} a v_{t+1} = \alpha_t a$, $w_{t+1} = w$, $c_{t+1} = 1$;
Else if $u_t \in u_{t+1} b \Sigma^*$, $x' \in u_{t+1} a \Sigma^*$, $c_t'' = 1$, $u_{t+1} b$ is a prefix of $\beta_t b$ and there exists a longest string v such that $x' \in Lang(u_{t+1} a v \pi)$, then, w_{t+1} is such that $u_{t+1} b w_{t+1} = \beta_t b$ and $v_{t+1} = v$, $c_{t+1} = 1$;

Otherwise, $c_{t+1} = 0$ and values of v_{t+1}, w_{t+1} are irrelevant and taken to be ε.
Case 2: Not case 1 (that is, $u_{t+1} = u_t$) and $c_t = 1$:
If $x' \in u_t a \Sigma^*$, then v_{t+1} is the longest prefix of v_t such that $x' \in Lang(u_t a v_{t+1} \pi)$. If such a v_{t+1} exists then let $c_{t+1} = 1$, $w_{t+1} = w_t$; otherwise $c_{t+1} = 0$ and values of v_{t+1}, w_{t+1} are irrelevant and set to ε.
If $x' \in u_t b \Sigma^*$, then w_{t+1} is the longest prefix of w_t such that $x' \in Lang(u_t b w_{t+1} \pi')$. If such a w_{t+1} exists then let $c_{t+1} = 1$, $v_{t+1} = v_t$; otherwise $c_{t+1} = 0$ and values of v_{t+1}, w_{t+1} are irrelevant and set to ε.
Case 3: Not case 1 (that is, $u_{t+1} = u_t$) and $c_t = 0$. In this case let $v_{t+1} = v_t$, $w_{t+1} = w_t$ and $c_{t+1} = c_t$.

This completes the description of how the memory of the learner is updated. It is easy to verify that the properties (P1) to (P3) are maintained (see (2) when the memory takes a non-? value for the first time, and the updates in (3)-(i), (ii) and (iii)). For properties, (P4) and (P5) note that in (3)-(iv) above: if $u_{t+1} = u_t$ and $c_t = 1$, then Case 2 updates $v_{t+1}, w_{t+1}, c_{t+1}$ to maintain properties (P4) and (P5). If u_{t+1} is proper prefix of u_t, then by the remark in Case 1, and using property (P2) and (P3), the construction assigns appropriate values to $v_{t+1}, w_{t+1}, c_{t+1}$.

Furthermore, using properties (P1), (P4) and (P5), it is easy to see that the learner is consistent, and it learns the class $\mathcal{L}$. Also, learner is confident as the values of u_t, α_t, β_t are monotonically non-increasing in t, and once u_t is stabilized to its final value, v_t and w_t are monotonically non-increasing. Furthermore, once $u_t, v_t, w_t, \alpha_t, \beta_t$ have reached their final values, c_t, c'_t, c''_t can only go from 1 to 0, and not the other way around. Thus, the memory gets stabilized on all inputs, and thus the learner is confident. □

Theorem 15. *For all $n > 0$, the class $\mathcal{P}_n \cup \{L \cup L' : L, L' \in \mathcal{P}_n \wedge L \cap L' = \emptyset\}$ has an automatic learner. Furthermore, this learner is consistent and confident, and is word length memory bounded.*

Proof. Note that for any two members $\pi_1, \pi_2 \in G_n - \{\varepsilon\}$ and any strings u and v with $Lang(u \cdot \pi_1) \cap Lang(v \cdot \pi_2) = \emptyset$, we must have that either the constant suffixes of π_1 and π_2 are not right-consistent or u, v are not left-consistent.

Thus, the theorem follows using Proposition 10, Theorem 9 (for learning $\mathcal{P}_n$), Proposition 11 and Theorem 12 (for learning languages $L \cup \{z\}$, with $L \in \mathcal{P}_n$ and $z \notin L$), Proposition 13, Proposition 14, the fact that G_n is finite, and $\mathcal{P}_n = \{u \cdot \pi : \pi \in G_n, u \in \Sigma^*\}$, (where the last two propositions above give the learnability of $L \cup L'$, with $L, L' \in \mathcal{P}_n$ and $L \cap L' = \emptyset$, L, L' are infinite). □

We now consider the general case of learning unions of pattern languages from $\mathcal{P}_n$. While the above results also hold for non-erasing pattern languages, the following results of this section hold only for erasing pattern languages.

Lemma 16. *Suppose $k > 1$, and Σ is a finite alphabet. Suppose $L_0, L_1, \ldots, L_k$ are erasing pattern languages (over the alphabet Σ) generated by regular patterns. If $card(\Sigma) \geq k + 1$, L_0 is infinite and the difference $L_0 - \bigcup_{i \in \{1,2,\ldots,k\}} L_i$ is not empty, then this difference is infinite.*

Proof. If $S = \{i : 1 \le i \le k, L_i \text{ is infinite}\}$, then infiniteness of $L_0 - \bigcup_{i \in S} L_i$ implies the infiniteness of $L_0 - \bigcup_{i \in \{1,2,\ldots,k\}} L_i$. Thus, without loss of generality, we can assume that each of $L_1, L_2, \ldots, L_k$ is infinite.

For j with $0 \le j \le k$, suppose $L_j = \mathit{Lang}(\pi_j)$, where $\pi_j = \alpha_j^1 @ \alpha_j^2 @ \ldots @ \alpha_j^{t_j}$, where $t_j > 1$, $\alpha_j^s \in \Sigma^+$ for all s with $1 < s < t_j$ and $\alpha_j^1, \alpha_j^{t_j} \in \Sigma^*$. Note that as L_j is a regular pattern language, there exists such a π_j.

Consider a string w in $L_0 - \bigcup_{i \in \{1,2,\ldots,k\}} L_i$. Let $w_1 = \alpha_0^1$ and w_2 be such that $w = w_1 w_2$. Below we will construct a string $y \in \Sigma^+$ such that $w_1 y w_2 \notin \bigcup_{i \in \{1,2,\ldots,k\}} L_i$. Note that $w_1 y w_2 \in L_0$. As this process can be repeated, we have that $L_0 - \bigcup_{i \in \{1,2,\ldots,k\}} L_i$ is infinite. Let

$$S_1 = \{j : 1 \le j \le k \wedge w_1 \text{ is a proper prefix of } \alpha_j^1\},$$
$$S_2 = \{j : 1 \le j \le k \wedge w_2 \text{ is a proper suffix of } \alpha_j^{t_j}\},$$
$$S_3 = \{j : 1 \le j \le k \wedge w_1 \text{ is not left-consistent with } \alpha_j^1 \text{ or } w_2 \text{ is not right-consistent with } \alpha_j^{t_j}\}.$$

For each $j \in S_1$, let $c_j = \alpha_j^1(|w_1|)$ (thus c_j is the character right after the prefix w_1 in α_j^1). For each $j \in S_2$, let $c_j = \alpha_j^{t_j}(|\alpha_j^{t_j}| - |w_2| - 1)$ (thus c_j is the character right before the suffix w_2 in $\alpha_j^{t_j}$). Let $\Sigma' = \Sigma - \{c_j : j \in S_1 \cup S_2\}$. The y we choose below will be a member of $(\Sigma')^+$. Thus, it easily follows that $w_1 y w_2 \notin L_j$, for $j \in S_1 \cup S_2 \cup S_3$. Furthermore, clearly $\mathrm{card}(\Sigma') > k - \mathrm{card}(S_1 \cup S_2 \cup S_3)$.

For $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3)$, let r_j be maximal, and correspondingly β_j be maximal prefix of $\alpha_j^{r_j}$ such that $w_1 \in \mathit{Lang}(\alpha_j^1 @ \alpha_j^2 @ \ldots @ \alpha_j^{r_j - 1} @ \beta_j)$. Note that there exists such a $r_j > 1$ as α_j^1 is a prefix of w_1. Furthermore, note that if $r_j \ne t_j$, then β_j is a proper prefix of $\alpha_j^{r_j}$. Similarly, let r'_j be minimal, and γ_j be corresponding maximal suffix of $\alpha_{r'_j}$ such that $w_2 \in \mathit{Lang}(\gamma_j @ \alpha_j^{r'_j+1} @ \alpha^{r'_j+2} @ \ldots @ \alpha_j^{t_j})$. Note that there exists such a $r'_j < t_j$ as $\alpha_j^{t_j}$ is a suffix of w_2. Furthermore, if $r'_j \ne 1$, then γ_j is a proper suffix of $\alpha_j^{r'_j}$.

Fix $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3)$. Now, $r_j \le r'_j$ (otherwise, $w_1 w_2 \in L_j$, as $w_1 \in \mathit{Lang}(\alpha_j^1 @ \alpha_j^2 \ldots @ \alpha_j^{r_j - 1} @)$ and $w_2 \in \mathit{Lang}(@ \alpha_j^{r_j} @ \alpha_j^{r_j+1} \ldots @ \alpha_j^{t_j})$). If $r_j < r'_j$, then $r_j \ne t_j$, and thus β_j must be a proper prefix of $\alpha_j^{r_j}$. Let $S_4 = \{j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3) : r_j < r'_j$ or $|\beta_j \gamma_j| < |\alpha_j^{r_j}|\}$. For $j \in S_4$, let $c_j = \alpha_j^{r_j}(|\beta_j|)$. Let $\Sigma'' = \Sigma' - \{c_j : j \in S_4\}$. We will make sure that $y \in (\Sigma'')^+$. Thus, we will have that $w_1 y w_2 \notin L_{j'}$, for $j' \in S_4$. Note that $\mathrm{card}(\Sigma'') \ge k - \mathrm{card}(S_1 \cup S_2 \cup S_3 \cup S_4)$.

Note that for $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$, $r_j = r'_j$ and $|\beta_j \gamma_j| > |\alpha_j^{r_j}|$ (here, $r_j = r'_j$ and $|\beta_j \gamma_j| = |\alpha_j^{r_j}|$ is not possible as otherwise, $w_1 w_2 \in L_j$). Furthermore, as $1 < r_j = r'_j < t_j$, $\beta_j \ne \varepsilon$, $\gamma_j \ne \varepsilon$, β_j is a proper prefix of $\alpha_j^{r_j}$ and γ_j is a proper suffix of $\alpha_j^{r_j}$. Thus, we have that $|\beta_j| \ge 1$, $|\gamma_j| \ge 1$, and $|\alpha_j^{r_j}| \ge 3$.

Let $r > |\alpha_j^{r_j}|$, for all $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$. Let $a, b \in \Sigma''$ be such that $a \ne \alpha_j^{r_j}(|\beta_j|)$ and $b \ne \alpha_j^{r_j}(|\alpha_j^{r_j}| - 1 - |\gamma_j|)$ (that is $\beta_j a$ is not a prefix of $\alpha_j^{r_j}$ and $b \gamma_j$ is not a suffix of $\alpha_j^{r_j}$), for all $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$.

Claim 17. *For $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$, for any $y \in a^r \Sigma^* b^r$, if $w_1 y w_2 \in L_j$, then $\alpha_j^{r_j}$ is a substring of y.*

The claim holds as, for the substitution θ such that $w_1 y w_2 = \pi_j \theta$, by definition of r_j, w_1 is a prefix of $(\alpha_j^1 @ \alpha_j^2 \ldots @ \alpha_j^{r_j - 1} @ \beta_j)\theta$. As, $\alpha_j^{r_j}(|\beta_j|) \neq a$, $w_1 a^r$ must be a proper prefix of $(\alpha_j^1 @ \alpha_j^2 \ldots @ \alpha_j^{r_j - 1} @ \alpha_j^{r_j})\theta$. Similarly, by definition of r'_j, w_2 is a suffix of $(\gamma_j @ \alpha_j^{r_j + 1} @ \ldots @ \alpha_j^{t_j})\theta$. As, $\alpha_j^{r_j}(|\alpha_j^{r_j}| - 1 - |\gamma_j|) \neq b$, $b^r w_2$ must be a proper suffix of $(\alpha_j^{r_j} @ \alpha_j^{r_j + 1} @ \ldots @ \alpha_j^{t_j})\theta$. Claim thus follows as $|\alpha_j^{r_j}| < r$.

If $w_1 a^r b^r w_2 \notin Lang(L_j)$, for all $j \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$ then one can take $y = a^r b^r$. Otherwise, suppose $w_1 a^r b^r w_2 \in Lang(L_{j_0})$, for some $j_0 \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$. Then, by Claim above, $\alpha_{j_0}^{r_{j_0}} = a^i b^{i'}$ for some i, i'. Furthermore, $i \neq 0$ and $i' \neq 0$, as $\alpha_{j_0}^{r_{j_0}}(|\beta_{j_0}|) \neq a$ and $\alpha_{j_0}^{r_{j_0}}(|\alpha_{j_0}^{r_{j_0}}| - 1 - |\gamma_{j_0}|) \neq b$. Thus

Claim 18. $(w_1(|w_1| - 1), w_2(0)) \in \{(a, a), (b, b), (b, a)\}$.

The claim holds as β_{j_0} is a suffix of w_1 and a nonempty proper prefix of $\alpha_{j_0}^{r_{j_0}}$, and γ_{j_0} is a prefix of w_2 and a nonempty proper suffix of $\alpha_{j_0}^{r_{j_0}}$, and $|\beta_{j_0}\gamma_{j_0}| > |\alpha_{j_0}^{r_{j_0}}|$.

If $\{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4) = \{j_0\}$, then taking $y = (ab)^r$, we have that $w_1 y w_2 \notin L_{j_0}$, as $\alpha_{j_0}^{r_{j_0}}$ must be a substring of $(ab)^r$, which would imply $\alpha_{j_0}^{r_{j_0}} = ab$, a contradiction to $|\alpha_{j_0}^{r_{j_0}}| \geq 3$.

So suppose $\text{card}(\{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)) \geq 2$. Thus, $\text{card}(\Sigma'') > 2$. Let $c \in \Sigma'' - \{a, b\}$. Then, we claim that $w_1 a^r c^r b^r w_2 \notin Lang(L_i)$ for any $i \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$. Suppose by way of contradiction that for some $i \in \{1, 2, \ldots, k\} - (S_1 \cup S_2 \cup S_3 \cup S_4)$, $w_1 a^r c^r b^r w_2 \in Lang(L_i)$. Then, by Claim 17 we must have that $\alpha_i^{r_i}$ is a substring of $a^r c^r b^r$. By choice of $a, b \in \Sigma''$, we have that $\alpha_i^{r_i} \notin a^*$ and $\alpha_i^{r_i} \notin b^*$. Furthermore, by Claim 18, $\alpha_i^{r_i} \notin c^*$, as $\alpha_i^{r_i}$ contains $w_1(|w_1| - 1)$ and $w_2(0)$. Now consider the following two cases.

Case 1: $\alpha_i^{r_i} \in a^+ c^+$. In this case, by Claim 18, we must have that $w_2(0) = a$, and some string in $a^+ c^+$ is a prefix of w_2. But this contradicts the fact that γ_{j_0} (which is a non-empty suffix of $\alpha_{j_0}^{r_{j_0}} \in a^+ b^+$) is a prefix of w_2.

Case 2: $\alpha_i^{r_i} = c^+ b^+$. In this case, by Claim 18, we must have that $w_1(|w_1| - 1) = b$, and some string in $c^+ b^+$ is a suffix of w_1. But this contradicts the fact that β_{j_0} (which is a non-empty prefix of $\alpha_{j_0}^{r_{j_0}} \in a^+ b^+$) is a suffix of w_1.

From the above analysis it follows that, for some $y \neq \varepsilon$, $w_1 y w_2 \notin \bigcup_{j \in \{1, 2, \ldots, k\}} L_j$. $\square$

Recall that $lleast(S)$ denotes the length-lexicographically least string in the set S.

Theorem 19. *Suppose $|\Sigma| \geq 3$, $n > 0$ and $\pi, \pi' \in G_n - \{\varepsilon\}$. Let $\mathcal{L} = \{Lang(u \cdot \pi) \cup Lang(v \cdot \pi') : u, v \in \Sigma^*, lleast(Lang(u \cdot \pi)) \leq_{ll} lleast(Lang(v \cdot \pi'))\}$. Then, there is a confident and automatic learner $M_{\pi, \pi'}$ using hypothesis space $\mathcal{H} = (H_\beta)_{\beta \in J}$ for some regular index set J and automatic family $\mathcal{H}$ such that:*

(1) $M_{\pi, \pi'}$ is word length memory bounded,

(2) $M_{\pi, \pi'}$ learns $\mathcal{L}$,

(3) for all texts T for a language $L \notin \mathcal{L}$, $M_{\pi, \pi'}$ converges on T to an index β such that $L - H_\beta$ is finite.

Proof. Note that by Lemma 16 different languages in $\mathcal{L}$ are pairwise infinitely different. The hypothesis space used by the learner is $\{H_{\mathrm{conv}(x,u,v,c)} : x, u, v \in \Sigma^*, c \in \{0, 1, 2\}\} \cup \{H_{\mathrm{emp}}\}$, where $H_{\mathrm{emp}} = \emptyset$, $H_{\mathrm{conv}(x,u,v,0)} = \Sigma^*$, $H_{\mathrm{conv}(x,u,v,1)} = Lang(u \cdot \pi)$ and $H_{\mathrm{conv}(x,u,v,2)} = Lang(u \cdot \pi) \cup Lang(v \cdot \pi')$.

If $content(T[k]) = \emptyset$, then the memory of the learner $M_{\pi,\pi'}$ will be ? and hypothesis of the learner will be emp; otherwise, the memory and hypothesis of the learner $M_{\pi,\pi'}$ will be the same. Thus we will just describe how the learner updates its memory.

Suppose T is the input text. Let mem_k^T denote the memory of the learner after having seen $T[k]$. If $content(T[k]) = \emptyset$, then $mem_k^T =?$. If $content(T[k]) \neq \emptyset$, then mem_k^T is of the form $\mathrm{conv}(x_k, u_k, v_k, c_k)$, where $x_k, u_k, v_k \in \Sigma^*, c_k \in \{0, 1, 2\}$.

The following properties will be satisfied by $mem_k^T = (x_k, u_k, v_k, c_k)$.

(P1) x_k is the length-lexicographically smallest string in $content(T[k])$. Below let k_0 be least such that $T(k_0) = x_k$.

(P2) If there does not exist a prefix u_k of x_k such that x_k is the length-lexicographically smallest string in $Lang(u_k \cdot \pi)$, then $c_k = 0$ and values of u_k, v_k are irrelevant.

For the following properties, assume that there exists a prefix u_k of x_k such that x_k is the length-lexicographically smallest string in $Lang(u_k \cdot \pi)$. Note that such a u_k is unique, if it exists.

(P3) u_k is the prefix of x_k such that x_k is the length-lexicographically smallest string in $Lang(u_k \cdot \pi)$.

(P4) If all strings in $\{T(s) : k_0 \leq s < k\} - \{\#\}$ belong to $Lang(u_k \cdot \pi)$, then $c_k = 1$ (in this case value of v_k is irrelevant).

(P5) If there exists a string in $\{T(s) : k_0 \leq s < k\} - \{\#\}$ which does not belong to $Lang(u_k \cdot \pi)$, then
If there exists a v_k such that $\{T(s) : k_0 \leq s < k\} - \{\#\} \subseteq Lang(u_k \cdot \pi) \cup Lang(v_k \cdot \pi')$, then v_k is the longest such string and $c_k = 2$. Otherwise, $c_k = 0$ and value of v_k is irrelevant.

Intuitively, $c = 0$ denotes that for the currently seen length-lexicographically minimal string x, there are no u, v such that x is the length-lexicographically smallest string in $Lang(u \cdot \pi)$, and all the strings seen after x belong to $Lang(u \cdot \pi) \cup Lang(v \cdot \pi')$.

The case of $c = 1$ denotes that, for the currently seen length-lexicographically minimal string x, x is also the length-lexicographically smallest string in $Lang(u \cdot \pi)$; furthermore, all strings seen in the input after x belong to $Lang(u \cdot \pi)$.

The case of $c = 2$ denotes that, for the currently seen length-lexicographically minimal string x, x is also the length-lexicographically smallest string in $Lang(u \cdot \pi)$; furthermore, at least one input string seen after x does not belong to $Lang(u \cdot \pi)$, and all strings seen in the input after x belong to $Lang(u \cdot \pi) \cup Lang(v \cdot \pi')$, and v is the longest such possible string.

The learner $M_{\pi,\pi'}$ can now be defined to satisfy the above properties as follows. Whenever $M_{\pi,\pi'}$ sees an input w which is length-lexicographically smaller than any previously seen input, it changes its memory to $\mathrm{conv}(w, u, v, c)$ satisfying the following conditions: $v = \varepsilon$; if there is a prefix s of w such that w is the length lexicographically least element of $Lang(s \cdot \pi)$, then $u = s$ and $c = 1$, else $u = w$ and $c = 0$.

In other cases, suppose the previous memory of $M_{\pi,\pi'}$ is $\mathrm{conv}(x, u, v, c)$ and the new input is w. Then, use the first case below which applies:

Case 1: $c = 0$ or $w \in \mathit{Lang}(u \cdot \pi)$. In this case new memory is $\mathrm{conv}(x, u, v, c)$.
Case 2: $w \notin \mathit{Lang}(\pi')$. In this case change the memory to $\mathrm{conv}(x, u, v, 0)$.
Case 3: $c = 1$. In this case let v' be the longest prefix of w such that $w \in \mathit{Lang}(v' \cdot \pi')$, and change the memory to $\mathrm{conv}(x, u, v', 2)$. If there is no such v', then change the memory to $(x, u, v, 0)$.
Case 4: $c = 2$. In this case let v' be the longest prefix of v such that $w \in \mathit{Lang}(v' \cdot \pi')$, and change the memory to $(x, u, v', 2)$. If there is no such v', then change the memory to $(x, u, v, 0)$.

Note that, on all input texts T, the memory/conjecture of $M_{\pi,\pi'}$ converges. To see this, note that value of $\lim_{k\to\infty} x_k$ clearly converges to the length-lexicographically least string in $\mathit{content}(T)$. Once final value $\lim_{k\to\infty} x_k$ is achieved, then $\lim_{k\to\infty} u_k$ and $\lim_{k\to\infty} c_k$ also converge (as c_k can only go from 1 to 2 to 0, after $\lim_{k\to\infty} x_k$ achieves its final value). Furthermore, v_k is monotonically non-increasing in length while $c_k = 2$.

Suppose $\mathit{content}(T) \neq \emptyset$ and the converged memory/conjecture is $\mathrm{conv}(x, u, v, c)$. Then, using the properties (P1) to (P5) above for different values of c, the final hypothesis of the learner is either for Σ^* (when $c = 0$), or $u \cdot \pi$ contains the length-lexicographically smallest string in the input and $\mathit{content}(T) \subseteq^* H_{\mathrm{conv}(x,u,v,c)}$ (when $c = 1$ or 2).

Furthermore, if $\mathit{content}(T) = L = \mathit{Lang}(s \cdot \pi) \cup \mathit{Lang}(s' \cdot \pi')$, for some $s, s' \in \Sigma^*$, where $\mathit{Lang}(s \cdot \pi)$ contains the length-lexicographically smallest string in L, then the following two statements hold:

(a) $M_{\pi,\pi'}$ on T converges to an index β such that $H_\beta \supseteq L$, as $L \subseteq^* H_\beta$, and thus by Lemma 16, $L \subseteq H_\beta$.

(b) $M_{\pi,\pi'}$ converges on T to an index β such that $H_\beta \subseteq L$ (and thus by (a) $H_\beta = L$). To see this, suppose $M_{\pi,\pi'}$ converges on T to index $\mathrm{conv}(x, u, v, c)$. Then x is the length-lexicographically least element of L, $\mathit{Lang}(s \cdot \pi)$ and $\mathit{Lang}(u \cdot \pi)$. Thus $s = u$. Furthermore, if $c = 2$, then we have that $\mathit{Lang}(s' \cdot \pi') \supseteq \mathit{Lang}(v \cdot \pi')$ (since in Cases 3 and 4, the algorithm chooses the longest possible prefix). So the theorem follows. □

Corollary 20. *Suppose $|\Sigma| \geq 3$ and $n > 0$. Let $\mathcal{L} = \{\mathit{Lang}(u \cdot \pi) \cup \mathit{Lang}(v \cdot \pi') : u, v \in \Sigma^*, \pi, \pi' \in G_n - \{\varepsilon\}\}$. Then, there is a confident and automatic learner M using hypothesis space $\mathcal{H} = (H_\beta)_{\beta \in J}$ for some regular index set J and automatic family $\mathcal{H}$ such that:*

(1) M is word length memory bounded;

(2) M learns $\mathcal{L}$;

(3) for all texts T for a language $L \notin \mathcal{L}$, M converges on T to an index β such that $L - H_\beta$ is finite.

Proof. For $\pi, \pi' \in G_n - \{\varepsilon\}$, let $M_{\pi,\pi'}$, be as given by Theorem 19. Define M which uses memory which is the convolution of the memories of all these $M_{\pi,\pi'}$, $\pi, \pi' \in G_n - \{\varepsilon\}$.

Hypothesis of M is the hypothesis of $M_{\pi,\pi'}$, where π, π' are chosen to be length-lexicographically least pair such that the conjecture of $M_{\pi,\pi'}$ is not a proper superset of the conjecture of any other $M_{\pi'',\pi'''}$. Now it follows using Theorem 19 and Lemma 16, that M learns $\mathcal{L}$. For any text T for $L \notin \mathcal{L}$, (3) holds as this holds for the limiting conjectures of each of the learners $M_{\pi,\pi'}$. □

Theorem 21. *Suppose $|\Sigma| \geq 3$ and $n > 0$. Let $\mathcal{L} = \{L_1 \cup L_2 : L_1, L_2 \in \mathcal{P}_n\}$. Then, there exists an automatic learner which is word length memory bounded such that:*

(1) The learner learns $\mathcal{L}$;

(2) For all texts T for a language $L \notin \mathcal{L}$, the learner converges to an index for a language L' such that $L - L'$ is finite.

Proof. For $\pi \in G_n$, let

$$\begin{aligned}
&\mathcal{L}_1^\pi = \{Lang(s \cdot \pi) \cup \{z\} : s \in \Sigma^*, z \in \Sigma^*, z \notin Lang(s \cdot \pi)\},\\
&\mathcal{L}_1 = \bigcup_{\pi \in G_n} \mathcal{L}_1^\pi \text{ and}\\
&\mathcal{L}_2 = \{L_1 \cup L_2 : L_1, L_2 \in \mathcal{P}_n, |L_1| > 1, |L_2| > 1\}.
\end{aligned}$$

Let M_1^π and M_2^π be the two learners given in the proof of Proposition 11. Note that (1) for every text T for $L \in \mathcal{L}_1^\pi$, at least one of M_1^π and M_2^π learns L from text T, and (2) for any input text T, if M_1^π (M_2^π) converges to the hypothesis different from that of Σ^*, say $Lang(s \cdot \pi) \cup S$, where S is either $\emptyset$ or a set containing one element not in $Lang(s \cdot \pi)$, then $S \subseteq content(T) \subseteq Lang(s \cdot \pi) \cup S$. (This property can be easily verified from the construction of M_1 and M_2 in Proposition 11).

The learner for $\mathcal{L}_2$ (say M) as given in Corollary 20 is confident (though may not be consistent).

Let N be a learner which on input text T has memory containing the convolution of the memories of the learner M for $\mathcal{L}_2$ (from Corollary 20) and the memories of the learners M_1^π and M_2^π, for each $\pi \in G_n$. Thus, N can simulate each of the above learners. Additionally, in memory it remembers if it has seen at most 2 elements along with the elements in the input text, if there are at most 2 elements in the input text.

The hypothesis space used by N contains the hypothesis spaces used by M_r^π (for $\pi \in G_n$, $r \in \{1, 2\}$) and by the learner M, as well as hypotheses for all languages containing at most two elements of Σ^*. The hypothesis of N on any input is based on the first case below which applies:

(a) The hypothesis for the input elements seen up to then, if it contains at most two elements;
(b) The hypothesis of the learner M_r^π, $\pi \in G_n$ $r \in \{1, 2\}$, if this hypothesis is not a proper superset of any of the hypothesis of $M_{r'}^{\pi'}$, $\pi' \in G_n$ and $r' \in \{1, 2\}$, and contains at most finitely many elements not in the hypothesis of the learner M;
(c) The hypothesis of the learner M, if no such learner M_r^π as in (b) above exists.

Clearly the above learner learns all languages of cardinality at most two.

Now consider the case of an input language $L \in \mathcal{L}_1$, of cardinality more than 2. Using Lemma 16 and the consistency of the learners $M_{r'}^{\pi'}$ as well as the property of the learner M for $\mathcal{L}_2$ that its final hypothesis misses out at most finitely many elements in the input language, one can see that the learner N converges as in (b) above to a correct hypothesis.

The last remaining case is that the input language L is in $\mathcal{L}_2 - \mathcal{L}_1$. Now, using Lemma 16, consistency of $M^{\pi'}_{r'}$ and the fact that a hypothesis $Lang(s \cdot \pi') \cup \{z\}$ by $M^{\pi'}_{r'}$ implies $\{z\} \subseteq L$, we have that (b) cannot hold in the limit or the hypothesis of the case (b) and the learner M are equivalent. Thus, N converges to the same language as the learner M for $\mathcal{L}_2$. Thus, N learns $\mathcal{L}_1 \cup \mathcal{L}_2$. $\square$

5 Character variables

In this section, we consider the following modification of pattern languages. We consider two types of variables: character variables which can be replaced by one symbol of Σ and string variables which can be replaced by any string, including the empty string. For such a pattern π, $Lang(\pi)$ denotes the set of all strings that can be obtained by replacing character variables by some character in Σ, and string variables by some string in Σ^*.

Note that one can simulate non-erasing pattern languages (as studied by Angluin [2]) by putting one character variable followed by one string variable. The above kind of languages are a special case of typed pattern languages considered by Koshiba [24]. The non-erasing pattern language associated with pattern $xyxz$ can be proven to be regular, by chosing the equivalent pattern $x'y'yx'z'z$ of character variables x', y', z' and erasing string variables y, z.

Definition 22. Suppose $n \in \mathbb{N}$. Let

$O_n = \{\pi : \pi$ contains only constants and character variables and for all $i, j < |\pi|$ with $\pi(i) = \pi(j)$ and $\pi(i)$ being a character variable, $\mathrm{card}(\{\pi(\ell) : i < \ell < j, \pi(\ell)$ is a character variable$\}) \leq n\}$

and $\mathcal{O}_n = \{Lang(\pi) : \pi \in O_n\}$.

For example, the pattern $abxaxyazba$ is in O_0 (where $\Sigma = \{a, b\}$ and x, y, z are character variables) and $axbxbybx$ is in O_2 but not in O_1.

Remark 23. In this remark we show that, for all $n \in \mathbb{N}$, $\mathcal{O}_n$ is an automatic family. Note that for $\pi \in O_n$, the number of variables in π might be large. This causes representation problems if we use O_n as indices for the automatic family — the corresponding alphabet set becomes infinite. The trick is to reuse variables, as at any point at most $n+1$ variables can be "active". Consider any pattern $\pi \in O_n$. We say that a variable $\pi(\ell)$ is active at ℓ', if both $\{\ell'' \leq \ell' : \pi(\ell) = \pi(\ell'')\}$ and $\{\ell'' \geq \ell' : \pi(\ell) = \pi(\ell'')\}$ are not empty. Note that by definition of O_n, at any ℓ', there can be at most $n+1$ active variables. Thus, we will code π by using only $n+1$ variables, where inactive variables are reused. The following sequence of definitions and arguments show that, for each $n \in \mathbb{N}$, $\mathcal{O}_n$ is an automatic family.

(a) For $i \leq n$, let s_i, v_i be $2n+2$ symbols not in Σ. Let $X = \{s_i : i \leq n\}$ and $Y = \{v_i : i \leq n\}$. Let $PP = \{\pi' \in (\Sigma \cup X \cup Y)^* : (\forall j < |\pi'|)\, [(\pi'(j) = v_i) \Rightarrow (\exists j' < j)\, [\pi'(j') = s_i]]\}$.
Intuitively, each occurrence of s_i indicates that the variable number i is being reused from that point onwards. Occurrence of v_i in π' then just corresponds to the variable which occurs at the most recent previous s_i in π'.

(b) For $\pi' \in PP$, let h be a function such that, if $\pi'(j) = v_i$, then $h(j) = j'$ for the largest $j' < j$ such that $\pi'(j') = s_i$. Note that the mapping $0^j \to 0^{h(j)}$ is automatic.
Let $LL(\pi') = \{y : (\forall \ell, \ell' < |\pi'|)\ [[(\pi'(\ell) \in \Sigma) \Rightarrow (y(\ell) = \pi'(\ell))]$ and $[(\pi'(\ell) = v_i) \Rightarrow (y(\ell) = y(h(\ell)))]]\}$.
As the above gives a first order definition for checking whether $y \in LL(\pi')$, we have that $\{(\pi', x) : x \in LL(\pi')\}$ is automatic.

(c) For any pattern $\pi \in O_n$ there exists a pattern $\pi' \in PP$ such that $Lang(\pi) = LL(\pi')$. To see this, define π' as follows.

BEGIN
Let Free $= \{i : i \leq n+1\}$.
For $j = 0$ to $|\pi| - 1$ do
  Beginfor
    If $\pi(j) \in \Sigma$,
    Then let $\pi'(j) = \pi(j)$.
    Else (*$\pi(j)$ is a variable*)
      If $\pi(j)$ does not appear in $\pi[j]$,
      Then let $i = \min(\text{Free})$, let Free $=$ Free $- \{i\}$ and let $\pi'(j) = s_i$.
      Else let $\pi'(j) = v_i$, for the unique $i \leq n$ such that for some $j' < j$, $\pi(j') = \pi(j)$ and $\pi'(j') = s_i$.
      Endif
      If $\pi(j)$ is a variable which does not appear in $\pi(j+1)\pi(j+2)\ldots\pi(|\pi|-1)$,
      Then let Free $=$ Free $\cup \{i\}$, where i satisfies $\pi'(j) \in \{s_i, v_i\}$.
      Endif
    Endif
  Endfor
END

It can be easily verified that $Lang(\pi) = LL(\pi')$. We say that π' above *represents* pattern π.

(d) For $\pi' \in PP$, it is easy to automatically check if it "represents" a pattern in O_n. To see this note that π' represents a pattern in O_n iff for all $\ell' < |\pi'|$, such that $\pi'(\ell') = v_i$, for $\ell = h(\ell')$, the following property is satisfied:
$\text{card}(\{h(j) : \pi'(j) \in Y, \ell < j < \ell'\} \cup \{j : \pi'(j) \in X, \ell < j < \ell'\}) \leq n$.

(e) Thus, we have that $\mathcal{O}_n = \{LL(\pi') : \pi' \in PP$ and π' represents a pattern in $O_n\}$. Thus, $\mathcal{O}_n$ is an automatic family.

Theorem 24. *For all $n \in \mathbb{N}$, $\mathcal{O}_n$ is learnable by an automatic learner with memory bounded by the length of the longest datum seen so far in the input.*

Proof. Suppose s and v are special symbols not in Σ.

The memory of the learner will be either ? or of the form $\text{conv}(x, x_0, x_1, \ldots, x_{2n})$, where $x \in \Sigma^*$ and each x_r is in $\Sigma^* \cdot (s \cdot (\Sigma \cup \{v\})^*)^*$, with $|x_r| = |x|$. Furthermore, $x_r(\ell) \in \{x(\ell), s, v\}$ for all $\ell < |x|$ and all $r \leq 2n$. Let MEM $= \{\text{conv}(x, x_0, x_1, \ldots, x_{2n}) : x \in \Sigma^*$ and for $r \leq 2n$,

$x_r \in \Sigma^* \cdot (s \cdot (\Sigma \cup \{v\})^*)^*$ and $|x_r| = |x|\}$, that is MEM is the set of possible values for the memory (besides ?).

The hypothesis of the learner will always be the same as its memory, where H_β, $\beta \in$ MEM is as defined below.

We say that $\beta = \mathrm{conv}(x, x_0, x_1, \ldots, x_{2n}) \in$ MEM is a *prepattern* iff, for every $\ell < |x|$, there is at most one $r \leq 2n$ with $x_r(\ell) \in \{s, v\}$. Intuitively, a prepattern codes a pattern π as described below.

For a prepattern $\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})$, let $\mathrm{PAT}(\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})) = \pi$ be such that

(a) $|\pi| = |x|$;
(b) $\pi(\ell) = x(\ell)$ iff $x_r(\ell) = x(\ell)$ for all $r \leq 2n$;
(c) $\pi(\ell)$ is the variable $v_{\ell'}$ iff for some $r \leq 2n$, $x_r(\ell) \in \{s, r\}$ and $\ell' \leq \ell$ is the largest number such that $x_r(\ell') = s$.

Intuitively, in a prepattern $\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})$ each x_i codes some of the variables appearing in the target pattern π: appearance of s and v's (before the next s) in x_i corresponds to a (distinct) variable which appears in the corresponding locations in π.

If β is a prepattern, then we let $H_\beta = \mathit{Lang}(\mathrm{PAT}(\beta))$, else we let $H_\beta = \emptyset$. Note that $\mathcal{H} = \{H_\beta : \beta \in \mathrm{MEM}\}$ is an automatic family, as one can automatically check whether $\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})$ is a prepattern, and for a prepattern β, automatically decide if $y \in \mathit{Lang}(\mathrm{PAT}(\beta))$. Note that one does not need to compute $\mathrm{PAT}(\beta)$ to do this, as for $\beta = \mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})$, $y \in \mathit{Lang}(\mathrm{PAT}(\beta))$ iff

$$(\forall 0^\ell, 0^{\ell'} : 0^\ell <_{ll} 0^{\ell'} \leq_{ll} x)\,(\forall r \leq 2n)$$
$$[[x_r(\ell) = s \text{ and } x_r(\ell') = v] \Rightarrow [(\exists 0^{\ell''} : 0^\ell <_{ll} 0^{\ell''} <_{ll} 0^{\ell'})[x_r(\ell'') = s] \text{ or } y(\ell) = y(\ell')]].$$

Thus, $\mathcal{H}$ is an automatic family.

For all $\pi \in O_n$, for a text T for $\mathit{Lang}(\pi)$, the learning algorithm will eventually give a prepattern β such that $\mathrm{PAT}(\beta) = \pi$ (except for possible renaming of variables).

The automatic learner initially has memory ? until it sees the first input $x \neq \#$; at which point its memory is $\mathrm{conv}(x, x, x, \ldots, x)$ (x appears $2n + 2$ times in the convolution).

In the following, suppose π is the target pattern. The invariants maintained by the learner on its memory $\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})$, after having seen input $T[k]$, with $\mathit{content}(T[k]) \neq \emptyset$ are as follows.

(I) x is the first element of $T[k]$ different from #.
(II) If, for all $w \in \mathit{content}(T[k])$, $w(\ell) = x(\ell)$, then $x_r(\ell) = x(\ell)$, for all $r \leq 2n$.
(* Intuitively, $x(\ell)$ appears to be a constant. *)
(III) For each $\ell < |x|$, the following are equivalent statements:
(a) There exists a $w \in \mathit{content}(T[k])$ such that ℓ is the least number for which $w(\ell) \neq x(\ell)$.
(b) There exists a $r \leq 2n$ such that $x_r(\ell) = s$.
Furthermore, in (b) above, such a r is unique and for all $r' \leq 2n$ with $r' \neq r$, $x_{r'}(\ell) = x(\ell)$.
Also, once $x_r(\ell) = s$, it never gets modified again.
Note that this property implies that, for any ℓ, there is at most one r such that $x_r(\ell) = s$.

(* Intuitively, exactly one of the x_r will be assigned the task of coding a variable in π. This assignment takes place when a w is received as input such that, for the least ℓ such that $\pi(\ell)$ is the corresponding variable, $w(\ell) \neq x(\ell)$ and $w(\ell') = x(\ell')$, for $\ell' < \ell$.*)

(IV) Suppose $x_r(\ell) = s$ and $\ell' > \ell$ is the least (if any) such that $x_r(\ell') = s$; if no such ℓ' exists, then let $\ell' = \infty$ for the following.
Let $S = \{w \in \mathit{content}(T[k]) : \ell$ is the least ℓ'' such that $w(\ell'') \neq x(\ell'')\}$.
Let $S' = \{\ell\} \cup \{\ell'' : \ell < \ell'' < \ell'$ and $x_r(\ell'') = v\}$.
(a) $\ell'' \in S'$ implies that for all $w \in S$, $[w(\ell'') = w(\ell)$ and $x(\ell'') = x(\ell)]$.
(b) $S' \supseteq \{\ell'' : \pi(\ell) = \pi(\ell'')\}$.
(* Intuitively, the set S' represents the possible locations where the variable $\pi(\ell)$ may appear in π.*)

Note that, if ℓ is the position where a variable $\pi(\ell)$ first appears in π, then once a string w is received as input such that the positions ℓ'' at which $w(\ell'') \neq x(\ell'')$ is exactly the positions in which a variable $\pi(\ell)$ appears in π, we will have by (IV) above that S' is exactly the set of positions at which variable $\pi(\ell)$ appears in π.

Once all such w, each corresponding to a variable in π, have been received, we will have by the above invariants that $\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n})$ is a prepattern and $\mathrm{PAT}(\mathrm{conv}(x, x_0, x_1, \ldots, x_{2n}))$ will be a pattern equivalent to π (except for variable renaming).

What remains is to show how the memory is updated to maintain the invariants. Recall that memory of M remains as ? until it first receives a non-# input x. At that time, the memory of the learner becomes $\mathrm{conv}(x, x, x, \ldots, x)$ (where x appears $2n+2$ times in the convolution). From then on, M does not update its memory on input # or input x. On receiving an input $y \neq x$, M does the following.

Let ℓ be least such that $y(\ell) \neq x(\ell)$.

(A) If there exists a r such that $x_r(\ell) = s$, then:
Let ℓ' be least such that $\ell' > \ell$ and $x_r(\ell') = s$; if no such ℓ' exists, then let $\ell' = |x|$.
Let $S'' = \{\ell'' : \ell < \ell'' < \ell', y(\ell'') = y(\ell)$ and $x(\ell'') = x(\ell)\}$.
For ℓ'' such that $\ell < \ell'' < \ell'$, if $\ell'' \notin S''$, then let $x_r(\ell'') = x(\ell'')$ (that is, if $x_r(\ell'') = v$, then it is reset to be $x(\ell'')$).

Note that invariants (I), (II), (III) and (IV) (a) are clearly maintained. For invariant (IV) (b) note that for all ℓ'' such that $\pi(\ell) = \pi(\ell'')$, we must have $y(\ell) = y(\ell'')$ and $x(\ell) = x(\ell'')$; thus, $\ell'' \in S''$ and (IV) (b) also holds.
(* Intuitively, for $\ell < \ell'' < \ell'$, this step removes the variable v at $x_r(\ell'')$, if it is found that $\pi(\ell)$ and $\pi(\ell'')$ cannot be the same variable. *)
Note that checking condition (A), and doing the update of memory as above is automatic.

(B) If there does not exist a r such that $x_r(\ell) = s$, then:
(* Intuitively, in this step we will assign an r to code the variable $\pi(\ell)$. Note that this is the first time that an input y has been received with ℓ being least ℓ' such that $y(\ell') \neq x(\ell')$. *)
Let $S = \{\ell'' : y(\ell'') = y(\ell)$ and $x(\ell'') = x(\ell)\}$. Note that S is a superset of ℓ'' such that $\pi(\ell) = \pi(\ell'')$.

For each r, let ℓ_r be the largest value $< \ell$, if any, such that $x_r(\ell_r) = s$ (if no such ℓ_r exists, then one takes ℓ_r to be -1). Similarly, for each r, let u_r be the least, if any, such that $u_r > \ell$ and $x_r(u_r) = s$ (if no such u_r exists, then take u_r to be $|x|$).
Without loss of generality assume that $\ell_0 \leq \ell_1 \leq \ldots \leq \ell_{2n}$.

(a) For r with $0 \leq r \leq n$ and for all ℓ' with $\ell \leq \ell' < u_r$, let $x_r(\ell') = x(\ell')$.
(* Intuitively, if $\ell_r \neq -1$, then variable $\pi(\ell_r)$ cannot appear in positions ℓ' with $\ell \leq \ell'$, as there are at least $n+1$ other variables $\pi(\ell_{r'})$, $n < r' \leq 2n$, and $\pi(\ell)$ which appear between locations ℓ_r (exclusive) and ℓ (inclusive). Thus, we can safely reset these places $\ell'' < u_r$ to $x(\ell')$, without violating invariant (IV) (b) for these variables. *)

(b) Let u be the median of u_r's.
Delete from S all elements $\geq u$.
(* Intuitively, if $u < |x|$, then variable $\pi(\ell)$ cannot appear in positions ℓ' with $u \leq \ell'$, as there are at least $n+1$ other variables $\pi(u_{r'})$, (for r' satisfying $\ell < u_{r'} \leq u$), which appear between locations ℓ (exclusive) and u (inclusive). Thus, we can safely deduce that $\pi(\ell)$ cannot appear in π at or beyond location u. *)

(c) Let r be such that $r \leq n$ and $u_r \geq u$.
(* Note that there exists such an r by u being median of $\{u_{r'} : r' \leq 2n\}$. *)
Now, by invariant (IV) (b) we have that, if $\ell_r \neq -1$, then variable $\pi(\ell_r)$ does not appear beyond location ℓ (using operation done in part (a) above, and invariant (IV) (b)). We thus assign x_r to code the variable $\pi(\ell)$.
For each $\ell' \in S - \{\ell\}$, let $x_r(\ell') = v$. Let $x_r(\ell) = s$.
For each $r' \neq r$, let $x_{r'}(\ell) = x(\ell)$. (* This is done to ensure invariant (III) — Note that this change is safe as $\pi(\ell)$ is not equal to any other variable $\pi(\ell''')$, with $\ell''' < \ell$, as ℓ is the least position at which y differs from x. *)

Based on the comments given above, it is easy to verify that the invariants are maintained. Now it follows, using the comment given after the invariants that the limiting value of the memory gives a prepattern β such that $\mathrm{PAT}(\beta)$ is π (except for possible renaming of variables).

Thus, M above learns $\mathcal{L}$. $\square$

Let U be a fixed plain universal Turing machine which maps strings to strings, that is, a partial-recursive function from strings to strings such that the complexity defined by it is optimal up to a constant; see the book of Li and Vitányi [28] for the existence of universal Turing machines and further background. Then, the plain Kolmogorov complexity of a string x is the length of the least string p such that $U(p) = x$. The plain Kolmogorov complexity of a string x relative to K, the halting problem, is the length of the least string p such that $U^K(p) = x$.

Theorem 25. *The class $\mathcal{L} = \{L \cup H : L \cap H = \emptyset \wedge L, H \in \mathcal{O}_0\}$ is not automatically learnable.*

Proof. Note that the patterns in which each character variable appears at most once are in $\mathcal{O}_0$. Let K denote the halting problem.

For any n, and $i \leq n$, let $\sigma_{i,n} \in \{0,1\}^n$ be such that the plain Kolmogorov complexity, relative to K, of $\sigma_{0,n}\sigma_{1,n} \ldots \sigma_{n,n}$ is at least n^2+n. Let $L_{i,n} = \{0^i 1 0^{n-i} \sigma_{i,n}\}$. Let $H_{i,n} = \{0,1\}^i \cdot 0 \cdot \{0,1\}^{2n-i}$.

The language $H_{i,n}$ is generated by a pattern which has i character variables followed by 0 followed by $2n-i$ character variables (where all the character variables in $H_{i,n}$ are distinct). Note that $L_{i,n}, H_{i,n} \in \mathcal{O}_0$ and $L_{i,n} \cap H_{i,n} = \emptyset$.

Fix an automatic learner M. We will show below that M fails to learn $L_{i,n} \cup H_{i,n}$, for some i, n with $i \leq n$.

Let $T'_{i,n}$ be a text for $H_{i,n}$ (obtained effectively from i, n). Let τ be a sequence of length $n+1$ such that $\tau(j) = 0^j 1 0^{n-j} \sigma_{j,n}$, for $j \leq n$. Let $T_{i,n} = \tau T'_{i,n}$. Note that $T_{i,n}$ is a text for $L_{i,n} \cup H_{i,n}$.

The memory of M, after receiving input τ, can be of length at most $2n + c(n+2)$, where c is a constant independent of n. To prove this, we inductively show that after receiving the m-th element of τ, the length of the memory of the learner is of length at most $2n + c \cdot (m+1)$, for c being greater than both the number of states of the automata accepting the graph of the learner M and the length of the initial memory of M. For $m = 0$, this claim clearly holds. Inductively, if the memory of M after receiving the m-th element is of length at most $2n + c \cdot (m+1)$, then as the automata accepts the graph of M, the length of the new memory can be at most the maximum of the length of the older memory and of the length of the new input plus the number of states of the automata. Thus, the memory of M after seeing $m+1$ elements of τ is bounded in length by $2n + c \cdot (m+2)$.

One can compute $\sigma_{i,n}$, using oracle K, by considering the final conjecture of M on input $T_{i,n}$. Hence, $\sigma_{i,n}$ can be computed, using oracle K, from i, n and the memory of M after seeing input τ. It follows that the plain Kolmogorov complexity of $\sigma_{0,n}\sigma_{1,n}\ldots\sigma_{n,n}$, relative to K is bounded by a function linear in n, a contradiction. Thus, no such learner M can exist. $\square$

We now consider the case of patterns having both string and character variables. We will only consider the case where each variable appears only once. Let $n \in \mathbb{N}$. Let $\mathcal{R}_n$ consist of the class of all pattern languages, where in the pattern each character variable or string variable occurs at most once and where there are at most $n-1$ items (characters or variables of either type) after the first occurrence of a string variable, if any. It can be shown that every automatic family of pattern languages, containing character and/or string variables, generated by patterns without repeating variables is contained in some $\mathcal{R}_n$.

We now show that $\mathcal{R}_n$ has an automatic learner. Let S_n denote the set of all patterns of length at most n, starting with a string variable and having each variable at most once. Let $\mathcal{O}'$ denote the set of all languages which are generated by patterns involving only character variables, each appearing at most once.

Let $\mathcal{R}_{n,\pi}$, for $\pi \in S_n$, contain all languages of the form $L \cdot Lang(\pi)$ with $L \in \mathcal{O}'$.

Then $\mathcal{R}_n$ is the union of $\mathcal{O}'$ and the classes $\mathcal{R}_{n,\pi}$, $\pi \in S_n$. Using Proposition 10, it suffices to give automatic consistent and confident learners for $\mathcal{O}'$ and $\mathcal{R}_{n,\pi}$, $\pi \in S_n$.

Proposition 26. *$\mathcal{O}'$ is learnable by a consistent and confident automatic learner.*

Proof. The hypothesis space $\mathcal{H}$ consists of the following languages: $H_{\text{emp}} = \emptyset$, $H_{\text{comp}} = \Sigma^*$ and, for $\pi \in (\Sigma \cup \{@\})^*$, $H_\pi = Lang(\pi)$, where each appearance of @ in π denotes a distinct character variable (that is all variables appearing in π are assumed to be distinct).

The learner N_0 for $\mathcal{O}'$ conjectures emp until it sees the first datum x. From then onwards, the learner maintains in memory a string $a_0a_1\ldots a_{i-1}$ of length $i = |x|$, where for $r < i$, $a_r = x(r)$ if all the strings y observed so far have $y(r) = x(r)$; otherwise a_r is @ representing a character variable. If only strings of length $|x|$ have been observed so far, then N_0 conjectures $a_0a_1\ldots a_{i-1}$, else N_0 conjectures comp. It is easy to verify that the above learner is automatic, consistent and confident and learns $\mathcal{O}'$. $\square$

Proposition 27. *For all $n > 0$, for $\pi \in S_n$, $\mathcal{R}_{n,\pi}$ is learnable by a consistent and confident automatic learner.*

Proof. Fix $n > 0$ and $\pi \in S_n$. The hypothesis space $\mathcal{H}$ consists of the following languages: $H_{\text{emp}} = \emptyset$, $H_{\text{comp}} = \Sigma^*$ and, for $\alpha \in (\Sigma \cup \{@\})^*$, $H_\alpha = Lang(\alpha) \cdot Lang(\pi)$, where each appearance of @ in α denotes a distinct character variable.

The learner N_π for languages in $\mathcal{R}_{n,\pi}$, $\pi \in S_n$ starts with the conjecture emp. After the first datum x is observed, the memory of the learner is of the form $a_0a_1\ldots a_{i-1}$, where i is the largest number such that $\Sigma^i \cdot Lang(\pi)$ contains all data observed so far. Furthermore, for $r < i$, $a_r = x(r)$, if all the strings y seen so far have $x(r) = y(r)$. Otherwise, a_r is @, representing a character variable. If all data observed so far are in $Lang(\pi)$, then the conjecture is $a_0a_1\ldots a_{i-1}$; otherwise the conjecture is comp. Here, note that $\Sigma^* \supseteq Lang(\pi) \supset \Sigma \cdot Lang(\pi) \supset \Sigma^2 \cdot Lang(\pi) \supset \ldots$, which permits the learner N_π to update the i monotonically: initially i is at most $|x|$; later, whenever a new datum w is observed, the new value of i is the minimum of the old value of i and the largest j with $\Sigma^j \cdot Lang(\pi)$ containing w. So the memory as above can be maintained by the learner automatically. Note that the language conjectured by N_k also grows monotonically. Furthermore, N_k is consistent and confident. $\square$

Learnability of $\mathcal{R}_n$ now follows using Propositions 26, Propositions 27 and Proposition 10.

Corollary 28. *For all $n > 0$, $\mathcal{R}_n$ is learnable by a consistent and confident automatic learner.*

6 Conclusion

In this paper we considered learnability of automatic subclasses of pattern languages. Such classes are contained in $\mathcal{P}_n$ for some n. We showed that each such class can be learnt by a consistent and confident automatic learner where the memory of the learner is bounded by the length of the first datum seen. We also investigated when the class of unions of two languages from $\mathcal{P}_n$ is automatically learnable and got an affirmative answer for the case that the alphabet size is at least three.

Additionally, we considered character variables and showed that the class $\mathcal{O}_n$, where the number of distinct character variables between any two same character variables is bounded by n, has an automatic learner. We showed that no automatic learner can learn the class of the unions of two languages from $\mathcal{O}_0$.

It is open at this point to which degree we can extend our result about learning of unions of languages in $\mathcal{P}_n$; in particular whether Theorem 21 has a counterpart for the learning of unions of three or more languages from $\mathcal{P}_n$.

Acknowledgment

We would like to thank the referees of both, the conference LATA 2011 and this journal, for their careful proofreading and their useful feedback.

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. Dana Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.
4. Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.
5. Janis Bārzdiņš. Inductive inference of automata, functions and programs. *Proceedings of the 20th International Congress of Mathematicians, Vancouver*, pages 455–560, 1974. In Russian. English translation in American Mathematical Society Translations: Series 2, 109:107–112, 1977.
6. Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
7. Achim Blumensath. *Automatic structures.* Diploma thesis, RWTH Aachen, 1999.
8. Achim Blumensath and Erich Grädel. Automatic structures. *15th Annual IEEE Symposium on Logic in Computer Science*, LICS 2000, pages 51–62, IEEE Computer Society, 2000.
9. John Case, Sanjay Jain, Rüdiger Reischuk, Frank Stephan and Thomas Zeugmann. Learning a subclass of regular patterns in polynomial time. *Theoretical Computer Science*, 364:115–131, 2006.
10. Henning Fernau. Identification of function distinguishable languages. *Theoretical Computer Science*, 290:1679–1711, 2003.
11. Rusins Freivalds, Efim Kinber and Carl H. Smith. On the impact of forgetting on learning machines. *Journal of the Association of Computing Machinery*, 42:1146–1168, 1995.
12. E. Mark Gold. Language identification in the limit. *Information and Control* 10:447–474, 1967.
13. Tom Head, Satoshi Kobayashi and Takashi Yokomori. Locality, reversibility, and beyond: learning languages from positive data. *Algorithmic Learning Theory, Ninth International Conference*, ALT 1998. Springer LNAI 1501:191–204, 1998.
14. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.
15. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
16. Sanjay Jain, Qinglong Luo, Pavel Semukhin and Frank Stephan. Uncountable automatic classes and learning. *Theoretical Computer Science*, 412(19):1805-1820, 2011.

17. Sanjay Jain, Qinglong Luo and Frank Stephan. Learnability of automatic classes. *Language and Automata Theory and Applications*, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings. Springer LNCS 6031:321-332, 2010. Also as Technical Report TRA1/09, School of Computing, National University of Singapore, 2009.
18. Sanjay Jain, Yuh Shin Ong, Shi Pu and Frank Stephan. *On automatic families.* In T. Arai, Q. Feng, B. Kim, G. Wu and Y. Yang, *Proceedings of the 11th Asian Logic Conference*, ALC 2009, pages 94–113, World Scientific, 2011.
19. Sanjay Jain, Eric Martin and Frank Stephan. Robust learning of automatic classes of languages. In J. Kivinen, C. Szepesvari, E. Ukkonen and T. Zeugmann, *Algorithmic Learning Theory, 22nd International Conference*, ALT 2011. Springer, LNAI 6925:55–69, 2011.
20. Sanjay Jain, Daniel N. Osherson, James S. Royer and Arun Sharma. *Systems That Learn.* MIT Press, 2nd Edition, 1999.
21. Bakhadyr Khoussainov and Mia Minnes. Three Lectures on Automatic Structures. *Proceedings of Logic Colloquium 2007. Lecture Notes in Logic*, 35:132-176, 2010.
22. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity*, LCC 1994. Springer LNCS 960:367–392, 1995.
23. Efim Kinber and Frank Stephan. Language learning from texts: mind changes, limited memory and monotonicity. *Information and Computation*, 123:224–241, 1995.
24. Takeshi Koshiba. Typed pattern languages and their learnability. In Paul Vitányi, editor, *Computational Learning Theory, second European Conference*, EuroCOLT 1995. Springer LNAI 904:367–379, 1995.
25. Steffen Lange and Rolf Wiehagen. Polynomial time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.
26. Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53:88–103, 1996.
27. Steffen Lange, Thomas Zeugmann and Sandra Zilles. Learning indexed families of recursive languages from positive data: a survey. *Theoretical Computer Science*, 397:194–232, 2008.
28. Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Third Edition, Springer, 2008.
29. 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.
30. 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.
31. Daniel Reidenbach. A non-learnable class of E-pattern languages. *Theoretical Computer Science*, 350:91–102, 2006.
32. Sasha Rubin. *Automatic Structures.* Ph.D. Thesis, The University of Auckland, 2004.
33. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.

34. 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.
35. Kenneth Wexler and Peter W. Culicover. *Formal Principles of Language Acquisition.* Cambridge, Massachusetts, The MIT Press, 1980.
36. Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik* (EIK) 12:93–99, 1976.