Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 397 (2008) 150–165

Theoretical Computer Science

www.elsevier.com/locate/tcs

Developments from enquiries into the learnability of the pattern languages from positive data

Yen Kaow Ng[a], Takeshi Shinohara[b,*]

[a] 26A, Jalan Lim Swee Aun, 34000, Taiping, Perak, Malaysia
[b] Kyushu Institute of Technology, Department of Artificial Intelligence, Iizuka, 820, Japan

Abstract

The pattern languages are languages that are generated from patterns, and were first proposed by Angluin as a non-trivial class that is inferable from positive data [D. Angluin, Finding patterns common to a set of strings, Journal of Computer and System Sciences 21 (1980) 46–62; D. Angluin, Inductive inference of formal languages from positive data, Information and Control 45 (1980) 117–135]. In this paper we chronologize some results that developed from the investigations on the inferability of the pattern languages from positive data.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Inductive inference; Positive data; Pattern languages; Chronology

1. In the beginning

The formal study of the so-called *inductive inference* started with Gold's paper in 1967 [18], from which we quote,

> A class of possible languages is specified, together with a method of presenting information to the learner about an unknown language, which is to be chosen from the class. The question is now asked, "Is the information sufficient to determine which of the possible languages is the unknown language?" Many definitions of learnability are possible, but only the following is considered here. Time is quantized and has a finite starting time. At each time the learner receives a unit of information and is to make a guess as to the identity of the unknown language on the basis of the information received so far. This process continues forever. The class of languages will be considered *learnable* with respect to the specified method of information presentation if there is an algorithm having the following property: Given any language of the class, there is some finite time after which the guesses will all be the same and they will be correct.

In the paper Gold introduced two information presentation models. In the first (called *inductive inference from text or positive data*), the learner is given, one by one, with or without repetition, ad infinitum, all the sentences in the

* Corresponding author.
E-mail addresses: kalngyk@gmail.com (Y.K. Ng), shino@ai.kyutech.ac.jp (T. Shinohara).

0304-3975/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.02.028

target language; and in the second (called *inductive inference from informant*), the learner is additionally informed of the sentences that are absent from the language. In both cases, the learner is said to *identify* the target language *in the limit* if and only if regardless of the order and frequency these sentences are presented, at some unspecified time the learner outputs a grammar for that language and then never changes its mind. The learner is said to *identify a class of languages* if and only if it identifies every language in that class. Finally, a class is said to be *learnable*, or *inferable*, if and only if there exists a learner that identifies the class. We are concerned only with learnability with respect to the former, inductive inference from positive data, in this paper.

Gold observed that in the case of inductive inference from positive data, even the class of languages that contains all of the finite languages and just one infinite language, is not learnable. Since the superclass of an unlearnable class is unlearnable, this result was misunderstood to mean that not many interesting classes are learnable from positive data alone. This was to remain so until Angluin's introduction of the *pattern languages* [1,2]. We quote an excerpt from linguist and computer scientist Costa Florêncio's doctorate thesis [16]:

> In the seventies (sometimes referred to as the Dark Age of formal learning theory) people like Bārzdiņš, Freivalds, Wiehagen and Jantke worked mostly on characterizing learnable classes under different learning criteria, most of this work was published in German or Russian. Work by Angluin in the early eighties rekindled interest in the paradigm. She presented non-trivial learnable classes that cross-cut the Chomsky Hierarchy, demonstrating that the pessimism following Gold's earlier results was due to misinterpretation.

The pattern languages are languages that are generated from *pattern*s. A pattern is a string over a finite set Σ of alphabet and an infinite set V of variables. The language generated by a pattern p, written $L(p)$, are the strings obtainable by substituting variables in p with non-empty strings over Σ. As an example let $\Sigma = \{a, b\}$, $V = \{x_1, x_2, \ldots\}$ and let pattern p be ax_1x_1b. $L(p)$ would then be all the strings that begin with a, followed by a string over Σ repeated twice, and then ending in b, for instance, $ababab$. Angluin showed that the class of pattern languages is incomparable with the class of regular languages and the class of context-free languages [1]. We note in particular these other results by Angluin (Theorem 3–11):

Definition 1 (*Indexed Families [2]*). A class of non-empty languages $\mathcal{L} = (L_i)_{i\in\mathbb{N}}$ is *an indexed family* if and only if there exists a computable function f such that for each $i \in \mathbb{N}$ and for each string w over Σ, $f(i, w) = 1$ if $w \in L_i$ and $f(i, w) = 0$ otherwise.

Note that the pattern languages form an indexed family. Any indexed family is inferable from informant — a result implied in Gold's paper [18] (see Section 3 in [70] for a more detailed discussion). Learnability of the indexed families from positive data, on the other hand, can be characterized using the following notion.

Definition 2 (*Finite Tell-tale [2]*). A *finite tell-tale* for a language L within a class $\mathcal{L}$ is a finite set $S \subseteq L$ such that there exists no language $L' \in \mathcal{L}$ where $S \subseteq L' \subset L$.

When a finite tell-tale for L is in the examples, then the examples cannot be for a proper subset of L.

Theorem 3 (*Theorem 1 in Angluin [2]*). *An indexed family $(L_i)_{i\in\mathbb{N}}$ is learnable from positive data if and only if there exists a procedure, effective in each i, that enumerates a finite tell-tale for L_i within the class.*

To see how a finite tell-tale helps in learning an indexed family, observe that given an indexed family $\mathcal{L} = (L_i)_{i\in\mathbb{N}}$, for any $L \in \mathcal{L}$, when presented with subsequently more and more examples from L one can find, in the limit, a language of the least index $L_j \in \mathcal{L}$ that is a superset of L — by simply finding the least j such that L_j contains all the examples presented so far. (This is because, for every $k < j$, if $L - L_k \neq \emptyset$, then some $x \in L - L_k$ will eventually be in the examples.) Hence to correctly identify L, it suffices for one to further be able to dismiss L_j that are proper supersets of L, and this can be done by accepting L_j only if its finite tell-tale is in the examples.

By applying Theorem 3, we can show classes with the following property of *finite thickness* to be learnable from positive data.

Definition 4 (*Finite Thickness [2,66]*). A class $\mathcal{L}$ of languages is said to have *finite thickness* if and only if for each string w over Σ, the set $\{L \in \mathcal{L} \mid w \in L\}$ is finite.

Theorem 5 (*Corollary 2 in [2]*). *Any indexed family with finite thickness is learnable from positive data.*

Since a string of length n cannot be generated by patterns longer than n:

Theorem 6 (*Example 1 in [2]*). *The class of pattern languages has finite thickness.*

By Theorems 5 and 6, we get:

Theorem 7. *The class of pattern languages is learnable from positive data.*

Hence Theorem 7 demonstrated that the pattern languages are learnable from positive data. We next look at methods for learning the class. Angluin suggested finding *descriptive languages* [2] (or *descriptive patterns* [1] in the case of the pattern languages) as a strategy for learning the indexed families from positive data [2].

Definition 8 (*Descriptive Pattern [1], Descriptive (or minimal) Language* [2,50]). (1) We say that a pattern p is *descriptive* of a set S of strings over Σ if and only if:

(i) $S \subseteq L(p)$, and
(ii) no other pattern p' where $S \subseteq L(p')$ has $L(p') \subset L(p)$.

Or more generally,
(2) Given an indexed family $\mathcal{L}$, we say that a language $L \in \mathcal{L}$ is a *minl* (minimal language) for a set S if and only if:

(i) $S \subseteq L$, and
(ii) no other language $L' \in \mathcal{L}$ where $S \subseteq L'$ has $L' \subset L$.

One of the reasons for using such languages as hypotheses being simply, as Angluin puts it [1]: "Intuitively, no pattern q gives a strictly "closer fit" to the sample S than p does." Furthermore,

Theorem 9 (*Modified from Theorem 5 in [2]*). *Given an indexed family $\mathcal{L}$ with finite thickness, then a learner that, on any non-empty finite input set S, outputs a language $L \in \mathcal{L}$ that is a minl for S, learns $\mathcal{L}$ from positive data.*

Note that the learner in Theorem 9 always output a language that contains all of the examples presented. Such a learner is said to be *consistent*. Furthermore, when given more examples, the learner will not change a language it hypothesized as long as the language continues to contain all of the examples presented. Such a learner is said to be *conservative*, since it does not change its mind as long as its hypothesis remains consistent. It is not difficult to see that any indexed family is learnable by a consistent learner (a consequence of Theorem 3), however, not every learnable indexed family is learnable by a conservative learner (see Theorem 4 of [2]).

Nevertheless, it may not be easy to compute *minl*s, as the following results suggest.

Theorem 10 (*Theorem 3.6 in [1]*). *Given any string s over Σ and pattern p, it is NP-complete to decide if $s \in L(p)$.*

Theorem 11 (*Theorem 4.2 in [1]*). *For any finite set S of strings over Σ, if $P \neq NP$, then there is no polynomial-time algorithm to find a pattern p of maximum possible length that is descriptive of S.*

Hence while Theorem 7 shows that the pattern languages are learnable from positive data, Theorems 10 and 11 suggest that it may be difficult to devise *efficient* learning algorithms for the class. (It is also worth noting that Jiang et al. showed that given two arbitrary patterns, it is in general undecidable whether the language of one is included in that of the other [21].) Here, a learning algorithm is said to be efficient if and only if it takes time polynomial to the input sample size in updating its hypothesis. Note that others have made the distinction between this and the total time required for a learning algorithm to converge (called *total learning time*) (cf. [15,46,69]).

For subclasses of the pattern languages, Angluin first showed that for the class of the languages generated by patterns with at most one variable (called *one-variable patterns*), there is an efficient algorithm for finding *minl*s [1]. Later, Shinohara found an efficient *minl* computation for the languages of the patterns where each variable appears at most once (called *regular patterns*) [50]. For these regular patterns, *minl*s can be efficiently computed even when substitutions for empty strings are allowed (the resultant languages are called *extended* or *erasing*) [50]. The languages generated by a superclass of the one-variable patterns and the regular patterns, called *non-cross patterns*, were also shown to have efficiently computable *minl*s [51].

Shinohara also found that the class which consists of all the unions of any two pattern languages, is learnable from positive data [52]. These results were subsequently presented at the conference "Mathematical Methods of

Specification and Synthesis of Software Systems" which took place in Wendisch Rietz, Germany, in the year 1985 [53]. A few problems interested researchers at that time, including:

(1) Are there efficient algorithms for learning the pattern languages?
(2) Can the unions of more than 2 pattern languages be learned from positive data?

The first question was answered in the positive by Lange and Wiehagen [28] with an interesting algorithm that occasionally output hypotheses that are inconsistent with the input. This algorithm aided many other enquiries, which we discuss in Section 3.

The second question was answered in 1989 by Keith Wright (a student of Robert Daley, who was present at the 1985 conference), using the following notion of *finite elasticity*.

Definition 12 (*Finite Elasticity [66,32]*). A class $\mathcal{L}$ of languages has *infinite elasticity* if and only if there exists an infinite sequence of pairwise distinct elements $w_0, w_1, w_2, \ldots$ and an infinite sequence of pairwise distinct languages $A_1, A_2, \ldots$ such that for each $k \in \mathbb{N}^+$, $\{w_i \mid i < k\} \subseteq A_k$, but $w_k \notin A_k$. $\mathcal{L}$ is said to have *finite elasticity* if and only if $\mathcal{L}$ does not have infinite elasticity.

Note that a class of languages with finite thickness has finite elasticity: if it has infinite elasticity then for each $k \in \mathbb{N}$, w_k will be included in the languages $A_{k+1}, A_{k+2}, \ldots$, which contradicts its finite thickness. The following are Wright's results on finite elasticity.

Theorem 13 (*Corollary 11 in [66]*). *If an indexed family $\mathcal{L}$ has finite elasticity, then for any $k \in \mathbb{N}^+$, the class consisting of the unions of up to k languages from $\mathcal{L}$ has finite elasticity.*

Theorem 14 (*Theorem 8 in [66]*). *An indexed family with finite elasticity is learnable from positive data.*

In fact, an indexed family with finite elasticity can be learned consistently and conservatively, via the learner in Theorem 9 (see Lemma 9 in [9]). Finally, by Theorems 6, 13 and 14, we get:

Theorem 15 (*Corollary 12 in [66]*). *For any $k \in \mathbb{N}^+$, the class of the unions of up to k pattern languages is learnable from positive data.*

At that time, others had also started to consider a superclass of the pattern languages, that is, the class of languages definable by the *elementary formal systems* (EFS) introduced by Smullyan [59]. An EFS is a logic program, like a Prolog program, but with patterns as terms. Shinohara showed the languages definable by a subclass of the EFSs called *primitive formal systems* [4] to be learnable from positive data [54]. (A primitive formal system consists of exactly two clauses defined over a same unary predicate: one without body, and one with terms in the body only as distinct variables appearing in the head.) At the time Wright was presenting the results on finite elasticity, Shinohara was presenting a paper with Arikawa and Yamamoto, showing a class of EFS definable languages in each level of the Chomsky hierarchy, and how the EFSs can be learned from informant [5] using Shapiro's model inference system [49]. Wright's result then enabled Shinohara to show a richer subclass of the EFSs, called *length-bounded EFSs*, to be learnable from positive data. Eventually, many others have shown through similar techniques that the classes definable by many interesting programming systems are learnable from positive data. We discuss these rather involving results only briefly in Section 2.

Learning by *minl*, as described in Theorem 9, continued to interest researchers, because there are properties, such as conservativeness and consistency, that we can expect from such a learner [2]. For this reason, in the early 90s, Arimura et al. worked on an efficient algorithm for finding such *minl*s for the class of the bounded unions of the *tree pattern languages* [9,10,12]. Their algorithm looks for minimality of a less demanding ordering, called *subsumption*, of the hypothesis space, than the ordering by the inclusion relation of *minl*. This finally resulted in a general algorithm for finding *minl* for any general hypothesis space where the subsumption relation can be well related to the inclusion relation. We discuss this in Section 5, and in Section 6 we discuss attempts on using the algorithm on biological samples.

The learnability of the pattern languages continued to interest researchers. While the class of the pattern languages was then known to be learnable from positive data, it was completely open on whether the erasing pattern languages are learnable from positive data or not. Learnabilities of unions, bounded or unbounded, of the pattern languages are also interesting since they are related to the inferability of the EFSs [54–56]. More precisely, unions of the

	Pattern language	Logic programming
‘79	Pattern languages introduced, efficient *minl* for one-variable pattern languages [1]	
‘80	Learnability of many indexed families (including pattern languages) [2]	
‘82	Efficient *minl* for the extended (or erasing) and non-erasing regular pattern languages [50]; and for the non-cross pattern languages [51]	
‘83	Learnability of the unions of two pattern languages [52]	
‘85	Invited talk on open problems by Shinohara [53]	
‘86		Learnability of primitive formal systems [54]
‘89	Finite elasticity shows learnability for bounded unions [66,32]	Subclasses of EFSs correspond to every class in Chomsky hierarchy [5]
‘90	Polynomial-time learning through Lange and Wiehagen algorithm (LWA) [28]	Bounded finite thickness as sufficient condition for learning [55,56] (classes include length-bounded EFSs, linear Prolog, CSGs of bounded productions)

Fig. 1. Chronological view of developments before 1991.

pattern languages form subclasses of the EFSs, and hence their learnability could provide insights on how to learn the EFSs (e.g. Wright’s results mentioned earlier), while their unlearnability would imply unlearnability for their EFS superclasses. Likewise, the inferability of the unions of the tree pattern languages is also interesting because it is related to the inferability of Prolog programs. In Section 4 we discuss efforts in showing the learnability of these classes. A timeline of the results surveyed in this paper is given in Figs. 1 and 3.

2. Programs that are learnable from positive data

The first result on the learnability of logic programs is due to [54], where it is shown that the *primitive formal systems* are learnable from positive data. After Wright’s work on finite elasticity, these results were extended to any EFS of (1) at most a bounded number of clauses, where (2) the left-hand side (head) of each clause is of length longer than its right-hand side (body) under every substitution [55,56]. Such an EFS is called *length-bounded*. We give a brief description of how learnability is shown for this class, without formally defining the EFSs:

It can be shown that for the length-bounded EFSs, for any given set of ground atoms S, there are only finitely many such EFSs with all non-redundant clauses, that generate S (such a property is called *bounded finite thickness*). This property allows one to show finite elasticity of the class. The length-bounded EFSs of a single clause has finite thickness and hence finite elasticity. By way of contradiction suppose that the length-bounded EFSs of up to n clauses has infinite elasticity (say witnessed by the sequences $w_0, w_1, \ldots,$ and $A_1, A_2, \ldots$), then one can show that its subset of up to only $n-1$ clauses has infinite elasticity also, by using the fact that there are only finitely many EFSs in $A_1, A_2, \ldots,$ that are not redundant with respect to any finite set from $w_0, w_1, \ldots$. That is, there must be infinitely many EFSs in $A_1, A_2, \ldots,$ that are redundant, and hence by removing a clause from each of these EFSs we can form another infinite sequence $A'_1, A'_2, \ldots,$ of EFSs of at most $n-1$ clauses.

Using similar techniques, it was shown at the same time that the class of Prolog programs with similar length bounds (called *linear Prolog* [49]), as well as the class of languages defined from context-sensitive grammars of up to at most a bounded number of productions, are all learnable from positive data. Furthermore, based on their work on the tree pattern languages (see Section 5), Ishizaka et al. considered single predicate Prolog programs with similar constraint as the primitive formal systems, called *primitive Prolog programs*, and showed the class of such programs to be efficiently inferable from positive data [19]. Finally, in [7], Arimura and Shinohara extended the class of learnable Prolog programs to a class called *linear covering Prolog programs*.

In 1995, Krishna Rao extended tremendously on these results and defined richer classes of Prolog programs (*hereditary*, *reductive* and *linearly-moded programs*) that are inferable from positive data [39]. The linearly-moded Prolog programs form a class that properly includes the linear Prolog programs and the linearly covering Prolog programs (which are incomparable), studied earlier by Shinohara and Arimura.

Extending on the EFS, Lange et al. [26] considered EFSs with the addition of a certain non-monotonic form of negation, resulting in programs called *advanced elementary formal systems* (AEFS). However, they found that for the AEFSs, the resultant class with the same constraints (length-bounded and bounded number of clauses) as the learnable EFS definable class reported in [55,56], turns out to be learnable from positive data if and only if the programs consist of only a single clause.

The next breakthrough in the learning of logic programs came from Martin and Sharma, who generalized earlier approaches using the notions of *strongly admissible* and *safe programs* [29], hence showing many less restrictive classes of Prolog programs to be learnable from positive data. Most recently, Krishna Rao provided yet another contribution with the notion of *recursion-bounded programs* [41], which includes many practical Prolog programs.

Programs other than EFSs and Prolog programs have also been actively researched. Mukouchi and Sato [34] showed some subclasses of the *simple formal systems* [4], the *regular formal systems* [54], and the *Simple H languages* [30], to be learnable from positive data. Meanwhile, Krishna Rao worked on the *term rewriting systems* (TRS) [14] and has demonstrated the learnability of several non-trivial subclasses of TRS definable languages [40,42].

3. Lange and Wiehagen algorithm

The *Lange and Wiehagen algorithm* (LWA) was proposed in 1990 as an algorithm that learns the pattern languages efficiently [28]. The algorithm has been studied extensively, and the discussion here should be taken to be only a brief reference to those studies. The algorithm works upon the observation that for any pattern language $L(p)$, p can be inferred using only the elements in $L(p)$ of the same length as p, that is, the elements of the shortest length in $L(p)$. In fact, on each given example, the algorithm uses only this example and its previous hypothesis to compute a new hypothesis. That is, in a sense, all the necessary information from the earlier examples had been "coded" in its hypothesis. The following lists the LWA algorithm, where π is the previous pattern hypothesized and s is the latest example. In the following, the length of a pattern p is denoted $|p|$, and $p(i)$ (where $1 \leq i \leq |p|$) denotes the ith letter in p.

```
LWA($s$)
 (1) If there is no previous hypothesis (i.e. $s$ is first example), hypothesize $L(s)$.
 (2) If $|s| > |\pi|$, continue to hypothesize $L(\pi)$.
 (3) If $|s| < |\pi|$, hypothesize $L(s)$. (All information in $\pi$ is abandoned.)
 (At this point $|s| = |\pi|$.)
 (4) Let $\varrho$ be a pattern of length $|\pi|$, where for $1 \leq i \leq |\pi|$,
     $\varrho(i) = \pi(i)$ if $\pi(i) = s(i)$
     $\varrho(i) = x_j$   if $\pi(i) \neq s(i)$ and $(\exists k < i)[\varrho(k) = x_j, s(k) = s(i), \pi(k) = \pi(i)]$
                (Use an existing variable to account for $s(i)$ if possible.)
     $\varrho(i) = x_m$   otherwise, where $m$ is the number of
                          different variables in $\varrho(1) \ldots \varrho(i-1)$.
                (Otherwise use a new variable to account for $s(i)$.)
     Hypothesize $L(\varrho)$.
```

The LWA is interesting in several aspects, most of all being, perhaps, the fact that it occasionally outputs inconsistent hypotheses, that is, hypotheses that do not account for all the examples presented. The question of consistency and learnability has been investigated since Bārzdiņš's results [13] that show that there are classes of recursive functions that can be learned only by inconsistent learners — mainly due to the undecidability of consistency in a general setting. Consistency is however decidable for the indexed families, and we have seen through Theorem 3 that any indexed family that is learnable from positive data can be learned by a consistent learner. The question then becomes: can a given class of indexed family be learned from positive data, efficiently and consistently? Ko and Hua made the conjecture that for the pattern languages, this is not possible even in the case of patterns that consist only of at most two variables [22]. In this context, LWA provided an excellent example of an inconsistent efficient learner for an indexed family where there is no known efficient and consistent learner. Further investigation of the LWA in this respect is pursued in [64,65]. Note that in contrast to learning from positive data, it has been shown that no efficient learner can infer the pattern languages consistently from an informant (Theorem 7(2) in [65]).

Another interesting aspect of the LWA is that its computation is based only on its previous hypothesis and the latest example given. Such a learner is called an *iterative learner* [63], and is of practical importance since it takes into account the limitation of space in a realistic computation.

The LWA also gives us the opportunity of a learning algorithm where its convergence can be determined via the input it has received, and this allows us to estimate the probability that a learner has converged [68,69].

Besides these, the LWA also provided good arguments to many other investigations, e.g. the study of *good examples* [17,27].

4. Learnability of the pattern languages

In this Section we discuss the learnability of subclasses of the pattern languages and their unions from positive data. We do not delve into the issues of efficiency since these are discussed via the two algorithms, LWA and MMG, respectively in Sections 3 and 5.

As mentioned in Section 1, the class of the pattern languages was first shown to be learnable by Angluin [2]. Shinohara extended this result to the class of the unions of two pattern languages [52], and initiated the investigations on languages that allow variables to be replaced with the empty string (that is, the extended or erasing pattern languages), as well as the languages that are generated by the patterns where each variable appears at most once (that is, the regular patterns) [50]. Wright extended the result on unions, showing that the class of unions of any bounded number of pattern languages is learnable [66]. This, however, does not include the unbounded unions, for which a negative result was obtained by Shinohara and Arimura [57]. To show the learnability of unbounded unions, the following sufficient condition was suggested in [57].

Definition 16 (*Infinite Anti-chain [24,57]*). A class of languages $\mathcal{L}$ *has no infinite anti-chain with respect to set inclusion* if and only if there does not exist an infinite collection of distinct languages $\{A_i \in \mathcal{L} \mid i \in \mathbb{N}\}$, such that for all $i, j \in \mathbb{N}$, $i \neq j$, $A_i \not\subseteq A_j$ and $A_j \not\subseteq A_i$.

Theorem 17 (*Lemma 22 in [57]*). *Let $\mathcal{L}$ be an indexed family with finite thickness. If $\mathcal{L}$ has no infinite anti-chain with respect to set inclusion, then the class of the unbounded unions of languages drawn from $\mathcal{L}$ is learnable from positive data.*

Using Theorem 17, it can be shown through Higmann's Theorem [24] that the following class is learnable from positive data.

Definition 18 (*Regular Patterns with Bounded Constant Segments [57]*). For $l \in \mathbb{N}^+$, let $\mathbf{RP}_l$ be the set of regular patterns of the form $w_0x_1w_1 \cdots w_{n-1}x_nw_n$ where $n \geq 0$, $x_0, \ldots, x_n$ are mutually distinct variables, and $w_0, \ldots, w_n$ are strings of up to length l over Σ.

Theorem 19 (*Theorem 27 and 28 in [57]*). *For any $l \in \mathbb{N}^+$, the class of unbounded unions of the languages (or erasing languages) generated by the patterns in $\boldsymbol{RP}_l$ is learnable from positive data.*

We note another property which is sufficient to show learnability. Wright noted in [66] that an indexed family with finite elasticity is learnable from positive data. Sato [47], Kobayashi and Yokomori [23] independently related finite elasticity to the following notion of *characteristic sets* introduced by Angluin.

Definition 20 (*Characteristic Set [3]*). A *characteristic set for a language L within a class $\mathcal{L}$* is a finite set S_L such that for each $L' \in \mathcal{L}$, $S_L \subseteq L' \Rightarrow L \subseteq L'$.

Hence a characteristic set S of a language L within a class $\mathcal{L}$ is a finite set of elements in L such that if any language in $\mathcal{L}$ includes S, then it must also include L.

Theorem 21 (*Theorem 2.24 in Sato [47], Proposition 7 in Kobayashi and Yokomori [23]*). *A class $\mathcal{L}$ of languages has finite elasticity if and only if every set of strings over Σ has a characteristic set within $\mathcal{L}$.*

Language class	learnable	fin. thick.	fin. elas.	char. set
bounded unions of the pattern languages	Yes	No	Yes	Yes
unbounded unions of languages generated from RP_l for any $l \in \mathbb{N}^+$ (erasing or non-erasing)	Yes	No	Yes	Yes
unbounded unions of the *substring pattern languages* (erasing or non-erasing)	Yes	No	No	Yes
unbounded unions of the proper one-variable pattern languages (erasing or non-erasing)	No	No	No	No
unbounded unions of the proper regular pattern languages (erasing or non-erasing)	No	No	No	No

Fig. 2. Learnability of unions of the pattern languages.

It is easy to see how the existence of a characteristic set for each language in a family $\mathcal{L}$ within $\mathcal{L}$ implies that the condition in Theorem 3 can be fulfilled, and this shows, via Theorem 21, that a class with finite elasticity is learnable from positive data. In fact, this also shows elegantly why a class with finite elasticity can be learned by the learner in Theorem 9. (More results on finite elasticity can be found in [47,61].)

Note that the requirement for each language in $\mathcal{L}$ to have a characteristic set within $\mathcal{L}$ is less strict than the requirement of finite elasticity, as witnessed by the class of unbounded unions of languages generated from patterns of the following form.

Definition 22 (*Substring Patterns*). A pattern is a *substring pattern* if and only if it is of the form $p = x_1 w x_2$ where x_1 and x_2 are distinct variables, and w is a string over Σ.

Theorem 23 (*Theorem 16 and 17 in [57]*). *The class of unbounded unions of the erasing or non-erasing languages generated from the substring patterns* (1) *has infinite elasticity, but* (2) *is nonetheless, learnable from positive data.*

However, the unbounded unions of the languages generated from the following patterns turned out to be unlearnable from positive data.

Definition 24 (*Proper Patterns [57]*). A pattern is said to be *proper* if and only if it contains at least one variable.

Theorem 25 (*Theorem 19 and 20 in [57]*). (1) *The class of unbounded unions of (erasing or non-erasing) languages generated from proper regular patterns is not learnable from positive data.*
(2) *The class of unbounded unions of (erasing or non-erasing) languages generated from proper one-variable patterns is not learnable from positive data.*

Part 2 of Theorem 25 is due to Frank Stephan. By these results, even the unbounded unions of the one-variable pattern languages, or the regular pattern languages, are not learnable from positive data. Fig. 2 summarizes the known results regarding learnability of classes obtained from unions of the pattern languages. An affirmation in the column of: (1) "fin. thick." implies that the class has finite thickness, (2) "fin. elas." implies that the class has finite elasticity, (3) "char. set." implies that each language in the class has a characteristic set within the class.

Learnability of the full class of the erasing pattern languages remained elusive. It was noted in [57] that the erasing pattern languages is learnable, if the set of variables is of finite cardinality (see Corollary 14 in [57]).

Denote the erasing pattern language of a pattern p by $L_\epsilon(p)$. Mitchell first obtained results in 1998 for a non-trivial subclass of the erasing pattern languages called **QRP**.

Definition 26 (*Erasing Languages of Quasi-regular Patterns [31]*). (1) A pattern p is *m-quasi-regular* if and only if each variable that occurs in p occurs exactly m times.
(2) $\mathbf{QRP}_m = \{L_\epsilon(p) \mid p \text{ is an } m\text{-quasi-regular pattern}\}$.
(3) $\mathbf{QRP} = \bigcup_{m\in\mathbb{N}} \mathbf{QRP}_m$.

Given a class of patterns $\mathcal{P}$, a pattern $p \in \mathcal{P}$ is *succinct* if and only if for all $p' \in \mathcal{P}$, $L_\epsilon(p') = L_\epsilon(p) \Rightarrow |p'| \geq |p|$. For a pattern p, let residual(p) denote the string obtained by substituting each variable in p with the empty string.

Theorem 27 (*Lemma 9 in [31]*). *For $m \in \mathbb{N}$, given p, $p' \in$ **QRP**$_m$ where* residual(p) = residual(p')*, if $L_\epsilon(p) \subseteq L_\epsilon(p')$ and $2|p'| < |p|$, then p is not succinct.*

Intuitively this is how Theorem 27 shows the learnability of **QRP**$_m$ from positive data. We first note that in the limit, the shortest example received must be the residual of the target pattern, p say. Now one can find, in the limit, a pattern p' that is (1) m-quasi-regular, where (2) residual(p') = residual(p), and (3) $L_\epsilon(p')$ is a superset of $L_\epsilon(p)$. The problem that remains is then to find, among subsets of $L_\epsilon(p')$, for the language $L_\epsilon(p)$. Theorem 27 shows that one need to look at only finitely many patterns for such subsets of $L(p')$, and hence $L_\epsilon(p)$ can be identified in the limit.

By further noting that m can be inferred from the shortest and the second shortest length of the examples, we have the following.

Theorem 28 (*Theorem 30 in [31]*). **QRP** *is learnable from positive data.*

Note that the erasing regular pattern languages are properly included in **QRP**. In [31], Mitchell also showed that the class of the erasing pattern languages is learnable from positive data in the case of an infinite or singular alphabet.

More results on the learnability of the erasing pattern languages from positive data did not emerge until 2002, when Reidenbach showed the following result for the erasing languages generated by the patterns which consist only of variables (called *terminal-free patterns*). It was noted that the class has infinite elasticity in [57], but its learnability remained open at that time.

Definition 29 (*Passe-partouts [43]*). Given an indexed family $\mathcal{L}$, $L \in \mathcal{L}$ and $S \subseteq L$. We say that $L' \in \mathcal{L}$ is a *passe-partout* (for L and S) if and only if $S \subseteq L'$ and $L' \subset L$.

Hence a passe-partout for L and S witnesses that S is not a finite tell-tale for L within $\mathcal{L}$.

Theorem 30 (*Lemma 1 in [43]*). *Let $p = x_1x_1x_2x_2x_3x_3$, where x_1, x_2 and x_3 are distinct variables, if Σ consists of exactly* 2 *letters, then for each finite $S \subseteq L_\epsilon(p)$, there exists a terminal-free pattern where its erasing language is a passe-partout for $L_\epsilon(p)$ and S.*

Hence no finite tell-tale exists for $L_\epsilon(p)$ within the class of the erasing languages generated by the terminal-free patterns. By Theorem 3,

Theorem 31 (*Theorem 1 in [43]*). *If Σ consists of exactly* 2 *letters, then the class of the erasing languages generated by the terminal-free patterns is not learnable from positive data.*

Corollary 32. *If Σ consists of exactly* 2 *letters, then the class of the erasing pattern languages is not learnable from positive data.*

However, when there are at least three letters in Σ, it turns out that for each terminal-free pattern p, there is a set of substitutions that transforms p into a set of strings, from which each variable in p can be inferred.

Theorem 33 (*Theorem 2 in [44]*). *If Σ consists of at least* 3 *letters, then the class of erasing languages generated by the terminal-free patterns is learnable from positive data.*

Hence the erasing languages of the terminal-free patterns are not learnable if and only if the alphabet consists of exactly 2 letters. Regrettably, the learnability in Theorem 33 did not follow with a learnability result for the full class of the erasing pattern languages. By showing that for some patterns, there are passe-partouts for every finite subset of the erasing languages they generate, Reidenbach showed the following.

Theorem 34 (*Theorem 1 in [45]*). *If Σ consists of exactly* 3 *or* 4 *letters, then the class of the erasing pattern languages is not learnable from positive data.*

Hence by these results of Reidenbach, the class of the erasing pattern languages is now known to be not learnable for the alphabet sizes of 2, 3 and 4.

5. The MMG Algorithm

Learning by *minl*, as in the fashion of Theorem 9, is desirable because this results in a learner that is *consistent* and *conservative*. As mentioned, a consistent learner never outputs a hypothesis that fails to account for some of the examples presented, while a conservative learner never changes its hypothesis as long as the current hypothesis continues to account for all of the examples presented.

In order to compute *minl* efficiently, Arimura et al. considered the classes of languages where inclusions can be computed efficiently via a form of subsumption relation [38] between languages in the class. They furthermore generalized Plotkin's idea of a *least general generalization* under the subsumption relation to the case of unions, called *minimal multiple generalizations* (*mmg*). We now define the subsumption relation and the minimal multiple generalizations, for the case of the pattern languages.

We say that a pattern q *refines* to a pattern p (written $p \preceq q$) if and only if p can be obtained from q by substituting variables in q with some non-empty patterns. We write $p \preceq_\epsilon q$ if and only if p can be obtained from q by substituting variables in q with some possibly empty patterns. We write $p \prec q$ if and only if $p \preceq q$ but not $q \preceq p$. Given two sets of patterns P and Q, we write $P \sqsubseteq Q$ if and only if for each $p \in P$, $p \preceq q$ for some $q \in Q$. $P \sqsubset Q$ if and only if $P \sqsubseteq Q$ but not $Q \sqsubseteq P$. Similarly one can define the relations $\prec_\epsilon$, $\sqsubseteq_\epsilon$, $\sqsubset_\epsilon$ from $\preceq_\epsilon$. Recall that the language of a pattern p is written $L(p)$, while its erasing language is written $L_\epsilon(p)$. For a set of patterns P, $L(P) = \bigcup_{p \in P} L(p)$ and $L_\epsilon(P) = \bigcup_{p \in P} L_\epsilon(p)$. An *mmg* is as follows.

Definition 35 (*Minimal Multiple Generalization [9]*). Given a class of pattern $\mathcal{P}$, a finite non-empty set S of strings over Σ and $k \in \mathbb{N}^+$, an *mmg*(S, k) is a set of up to k patterns $P \subseteq \mathcal{P}$ such that $S \subseteq L(P)$ and there are no other set of up to k patterns $P' \subseteq \mathcal{P}$ where $S \subseteq L(P')$ such that $P' \sqsubset P$.

For comparison with the *mmg* we extend the definition of *minl* to unions, as follows.

Definition 36 (*Minl For Unions [9]*). Given a class of pattern $\mathcal{P}$, a finite non-empty set S of strings over Σ and $k \in \mathbb{N}^+$, a *minl*(S, k) is a set of up to k patterns $P \subseteq \mathcal{P}$ such that $S \subseteq L(P)$ and there are no other set of up to k patterns $P' \subseteq \mathcal{P}$ where $S \subseteq L(P')$ such that $L(P') \subset L(P)$.

Similarly one can define *mmg*s and *minl*s for the erasing case. The following result, which holds equivalently for the erasing case, gives a condition for which an *mmg* would be a *minl*.

Theorem 37 (*[9], Also See Proposition 10 in [36]*). *For a class of patterns $\mathcal{P}$, if for every collection of up to k patterns $P, Q \subseteq \mathcal{P}$ we have $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$, then for any non-empty finite set S of strings over Σ and $k \in \mathbb{N}^+$, a set of patterns is an mmg(S, k) if and only if it is a minl(S, k).*

Note that the condition $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$ is fulfilled if for any set of $k + 1$ patterns $p_0, \dots, p_k$ from the class, we have $L(p_0) \subseteq \bigcup_{1 \le i \le k} L(p_i) \Rightarrow L(p_0) \subseteq L(p_i)$ for some $1 \le i \le k$, in which case we say that the class has *compactness with respect to containment*.

Mukouchi [33] first formally studied the condition with respect to the pattern languages. At the same time, Arimura et al. gave an efficient algorithm for finding *mmg*s for the bounded unions of the tree pattern languages and showed that when the number of languages allowed in a union is less than the alphabet size, compactness with respect to containment can be achieved. This resulted in an efficient algorithm for finding *minl*s for the tree pattern languages [9,10,12].

The next breakthrough came when Arimura et al. [11] gave an efficient algorithm (**MMG**) for finding *mmg*s for any class $\mathcal{P}$ of pattern languages where an *efficient complete refinement operator* exists for the class. A *complete refinement operator* ρ is a subrelation of $\prec$ where for any patterns $p, q \in \mathcal{P}$, $p \preceq q \iff p \in \rho^+(q)$. A complete refinement operator is *efficient* if for any $p \in \mathcal{P}$, $\rho(p)$ is polynomial-time computable. Such an operator exists for the regular patterns, both for their non-erasing languages [11], as well as their erasing languages [6].

Given a class $\mathcal{P}$ of patterns, a finite non-empty set $S \subseteq \Sigma^*$ and $k \in \mathbb{N}^+$, the following lists the **MMG** algorithm. All patterns in the following are implicitly elements of $\mathcal{P}$. The algorithm assumes that there is a pattern $\perp$ in the class $\mathcal{P}$ such that every pattern $p \in \mathcal{P}$ can be obtained from $\perp$ (i.e. $p \preceq \perp$). (One can adapt the following algorithm straightforwardly to use a complete refinement operator defined on erasing substitutions.)

```
MMG($\mathcal{P}$, $S$, $k$)
  (1) Let $P \leftarrow \{\bot\}$.
  (2) Let $\Delta k \leftarrow k$.
  (3) While $\Delta k \geq 2$ and there exists $p \in P$, and a set
            $Q \sqsubseteq \{p\}$ of more than one and up to $\Delta k$ patterns where
        (i) $S - L(P - \{p\}) \subseteq L(Q)$, but
        (ii) no $Q' \subset Q$ has $S - L(P - \{p\}) \subseteq L(Q')$ (i.e. no pattern in $Q$
             is redundant),
      (3.1) Replace each $q \in Q$ with a more specific $q' \prec q$ until any further
            such replacement will result in $S \not\subseteq L(P) - L(\{p\}) \cup L(Q)$.
      (3.2) Let $P \leftarrow P - \{p\} \cup Q$.
      (3.3) Let $\Delta k \leftarrow \Delta k - |Q| + 1$.
  (4) Output $P$.
```

Arimura et al. showed that the condition of compactness with respect to containment holds for the regular pattern languages when the set of variables is of a finite size m, and the alphabet is of size larger than $2km$, where k is the number of languages allowed in a union. Hence by Theorem 37, in such cases, the output of **MMG** would be a *minl*.

A more complete result was later given by Arimura and Shinohara, showing that for the class of regular pattern languages, compactness with respect to containment holds if the alphabet size is at least $2k + 1$, but fails to hold if the alphabet size is less than or equal to $k + 1$ [8]. Whether compactness with respect to containment holds for alphabet sizes within $k + 2$ to $2k$ remained open until Sato et al. [48] showed the following results.

Theorem 38 (*Theorem 20 in [48]*). *If $k \geq 3$, then the class of the regular pattern languages has compactness with respect to containment if and only if the alphabet contains at least $2k - 1$ letters.*

Theorem 39 (*Corollary 23 in [48]*). *If $k = 2$, then the class of the regular pattern languages has compactness with respect to containment if the alphabet consists of at least 4 letters.*

For the erasing regular pattern languages, Uemura and Sato [62] completely characterized the condition of compactness with respect to containment with the following.

Theorem 40 (*Theorem 4 in [62]*). *If $k \geq 1$, then the class of the erasing regular pattern languages has compactness with respect to containment if and only if the alphabet contains at least $k + 2$ letters.*

Besides the **MMG** algorithm, *minl* computation has also been pursued by Jantke [20] (for the k-variable pattern languages), and Erlebach et al. [15] (for one-variable pattern languages). The results by Erlebach et al. is particularly interesting in the sense that they investigated the *total expected learning time* of their algorithm and showed it to be polynomial with respect to the expected string length.

6. Using the MMG algorithm

Arimura et al. first examined practical uses of the **MMG** algorithm [6]. The tests were performed on the two samples of biological sequences: transmembrane domains and signal peptide sequences. These samples are short strings of lengths 30 to 50, over an alphabet of 20 amino acids. In their experiments, they used a small portion of these biosequences as input to the **MMG** algorithm, and then tested the obtained *mmg*s for their accuracies in (1) matching all the biosequences, and (2) excluding randomly selected biosequences. There are places in the **MMG** algorithm where choices are arbitrary over all suitable patterns (at line 3 and 3.1 of the **MMG** algorithm listed), and in their study, they investigated how additional preferences placed on these choices of patterns affect the accuracies of the obtained *mmg*s. The accuracies obtained in their tests compared favorably to results from another system which uses both positive and negative data.

Furthermore, by arranging the amino acids into 3 hydropathy groups (according to the schema by Kyte and Doolittle [25]), they processed the biosequences into strings over an alphabet of only three letters (i.e. each for one of the hydropathy groups). This reduction in alphabet size greatly reduced the pattern search space, but nonetheless produced *mmg*s with similar accuracies as the *mmg*s obtained from unprocessed sequences [6]. Yamaguchi et al. [67] continued with this line of enquiry. In their tests they did without the assumption of the schema by Kyte and Doolittle,

	Pattern language	Logic programming	MMG
'91			First study on compactness for the pattern languages [33]; Polynomial time inference of bounded unions of tree pattern languages [9,10] (summarized and extended in [12])
'92		Efficient learning of primitive Prolog [19]	
'93	Properties of finite elasticity [61]		
'94		Linearly covering Prolog [7]	MMG for pattern languages [11]; Experimental results on biosequences [6]
'95	Studies on various structures including finite elasticity and characteristic sets [47]; Expected total learning time analysis of LWA [68]		
'96	Unbounded unions [57]	Hereditary, reductive and linearly-moded Prolog [39]	Results on compactness for the regular pattern languages [8]; Experimental results on biosequences [67]
'97	Polynomial expected total learning time *minl* learner for one-variable pattern languages [15]		
'98	Learnability of the subclass QRP [31]; Linear total learning time one-variable pattern languages learner [46]		Complete study on compactness for the regular pattern languages [48]; Experimental results with high accuracies [60]
'99		Sufficient conditions for learnability: strongly admissible and safe programs [29]	
'01		EFSs with negation (AEFSs) studied [26]	
'02	Erasing pattern languages not learnable for alphabet size 2 [43]		Complete study on compactness of the erasing regular pattern languages [62]
'04	Terminal-free erasing pattern languages learnable for all alphabet sizes ≥ 3 [44]; Erasing pattern languages not learnable for alphabet sizes 3 and 4 [45]	Learnability of simple flat Term Rewriting Systems (TRS) [40]; Learnability [34] of subclasses of simple formal systems [4], regular formal systems [54] Simple H (SH) languages [30]	
'05	Learning by fittest covers [36]	Sufficient condition for learnability: recursion bounded programs [41]	Success of [60] explained [35]
'06		Length-bounded TRSs [42]	More experimental results [37]

Fig. 3. Chronological view of developments from 1991.

and searched for a grouping that, in a sense, minimizes the overlap between the resultant strings of the positive and negative samples. This resulted in a grouping that resembled that of Kyte and Doolittle, which subsequently gave results similar to that obtained in the previous work.

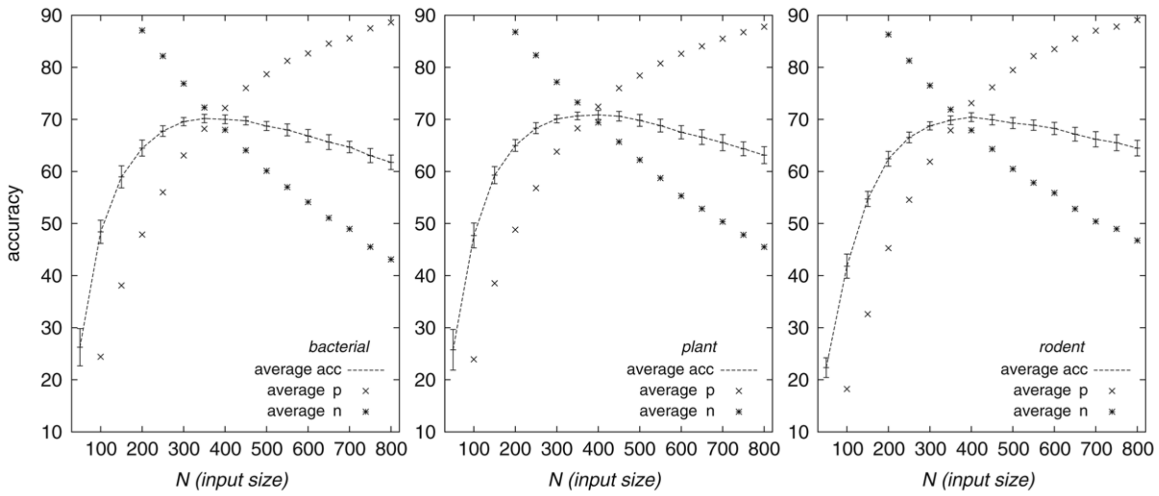

Fig. 4. accuracies vs size of sample used as **MMG** input.

The next breakthrough in the use of the **MMG** algorithm came from Takae et al. [60], where they found that the *mmg*s of regular patterns which contain *sort symbols* (called *sort regular patterns*), achieved higher accuracies than plain regular patterns. A *sort symbol* is a letter associated with a non-empty set $S \subseteq \Sigma$, and is written $[S]$, where in a substitution for a sort regular pattern, a sort symbol $[S]$ may be replaced with another sort symbol $[S']$ where $S' \subseteq S$, or with any letter in S. This allowed more ambiguous expression of patterns and also made somewhat redundant to preprocess the patterns, as in the two earlier studies. Such symbols have previously been used by others, for example in the PROSITE database [58], under other names.

In Fig. 4 we show essentially the same results reported in [60], replotted using updated data in 2005. The plots are of the averaged accuracies of *mmg*s (obtained from samples of sizes $N = 50, 100, 150, \ldots, 800$) in:

1. matching the (remaining) positive biosequences (p, legend $\times$)
2. excluding the negative biosequences (n, legend $*$), and
3. $\sqrt{p \cdot n}$ (acc, dashed line)

against the sizes N. Each of these plots is for the signal peptide sequences for one of the following three types of samples: bacteria, plant and rodent. About 1800 positive sequences were available for each of these samples, and samples of about 10 times those sizes were carefully generated and used as negative sequences in the accuracy tests.

Ng et al. [35] attempted at an explanation for these high accuracies obtained by [60]. They noted that the language of any sort regular pattern can be expressed as the union of a few languages of the regular patterns. For example, let $\Sigma = \{a, b\}$ and let a sort regular pattern $p = x_1[\{a, b\}][\{a, b\}]x_2$ be where x_1, x_2 are variables. The language of p is then equivalent to the union of the languages of the following regular patterns: x_1aax_2, x_1abx_2, x_1bax_2, x_1bbx_2. By a notion of *coverage*, defined to be the number of strings of up to a suitable length generated by a language, they showed that regardless of the types of patterns used, the *mmg*s of similar coverages always scored about the same accuracies. Intuitive, such coverages correspond to the amount of over-generalization committed by the *mmg*s. Ng et al. hence argues that the higher accuracies obtained by the *mmg*s of the sort regular patterns were due to the more specific languages that these patterns are able to express, and the same effects can be as well achieved by the regular patterns, by allowing more languages to be included in a union. However, they also noted that the runtime of the **MMG** algorithm increases more on the number of unions allowed than on the use of additional symbols, and hence for the **MMG** algorithm the use of sort symbols seems desirable.

Some work on the inductive inference from positive data using hypotheses which minimize such coverages can be found in [36], and further experimental studies of the use of such coverages can be found in [37].

7. Concluding remarks

The inductive inference of the pattern languages from positive data has come a long way from the original propositions by Angluin.

Theoretically, the class has provided a very fertile ground for the testing and development of many ideas in inductive inference, such as those witnessed by the Lange and Wiehagen algorithm. Investigating the learnability of this seemingly very simple class and its variants has turned out to be a most testing, but also rewarding task. The subsequent extensions to the more expressive logic programs also yielded many interesting results. Experimentally, this class of languages has provided a very simple and natural framework for the study of strings. The little assumptions made of the underlying hypothesis space allows for very flexible adaptations, as is witnessed by the **MMG** algorithm, which accepts many hypothesis spaces as input.

For all these reasons and more, studies that involve the pattern languages have become too numerous for one to keep account of, and the omissions of important works we failed to mention here – if any – are not intentional.

References

[1] D. Angluin, Finding patterns common to a set of strings, Journal of Computer and System Sciences 21 (1980) 46–62. First appeared in STOC'79.

[2] D. Angluin, Inductive inference of formal languages from positive data, Information and Control 45 (1980) 117–135.

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

[4] S. Arikawa, Elementary formal systems and formal languages — simple formal systems, in: Memoirs of the Faculty of Science, Kyushu University Series A, Mathematics, vol. 24, 1970, pp. 47–75.

[5] S. Arikawa, T. Shinohara, A. Yamamoto, Learning elementary formal systems, Theoretical Computer Science 95 (1992) 97–113. First appeared in COLT'89.

[6] H. Arimura, R. Fujino, T. Shinohara, S. Arikawa, Protein motif discovery from positive examples by Minimal Multiple Generalization over regular patterns, in: Proceedings of the Genome Informatics Workshop 1994, Universal Academy Press, 1994, pp. 39–48.

[7] H. Arimura, T. Shinohara, Inductive inference of Prolog programs with linear data dependency from positive data, in: Proc. Information Modelling and Knowledge Bases V, IOS Press, 1994, pp. 365–375.

[8] H. Arimura, T. Shinohara, Compactness for unions of regular pattern languages, in: Proceedings of Symposium on Language and Automaton, Research on Computational Models and Complexity, RIMS Koukyuroku, vol. 950, 1996, pp. 246–249 (in Japanese).

[9] H. Arimura, T. Shinohara, S. Otsuki, A polynomial time algorithm for finding finite unions of tree pattern languages, in: Proceedings of Nonmonotonic and Inductive Logic, Second International Workshop, in: Lecture Notes in Artificial Intelligence, vol. 659, Springer-Verlag, 1991, pp. 118–131.

[10] H. Arimura, T. Shinohara, S. Otsuki, Polynomial time inference of unions of two tree pattern languages, IEICE Transactions on Information and Systems E75-D (7) (1992) 426–434. First appeared in ALT'91.

[11] H. Arimura, T. Shinohara, S. Otsuki, Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data, in: Proceedings of STACS 94, 11th Annual Symposium on Theoretical Aspects of Computer Science, in: Lecture Notes in Computer Science, vol. 775, Springer-Verlag, 1994, pp. 649–660.

[12] H. Arimura, T. Shinohara, S. Otsuki, H. Ishizaka, A generalization of the least general generalization, Machine Intelligence 13 (1994) 59–85. First appeared as RIFIS Technical Report in 1992.

[13] J. Bārzdiņš, Inductive inference of automata, functions and programs, in: Int. Math. Congress, Vancouver, 1974, pp. 771–776.

[14] N. Dershowitz, J.-P. Jouannaud, Rewrite systems, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B), North-Holland, 1990, pp. 243–320.

[15] T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger, T. Zeugmann, Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries, Theoretical Computer Science 261 (1) (2001) 119–156 (special issue for ALT'97).

[16] C. Costa Florêncio, Learning Categorial Grammars. Ph.D. Thesis, Utrecht institute of Linguistics OTS, Utrecht University, 2003.

[17] R. Freivalds, E. Kinber, R. Wiehagen, Inductive inference from good examples, in: Analogical and Inductive Inference, Proceedings of the Second International Workshop, in: Lecture Notes in Artificial Intelligence, vol. 397, Springer-Verlag, 1989, pp. 1–17.

[18] E.M. Gold, Language identification in the limit, Information and Control 10 (1967) 447–474.

[19] H. Ishizaka, H. Arimura, T. Shinohara, Efficient inductive inference of primitive prologs from positive data, in: Proceedings of Algorithmic Learning Theory, Third International Workshop, ALT'92, in: Lecture Notes in Artificial Intelligence, vol. 743, Springer-Verlag, 1992, pp. 135–146.

[20] K.P. Jantke, Polynomial time inference of general pattern languages, in: Proceedings of STACS 84, Symposium on Theoretical Aspects of Computer Science, in: Lecture Notes in Computer Science, vol. 166, Springer-Verlag, 1984, pp. 314–325.

[21] T. Jiang, A. Salomaa, K. Salomaa, S. Yu, Decision problems for patterns, Journal of Computer and System Sciences 50 (1995) 53–63.

[22] K.-I. Ko, C.-H. Hua, A note on the two-variable pattern-finding problem, Journal of Computer and System Sciences 34 (1) (1987) 75–86.

[23] S. Kobayashi, T. Yokomori, Identifiability of subspaces and homomorphic images of zero-reversible languages, in: Proceedings of Algorithmic Learning Theory, Eighth International Conference, ALT'97, in: Lecture Notes in Artificial Intelligence, vol. 1316, Springer-Verlag, 1997, pp. 48–61.

[24] J.B. Kruskal, The theory of well-quasi-ordering: A frequently discovered concept, Journal of Combinatorial Theory (A) 13 (1972) 297–305.

[25] J. Kyte, R.F. Doolittle, A simple method for displaying the hydropathic character of a protein, Journal of Molecular Biology 157 (1982) 105–132.

[26] S. Lange, G. Grieser, K.P. Jantke, Advanced elementary formal systems, Theoretical Computer Science 1 (298) (2003) 51–70 (special issue on ALT 2001).

[27] S. Lange, J. Nessel, R. Wiehagen, Language learning from good examples, in: Algorithmic Learning Theory: Fourth International Workshop on Analogical and Inductive Inference, AII '94, and Fifth International Workshop on International Workshop, ALT'94, in: Lecture Notes in Artificial Intelligence, vol. 872, Springer-Verlag, 1994, pp. 423–437.

[28] S. Lange, R. Wiehagen, Polynomial-time inference of arbitrary pattern languages, New Generation Computing 8 (1991) 361–370. First appeared in ALT'90.

[29] E. Martin, A. Sharma, On sufficient conditions for learnability of logic programs from positive data, in: Proceedings of Inductive Logic Programming: 9th International Workshop, ILP'99, in: Lecture Notes in Computer Science, vol. 1634, Springer-Verlag, 1999, pp. 198–209.

[30] A. Mateescu, G. Paun, G. Rozenberg, A. Salomaa, Simple splicing systems, Discrete Applied Mathematics 84 (1–3) (1998) 145–163.

[31] A. Mitchell, Learnability of a subclass of extended pattern languages, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, ACM Press, 1998.

[32] T. Motoki, T. Shinohara, K. Wright, The correct definition of finite elasticity: Corrigendum to identification of unions, in: L. Valiant, M. Warmuth (Eds.), Proceedings of the Fourth Annual Workshop on Computational Learning Theory, Morgan Kaufmann, 1991, p. 375.

[33] Y. Mukouchi, Containment problems for pattern languages, IEICE Transactions on Information and Systems E75-D (7) (1992) 420–425 (special issue for ALT'01).

[34] Y. Mukouchi, M. Sato, Learning languages generated by elementary formal systems and its application to SH languages, in: Proceedings of Algorithmic Learning Theory, Fifteenth International Conference, ALT'04, in: Lecture Notes in Artificial Intelligence, vol. 3244, Springer-Verlag, 2004, pp. 380–394.

[35] Y.K. Ng, H. Ono, T. Shinohara, Measuring over-generalization in the minimal multiple generalizations of biosequences, in: Proceedings of Discovery Science, 8th International Conference, DS 2005, in: Lecture Notes in Artificial Intelligence, vol. 3735, Springer-Verlag, 2005, pp. 176–188.

[36] Y.K. Ng, T. Shinohara, Inferring unions of the pattern languages by the most fitting covers, in: Proceedings of Algorithmic Learning Theory, Sixteenth International Conference, ALT'05, in: Lecture Notes in Artificial Intelligence, vol. 3734, Springer-Verlag, 2005, pp. 269–282.

[37] Y.K. Ng, T. Shinohara, Finding consensus patterns in very scarce biosequence samples from their minimal multiple generalizations, in: Proceedings of Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference, PAKDD 2006, in: Lecture Notes in Artificial Intelligence, vol. 3918, Springer-Verlag, 2006, pp. 540–545.

[38] G.D. Plotkin, A note on inductive generalization, in: Machine Intelligence, vol. 5, Edinburgh University Press, 1970, pp. 153–163.

[39] M.R.K. Krishna Rao, Some classes of prolog programs inferable from positive data, Theoretical Computer Science A 241 (2000) 211–234 (special issue for ALT'96).

[40] M.R.K. Krishna Rao, Inductive inference of term rewriting systems from positive data, in: Proceedings of Algorithmic Learning Theory, Fifteenth International Conference, ALT'04, in: Lecture Notes in Artificial Intelligence, vol. 3244, Springer-Verlag, 2004, pp. 69–82.

[41] M.R.K. Krishna Rao, A class of prolog programs with non-linear outputs inferable from positive data, in: Proceedings of Algorithmic Learning Theory, Sixteenth International Conference, ALT'05, in: Lecture Notes in Artificial Intelligence, vol. 3734, Springer-Verlag, 2005, pp. 312–326.

[42] M.R.K. Krishna Rao, Learnability of term rewrite systems from positive examples, in: Proceedings of the Twelfth Computing: The Australasian Theory Symposium, CATS 2006, in: Conferences in Research and Practice in Information Technology, vol. 51, Australian Computer Society, 2006, pp. 133–137.

[43] D. Reidenbach, A negative result on inductive inference of extended pattern languages, in: Proceedings of Algorithmic Learning Theory, Thirteenth International Conference, ALT'02, in: Lecture Notes in Artificial Intelligence, vol. 2533, Springer-Verlag, 2002, pp. 308–320.

[44] D. Reidenbach, A discontinuity in pattern inference, in: Proceedings of STACS 04, 21st Annual Symposium on Theoretical Aspects of Computer Science, in: Lecture Notes in Computer Science, vol. 2996, Springer-Verlag, 2004, pp. 129–140.

[45] D. Reidenbach, On the learnability of e-pattern languages over small alphabets, in: Proceedings of the Seventeenth Annual Conference on Computational Learning Theory, in: Lecture Notes in Computer Science, vol. 3120, Springer-Verlag, 2004, pp. 140–154.

[46] R. Reischuk, T. Zeugmann, An average case optimal one-variable pattern language learner, Journal of Computer and System Sciences 60 (2) (1998) 302–335 (special issue for COLT'98).

[47] M. Sato, Inductive inference of formal languages, Bulletin of Informatics and Cybernetics 27 (1) (1995) 85–106.

[48] M. Sato, Y. Mukouchi, D. Zheng, Characteristic sets for unions of regular pattern languages and compactness, in: Proceedings of Algorithmic Learning Theory, Ninth International Conference, ALT'98, in: Lecture Notes in Artificial Intelligence, vol. 1501, Springer-Verlag, 1998, pp. 220–233.

[49] E. Shapiro, Algorithmic Program Debugging, MIT Press, 1983.

[50] T. Shinohara, Polynomial time inference of extended regular pattern languages, in: RIMS Symposia on Software Science and Engineering, Kyoto, Japan, in: Lecture Notes in Computer Science, vol. 147, Springer-Verlag, 1982, pp. 115–127.

[51] T. Shinohara, Polynomial time inference of pattern languages and its application, in: Proceedings of the 7th IBM Symposium on Mathematical Foundations of Computer Science, 1982, pp. 192–209.

[52] T. Shinohara, Inferring unions of two pattern languages, Bulletin of Informatics and Cybernetics 20 (1983) 83–88.

[53] T. Shinohara, Some problems on inductive inference from positive data, in: Mathematical Methods of Specification and Synthesis of Software Systems, Wendisch-Rietz, GDR, in: Lecture Notes in Computer Science, vol. 215, Springer-Verlag, 1985, pp. 41–58.

[54] T. Shinohara, Inductive inference of formal systems from positive data, Bulletin of Informatics and Cybernetics 22 (1986) 9–18.

[55] T. Shinohara, Inductive inference of monotonic formal systems from positive data, New Generation Computing 8 (1991) 371–384. First appeared in ALT'90.

[56] T. Shinohara, Rich classes inferable from positive data: Length–bounded elementary formal systems, Information and Computation 108 (1994) 175–186. First appeared in COLT'90.

[57] T. Shinohara, H. Arimura, Inductive inference of unbounded unions of pattern languages from positive data, Theoretical Computer Science A 241 (2000) 191–209 (special issue for ALT'96).
[58] C.J. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, P. Bucher, PROSITE: A documented database using patterns and profiles as motif descriptors, Brief Bioinformatics 3 (2002) 265–274.
[59] R. Smullyan, Theory of formal systems, Annals of Mathematical Studies 47 (1961).
[60] T. Takae, T. Kasai, H. Arimura, T. Shinohara, Knowledge discovery in biosequences using sort regular patterns. in: Workshop on Applied Learning Theory, 1998.
[61] M. Takashi, M. Sato, Properties of language classes with finite elasticity, in: Proceedings of Algorithmic Learning Theory, Fourth International Workshop, ALT'93, in: Lecture Notes in Artificial Intelligence, vol. 744, Springer-Verlag, 1993, pp. 187–196.
[62] J. Uemura, M. Sato, Compactness and learning of classes of unions of erasing regular pattern languages, in: Proceedings of Algorithmic Learning Theory, Thirteenth International Conference, ALT'02, in: Lecture Notes in Artificial Intelligence, vol. 2533, Springer-Verlag, 2002, pp. 293–307.
[63] R. Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, Journal of Information Processing and Cybernetics (EIK) 12 (1976) 93–99.
[64] R. Wiehagen, T. Zeugmann, Ignoring data may be the only way to learn efficiently, Journal of Experimental and Theoretical Artificial Intelligence 6 (1994) 131–144. First appeared in AII'92.
[65] R. Wiehagen, T. Zeugmann, Learning and consistency, in: Algorithmic Learning for Knowledge-Based Systems, in: Lecture Notes in Artificial Intelligence, vol. 961, Springer-Verlag, 1995, pp. 1–24.
[66] K. Wright, Identification of unions of languages drawn from an identifiable class, in: R. Rivest, D. Haussler, M. Warmuth (Eds.), Proceedings of the Second Annual Workshop on Computational Learning Theory, Morgan Kaufmann, 1989, pp. 328–333.
[67] M. Yamaguchi, S. Shimozono, T. Shinohara, Finding minimal multiple generalization over regular patterns with alphabet indexing, in: Proceedings of the Seventh Workshop on Genome Informatics, vol. 7, Universal Academy Press, 1996, pp. 51–60.
[68] T. Zeugmann, Lange and Wiehagen's pattern language learning algorithm: An average-case analysis with respect to its total learning time, Annals of Mathematics and Artificial Intelligence 23 (1–2) (1998) 117–145. First appeared as a RIFIS Technical Report in 1995.
[69] T. Zeugmann, From learning in the limit to stochastic finite learning, Theoretical Computer Science 364 (1) (2006) 77–97.
[70] T. Zeugmann, S. Lange, A guided tour across the boundaries of learning recursive languages, in: Algorithmic Learning for Knowledge-Based Systems, in: Lecture Notes in Artificial Intelligence, vol. 961, Springer-Verlag, 1995, pp. 190–258.