

Mind Change Complexity of Learning Logic Programs

Sanjay Jain^a and Arun Sharma^b

^a*School of Computing, National University of Singapore, Singapore 119260, Republic of Singapore, Email: sanjay@comp.nus.edu.sg*

^b*School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia, Email: arun@cse.unsw.edu.au*

Abstract

The present paper motivates the study of mind change complexity for learning minimal models of length-bounded logic programs. It establishes ordinal mind change complexity bounds for learnability of these classes both from positive facts and from positive and negative facts.

Building on Angluin's notion of finite thickness and Wright's work on finite elasticity, Shinohara defined the property of bounded finite thickness to give a sufficient condition for learnability of indexed families of computable languages from positive data. This paper shows that an effective version of Shinohara's notion of bounded finite thickness gives sufficient conditions for learnability with ordinal mind change bound, both in the context of learnability from positive data and for learnability from complete (both positive and negative) data.

Let ω be a notation for the first limit ordinal. Then, it is shown that if a language defining framework yields a uniformly decidable family of languages and has effective bounded finite thickness, then for each natural number $m > 0$, the class of languages defined by formal systems of length $\leq m$:

- is identifiable in the limit from positive data with a mind change bound of ω^m ;
- is identifiable in the limit from both positive and negative data with an ordinal mind change bound of $\omega \times m$.

The above sufficient conditions are employed to give an ordinal mind change bound for learnability of minimal models of various classes of length-bounded Prolog programs, including Shapiro's linear programs, Arimura and Shinohara's depth-bounded linearly-covering programs, and Krishna Rao's depth-bounded linearly-moded programs. It is also noted that the bound for learning from positive data is tight for the example classes considered.

Keywords: Inductive Inference, Computational Learning Theory, Mind Change Complexity, Logic Programs.

1 Motivation and Introduction

Machine learning in the context of first-order logic and its subclasses can be traced back to the work of Plotkin [Plo71] and Shapiro [Sha81]. In recent years, this work has evolved into the very active field of Inductive Logic Programming (ILP). Numerous practical systems have been built to demonstrate the feasibility of learning logic programs as descriptions of complex concepts. The utility of these systems has been demonstrated in many domains including drug design, protein secondary structure prediction, and finite element mesh design (see Muggleton and DeRaedt [MDR94], Lavrac and Dzeroski [LD94], Bergadano and Gunetti [BG96], and Nienhuys-Cheng and de Wolf [NCdW97] for a survey of this field).

Together with practical developments, there has also been some interest in deriving learnability theorems for ILP. Several results in the PAC setting have been established; we refer the reader to Dzeroski, Muggleton, and Russell [DMR92a] [DMR92b], Cohen [Coh95a,Coh95b,Coh95c], De Raedt and Dzeroski [DRD94], Haussler [Hau89], Frisch and Page [FP91], Yamamoto [Yam93], Kietz [Kie93], and Maass and Turán [MT96].

Insights about which classes of logic programs are suitable as hypothesis spaces from the learnability perspective are likely to be very useful to ILP. Unfortunately, the few positive results that have been demonstrated in the PAC setting are for very restricted classes of logic programs. Hence, it is useful to consider more general models to analyze learnability of logic programs.¹ In the present paper, we develop tools to investigate identifiability in the limit with “mind change bounds” of minimal models of logic programs.

The first identification in the limit result about learnability of logic programs is due to Shapiro [Sha81]. He showed that the class of h -easy models is identifiable in the limit from both positive and negative facts. Adapting the work on learnability of subclasses of elementary formal systems², Shinohara [Shi91]

¹ The learnability analysis of ILP in the learning by query model is able to overcome some of the restrictive nature of the PAC model by allowing the learner queries to an oracle. For examples of such analysis, see Khardon [Kha98] and Krishna-Rao and Sattar [KRS98].

² Arikawa [Ari70] adapted Smullyan’s [Smu61] elementary formal systems (EFS) for investigation of formal languages. Later, Arikawa et al. [ASY92] showed that EFS can be viewed as a logic programming language over strings. Recently, various subclasses of EFS have been investigated in the context of learnability (e.g., see

showed that the class of minimal models of *linear* Prolog programs consisting of at most m clauses is identifiable in the limit from only positive facts. Unfortunately, linear logic programs are very restricted as they do not even allow local variables (i.e., each variable in the body must appear in the head). Arimura and Shinohara [AS94] introduced a class of *linearly-covering* logic programs that allows local variables in a restricted sense. They showed that the class of minimal models of linearly-covering Prolog programs consisting of at most m clauses of bounded body length is identifiable in the limit from only positive facts. Krishna Rao [KR96] noted that the class of linearly-covering programs is very restrictive as it did not even include the standard programs for `reverse`, `merge`, `split`, `partition`, `quick-sort`, and `merge-sort`. He proposed the class of *linearly-moded* programs that included all these standard programs and showed the class of minimal models of such programs consisting of at most m clauses of bounded body length to be identifiable in the limit from positive facts.

While the above results are positive, it may be argued that the model is too general as the number of mind changes allowed is unbounded. Some authors have considered a polynomial time bound on the update of hypotheses in the identification in the limit setting. However, this restriction may not be very meaningful if the number of mind changes (and consequently the number of updates) is unbounded. Recently, a number of approaches for modeling mind change bounds have been proposed [FS93,JS97,AJS97,Amb95,SSV97,AFS96]. The present paper employs constructive ordinals as mind change counters to model the mind change complexity of learning classes of logic programs in identification in the limit setting. We illustrate this notion in the context of identification in the limit of languages from positive data.

TextEx denotes the collection of language classes that can be identified in the limit from positive data. An obvious approach to bounding the number of mind changes is to require that the learning machine makes no more than a constant number of mind changes. This approach of employing natural numbers as mind change bounds was first considered by Bārzdīņš and Podnieks [BP73] (see also Case and Smith [CS83]). For each natural number m , **TextEx** _{m} denotes the set of language classes that can be identified in the limit from positive data with no more than m mind changes. However, a constant mind change bound has several drawbacks:

- it places the same bound on each language in the class irrespective of its “complexity”;
- it does not take into account scenarios in which a learner, after examining an element of the language, is in a position to issue a bound on the number of mind changes (i.e., the learner computes and updates mind change bounds

Shinohara [Shi91,Shi94]).

based on the incoming data).

To model situations where a mind change bound can be derived from data and updated as more data becomes available, constructive ordinals have been employed as mind change counters by Freivalds and Smith [FS93], and by Jain and Sharma [JS97]. We describe this notion next.

\mathbf{TxtEx}_α denotes the set of language classes that can be identified in the limit from positive data with an ordinal mind change bound α . We illustrate the interpretation of this notion with a few examples. Let ω denote a notation for the least limit ordinal. Then a mind change bound of $\alpha \prec \omega$ is the earlier notion of mind change identification where the bound is a natural number. For $\alpha = \omega$, \mathbf{TxtEx}_ω denotes learnable classes for which there exists a machine which, by the time it has made its first mind change, has also announced an upper bound on the number of mind changes it will make before the onset of successful convergence. Angluin's [Ang80b,Ang80a] class of pattern languages is a member of \mathbf{TxtEx}_ω . Proceeding on, the class $\mathbf{TxtEx}_{\omega \times 2}$ contains classes for which there is a learning machine that, as above, announces an upper bound on the number of mind changes, but reserves the right to revise its upper bound once. In $\mathbf{TxtEx}_{\omega \times 3}$, the machine reserves the right to revise its upper bound twice, and in $\mathbf{TxtEx}_{\omega \times 3+1}$, the machine can make upto one extra mind change before it conjectures an upper bound (which it is allowed to revise twice). $\mathbf{TxtEx}_{\omega^2}$ contains classes for which the machine announces an upper bound on the number of times it may revise its conjectured upper bound on the number of mind changes, and so on.

The notion of ordinal mind change bound has been employed to give learnability results for unions of pattern languages and subclasses of elementary formal systems (see [JS97,AJS97]). In the present paper, we generalize these results to establish a sufficient condition for learnability with ordinal mind change bounds and apply the results to obtain mind change bounds for learning subclasses of length-bounded logic programs. We discuss this sufficient condition briefly.

Let U be a recursively enumerable set of objects. A *language* is any subset of U ; a typical variable for languages is L . Let R be a recursively enumerable set of rules (these could be productions in the context of formal languages or clauses in the context of logic programs). A finite subset of R is referred to as a *formal system*; a typical variable for formal systems is Γ . Let \mathbf{Lang} be a mapping from the set of formal systems to languages. For technical convenience, we assume that $\mathbf{Lang}(\emptyset) = \emptyset$. We call the triple $\langle U, R, \mathbf{Lang} \rangle$ a *language defining framework*. In the sequel, we only consider those language defining frameworks for which the class $\{\mathbf{Lang}(\Gamma) \mid \Gamma \text{ is a finite subset of } R\}$ is a uniformly decidable family of computable languages. Furthermore, we suppose that a decision procedure for $\mathbf{Lang}(\Gamma)$ can be found effectively from

Γ .

A semantic mapping from formal systems to languages is *monotonic* just in case $\Gamma \subset \Gamma'$ implies $\mathbf{Lang}(\Gamma) \subseteq \mathbf{Lang}(\Gamma')$. A formal system Γ is said to be reduced with respect to a finite $X \subseteq U$ just in case X is contained in $\mathbf{Lang}(\Gamma)$ but not in any language defined by a proper subset of Γ . We assume, without loss of generality for this paper, that for all finite sets $X \subseteq U$, there exists a finite $\Gamma \subseteq R$, such that $X \subseteq \mathbf{Lang}(\Gamma)$.

Building on Angluin's [Ang80b] work on finite thickness and Wright's [Wri89] work on finite elasticity, Shinohara [Shi91] defined a language defining framework to have *bounded finite thickness* just in case

- (a) it is monotonic and
- (b) for each finite $X \subseteq U$ and for each natural number $m > 0$, the set of languages defined by formal systems that
 - (i) are reduced with respect to X and
 - (ii) that are of cardinality $\leq m$,

is finite. He showed that if a language defining framework has bounded finite thickness, then for each $m > 0$, the class of languages definable by formal systems of cardinality $\leq m$ is identifiable in the limit from positive data.

The present paper places a further requirement on Shinohara's notion of bounded finite thickness to derive sufficient conditions for learnability with mind change bounds. A language defining framework is said to have *effective bounded finite thickness* just in case the set of formal systems that are reduced with respect to X in the definition of bounded finite thickness can be obtained effectively in X . We show that the notion of effective bounded finite thickness gives an ordinal mind change bound for both learnability from positive data and for learnability from positive and negative data. In particular, we establish that if a language defining framework has effective bounded finite thickness, then for each natural number $m > 0$, the class of languages defined by formal systems of cardinality $\leq m$:

- is identifiable in the limit from positive data with an ordinal mind change bound of ω^m ;
- is identifiable in the limit from both positive and negative data with an ordinal mind change bound of $\omega \times m$.

We employ the above results to give mind change bounds for the following classes of Prolog programs:

- (a) Shapiro's linear logic programs (similar result can be shown for the class of hereditary logic programs [MSS91,MSS93] and reductive logic programs [KR96]);

- (b) Krishna Rao's linearly-moded logic programs with bounded body length (similar result holds for Arimura and Shinohara's linearly-covering logic programs with bounded body length [AS94]).

In the sequel we proceed as follows. Section 2 introduces the preliminaries of ordinal mind change identification. Section 3 establishes sufficient condition for learnability with ordinal mind change bound for both positive data and positive and negative data. In Section 4, we introduce preliminaries of logic programming and apply the results from Section 3 to establish mind change bounds for learnability of minimal models of various subclasses of length-bounded Prolog programs.

2 Ordinal Mind Change Identification

N denotes the set of natural numbers, $\{0, 1, 2, \dots\}$. Any unexplained recursion theoretic notation is from [Rog67]. Cardinality of a set S is denoted $\text{card}(S)$. The maximum and minimum of a set are represented by $\max(\cdot)$ and $\min(\cdot)$, respectively. The symbols $\subseteq, \supseteq, \subset, \supset, \emptyset$ respectively stand for subset, superset, proper subset, proper superset, and the emptyset. Λ denotes the empty sequence.

Definition 1 A class of languages $\mathcal{L} = \{L_i \mid i \in N\}$ is a *uniformly decidable family of computable languages* just in case there exists a computable function f such that for each $i \in N$ and for each $x \in U$,

$$f(i, x) = \begin{cases} 1, & \text{if } x \in L_i, \\ 0, & \text{otherwise.} \end{cases}$$

As noted in the introduction, we only consider uniformly decidable families of computable languages. In the next three definitions we introduce texts (positive data presentation), informants (positive and negative data presentation), and learning machines.

Definition 2 [Gol67]

- (a) A *text* T is a mapping from N into $U \cup \{\#\}$. (The symbol $\#$ models pauses in data presentation.)
- (b) $\text{content}(T)$ denotes the intersection of U and the range of T .
- (c) A text T is for a language L iff $L = \text{content}(T)$.
- (d) The initial sequence of text T of length n is denoted $T[n]$.
- (e) The set of all finite initial sequences of U and $\#$'s is denoted SEQ. We let σ and τ range over SEQ.

Definition 3 [Gol67]

- (a) An *informant* I is an infinite sequence over $U \times \{0, 1\}$ such that for each $x \in U$ either $(x, 1)$ or $(x, 0)$ (but not both) appear in the sequence.
- (b) I is an informant for L iff $(x, 1)$ appears in I if $x \in L$ and $(x, 0)$ appears in I if $x \notin L$.
- (c) $I[n]$ denotes the initial sequence of informant I with length n .
- (d) $\text{content}(I) = \{(x, y) \mid (x, y) \text{ appears in sequence } I\}$; $\text{content}(I[n])$ is defined similarly.
- (e) $\text{PosInfo}(I[n]) = \{x \mid (x, 1) \in \text{content}(I[n])\}$; $\text{NegInfo}(I[n]) = \{x \mid (x, 0) \in \text{content}(I[n])\}$.
- (f) $\text{InfSEQ} = \{I[n] \mid I \text{ is an informant for some } L \subseteq U\}$. We let σ and τ also range over InfSEQ .

We let $\sigma \subseteq \tau$ denote “ σ is an initial sequence of τ .”

Definition 4 Let \mathcal{F} denote the set of all formal systems.

- (a) A learning machine from texts (informants) is an algorithmic mapping from SEQ (InfSEQ) into $\mathcal{F} \cup \{?\}$. A typical variable for learning machines is \mathbf{M} .
- (b) \mathbf{M} is said to *converge* on text T to Γ (written: $\mathbf{M}(T)$ converges to Γ or $\mathbf{M}(T) \downarrow = \Gamma$) just in case for all but finitely many n , $\mathbf{M}(T[n]) = \Gamma$. A similar definition holds for informants.

A conjecture of “?” by a machine is interpreted as “no guess at this moment.” This is useful to avoid biasing the number of mind changes of a machine. For this paper, we assume, without loss of generality, that $\sigma \subseteq \tau$ and $\mathbf{M}(\sigma) \neq ?$ implies $\mathbf{M}(\tau) \neq ?$.

We next introduce ordinals as models for mind change counters. We assume a fixed notation system, O , and partial ordering of ordinal notations as used by, for example, Kleene [Kle38,Rog67,Sac90]. \preceq, \prec, \succeq and \succ on ordinal notations below refer to the partial ordering of ordinal notations in this system. We do not go into the details of the notation system used, but instead refer the reader to [Kle38,Rog67,Sac90,CJS95,FS93]. In the sequel, we are somewhat informal and use $m \in N$ as notation for the corresponding ordinals. We let $+$ and \times also denote the addition and multiplication over ordinals.

Definition 5 \mathbf{F} , an algorithmic mapping from SEQ (or InfSEQ) into ordinal notations, is an *ordinal mind change counter function* just in case $(\forall \sigma \subseteq \tau)[\mathbf{F}(\sigma) \succeq \mathbf{F}(\tau)]$.

Definition 6 [FS93,JS97] Let α be an ordinal notation.

- (a) We say that \mathbf{M} , with associated ordinal mind change counter function

\mathbf{F} , \mathbf{TxtEx}_α -*identifies* a text T just in case the following three conditions hold:

- (i) $\mathbf{M}(T)\downarrow = \Gamma$ and $\mathbf{Lang}(\Gamma) = \text{content}(T)$,
 - (ii) $\mathbf{F}(\Lambda) = \alpha$, and
 - (iii) $(\forall n)[? \neq \mathbf{M}(T[n]) \neq \mathbf{M}(T[n+1]) \Rightarrow \mathbf{F}(T[n]) \succ \mathbf{F}(T[n+1])]$.
- (b) \mathbf{M} , with associated ordinal mind change counter function \mathbf{F} , \mathbf{TxtEx}_α -*identifies* L (written: $L \in \mathbf{TxtEx}_\alpha(\mathbf{M}, \mathbf{F})$) just in case \mathbf{M} , with associated ordinal mind change counter function \mathbf{F} , \mathbf{TxtEx}_α -*identifies* each text for L .
- (c) $\mathbf{TxtEx}_\alpha = \{\mathcal{L} \mid (\exists \mathbf{M}, \mathbf{F})[\mathcal{L} \subseteq \mathbf{TxtEx}_\alpha(\mathbf{M}, \mathbf{F})]\}$.

Definition 7 [FS93,AJS97] Let α be an ordinal notation.

- (a) We say that \mathbf{M} , with associated ordinal mind change counter function \mathbf{F} , \mathbf{InfEx}_α -*identifies* an informant I just in case the following three conditions hold:
 - (i) $\mathbf{M}(I)\downarrow = \Gamma$ and $\mathbf{Lang}(\Gamma) = \text{PosInfo}(I)$,
 - (ii) $\mathbf{F}(\Lambda) = \alpha$, and
 - (iii) $(\forall n)[? \neq \mathbf{M}(I[n]) \neq \mathbf{M}(I[n+1]) \Rightarrow \mathbf{F}(I[n]) \succ \mathbf{F}(I[n+1])]$.
- (b) \mathbf{M} , with associated ordinal mind change counter function \mathbf{F} , \mathbf{InfEx}_α -*identifies* L (written: $L \in \mathbf{InfEx}_\alpha(\mathbf{M}, \mathbf{F})$) just in case \mathbf{M} , with associated ordinal mind change counter function \mathbf{F} , \mathbf{InfEx}_α -*identifies* each informant for L .
- (c) $\mathbf{InfEx}_\alpha = \{\mathcal{L} \mid (\exists \mathbf{M}, \mathbf{F})[\mathcal{L} \subseteq \mathbf{InfEx}_\alpha(\mathbf{M}, \mathbf{F})]\}$.

We refer the reader to Ambainis [Amb95] for a discussion on how the learnability classes depend on the choice of the ordinal notation.

3 Conditions for learnability with mind change bound

We now formally define what it means for a language defining framework to have the property of *effective bounded finite thickness*. Recall that a semantic mapping \mathbf{Lang} is *monotonic* just in case for any two formal systems Γ and Γ' , $\Gamma \subseteq \Gamma' \Rightarrow \mathbf{Lang}(\Gamma) \subseteq \mathbf{Lang}(\Gamma')$. Also, recall from the introduction that we only consider language defining frameworks that yield uniformly decidable families of computable languages.

Definition 8 Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework such that \mathbf{Lang} is monotonic. For any $X \subseteq U$, let

$$\text{Gen}_X \stackrel{\text{def}}{=} \{\Gamma \mid \Gamma \subseteq R \wedge \text{card}(\Gamma) < \infty \wedge X \subseteq \mathbf{Lang}(\Gamma)\},$$

$$\text{Min}_X \stackrel{\text{def}}{=} \{\Gamma \in \text{Gen}_X \mid (\forall \Gamma' \in \text{Gen}_X)[\Gamma' \not\subseteq \Gamma]\},$$

and

$$\text{Min}_X^m \stackrel{\text{def}}{=} \{\Gamma \in \text{Min}_X \mid \text{card}(\Gamma) \leq m\}.$$

$\langle U, R, \mathbf{Lang} \rangle$ is said to have *effective m -bounded finite thickness* just in case for all finite $X \subseteq U$, Min_X^m is finite and can be obtained effectively in X (i.e., there are functions, recursive in X , for enumerating Min_X^m , and for finding cardinality of Min_X^m).

$\langle U, R, \mathbf{Lang} \rangle$ is said to have *effective bounded finite thickness* just in case it has effective m -bounded finite thickness for each $m \in \mathbb{N}$.

Note that $\text{Min}_X = \bigcup_m \text{Min}_X^m$. Also, if $\langle U, R, \mathbf{Lang} \rangle$ has effective $(m + 1)$ -bounded finite thickness, then it has effective m -bounded finite thickness.

Proposition 9 *Suppose $X \subseteq X' \subseteq U$, such that $\text{Min}_{X'}$ is nonempty. Then, Min_X is not empty, and for every $\Gamma' \in \text{Min}_{X'}$, there exists a $\Gamma \in \text{Min}_X$ such that $\Gamma \subseteq \Gamma'$.*

PROOF. Suppose $\Gamma' \in \text{Min}_{X'}$. Then clearly, $X \subseteq X' \subseteq \mathbf{Lang}(\Gamma')$. Since Γ' is finite, there exists a finite subset Γ of Γ' such that $X \subseteq \mathbf{Lang}(\Gamma)$, but $X \not\subseteq \mathbf{Lang}(\Gamma'')$ for any $\Gamma'' \subset \Gamma$. It follows that $\Gamma \in \text{Min}_X$. ■

Proposition 10 *Suppose $X \subseteq U$, such that Min_X is nonempty. Then, for any $\Gamma \in \text{Min}_X$, there exists a finite $X' \subseteq X$ such that $\Gamma \in \text{Min}_{X'}$.*

PROOF. Proposition is trivial for finite X . So let X be infinite. Let $\Gamma \in \text{Min}_X$. Let x_0, x_1, \dots be a listing of elements of X . Let $S_i = \{\Gamma' \mid \Gamma' \subseteq \Gamma \wedge \{x_0, \dots, x_i\} \subseteq \mathbf{Lang}(\Gamma')\}$. Note that each S_i is nonempty (since Γ belongs to every S_i). Moreover, $S_i \supseteq S_{i+1}$. Thus, $\lim_{i \rightarrow \infty} S_i$ converges to a set S . Now, for every $\Gamma' \in S$, $X \subseteq \mathbf{Lang}(\Gamma')$ (by definition of S_i). Thus, $S = \{\Gamma\}$ (since $\Gamma \in \text{Min}_X$).

Let i be such that $S = S_i$. Hence, it follows that $\{x_0, \dots, x_i\} \subseteq X \subseteq \mathbf{Lang}(\Gamma)$, and for all $\Gamma' \subset \Gamma$, $\{x_0, \dots, x_i\} \not\subseteq \mathbf{Lang}(\Gamma')$ (by definition of S_i). It follows that $\Gamma \in \text{Min}_{\{x_0, \dots, x_i\}}$. ■

3.1 Learnability from positive data

We now show that if a language defining framework has effective m -bounded finite thickness then the class of languages defined by formal systems of cardinality $\leq m$ can be $\mathbf{TextEx}_{\omega, m}$ -identified. This result is a generalization of a lemma from [JS97]. To state this result, we need some technical machinery which we describe next.

Definition 11 A *search tree* is a finite labeled rooted tree. We denote the label of node, v , in search tree H by $C_H(v)$.

Intuitively, the labels on the nodes are formal systems. We next introduce a partial order on search trees.

Definition 12 Suppose H_1 and H_2 are two search trees. We say that $H_1 \preceq H_2$ just in case the following properties are satisfied:

- (A) the root of H_1 has the same label as the root of H_2 ;
- (B) H_1 is a labeled subgraph of H_2 ; and
- (C) all nodes of H_1 , except the leaves, have exactly the same children in both H_1 and H_2 .

Essentially, $H_1 \preceq H_2$ means that H_2 is obtained by attaching some (possibly empty) trees to some of the leaves of the search tree H_1 . It is helpful to formalize the notion of *depth* of a search tree as follows: the depth of the root is 0; the depth of a child is 1 + the depth of the parent; and the depth of a search tree is the depth of its deepest leaf.

Q , an algorithmic mapping from SEQ to search trees, is called an m -Explorer iff the following properties are satisfied:

- (A) $\sigma \subseteq \tau \Rightarrow Q(\sigma) \preceq Q(\tau)$;
- (B) $(\forall \sigma)[\text{depth}(Q(\sigma)) \leq m]$; and
- (C) for all T , $Q(T) \downarrow$, i.e., $(\forall n)[Q(T[n]) = Q(T[n+1])]$.

(The reader should note that (C) is actually implied by (A) and (B); (C) has been included to emphasize the point.)

We can now state the lemma from [JS97] that links the existence of an m -Explorer to $\mathbf{TextEx}_{\omega^m}$ -identification.

Lemma 13 *Suppose Q is an m -Explorer. Then there exists a machine \mathbf{M} and an associated ordinal mind change counter \mathbf{F} such that the following properties are satisfied:*

- (A) $(\forall \text{ texts } T)[\mathbf{M}(T) \downarrow]$;
- (B) $\mathbf{F}(\Lambda) = \omega^m$; and
- (C) *if there exists a node v in $Q(T)$ such that $\mathbf{Lang}(C_{Q(T)}(v)) = \text{content}(T)$, then \mathbf{M} , with associated mind change counter \mathbf{F} , $\mathbf{TextEx}_{\omega^m}$ -identifies T .*

We now establish a theorem that bridges Lemma 13 with the notion of effective bounded finite thickness and $\mathbf{TxtEx}_{\omega^m}$ -identifiability.

Theorem 14 *Suppose $m > 0$. Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework which has effective m -bounded finite thickness. Let*

$$\mathcal{L}^m \stackrel{\text{def}}{=} \{\mathbf{Lang}(\Gamma) \mid \Gamma \subseteq R \wedge \text{card}(\Gamma) \leq m\}.$$

Then $\mathcal{L}^m \in \mathbf{TxtEx}_{\omega^m}$.

PROOF. Fix $m > 0$, and assume the hypothesis. We construct an m -Explorer Q as follows. Let T be a text. Let $Q(\Lambda)$ be just a root with label \emptyset . $Q(T[n+1])$ is obtained from $Q(T[n])$ as follows. For each leaf v in $Q(T[n])$ with label Γ such that $\text{depth}(v) < m$ and $\text{content}(T[n+1]) \not\subseteq \mathbf{Lang}(\Gamma)$, do the following:

for each $\Gamma' \in \text{Min}_{\text{content}(T[n+1])}^m$, such that $\Gamma \subset \Gamma'$, add a child to v with label Γ' .

It is easy to verify that Q is an m -Explorer. Also note that if node u is a parent of node v in $Q(T[n])$, where label on node u is Γ and label on node v is Γ' , then $\Gamma \subset \Gamma'$, and in particular $\text{card}(\Gamma) < \text{card}(\Gamma')$. Thus, label of a node at depth d in $Q(T[n])$, must have cardinality at least d .

We now claim, inductively on n , that

$$(\forall \Gamma \in \text{Min}_{\text{content}(T[n])}^m)[\Gamma \text{ is a label of some leaf in } Q(T[n])].$$

For $n = 0$, clearly, $\text{Min}^m(\Lambda) = \{\emptyset\}$, and \emptyset is the label of the root (which is also a leaf) of $Q(\Lambda)$.

Suppose the induction hypothesis holds for $n = k$. We show that the induction hypothesis holds for $n = k + 1$. Consider any element Γ of $\text{Min}_{\text{content}(T[k+1])}^m$. Let $\Gamma' \in \text{Min}_{\text{content}(T[k])}^m$ be such that $\Gamma' \subseteq \Gamma$ (by Proposition 9, there exists such a Γ'). Now, either

(A) $\Gamma' = \Gamma$ is a leaf of both $Q(T[k])$ and $Q(T[k+1])$ or

(B) $\mathbf{Lang}(\Gamma')$ does not contain $\text{content}(T[k+1])$, $\text{card}(\Gamma') < \text{card}(\Gamma) \leq m$, and thus, in construction of $Q(T[k+1])$, a node with label Γ would be added to each leaf with label Γ' in $Q(T[k])$ (there exists at least one such leaf by the induction hypothesis).

Thus, induction hypothesis holds for $n = k + 1$. It thus follows by Proposition 10 that, for $L \in \mathcal{L}^m$, for any text T for L , every Γ in Min_L^m , is a leaf of $Q(T)$. The theorem now follows from Lemma 13. ■

Corollary 15 *Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework which has effective bounded finite thickness. For each $m > 0$, let*

$$\mathcal{L}^m \stackrel{\text{def}}{=} \{\mathbf{Lang}(\Gamma) \mid \Gamma \subseteq R \wedge \text{card}(\Gamma) \leq m\}.$$

Then $\mathcal{L}^m \in \mathbf{TxtEx}_{\omega^m}$.

The above theorem's proof gives a new algorithm for learning the classes discussed in Section 4. Previous learnability results for these classes relied on Angluin's algorithm [Ang80b]. It should be noted that the same technique was used in [JS97] to show that language classes formed by taking union of at most n pattern languages and language classes formed by using at most n clauses of length bounded elementary formal systems [ASY92, Shi94] belong to $\mathbf{TxtEx}_{\omega^n}$.

3.2 Learnability from positive and negative data

In this section we show that if a language defining framework has effective bounded finite thickness then the class of languages defined by formal systems of cardinality $\leq m$ can be \mathbf{InfEx} -identified with an ordinal mind change bound of $\omega \times m$. This result is a generalization of a result about unions of pattern languages from [AJS97]. We first introduce some technical machinery.

Let $\mathbf{Pos} \subseteq U$ and $\mathbf{Neg} \subseteq U$ be two disjoint finite sets such that $\mathbf{Pos} \neq \emptyset$. Then let

$$Z_i^{\mathbf{Pos}, \mathbf{Neg}} \stackrel{\text{def}}{=} \{\Gamma \subseteq R \mid \text{card}(\Gamma) = i \wedge [\mathbf{Pos} \subseteq \mathbf{Lang}(\Gamma)] \wedge [\mathbf{Neg} \subseteq U - \mathbf{Lang}(\Gamma)]\}.$$

The next lemma and corollary shed light on computation of $Z_i^{\mathbf{Pos}, \mathbf{Neg}}$.

Lemma 16 *Suppose $i \in \mathbb{N}$. Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework with effective $(i+1)$ -bounded finite thickness. Let $\mathbf{Pos} \neq \emptyset$ and \mathbf{Neg} be two disjoint finite subsets of U . Suppose $(\forall j \leq i)[Z_j^{\mathbf{Pos}, \mathbf{Neg}} = \emptyset]$. Then, $Z_{i+1}^{\mathbf{Pos}, \mathbf{Neg}}$ can be computed effectively from \mathbf{Pos} and \mathbf{Neg} . (Note that $Z_{i+1}^{\mathbf{Pos}, \mathbf{Neg}}$ must be finite in this case!)*

PROOF. Let \mathbf{Pos} , \mathbf{Neg} , and i be as given in the hypothesis of the lemma.

We claim that $Z_{i+1}^{\mathbf{Pos}, \mathbf{Neg}} \subseteq \{\Gamma \mid \Gamma \in \text{Min}_{\mathbf{Pos}}^{i+1}\}$. To see this, suppose $\Gamma \in Z_{i+1}^{\mathbf{Pos}, \mathbf{Neg}}$. Clearly, $\mathbf{Pos} \subseteq \mathbf{Lang}(\Gamma)$. Suppose there exists a $\Gamma' \subset \Gamma$ such that $\mathbf{Pos} \subseteq \mathbf{Lang}(\Gamma')$. Then, clearly, $\mathbf{Lang}(\Gamma') \subseteq \mathbf{Lang}(\Gamma)$. Thus, $\mathbf{Neg} \cap \mathbf{Lang}(\Gamma') \subseteq \mathbf{Neg} \cap \mathbf{Lang}(\Gamma) = \emptyset$. Thus, $\Gamma' \in Z_{\text{card}(\Gamma')}^{\mathbf{Pos}, \mathbf{Neg}}$, a contradiction to the hypothesis of the lemma. Thus, for all $\Gamma' \subset \Gamma$, $\mathbf{Pos} \not\subseteq \mathbf{Lang}(\Gamma')$. Thus, $\Gamma \in \text{Min}_{\mathbf{Pos}}^{i+1}$.

It follows that $Z_{i+1}^{\mathbf{Pos}, \mathbf{Neg}} = \{\Gamma \in \text{Min}_{\mathbf{Pos}}^{i+1} \mid \mathbf{Neg} \cap \mathbf{Lang}(\Gamma) = \emptyset\}$.

Since $\text{Min}_{\text{Pos}}^{i+1}$ is obtainable effectively from Pos , it follows that $Z_{i+1}^{\text{Pos}, \text{Neg}}$ is obtainable effectively from Pos and Neg . \blacksquare

Corollary 17 *Suppose $m > 0$, $m \in N$. Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework with effective m -bounded finite thickness. Let $\text{Pos} \neq \emptyset$ and Neg be two disjoint finite subsets of U . Then, effectively from Pos , Neg one can determine $i = \min(\{i \mid Z_j^{\text{Pos}, \text{Neg}} \neq \emptyset\} \cup \{m + 1\})$, and corresponding $Z_i^{\text{Pos}, \text{Neg}}$ (which must be finite).*

PROOF. Note that $\mathbf{Lang}(\emptyset)$ is empty. The corollary now follows by repeated use of Lemma 16 until one finds an i such that $Z_i^{\text{Pos}, \text{Neg}} \neq \emptyset$ or discovers that $Z_m^{\text{Pos}, \text{Neg}} = \emptyset$. \blacksquare

We now show our result for identification with ordinal mind change bound from informants.

Theorem 18 *Suppose $m > 0$. Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework with effective m -bounded finite thickness. Let*

$$\mathcal{L}^m \stackrel{\text{def}}{=} \{\mathbf{Lang}(\Gamma) \mid \Gamma \subseteq R \wedge \text{card}(\Gamma) \leq m\}.$$

Then $\mathcal{L}^m \in \mathbf{InfEx}_{\omega \times m}$.

PROOF. Fix m . Let I be an informant. Then for $n \in N$, $\mathbf{M}(I[n])$ and $\mathbf{F}(I[n])$ are defined as follows.

Let $\text{Pos} = \text{PosInfo}(I[n])$ and $\text{Neg} = \text{NegInfo}(I[n])$.

If $\text{Pos} = \emptyset$, then $\mathbf{M}(I[n]) = ?$ and $\mathbf{F}(I[n]) = \omega \times m$.

If $\text{Pos} \neq \emptyset$, then let $j = \min(\{j' \mid Z_{j'}^{\text{Pos}, \text{Neg}} \neq \emptyset\} \cup \{m + 1\})$. Note that j (and corresponding $Z_j^{\text{Pos}, \text{Neg}}$) can be found effectively from $I[n]$, using Corollary 17. Now define $\mathbf{M}(I[n])$ and $\mathbf{F}(I[n])$ based on the following cases.

If $j > m$, then let $\mathbf{M}(I[n]) = \mathbf{M}(I[n - 1])$, and $\mathbf{F}(I[n]) = \mathbf{F}(I[n - 1])$.

If $j \leq m$, then let $\mathbf{M}(I[n]) = \Gamma$, where Γ is the lexicographically least element in $Z_j^{\text{Pos}, \text{Neg}}$, and let $\mathbf{F}(I[n]) = \omega \times k + \ell$, where $k = m - j$, and $\ell = \text{card}(Z_j^{\text{Pos}, \text{Neg}}) - 1$.

It is easy to verify that \mathbf{M}, \mathbf{F} witness that $\mathcal{L}^m \in \mathbf{InfEx}_{\omega \times m}$. \blacksquare

Corollary 19 *Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework with effec-*

tive bounded finite thickness. For $m > 0$, let

$$\mathcal{L}^m \stackrel{\text{def}}{=} \{\mathbf{Lang}(\Gamma) \mid \Gamma \subseteq R \wedge \text{card}(\Gamma) \leq m\}.$$

Then $\mathcal{L}^m \in \mathbf{InfEx}_{\omega \times m}$.

4 Classes of logic programs

In this section, we describe the application of Theorem 14 (and of Theorem 18) to certain classes of logic programs. These classes are known to have bounded finite thickness. It turns out that the proof of bounded finite thickness can easily be modified to show effective bounded finite thickness. However, for the sake of completeness, we present the definitions and the results for two representative classes. We first describe the preliminaries from logic programming; the reader is referred to Lloyd [Llo87] for any unexplained notation.

Let Π, Σ, \mathcal{X} be mutually disjoint sets such that Π and Σ are finite. Π is the set of *predicate symbols*, Σ is the set of *function symbols*, and \mathcal{X} is the set of *variables*. The arity of a function or a predicate symbol p is denoted $\text{arity}(p)$. The set of terms constructed from the function symbols in Σ and variables in \mathcal{X} is denoted $\text{Terms}(\Sigma, \mathcal{X})$. $\text{Atoms}(\Pi, \Sigma, \mathcal{X})$ denotes the set of atoms formed from predicate symbols in Π and terms in $\text{Terms}(\Sigma, \mathcal{X})$. The set of ground atoms for a predicate symbol p , then is $\text{Atoms}(\{p\}, \Sigma, \emptyset)$; we denote this set by $B(p)$. The size of a term t , denoted $|t|$, is the number of symbols other than punctuation symbols in t . The *body length* of a definite clause is the number of literals in its body. The *length of a logic program* P , denoted $\text{Length}(P)$, is just the number of clauses in P .

Following the treatment of [KR96], we take the least Herbrand model semantics of logic programs as our monotonic semantic mapping in the present paper. We will refer to the target predicate being learned by the symbol p . It should be noted that the treatment can be generalized to take into account the situation of multiple predicates in an obvious way. Then our language defining frameworks will be of the form $\langle B(p), LP, M_p \rangle$, where LP is the class of Prolog clauses being considered and M_p denotes the semantic mapping such that $M_p(P)$ is the set of all atoms of the target predicate p in the least Herbrand model of P .

We next describe linear Prolog programs introduced by Shapiro [Sha81].

Definition 20 [Sha81] A definite clause $p(t_1, \dots, t_n) \leftarrow q_1(s_{1_1}, \dots, s_{1_{n_1}}), \dots, q_k(s_{k_1}, \dots, s_{k_{n_k}})$ is called *linear* just in case for each i , $1 \leq i \leq k$, $|t_1\sigma| + \dots + |t_n\sigma| \geq |s_{i_1}\sigma| + \dots + |s_{i_{n_i}}\sigma|$ for any substitution σ . A logic program P is said to be *linear* just in case each clause in P is linear.

Shinohara [Shi91] showed the following.

Theorem 21 [Shi91] *The class of least Herbrand models of linear Prolog programs is a uniformly decidable family of computable languages.*

Let \mathbf{LC} denote the class of all linear clauses and M_p be a semantic mapping such that $M_p(P)$ is the set of all atoms of the target predicate p in the least Herbrand model of P . Then we have the following.

Theorem 22 *The language defining framework $\langle B(p), \mathbf{LC}, M_p \rangle$ has effective bounded finite thickness.*

PROOF. Shinohara's proof of $\langle B(p), \mathbf{LC}, M_p \rangle$ having bounded finite thickness can easily be modified to show that it is effective. \blacksquare

We note that a similar result can be shown for the class of hereditary logic programs [MSS91, MSS93] and reductive logic programs [KR96].

As a consequence of the above theorem and the results in Section 3, for each $m \geq 1$, the class of languages $\mathcal{L}^m = \{M_p(P) \mid P \in \mathbf{LC} \wedge \mathbf{Length}(P) \leq m\}$ is a member of $\mathbf{TxtEx}_{\omega^m}$ and of $\mathbf{InfEx}_{\omega \times m}$.

The above results were for classes of logic programs that did not allow local variables. We now turn our attention to the mind change complexity of learning classes of logic programs that allow local variables. We show that the language defining frameworks associated with the class of linearly-covering Prolog programs of Arimura and Shinohara and the class of linearly-moded Prolog programs of Krishna Rao have effective bounded finite thickness if the body length of the clauses is bounded. Since the class of linearly-covering programs are subsumed by the class of linearly-moded programs, we show the result for only the latter class. But, first we introduce some terminology about parametric size of terms, and moded logic programs.

Let $\langle \rangle$ denote an empty list.

Definition 23 The *parametric size* of a term t , denoted $\mathbf{Psize}(t)$, is defined inductively as follows:

- (a) if t is a variable x then $\mathbf{Psize}(t)$ is the linear expression x ;
- (b) if t is the empty list, then $\mathbf{Psize}(t)$ is 0;
- (c) if $t = f(t_1, \dots, t_n)$ and $f \in \Sigma - \{\langle \rangle\}$, then $\mathbf{Psize}(t)$ is the linear expression $1 + \mathbf{Psize}(t_1) + \dots + \mathbf{Psize}(t_n)$.

We usually denote a sequence of terms t_1, \dots, t_n by \mathbf{t} . The parametric size of a sequence of terms t_1, \dots, t_n is the sum $\mathbf{Psize}(t_1) + \dots + \mathbf{Psize}(t_n)$.

The definition of linearly-moded programs requires the notion of modes associated with each argument in a predicate.

Definition 24 (a) A *mode declaration* for an n -ary predicate p is a mapping from $\{1, \dots, n\}$ to the set $\{+, -\}$. (b) Let \mathbf{md} be a mode declaration for the predicate p . Then the sets $+(p) = \{j \mid \mathbf{md}(j) = +\}$ and $-(p) = \{j \mid \mathbf{md}(j) = -\}$ are the sets of input and output positions of p , respectively.

If each predicate in a logic program has a unique mode declaration, the program is referred to as a moded program. In dealing with moded programs, it is useful to group together the input and output arguments, i.e., $p(\mathbf{s}; \mathbf{t})$ is an atom with input terms \mathbf{s} and output terms \mathbf{t} .

The definition of linearly-moded logic programs requires the following technical notion.

Definition 25 [KR96] Let P be a moded logic program and let I be a mapping from the set of predicates occurring in P to sets of input positions such that $I(p) \subseteq +(p)$ for each predicate p in P . Then for an atom $A = p(\mathbf{s}; \mathbf{t})$, the following linear inequality is denoted $\text{LI}(A, I)$.

$$\sum_{i \in I(p)} \text{Psize}(s_i) \geq \sum_{j \in -(p)} \text{Psize}(t_j).$$

We now define Krishna Rao's notion of what it means for a logic program to be linearly-moded.

Definition 26 [KR96]

- (a) Let P be a moded logic program and let I be a mapping from the set of predicates in P to the sets of input positions satisfying $I(p) \subseteq +(p)$ for each predicate p in P . P is said to be *linearly-moded with respect to I* if each clause

$$p_0(\mathbf{s}_0; \mathbf{t}_0) \leftarrow p_1(\mathbf{s}_1; \mathbf{t}_1), \dots, p_k(\mathbf{s}_k; \mathbf{t}_k)$$

in P satisfies the following two conditions:

- (i) $\text{LI}(A_1, I), \dots, \text{LI}(A_{j-1}, I)$ together imply $\text{Psize}(\mathbf{s}_0) \geq \text{Psize}(s_j)$, for each $j \geq 1$, and
 - (ii) $\text{LI}(A_1, I), \dots, \text{LI}(A_k, I)$ together imply $\text{LI}(A_0, I)$, where A_j is the atom $p_j(\mathbf{s}_j; \mathbf{t}_j)$ for each $j \geq 0$.
- (b) A logic program P is said to be *linearly-moded* just in case it is linearly-moded with respect to some mapping I .

We now introduce the language defining framework of linearly-moded clauses. For $k > 0$, let LMC_k denote the set of all linearly-moded clauses of body length at most k . Then the language defining framework associated with linearly-moded clauses is $\langle B(p), \text{LMC}_k, M_p \rangle$.

Theorem 27 [KR96] *For $k \geq 1$, the class of least Herbrand models of logic programs with clauses in LMC_k is an indexed family of recursive languages.*

Theorem 28 *For $k \geq 1$, the language defining framework $\langle B(p), \text{LMC}_k, M_p \rangle$ has effective bounded finite thickness.*

PROOF. Krishna Rao's [KR96] proof of $\langle B(p), \text{LMC}_k, M_p \rangle$ having bounded finite thickness can easily be made effective. ■

As a consequence of the above theorem and the results in Section 3, for each $m \geq 1$, for each $k \geq 1$, the class of languages $\mathcal{L}_k^m = \{M_p(P) \mid P \in \text{LMC}_k \wedge \text{Length}(P) \leq m\}$ is a member of $\mathbf{TxtEx}_{\omega^m}$ and of $\mathbf{InfEx}_{\omega \times m}$. The reader should note that the bound k on the body length of clauses is crucial for the effective bounded thickness property. It can be shown that without such a restriction the class of least Herbrand models of length-bounded linearly-moded programs contains a superfinite subclass, thereby ruling out its learnability from positive data. Krishna Rao [KR96] has shown that both the classes of linear clauses and the class of linearly-covering clauses is included in the class of linearly-moded clauses, but the classes of linear clauses and linearly-covering clauses are incomparable to each other.

5 Some Final Remarks

A natural remaining question is whether the bounds of ω^m and $\omega \times m$ are tight. It can be shown for the example classes in this paper that for identification from positive data, the ordinal bound of ω^m is tight. The reason being that unions of upto m -pattern languages is contained in all these classes, but cannot be identified in \mathbf{TxtEx}_α , for $\alpha \prec \omega^m$ [JS97]. For identification from both positive and negative data, it is still open if the bound of $\omega \times m$ is tight. However, we can show an improvement on the bound $\omega \times m$ under certain conditions if a restricted version of the language equivalence problem is decidable. In particular, we can show the following. Let $a \div b$ denote $a - b$, if $a \geq b$, and 0 otherwise.

Theorem 29 *Suppose $m > 0$. Let $\langle U, R, \mathbf{Lang} \rangle$ be a language defining framework with effective m -bounded finite thickness. Let*

$$\mathcal{L}^m \stackrel{\text{def}}{=} \{\mathbf{Lang}(\Gamma) \mid \Gamma \subseteq R \wedge \text{card}(\Gamma) \leq m\}.$$

Suppose $q \leq m$, and the equivalence of $\mathbf{Lang}(\Gamma)$ and $\mathbf{Lang}(\Gamma')$ is decidable for $\text{card}(\Gamma) = \text{card}(\Gamma') \leq q$.

Then $\mathcal{L}^m \in \mathbf{InfEx}_{\omega \times (m-q) + (q \div 1)}$.

PROOF. The proof of this theorem is on the same lines as that of Theorem 18. We just modify that proof, when $j \leq q$, to exploit the decidability of equivalence of decision procedures.

Fix m . Let I be an informant. Then for $n \in N$, $\mathbf{M}(I[n])$ and $\mathbf{F}(I[n])$ are defined as follows.

Let $\text{Pos} = \text{PosInfo}(I[n])$ and $\text{Neg} = \text{NegInfo}(I[n])$.

If $\text{Pos} = \emptyset$, then $\mathbf{M}(I[n]) = ?$ and $\mathbf{F}(I[n]) = \omega \times (m - q) + (q \div 1)$.

If $\text{Pos} \neq \emptyset$, then let $j = \min(\{j' \mid Z_{j'}^{\text{Pos}, \text{Neg}} \neq \emptyset\} \cup \{m + 1\})$. Note that j (and corresponding $Z_j^{\text{Pos}, \text{Neg}}$) can be found effectively from $I[n]$, using Corollary 17. Now define $\mathbf{M}(I[n])$ and $\mathbf{F}(I[n])$ based on the following cases.

If $j > m$, then let $\mathbf{M}(I[n]) = \mathbf{M}(I[n - 1])$, and $\mathbf{F}(I[n]) = \mathbf{F}(I[n - 1])$.

If $q < j \leq m$, then let $\mathbf{M}(I[n]) = \Gamma$, where Γ is the lexicographically least element in $Z_j^{\text{Pos}, \text{Neg}}$, and let $\mathbf{F}(I[n]) = \omega \times k + \ell$, where $k = m - j$, and $\ell = \text{card}(Z_j^{\text{Pos}, \text{Neg}}) - 1$.

If $0 < j \leq q$, and all decision procedures in $Z_j^{\text{Pos}, \text{Neg}}$ are equivalent then let $\mathbf{M}(I[n]) =$ the lexicographically least decision procedure in $Z_j^{\text{Pos}, \text{Neg}}$. Let $\mathbf{F}(I[n]) = \omega \times (m - q) + (q - j)$.

If $0 < j \leq q$, and there exist nonequivalent decision procedures in $Z_j^{\text{Pos}, \text{Neg}}$, then let $\mathbf{M}(I[n]) = \mathbf{M}(I[n - 1])$, and $\mathbf{F}(I[n]) = \mathbf{F}(I[n - 1])$.

It is easy to verify that \mathbf{M}, \mathbf{F} witness that $\mathcal{L}^m \in \mathbf{InfEx}_{\omega \times (m-q) + (q \div 1)}$. ■

Note that for pattern languages, it can be effectively tested whether $\mathbf{Lang}(p) = \mathbf{Lang}(p')$ for patterns p and p' . This property was implicitly used in [AJS97] to show that, $\mathbf{PATTERN}^{n+1}$, the language class formed by taking union of at most $n + 1$ pattern languages, belongs to $\mathbf{InfEx}_{\omega \times n}$. It would be interesting to determine whether the equivalence problem for logic program classes discussed in Section 4 can be effectively solved for program length q for various values of q .

6 Acknowledgements

We thank the anonymous referees for several helpful comments which improved the presentation of the paper. The research of Sanjay Jain was supported in part by NUS grant number RP3992710. The research of Arun Sharma was supported in part by Australian Research Council Grant A49803051. Preliminary version of this paper appeared in Fourth European Conference on Computational Learning Theory 1999.

References

- [AFS96] A. Ambainis, R. Freivalds, and C. Smith. General inductive inference types based on linearly-ordered sets. In *Proceedings of Symposium on Theoretical Aspects of Computer Science*, volume 1046 of *Lecture Notes in Computer Science*, pages 243–253. Springer-Verlag, 1996.
- [AJS97] A. Ambainis, S. Jain, and A. Sharma. Ordinal mind change complexity of language identification. *Theoretical Computer Science*, 220:323–343, 1999. Special issue on Australasian Computer Science.
- [Amb95] A. Ambainis. Power of procrastination in inductive inference: How it depends on used ordinal notations. In Paul Vitányi, editor, *Second European Conference on Computational Learning Theory*, volume 904 of *Lecture Notes in Artificial Intelligence*, pages 99–111. Springer-Verlag, 1995.
- [Ang80a] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
- [Ang80b] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
- [Ari70] S. Arikawa. Elementary formal systems and formal languages—simple formal systems. *Memoirs of the Faculty of Science, Kyushu University Series A*, 24:47–75, 1970.
- [AS94] H. Arimura and T. Shinohara. Inductive inference of Prolog programs with linear data dependency from positive data. In H. Jaakkola, H. Kangassalo, T. Kitahashi, and A. Markus, editors, *Proc. Information Modelling and Knowledge Bases V*, pages 365–375. IOS Press, 1994.
- [ASY92] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, 95:97–113, 1992.
- [BG96] F. Bergadano and G. Gunetti. *Inductive Logic Programming: from Machine Learning to Software Engineering*. MIT Press, 1996.

- [BP73] J. Bārzdīņš and K. Podnieks. The theory of inductive inference. In *Second Symposium on Mathematical Foundations of Computer Science*, pages 9–15. Math. Inst. of the Slovak Academy of Sciences, 1973.
- [CJS95] J. Case, S. Jain, and M. Suraj. Not-so-nearly-minimal-size program inference. In K. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 77–96. Springer-Verlag, 1995.
- [Coh95a] W. W. Cohen. PAC-Learning non-recursive Prolog clauses. *Artificial Intelligence*, 79:1–38, 1995.
- [Coh95b] W. W. Cohen. PAC-Learning recursive logic programs: Efficient algorithms. *Journal of Artificial Intelligence Research*, 2:501–539, 1995.
- [Coh95c] W. W. Cohen. PAC-Learning recursive logic programs: Negative results. *Journal of Artificial Intelligence Research*, 2:541–573, 1995.
- [CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [DMR92a] S. Dzeroski, S. Muggleton, and S. Russell. PAC-Learnability of constrained nonrecursive logic programs. In *Proceedings of the Third International Workshop on Computational Learning Theory and Natural Learning Systems*, 1992. Wisconsin, Madison.
- [DMR92b] S. Dzeroski, S. Muggleton, and S. Russell. PAC-Learnability of determinate logic programs. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 128–135. ACM Press, July 1992.
- [DRD94] L. De Raedt and S. Dzeroski. First-order jk -clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.
- [FP91] A. Frisch and C.D. Page. Learning constrained atoms. In *Proceedings of the Eighth International Workshop on Machine Learning*. Morgan Kaufmann, 1991.
- [FS93] R. Freivalds and C. Smith. On the role of procrastination in machine learning. *Information and Computation*, pages 237–271, 1993.
- [Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [Hau89] D. Hausler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4(1), 1989.
- [JS97] S. Jain and A. Sharma. Elementary formal systems, intrinsic complexity, and procrastination. *Information and Computation*, 132:65–84, 1997.
- [Kha98] R. Khardon. Learning first order universal Horn expressions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 154–165. ACM Press, 1998.

- [Kie93] J-U. Kietz. Some computational lower bounds for the computational complexity of inductive logic programming. In *Proceedings of the 1993 European Conference on Machine Learning*, 1993. Vienna.
- [Kle38] S. Kleene. Notations for ordinal numbers. *Journal of Symbolic Logic*, 3:150–155, 1938.
- [KR96] M. Krishna Rao. A class of Prolog programs inferable from positive data. In A. Arikawa and A. Sharma, editors, *Algorithmic Learning Theory: Seventh International Workshop (ALT '96)*, volume 1160 of *Lecture Notes in Artificial Intelligence*, pages 272–284. Springer-Verlag, 1996.
- [KRS98] M. Krishna Rao and A. Sattar. Learning from entailment of logic programs with local variables. In M. Richter, C. Smith, R. Wiehagen, and T. Zeugmann, editors, *Algorithmic Learning Theory: Ninth International Workshop (ALT '97)*, *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1998. To appear.
- [LD94] N. Lavarač and S. Džeroski. *Inductive Logic Programming*. Ellis Horwood, New York, 1994.
- [Llo87] J.W. Lloyd. *Foundation of Logic Programming (Second Edition)*. Springer, New York, 1987.
- [MDR94] S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19(20):629–679, 1994.
- [MSS91] S Miyano, A. Shinohara, and T. Shinohara. Which classes of elementary formal systems are polynomial-time learnable? In *Proceedings of the Second Workshop on Algorithmic Learning Theory*, pages 139–150, 1991.
- [MSS93] S. Miyano, A. Shinohara, and T. Shinohara. Learning elementary formal systems and an application to discovering motifs in proteins. Technical Report RIFIS-TRCS-37, RIFIS, Kyushu University, 1993.
- [MT96] W. Maass and Gy. Turán. On learnability and predicate logic. Technical Report NC-TR-96-023, NeuroCOLT Technical Report, 1996.
- [NCdW97] S.H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. LNAI Tutorial 1228. Springer-Verlag, 1997.
- [Plo71] G. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, University of Edinburgh, 1971.
- [Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted, MIT Press 1987.
- [Sac90] G. Sacks. *Higher Recursion Theory*. Springer-Verlag, 1990.
- [Sha81] E. Shapiro. Inductive inference of theories from facts. Technical Report 192, Computer Science Department, Yale University, 1981.

- [Shi91] T. Shinohara. Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, 8:371–384, 1991.
- [Shi94] T. Shinohara. Rich classes inferable from positive data: Length-bounded elementary formal systems. *Information and Computation*, 108:175–186, 1994.
- [Smu61] R. Smullyan. *Theory of Formal Systems, Annals of Mathematical Studies, No. 47*. Princeton, NJ, 1961.
- [SSV97] A. Sharma, F. Stephan, and Y. Ventsov. Generalized notions of mind change complexity. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pages 96–108. ACM Press, 1997.
- [Wri89] K. Wright. Identification of unions of languages drawn from an identifiable class. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 328–333. Morgan Kaufmann, 1989.
- [Yam93] A. Yamamoto. Generalized unification as background knowledge in learning logic programs. In K. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Algorithmic Learning Theory: Fourth International Workshop (ALT '93)*, volume 744 of *Lecture Notes in Artificial Intelligence*, pages 111–122. Springer-Verlag, 1993.