Learning Elementary Formal Systems with Queries

Hiroshi Sakamoto [a] Kouichi Hirata [b] Hiroki Arimura [a,c]

[a] *Department of Informatics, Kyushu University, Hakozaki 6-10-1, Fukuoka 812-8581, Japan*

[b] *Department of Artificial Intelligence, Kyushu Institute of Technology, Kawazu 680-4, Iizuka 820-8502, Japan,*

[c] *PRESTO, Japan Science and Technology Co., Japan*

Abstract

The *elementary formal system* (*EFS*) is a kind of logic programs which directly manipulates strings, and the learnability of the subclass called *hereditary* EFSs (*HEFSs*) has been investigated in the frameworks of the PAC-learning, query-learning, and inductive inference models. The hierarchy of HEFS is expressed by HEFS(m, k, t, r), where m, k, t and r denote the number of clauses, the occurrences of variables in the head, the number of atoms in the body, and the arity of predicate symbols. The present paper deals with the learnability of HEFS in the query learning model using *equivalence* queries and additional queries such as *membership*, *predicate membership*, *entailment membership*, and *dependency* queries. We show that the class HEFS$(*, k, t, r)$ is polynomial-time learnable with the equivalence and predicate membership queries and the class HEFS$(*, k, *, r)$ *with termination property* is polynomial-time learnable with the equivalence, entailment membership, and dependency queries for the unbounded parameter $*$. A lowerbound on the number of queries is presented. We also show that the class HEFS$(*, k, t, r)$ is hard to learn with the equivalence and membership queries under the cryptographic assumptions. Furthermore, the learnability of the class of unions of regular pattern languages, which is a subclass of HEFSs, is investigated. The bounded unions of regular pattern languages are *polynomial-time predictable with membership query*. However, all unbounded unions of regular pattern languages are not polynomial-time predictable with membership queries if neither are the DNF formulas.

1 Introduction

The *elementary formal system* (EFS, for short) was originally invented by Smullyan [40] in early 1960s to develop his recursive function theory. Professor Arikawa is a pioneer to employ such an EFS for studying formal language theory [7] in 1970. After about 20 years later, he and his partners [8,9] characterized the EFSs as logic programs over strings and introduced a new hierarchy of various language classes, which includes the four classes of Chomsky hierarchy, the class of pattern languages, and many others. Furthermore, he enhanced EFSs as a unifying framework for language learning, by devising inductive inference algorithms (MIEFS) for these EFS classes based on Shapiro's model inference system [35].

Stimulated by the series of Arikawa's works, many researchers investigated the EFSs in the various models of algorithmic/computational learning theory. Shinohara [38] showed that the *length-bounded* EFSs belonging to the above hierarchy is inferable in the limit from positive examples alone. This result is a valuable extension of the previous inferability of bounded unions of pattern languages [1,37,38,44]. Mukouchi and Arikawa [29] showed that the class of length-bounded EFSs is also refutablly inferable. This notion is a new criterion introduced by Mukouchi and Arikawa [29] that a learner can refute each hypothesis space if it turns out to be insufficient for identification. Many other researchers such as [21,22,27,28] enjoyed various topological properties of EFSs on inductive inference. Jain and Sharma [19] analyzed the mind change complexity and the intrinsic complexity of EFSs.

In contrast to the learnability of EFSs on inductive inference, the polynomial-time learnability is another interesting theme on learning EFSs. For this purpose, Miyano *et al.* [25,26] introduced the subclass *hereditary EFS*, denoted by HEFS. An HEFS consists of clauses that satisfy a substring property such that any pattern appearing in the body also appears as a substring of some argument of the head. This class is rich enough to include the class of pattern languages and class of context-free languages, while the syntax is restricted to allow efficient learning. Actually, this class exactly defines the complexity class PTIME [18]. Miyano *et al.* consider the learnability of the hierarchy HEFS(m, k, t, r) with the parameters such that m, k, t and r are the maximum number of clauses, the maximum number of occurrences of variables in the head, the maximum number of atoms in the body, and the maximum arity of predicate symbols, respectively. They showed that the HEFS(m, k, t, r) is PAC-learnable for every fixed $m, k, t, r \geq 0$.

Other result was shown in the query learning model introduced by Angluin [4]. In this learning model, an algorithm can ask the equivalence, membership, and other types of queries to efficiently learn a target concept. As an interesting relationship between the PAC and query models, it is known that if a concept class is learnable in polynomial time with equivalence queries (and membership queries, resp.) and the membership decision is polynomial time decidable, then it is also PAC-learnable (with membership queries, resp.) [4]. Sakakibara [34] studied the query learnability of the subclass of HEFSs called *extended sim-*

ple EFS (ESEFS, for short). He showed that the class k-*bounded* ESEFS is learnable in polynomial time using the equivalence and *predicate membership* queries, an augmented version of membership queries. The class k-bounded ESEFS is a proper subclass of $\mathrm{HEFS}^{-}(*, k, k, 1)$, where $\mathrm{HEFS}^{-}(m, k, t, r)$ denotes the $\mathrm{HEFS}(m, k, t, r)$ of which the facts are always ground.

In the present paper, we investigate the learnability of the HEFSs w.r.t. the query learning model. Two classes are shown to be learnable in polynomial time using the queries mentioned below with presenting learning algorithms. Moreover, other classes are shown to be hard to learn in the sense of prediction preserving reductions [5,33].

First we extend the Sakakibara's result [34] to whole class of $\mathrm{HEFS}(*, k, t, r)$. The learning algorithm uses a top-down search strategy based on the controlled generation of candidate clauses and the contradiction backtracing algorithm of Shapiro [35]. This algorithm can be regarded as a polynomial time counterpart of the MIEFS of Arikawa, Shinohara, and Yamamoto [9]. We show that this algorithm learns all hypotheses H_* of $\mathrm{HEFS}(*, k, t, r)$ in polynomial time using $O(p^t mn^{2k+2rt}k^k)$ equivalence queries and $O(p^{t+1}mn^{2k+2r(t+1)}k^k)$ predicate membership queries for every $k, t, r \geq 0$, where p is the number of predicate symbols, m is the cardinality of H_*, and n is the size of the longest counterexample seen so far. Unfortunately, the running time is exponential in the maximum length t of the bodies.

To overcome this difficulty, we consider a subclass of HEFS called terminating HEFS (THEFS, for short). Arikawa *et al.* [9] and Yamamoto [43] showed that the standard SLD-resolution procedure can be used as the decision procedure for EFS languages. However, this procedure may not terminate in case of goals. Thus, we consider the *dependency relation* of an EFS H that is a smallest transitive relation over atoms $>_H$ such that $A >_H B$ if A and B appear, respectively, in the head and the body of an instance of a clause in H. An HEFS H is called *terminating* if there exists a well-founded relation $>$, i.e., there exists no infinite decreasing chain, on atoms that bounds $>_H$. It is obvious that, for a terminating HEFS H, the SLD-resolution procedure for $H \models C$ always terminates for every clause C. Hence, we define the hierarchy $\mathrm{THEFS}(m, k, t, r)$ of terminating HEFSs similary to $\mathrm{HEFS}(m, k, t, r)$.

We also allow a learner to use two types of additional queries for the target EFS H_*. The first type of queries is the *entailment membership query* to ask if a given clause is entailed from a target hypothesis. The model with entailment equivalence query and entailment membership queries is called the *learning from entailment* [15,32], which is particularly suitable for learning the first-order logic and logic programs [10,11,16,20,32]. The second type of queries is the *dependency query* to determine if a pair of atoms are in the dependency relation of a target program.

We design a learning algorithm for THEFS$(*, k, *, r)$ with equivalence, entailment membership, and dependency queries. This algorithm adopts the bottom-up search strategy by combining three generalization techniques, i.e., *saturation*, *rewind* and *maximal common subsumer* [10,11,15,16,20,32]. We show that this algorithm exactly learns the class THEFS$(*, k, *, r)$ in polynomial time using $O(pmn^{2r+1})$ equivalence queries, $O(p^2m^2n^{4k+4r+1}k^k)$ entailment membership queries, and $O(p^2m^2n^{4k+4r+1}k^k)$ dependency queries, where m is the number of clauses and n is the length of the longest counterexample seen so far. The number $O(pmn^{2r+1})$ of equivalence queries for this algorithm is significantly smaller than the number $O(p^tmn^{2k+2rt}k^k)$ for the previous top-down algorithm for HEFS$(*, k, t, r)$. Also we show that, by analyzing the VC-dimension, lower bound of the queries to learn THEFS$(*, k, *, r)$ is $\Omega(mn^{r/2})$ for some ordering $>$, which implies that the number of equivalence queries of this algorithm is nearly optimal.

Furthermore, we present the series of representation-independent hardness results for predicting HEFSs by adopting the prediction-preserving reduction without or with membership queries [5,33]. For classes with polynomial time evaluation problems, it is known that the prediction hardness derives both the hardness for PAC-learning and exact learning. We denote by $\mathcal{R}P$, $\cup_m\mathcal{R}P$ and $\cup\mathcal{R}P$ the class of regular pattern languages, at most m unions of regular pattern languages, and all finite unions of regular pattern languages, respectively [12,17,25,26,36,37,39]. Shinohara and Arimura [39] showed that $\mathcal{R}P$ and $\cup_m\mathcal{R}P$ are inferable from positive data while ∪RP is not. Along this line of studies, we show the hardness of learning of these classes. The class $\mathcal{R}P$ is not polynomial-time predictable if neither are DNF formulas and the class $\cup\mathcal{R}P$ is not polynomial-time predictable with membership queries if neither are DNF formulas. Note that the class $\cup_m\mathcal{R}P$ is polynomial-time predictable with membership queries [17] but it is open whether it is learnable with the equivalence and membership queries.

Our results on the hardness for pattern languages improves the previous non-learnability results for $\mathcal{R}P$ and $\cup\mathcal{R}P$ [26] in the representation-dependent manner. Furthermore, the third result extends the learnability of $\mathcal{R}P$ with a single positive example and membership queries [24]. The $\mathcal{R}P$, $\cup_m\mathcal{R}P$ and $\cup\mathcal{R}P$ are corresponding to the HEFS$(1, *, 0, 1)$, HEFS$(m, *, 0, 1)$ and HEFS$(*, *, 0, 1)$, respectively. Hence, we can conclude that the bound on k is necessary to efficiently learn HEFS$(*, k, t, r)$ with equivalence and membership queries. Other hardness results indicate that the HEFS$^-(*, k, t, r)$ is not polynomial-time predictable with membership queries under the cryptographic assumptions, even if $k = t = r = 1$.

Finally, concerning the learnability of k-bounded ESEFSs which is a subclass of HEFS$^-(*, k, k, 1)$, with the equivalence and predicate membership queries [34], we show that the bound k is essential for the efficient learnability,

Fig. 1. The summary of the learnability of a hierarchy HEFS(m, k, t, r) of HEFSs presented in this paper. In the all tables, the first row indicates the types of queries used. The types of queries assumed in this paper are the equivalence (EQ), membership (MQ), predicate membership (PMQ), entailment membership (EntMQ), and dependency (DQ) queries. Each label "poly" means that the class is polynomial-time exact learnable with EQs and the indicated queries. The label "hard" means that learning the class with the queries is as hard as learning the class of DNF formulas, while the label "hard*" means the class is not polynomial-time learnable under the cryptographic assumptions. The "pred" means that the class is polynomial-time predictable with the indicated queries. The "PAC" and "not PAC" mean the class is and is not polynomial-time PAC-learnable, respectively. Finally, each arrow in the tables means that the result of the cell containing the arrow is directly derived from the neighbor pointed by the arrow.

(a) Learnability of HEFSs

Class	EQ	EQ+MQ	EQ+PMQ
HEFS(m, k, t, r)	PAC [25,26]	←	←
k-bounded ESEFSs	→	hard* (Th49)	poly [34]
HEFS$(*, k, t, r)$	→	hard* (Th49)	poly (Th31)
HEFS$^{-}(*, *, *, r)$	→	→	hard (Th50)

(b) Learnability of terminating HEFSs

Class	EQ+MQ	EQ+EntMQ	EQ+EntMQ+DQ
THEFS$(*, k, *, r)$	hard* (Th49)	open	poly (Th42)

(c) Learnability of regular pattern languages and their unions

Class	EQ	EQ+MQ
$\mathcal{R}P$	not PAC [25,26] / hard (Th46)	poly [24]
$\cup_m \mathcal{R}P$	↑ / ↑	pred [17]
$\cup \mathcal{R}P$	↑ / ↑	hard (Th47)

i.e., the HEFS$^{-}(*, *, *, r)$ is not polynomial-time predictable with the membership or predicate membership queries if neither are the DNF formulas, even if $r = 1$. All results in this paper are summarized in Fig. 1.

2 Preliminaries

In this section, we give the definitions and theorems on elementary formal systems, learning models, and prediction-preserving reductions necessary for the later discussion.

2.1 Elementary formal systems and their languages

For a set S, $\#S$ denotes the cardinality of S. Let Σ be a finite alphabet of *constant symbols*, X be a countable set of *variables*, and for every $r \geq 0$, Π_r be a finite alphabet of r-ary *predicate symbols*. Moreover, let $\Pi = \cup_{i \geq 0} \Pi_i$. We assume that Σ, X and Π are mutually disjoint. We call the pair $\mathcal{S} = (\Sigma, \Pi)$ a *signature*.

For each predicate symbol $p \in \Pi_r$, r is called an *arity* of p. We denote by $\text{arity}(\Pi)$ the maximum arity of the predicate symbols in Π. By Σ^*, Σ^+ and $\Sigma^{[n]}$, we denote the sets of all finite strings, all nonempty finite strings, and all strings of length n or less respectively, over Σ.

A *pattern* over $\mathcal{S}$ is an element of $(\Sigma \cup X)^+$. A pattern over $\mathcal{S}$ is called *regular* if each variable appears at most once in it. An *atom* over $\mathcal{S}$ is an expression of the form $p(\pi_1, \ldots, \pi_r)$, where $r \geq 0$, $p \in \Pi_r$ and each π_i is a pattern over $\mathcal{S}$ $(1 \leq i \leq n)$. A *definite clause* (*clause*, for short) over $\mathcal{S}$ is an expression of the form:

$$C = A \leftarrow A_1, \ldots, A_m,$$

where $m \geq 0$ and $A, A_1, \ldots, A_m$ are atoms over $\mathcal{S}$. The atom A and the set $\{A_1, \ldots, A_m\}$ of atoms are called the *head* and the *body* of C and denoted by $hd(C)$ and $bd(C)$, respectively. In case that $m = 0$ (resp., $m > 0$), a clause is called a *fact* (resp., a *rule*). A clause or an atom over $\mathcal{S}$ is *ground* if it contains no variable.

Definition 1 Let $\mathcal{S} = (\Sigma, \Pi)$ be a signature. An *elementary formal system* (*EFS*, for short) over $\mathcal{S}$ is a finite set of clauses over $\mathcal{S}$.

For a signature $\mathcal{S} = (\Sigma, \Pi)$, $Atom_{\mathcal{S}}$ and $Clause_{\mathcal{S}}$ denote the sets of all atoms and all clauses over $\mathcal{S}$, respectively. In particular, the set of all ground atoms over $\mathcal{S}$ is called the *Herbrand base* over $\mathcal{S}$ and denoted by $Base_{\mathcal{S}}$.

A *substitution* is a homomorphism $\theta : (\Sigma \cup X)^+ \rightarrow (\Sigma \cup X)^+$ such that $\theta(a) = a$ for each symbol $a \in \Sigma$. For a substitution θ and a pattern π, the $\pi\theta$ denotes the image of π by θ. For an atom $A = p(\pi_1, \ldots, \pi_n)$ and a clause $C = A \leftarrow$

$A_1, \ldots, A_m$, we define $A\theta = p(\pi_1\theta, \ldots, \pi_n\theta)$ and $C\theta = A\theta \leftarrow A_1\theta, \ldots, A_m\theta$. Then, we say that $A\theta$ and $C\theta$ are *instances* of A and C, respectively. In particular, if $A\theta$ or $C\theta$ becomes ground, then θ is called a *ground substitution*.

We end this subsection by introducing the notion of *subsumption*, denoted by $\sqsupseteq$ which plays an important role in Section 3. For atoms A and B over $\mathcal{S}$, we define A *subsumes* B, denoted by $A \sqsupseteq B$, if there exists a substitution θ such that $A\theta = B$, that is, B is an instance of A.

For clauses C and D over $\mathcal{S}$, we define C *subsumes* D, denoted by $C \sqsupseteq D$, if there exists a substitution θ such that $hd(C\theta) = hd(D)$ and $bd(C\theta) \subseteq bd(D)$. We define C *properly subsumes* D, denoted by $C \sqsupset D$, if $C \sqsupseteq D$ but $D \not\sqsupseteq C$.

For EFSs H and G over $\mathcal{S}$, we define H *subsumes* G, denoted by $H \sqsupseteq G$, if for every $D \in G$, there exists a clause $C \in H$ such that $C \sqsupseteq D$. Then we say that H is a *generalization* of G or G is a *refinement* of H. Furthermore, a refinement G of H is *conservative* if, for every $D \in G$, there exists at most one clause $C \in H$ such that $C \sqsupseteq D$. We define $H \sqsupset G$ if $H \sqsupseteq G$ but $G \not\sqsupseteq H$.

2.2 Three semantics for EFSs

In this subsection, we first introduce a model theory for EFSs as follows for uniformly dealing with three semantics. Let us identify a given signature $\mathcal{S} = (\Sigma, \Pi)$ with the first-order signature $(\Sigma, \{\cdot\}, \Pi)$, where "$\cdot$" is a string concatenation operator satisfying the associativity $\forall x \forall y \forall z[x \cdot (y \cdot z) = (x \cdot y) \cdot z]$.

An *interpretation* $\mathcal{I}$ over $\mathcal{S}$ is a triple (U, I, α), where U is a set, I is a mapping that maps $p \in \Pi_r$ $(r \geq 0)$, "$\cdot$" and $a \in \Sigma$ to an r-ary relation over U, a binary associative function over U and an element of U, respectively, and α is a variable-assignment to U. Then, the *satisfaction* relation $\models$ is defined in a standard manner (*cf.*, [14,31]). A *model* of an atom A or a clause C over $\mathcal{S}$ is an interpretation $\mathcal{I}$ over $\mathcal{S}$ such that $\mathcal{I} \models A$ and $\mathcal{I} \models C$, respectively. We assume that any variable in a clause is universally quantified. A *model* of an EFS H over $\mathcal{S}$ is a model of every clause in H over $\mathcal{S}$.

For an EFS H and a clause C over $\mathcal{S}$, we say that H *entails* C, denoted by $H \models C$, if every model of H is a model of C. For EFSs H and G over $\mathcal{S}$, we say that H *entails* G, denoted by $H \models G$, if every model of H is a model of G.

Originally, the semantics of EFSs is defined by the provability relation $\vdash$ defined in [9]. For an EFS H and a clause C over $\mathcal{S}$, respectively, the relation $H \vdash C$ which means that C is *provable* from H is defined inductively as follows:

(1) If $C \in H$, then $H \vdash C$.
(2) If $H \vdash C$, then $H \vdash C\theta$ for a substitution θ.
(3) If $H \vdash A \leftarrow A_1, \ldots, A_m, A_{m+1}$ and $H \vdash A_{m+1}$, then $H \vdash A \leftarrow A_1, \ldots, A_m$.

The following lemma gives the relationship between $\vdash$ and $\models$.

Lemma 2 (Arikawa *et al.* [9]) For every atom A and EFS H, $H \models A$ iff $H \vdash A \leftarrow$.

The language semantics is a standard semantics of EFSs (*cf.* [8,9,25,26]). Let H be an EFS over $\mathcal{S} = (\Sigma, \Pi)$ and $p_0 \in \Pi$ be a distinguished predicate symbol. Then, the *language defined by H and p_0 over $\mathcal{S}$* is the set

$$L_{\mathcal{S}}(H, p_0) = \{\, w \in \Sigma^+ \mid H \models p_0(w) \,\}.$$

A language $L \subseteq \Sigma^+$ is *definable by an EFS over $\mathcal{S}$* or it is an *EFS language over $\mathcal{S}$* if there exists an EFS H over $\mathcal{S}$ and $p_0 \in \Pi$ such that $L = L_{\mathcal{S}}(H, p_0)$.

The least Herbrand model semantics [9,43] is based on all of the ground atoms provable from a given EFS. The *least Herbrand model* of an EFS H over $\mathcal{S}$ is the set $M_{\mathcal{S}}(H) = \{\, A \in Base_{\mathcal{S}} \mid H \models A \,\}$ [9,43].

The entailment semantics is based on all clauses entailed by a given EFS. The *entailment set* of an EFS H over $\mathcal{S}$, denoted by $Ent_{\mathcal{S}}(H)$, is the set of all clauses over $\mathcal{S}$ entailed by H, i.e., $Ent_{\mathcal{S}}(H) = \{\, C \in Clauses_{\mathcal{S}} \mid H \models C \,\}$.

Formally, a *semantics* for a class $\mathcal{H}$ of EFSs is a pair $(U, \hat{L}(\cdot))$, where U is a set of objects, called the *domain*, and a mapping $\hat{L} : \mathcal{H} \to 2^U$, called the *language mapping*.

Definition 3 Let $\mathcal{S}$ be a signature (Σ, Π) and $p_0 \in \Pi_1$ is the distinguished predicate.

- The *language semantics* on $\mathcal{S}$ is a pair $(Atom_{\mathcal{S}}, L_{\mathcal{S}}(\cdot, p_0))$.
- The *least Herbrand model semantics* on $\mathcal{S}$ is a pair $(Base_{\mathcal{S}}, M_{\mathcal{S}}(\cdot))$.
- The *entailment semantics* on $\mathcal{S}$ is a pair $(Clause_{\mathcal{S}}, Ent_{\mathcal{S}}(\cdot))$.

We introduce a *proof-DAG* by extending the parse-DAG for k-bounded CFGs by Angluin [3] and the ground proof-DAG for EFS by Sakakibara [34].

Definition 4 A *proof-DAG* for a clause C by an EFS H is a finite directed acyclic graph T with the following properties. Nodes in T are atoms possibly containing variables. The node $A = hd(C)$ is the unique node with in-degree zero, called the *root*. For each node B in T, let $Succ(B)$ be the set of nodes B' with edges from B to B'. Then for every node B in T, either $B \in bd(C)$ or $(B \leftarrow Succ(B))$ is an instance of a clause in H.

A proof-DAG T of C by H is *minimal* if no proper subgraph of T is also a proof-DAG for C by H. A minimal proof-DAG T for a clause C by H is said to be *trivial* if all nodes but the root in T are contained in $bd(C)$, and *non-trivial* otherwise. Note that if T is trivial then all nodes appear in C. We will assume that a proof-DAG is always minimal.

The *Skolem substitution for C w.r.t. H* is a substitution σ that replaces the variables x in C with mutually distinct fresh constants c_x not appearing in H and C.

Lemma 5 Let H be an EFS and C a clause. For the Skolem substitution σ for C w.r.t. H, $H \models \forall(C)$ iff $H \models C\sigma$.

Lemma 6 Let $\mathcal{S}$ be a signature, H an EFS consisting of ground clauses, and $A \in Base_{\mathcal{S}}$ a ground atom. Then, $H \models A$ iff there exists a minimal proof-DAG T for $A \leftarrow$ by H.

PROOF. The if direction of the lemma is easily proved by induction on the size $n \geq 1$ of the proof-DAG for A by H. Next, we will show the only-if direction. Suppose that $H \models A$. Let $M = M_{\mathcal{S}}(H)$. First, since M is the smallest among the Herbrand models of H, we can show that M is the *supported model*, that is, if $M \models A$ then there is some $C \in H$ such that $A = hd(C)$ and $M \models bd(C)$. Then, we show the lemma by induction on the cardinality $n = \#H$. If $n = 1$ then H consists of the fact $A \leftarrow$, and thus, the lemma immediately follows. Suppose that $\#H = n+1$ and the lemma holds for any EFS of cardinality no more than n. By the claim shown above, there is some clause $C = (A \leftarrow B_1, \ldots, B_m) \in H$ such that $A = hd(C)$ and $M \models B_1 \wedge \ldots \wedge B_m$. Let $H' = H - \{C\}$ and $M' = M_{\mathcal{S}}(H')$. We will show that $M' \models B_1 \wedge \ldots \wedge B_m$. Suppose to the contrary that there is some interpretation I such that $I \models H - \{C\}$ but $I \not\models B_1 \wedge \ldots \wedge B_m$. Since $B_1 \wedge \ldots \wedge B_m$ is the body of C, we see that $I \models C$ regardless the truth value of A. Therefore, I is a model of both $H - \{C\}$ and C, and thus that $I \models M$ but $I \not\models B_1 \wedge \ldots \wedge B_m$. However, this contradicts the assumption. Hence, $M' \models B_1 \wedge \ldots \wedge B_m$. Since $\#H' \leq n$, by induction hypothesis, we have that for every $1 \leq i \leq m$, there exists a proof-DAG T_i for B_i by H'. Hence, we have a proof-DAG for A by H by merging $T_1, \ldots, T_m$ and by adding the root node A and the edges $\{(A, B_i) \mid 1 \leq i \leq m\}$. It is not hard to see that the resulting graph T is acyclic. □

The following lemma, an EFS counterpart of the subsumption theorem in clausal logic [30], characterizes the entailment relation $\models$ for EFS in terms of a proof-DAG. Since the theorem is essential in our learnability results in Chapter 3, we will give a complete proof of our version of the lemm with proof-DAGs though [30] have given an indirect proof using the completeness of SLD-resolutions for definite logic programs.

Lemma 7 (The subsumption theorem) Let H be an EFS and C a clause. Then, $H \models C$ if and only if one of the following statements holds:

(i) C is a tautology.
(ii) C is subsumed by some clause in H.
(iii) There exists a non-trivial minimal proof-DAG for C by H.

PROOF. The only-if direction is straightforward. We will show the converse direction. Let $C\sigma$ be the ground clause obtained from C by applying the Skolem substitution σ for C w.r.t. H. Suppose that $H \models C$. Then, it follows from Lemma 5 and the deduction theorem of first-order logic that $H \models C\sigma$ implies $H' \models A'$, where we put $A' = hd(C\sigma)$ and $H' = H \cup bd(C\sigma)$. From Lemma 6, we have a proof-DAG T' for A' by H'. By applying the inverse mapping σ^{-1} to T', we obtain a proof-DAG T for C by H Since σ is a one-to-one mapping that introduces only fresh constants C not appearing in $\{C\} \cup H$.

Now, we will show that if neither (i) nor (ii) holds then (iii) there exists a non-trivial minimal proof-DAG for C by H. Assume that C is neither a tautology nor subsumed by any clause in H. We further assume without loss of generality that T is minimal. Suppose to contradict that T is trivial. Then, we can show that the height of T is at most two, that is, T consists of the root $A = hd(C)$ and (possibly empty) leaves $Succ(A) = \{B_1, \ldots, B_n\}$ $(n \geq 0)$. If the set $Succ(A)$ is empty then A is both the root and the unique leaf of T. Then, there are two cases below. If A is an instance of some fact D in H then we have that $A \leftarrow$ is subsumed by D. Otherwise, $A = hd(C)$ appears in $bd(C)$, and this means that the clause C is a tautology. In both cases, the contradiction is derived. We assume that $Succ(A)$ is not empty. By the definition of a proof-DAG, the clause $A \leftarrow Succ(A)$ is an instance of some clause D in H. Suppose that $D\theta = (A \leftarrow Succ(A))$ for some θ. On the other hand, since T is trivial, $Succ(A)$ must be a subset of $bd(C)$. Therefore, it follows that $hd(D\theta) = hd(D\theta)$ and $bd(D\theta) \subseteq bd(C)$, and thus we know that C is subsumed by D. This contradicts the assumption. Hence, we conclude that T is non-trivial , and this completes the proof. □

In the remainder of this paper, we will omit the subscript $\mathcal{S}$ if it is not necessary to explicitly disignate it. In Section 3, a signature is explicitly given to a learner before the learning session starts. In Section 4, a signature is implicitly assumed to contain all predicate and constant symbols occurring in EFSs.

In this subsection, we introduce the several subclasses of EFSs, which are developed by many researchers [7–9,18,25,26,34,38,43].

First, we prepare the notations necessary to define the subclasses. The *size* of a pattern π, denoted by $|\pi|$, is the length of the string π as a string over $\Sigma \cup X$. The *variable-occurrence* of π, denoted by $o(\pi)$, is the total number of the occurrences of variables from X appearing in π. We denote by $var(\pi)$ the set of variables in X appearing in π. For example, if $\Sigma = \{a, b\}$, $X = \{x, y, \ldots\}$ and $\pi = abxbxyab$, then $|\pi| = 8$ and $o(\pi) = 3$. For an expression E, we define the representation length $||E||$ and the occurrences of variables x in E as follows. For an atom $A = p(\pi_1, \ldots, \pi_n)$, we define $||A|| = |\pi_1| + \cdots + |\pi_n|$ and $o(A) = o(\pi_1) + \cdots + o(\pi_n)$. For a clause $C = A_0 \leftarrow A_1, \ldots, A_m$, we define $||C|| = ||A_0|| + \cdots + ||A_m||$ and $o(C) = o(A_0) + \cdots + o(A_m)$. For an EFS H, the *size* of H, written $||H||$, is $\sum_{C \in H} ||C||$.

Definition 8 We introduce the following restrictions of clauses.

(1) A clause $A \leftarrow A_1, \ldots, A_m$ is called *variable-bounded* [9] if every variable appearing in the body $A_1, \ldots, A_m$ also appears in the head A.
(2) A clause $A \leftarrow A_1, \ldots, A_m$ is called *length-bounded* [9] if $|A\theta| \geq |A_1\theta| + \ldots + |A_m\theta|$ for each substitution θ.
(3) A clause $p(\pi) \leftarrow q_1(x_1), \ldots, q_m(x_m)$ is called *extended simple* [34] if $p, q_1, \ldots, q_m$ are unary predicate symbols and $x_1, \ldots, x_m$ are all variables appearing in π.
(4) A clause is called *simple* [9] if it is of the form $p(\pi) \leftarrow q_1(x_1), \ldots, q_m(x_m)$, where $p, q_1, \ldots, q_m$ are unary predicate symbols and $x_1, \ldots, x_m$ are mutually distinct variables appearing in π.
(5) A simple clause is called *regular* [7] if the pattern in its head is regular.
(6) A regular clause is called *left-linear* (*resp.*, *right-linear*) [7] if the pattern in its head is of the form wx (*resp.*, xw) for some string $w \in \Sigma^*$.
(7) A clause is *hereditary* [26] if it is of the form
$p(\pi_1, \ldots, \pi_n) \leftarrow q_1(\tau_1, \ldots, \tau_{t_1}), q_2(\tau_{t_1+1}, \ldots, \tau_{t_2}), \ldots, q_m(\tau_{t_{m-1}+1}, \ldots, \tau_{t_m})$,
and each pattern τ_j $(1 \leq j \leq t_m)$ is a substring of some π_i $(1 \leq i \leq n)$.

The extended simple clause was introduced in the context of simple formal systems (SFSs) [34], so an extended simple clause is an extension of a simple clause in SFSs [7]. In contrast, the above extended simple clause is not an extension of a simple clause in EFSs. In particular, there exists no extended simple clause that is a non-ground fact and that has variables only occurring in the head.

Definition 9 An EFS H is called *variable-bounded* (*resp.*, *length-bounded*, *extended simple*, *simple*, *regular*, *left-linear*, *right-linear*, *hereditary*) if each clause in H is variable-bounded (*resp.*, length-bounded, extended simple, simple, regular, left-linear, right-linear, hereditary).

For example, let $\Pi = \{p_0, q\}$ and $\Sigma = \{a, b, c\}$. Then, the following simple EFS H_0 and hereditary EFS H_1 define the languages $L(H_0, p_0) = \{ w \in \{a, b\}^+ \mid w \text{ is a string of the balanced parentheses} \}$ and $L(H_1, p_0) = \{ a^n b^n c^n \mid n \geq 1 \}$, respectively.

$$H_0 = \left\{ \begin{array}{l} p_0(xy) \leftarrow p_0(x), p_0(y) \\ p_0(axb) \leftarrow p_0(x) \\ p_0(ab) \leftarrow \end{array} \right\}, \ H_1 = \left\{ \begin{array}{l} p_0(xyz) \leftarrow q(x, y, z) \\ q(ax, by, cz) \leftarrow q(x, y, z) \\ q(a, b, c) \leftarrow \end{array} \right\}.$$

We abbreviate an extended simple EFS and a hereditary EFS as an ESEFS and an HEFS, respectively. The following hierarchy HEFS(m, k, t, r) of HEFSs introduced by [26] gives a useful framework for polynomial-time learnability.

Definition 10 (Miyano *et al.* [25,26]) HEFS(m, k, t, r) is the class of all HEFSs consisting of at most m clauses each of which satisfies the following conditions (a)–(c). HEFS$^-(m, k, t, r)$ is the subclass of HEFS(m, k, t, r) consisting of at most m clauses each of which satisfies the following conditions (a)–(d).

(a) The variable-occurrence in the head is at most k.
(b) The number of atoms in the body is at most t.
(c) The arity of each predicate symbol is at most r.
(d) All facts are ground.

In this hierarchy, the symbol '$*$' indicates that there is no bound on this parameter.

The HEFSs H_0 and H_1 in the above example belong to HEFS$^-(3, 2, 2, 1)$ and HEFS$^-(3, 3, 1, 3)$, respectively. We can give the correspondence of the EFS languages to Chomsky's hierarchy and complexity classes.

Theorem 11 The following relations hold for the EFS languages above.

(1) **(Arikawa [7], Arikawa *et al.* [9])** A language is recursively enumerable, (resp., context-sensitive, context-free, regular) iff it is definable by a variable-bounded (resp., length-bounded, regular, left/right-linear) EFS.
(2) **(Ikeda, Arimura [18])** A language is accepted by a polynomial time deterministic Turing machine iff it is definable by a hereditary EFS.
(3) **(Arikawa *et al.* [9])** Any regular pattern language, (resp., union of regular pattern languages, regular language, context-free language) is defined by an EFS in HEFS$(1, *, 0, 1)$, (resp. HEFS$(*, *, 0, 1)$, HEFS$(*, 1, 1, 1)$,

HEFS$(*, 2, 2, 1)$).

Finally, we formulate the termination for HEFSs, which are motivated by the acyclicity of EFSs [6,10,13].

Definition 12 *Let $\mathcal{S}$ be a signature and H be an EFS over $\mathcal{S}$. The* dependency graph *of H is a possibly infinite directed graph $G_H = (Atom_{\mathcal{S}}, E)$ such that there exists an edge from A to B, i.e., $(A, B) \in E$, iff there exist a ground instance C of some clause in H such that $A = hd(C)$ and $B \in bd(C)$.*

Definition 13 Let $\mathcal{S}$ be a signature and H be an EFS over $\mathcal{S}$. The *dependency relation* of H is a binary relation $>_H$ on $Atom_{\mathcal{S}}$ such that $A >_H B$ iff there exists a path of non-zero length from A to B in the dependency graph G_H of H.

A binary relation R on S is *transitive* if aRb and bRc implies aRc for every $a, b, c \in S$. Also R is *well-founded* if there exists no infinite decreasing chain from a such as $aRa_1, a_1Ra_2, a_2Ra_3, \cdots$, for every $a \in S$.

Definition 14 Let $\mathcal{S}$ be a signature, H be an EFS over $\mathcal{S}$ and $>$ be a transitive binary relation on $Atom_{\mathcal{S}}$. The dependency relation $>_H$ of H is *bounded by* $>$ if $A >_H B$ implies $A > B$ for every atoms $A, B \in Atom_{\mathcal{S}}$.

Definition 15 Let $\mathcal{S}$ be a signature and H be an EFS over $\mathcal{S}$ Then, H is *terminating* if there exists a well-founded transitive binary relation $>$ on $Atom_{\mathcal{S}}$ that bounds the dependency relation $>_H$ of H.

Let $\mathcal{S}$ be a signature, $\mathcal{H}$ be a class of EFSs over $\mathcal{S}$, and $>$ be a transitive binary relation on $Atom_{\mathcal{S}}$. We say that $\mathcal{H}$ is *uniformly bounded by* $>$ if the dependency relation $>_H$ is bounded by $>$ for every $H \in \mathcal{H}$. We denote by $\mathcal{H}(>)$ the maximal subclass of $\mathcal{H}$ whose dependency relation is uniformly bounded by $>$, i.e., $\mathcal{H}(>) = \{ H \in \mathcal{H} \mid >_H \text{ is bounded by } > \}$.

As similar as HEFS(m, k, t, r), we can introduce a class THEFS(m, k, t, r) of terminating HEFSs with the same parameters m, k, t and r. In particular, we denote (THEFS(m, k, t, r))$(>)$ by THEFS$(>, m, k, t, r)$.

2.4 Learning models

In this subsection, we introduce the learning models. Here, a class $\mathcal{H}$ of grammars, called a *hypothesis space*, is always assumed. If a hypothesis space $\mathcal{H}$ is a class of EFSs, then a signature is assumed to be in common.

Let $(U, \hat{L}(\cdot))$ be the semantics for $\mathcal{H}$. Each element of U is called an *example*.

The language $\hat{L}(H)$ is also called the *concept defined by* H. We say that two hypotheses H and H_* are *equivalent* under the semantics $(U, \hat{L}(\cdot))$ if $\hat{L}(H) = \hat{L}(H_*)$.

Let $H_* \in \mathcal{H}$ be a *target hypothesis*. An example w is called *positive* for H_* if $w \in \hat{L}(H_*)$ and *negative* otherwise. Many researchers have been developed several different learning models to capture the efficient learnability from the viewpoints of the criterion of identification and the protocol of receiving examples and queries. In this paper, we employ the following two learning models. First, we define the exact learning model, where a learning algorithm makes the following queries to collect the information on H_* [4].

Definition 16 (Angluin [4]) Let $H_* \in \mathcal{H}$ be a target hypothesis.

(1) An *equivalence query* for H_* (EQ, for short) takes $H \in \mathcal{H}$ as input, denoted by EQ(H). The answer is "yes" if $\hat{L}(H) = \hat{L}(H_*)$ and a *counterexample* $w \in (\hat{L}(H_*) - \hat{L}(H)) \cup (\hat{L}(H) - \hat{L}(H_*))$ is returned otherwise. A counterexample w is called *positive* if $w \in \hat{L}(H_*)$ and called *negative* if $w \notin \hat{L}(H_*)$.
(2) A *membership query* for H_* (MQ, for short) takes $w \in \Sigma^+$ as input, denoted by MQ(w). The answer is "yes" if $w \in L(H_*)$ and "no" otherwise.

Definition 17 (Angluin [4]) A *polynomial-time exact learning algorithm* **A** for $\mathcal{H}$ is an algorithm that identifies the target hypothesis $H_* \in \mathcal{H}$ making equivalence and membership queries for H_*, **A** must halt and output a hypothesis $H \in \mathcal{H}$ that is equivalent to H_*, i.e., $\hat{L}(H) = \hat{L}(H_*)$, and, at any stage in the learning algorithm, the running time of **A** must be bounded by a polynomial in the size of H_* and of the longest counterexample returned by equivalence queries so far. $\mathcal{H}$ is called *polynomial-time exact learnable* if there exists a polynomial-time exact learning algorithm for $\mathcal{H}$.

On the other hand, we introduce the prediction model according to Pitt and Warmuth [33] and Angluin and Kharitonov [5].

Definition 18 (Pitt & Warmuth [33], Angluin & Kharitonov [5]) An algorithm **A** is called a *prediction algorithm* for $\mathcal{H}$ that takes s (a bound on the size of $\mathcal{H}$), n (a bound on the length of examples), ε (an accuracy bound), a collection of labeled examples such that each positive (*resp.*, negative) example is labeled by $+$ (*resp.*, $-$), and an unlabeled example w of H_* as input, and outputs either $+$ or $-$ indicating its prediction for w. The **A** is called a *polynomial-time prediction algorithm* if the running time of **A** is bounded by a polynomial in s, n and $1/\varepsilon$. For some polynomial p, for all input parameters s, n and ε and for all probability distributions on examples, if **A** is given at least $p(s, n, 1/\varepsilon)$ randomly generated examples of H_* and randomly generated unlabeled example w, and the probability that **A** incorrectly predicts the label of w for H_* is at most ε, then we say that **A** *successfully predicts* $\mathcal{H}$. Moreover, $\mathcal{H}$ is called *polynomial-time predictable* if there exists a polynomial-time prediction algorithm for $\mathcal{H}$ that successfully predicts $\mathcal{H}$.

The algorithm **A** is called a *prediction with membership queries algorithm* (*pwm-algorithm*, for short) if it is a prediction algorithm which is allowed to make membership queries. The *polynomial-time pwm-algorithm* is similarly defined as above.

Definition 19 (Valiant [42]) A polynomial time PAC learning algorithm $\mathcal{A}$ for $\mathcal{H}$ is an algorithm that takes parameters s, n, ε and a collection of randomly generated labeled examples, chosen according to an unknown probability distribution D, as in the prediction learning model above, and outputs with hight probability a hypothesis $H \in \mathcal{H}$ that approximates the target hypothesis H_* with true error at most ε w.r.t. D. The time and the number of examples that algorithm $\mathcal{A}$ requires are bounded by polynomials in $s, n, 1/\varepsilon$, and $\mathcal{A}$ have to work regardless of the distribution D. We can also define a variant of PAC-learning model in which a learning algorithm is allowed to make membership queries in addition to random examples [5].

There is a close relationship among exact learning with equivalence queries, PAC-learning and prediction models without or with membership queries.

Theorem 20 (Angluin [4], Angluin & Kharitonov [5]) If a hypothesis space $\mathcal{H}$ is polynomial-time exact learnable with equivalence queries, then it is polynomial-time PAC learnable. If $\mathcal{H}$ is polynomial-time PAC learnable, then it is polynomial-time predictable. Furthermore, these statements also hold with membership queries.

In this paper, we also introduce the following extension of membership queries based on the non-standard semantics of EFSs.

Definition 21 Let $H_* \in \mathcal{H}$ be a target hypothesis.

(1) **(Angluin [3], Sakakibara [34])** A *predicate membership query* for H_* (PMQ, for short) takes a ground atom $A = p(w_1, \ldots, w_n)$ for $p \in \Pi$ and $w_i \in \Sigma^+$ $(1 \leq i \leq n)$ as input, denoted by PMQ(A). The answer is "yes" if $H_* \models A$, i.e., $A \in M(H_*)$ and "no" otherwise.
(2) **(Frazier & Pitt [15])** An *entailment membership query* for H_* (EntMQ, for short) takes a (possibly non-ground) clause C as input, denoted by EntMQ(C). The answer is "yes" if $H_* \models C$, i.e., $C \in Ent(H_*)$ and "no" otherwise.

The PMQs and EntMQs coincide with exactly the membership queries under the least Herbrand model semantics $(Base, M(\cdot))$ and the entailment semantics $(Clause_{\mathcal{S}}, Ent(\cdot))$, respectively. We can observe that an MQ is simulated by a PMQ and then a PMQ is by an EntMQ.

Furthermore, we can define the *entailment equivalence query* (EntEQ, for short) as the equivalence query under the semantics $(Clause_{\mathcal{S}}, Ent(\cdot))$, where a counterexample is a clause. The learning model with EntEQ and EntMQ, called *learning from entailment* [15], gives a valuable framework for the efficient learnability of first-order logic or logic programs [10,11,16,20,32]. For all subclasses of HEFSs studied in Chapter 3 and Chapter 4 except Theorem 50, all types of queries, namely, MQ, PMQ, EntMQ, EQ, and EntEQ, introduced above are polynomial-time computable.

Finally, we define the query to ask about the termination information.

Definition 22 A *dependency query* for H_* (DQ, for short) takes a pair (A, B) of atoms as input, denoted by DQ(A, B). The answer is "yes" if $A >_{H_*} B$ holds and "no" otherwise.

2.5 Prediction-preserving reduction

Pitt and Warmuth [33] have introduced the notion of reducibility between prediction problems. *Prediction-preserving reducibility* is essentially a method of showing that one hypothesis space is not harder to predict than another. Furthermore, Angluin and Kharitonov [5] have extended the prediction-preserving reduction to the notion of reducibility between prediction problems *with membership queries*.

Definition 23 (Pitt & Warmuth [33], Angluin & Kharitonov [5]) Let $\mathcal{H}_i$ be a hypothesis space over a domain U_i $(i = 1, 2)$. For every nonnegative integers n and s, we define $U_i^{[n]} = \{w \in U_i \mid |w| \leq n\}$ and $\mathcal{H}_i^{[s]} = \{H \in \mathcal{H}_i \mid ||H|| \leq s\}$. We say that *predicting* $\mathcal{H}_1$ *reduces to predicting* $\mathcal{H}_2$, denoted by $\mathcal{H}_1 \trianglelefteq \mathcal{H}_2$, if there exists a function $f : \mathbf{N} \times \mathbf{N} \times U_1 \to U_2$ (called an *instance mapping*) and a function $g : \mathbf{N} \times \mathbf{N} \times \mathcal{H}_1 \to \mathcal{H}_2$ (called a *concept mapping*) satisfying the following conditions:

(1) for each $w \in U_1^{[n]}$ and $H \in \mathcal{H}_1^{[s]}$, $w \in \hat{L}(H)$ iff $f(n, s, w) \in \hat{L}(g(n, s, H))$.
(2) the representation length of $g(n, s, H)$ is polynomial in the representation length of H; That is, $||g(n, s, H)|| \leq q(||H||)$ for some polynomial q.
(3) $f(n, s, w)$ can be computed in polynomial time.

Furthermore, we say that *predicting* $\mathcal{H}_1$ *reduces to predicting* $\mathcal{H}_2$ *with membership queries* (*pwm-reduces*, for short), denoted by $\mathcal{H}_1 \trianglelefteq_{\mathrm{pwm}} \mathcal{H}_2$, if there exists a function $f : \mathbf{N} \times \mathbf{N} \times U_1 \to U_2$, a function $g : \mathbf{N} \times \mathbf{N} \times \mathcal{H}_1 \to \mathcal{H}_2$, and a function $h : \mathbf{N} \times \mathbf{N} \times U_2 \to U_1 \cup \{\top, \bot\}$ (called a *membership query mapping*) satisfying the above and the following conditions:

4. for each $w' \in U_2$ and $H \in \mathcal{H}_1^{[s]}$, if $h(n,s,w') = \top$ then $w' \in \hat{L}(g(n,s,H))$; if $h(n,s,w') = \bot$ then $w \notin \hat{L}(g(n,s,H))$; if $h(n,s,w') = w \in U_1$, then it holds that $w' \in \hat{L}(g(n,s,H))$ iff $w \in \hat{L}(H)$;
5. $h(n,s,w')$ can be computed in polynomial time.

Theorem 24 (Pitt & Warmuth [33], Angluin & Kharitonov [5]) Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be hypothesis spaces and suppose that $\mathcal{H}_1 \trianglelefteq \mathcal{H}_2$ ($\mathcal{H}_1 \trianglelefteq_{\rm pwm} \mathcal{H}_2$). If $\mathcal{H}_2$ is polynomial-time predictable (with membership queries), then so is $\mathcal{H}_1$.

We deal with the following hypothesis spaces to reduce the prediction problem to several EFS subclasses: $\mathcal{DFA}$ and $\cup\mathcal{DFA}$ denote the class of all languages accepted by the DFAs and the finite union of DFAs, respectively. $\mathcal{DNF}_n$ denotes the class of all DNF formulas over n Boolean variables, Let $\mathcal{DNF} = \cup_{n\geq 1}\mathcal{DNF}_n$.

Theorem 25 The following statements hold.

(1) **(Angluin [2])** $\mathcal{DFA}$ is polynomial-time exactly learnable with equivalence and membership queries.
(2) **(Angluin & Kharitonov [5])** $\cup\mathcal{DFA}$ is not polynomial-time predictable with membership queries under the cryptographic assumptions that inverting the RSA encryption function, recognizing quadratic residues and factoring Blum integers are not solvable in polynomial time.
(3) **(Angluin & Kharitonov [5])** $\mathcal{DNF}$ is neither polynomial-time predictable or not polynomial-time predictable with membership queries, if there exist one-way functions that can not be inverted by polynomial-sized circuits.

3 Learning HEFSs

We study the polynomial-time learnability of subclasses of HEFSs using various types of queries. We first show that the class HEFS$(*, k, t, r)$ of HEFSs is polynomial-time exact learnable with equivalence and predicate membership queries. Next, we show that the class THEFS$(*, k, *, r)$ of terminating HEFSs is polynomial-time exact learnable with equivalence, entailment membership, and dependency queries, where the last type of queries askes about the termination information.

Sakakibara [34] showed that, for every $k \geq 0$, the class of k-bounded ESEFSs, which is a subclass of $\text{HEFS}^-(*, k, k, 1)$, is polynomial-time exact learnable with equivalence and predicate membership queries. In this subsection, we extend this result to the whole class $\text{HEFS}(*, k, t, r)$ for every $k, t, r \geq 0$.

In general, the entailment relation is undecidable for variable-bounded EFSs [9] and deterministic exponential-time complete for HEFSs [18]. The following lemma claims that the entailment relation in $\text{HEFS}(*, k, *, r)$ is polynomial-time decidable.

Lemma 26 For a clause C and an EFS H, suppose $H \cup \{C\} \in \text{HEFS}(*, k, *, r)$. Then, a proof-DAG for $H \models C$ is polynomial-time computable in $|C|$ and $|H|$ if it exists.

PROOF. Let θ be the ground substitution that maps each variable x in C to a new constant c_x. Then, we can see that $H \models C$ if $H \cup bd(C\theta) \models hd(C\theta)$ under the extended alphabet $\Sigma \cup \{ c_x \mid x \in var(\pi) \}$. The result immediately follows from Miyano *et al.* [26]. □

For a signature $\mathcal{S} = (\Sigma, \Pi)$ and an atom $A = p(\pi_1, \ldots, \pi_r)$, we define the subset $Atom_{\mathcal{S}}(A)$ of $Atom_{\mathcal{S}}$ as:

$$Atom_{\mathcal{S}}(A) = \left\{ q(\tau_1, \ldots, \tau_s) \in Atom_{\mathcal{S}} \;\middle|\; \begin{array}{l} \text{every } \tau_i (1 \leq i \leq s) \text{ is a substring} \\ \text{of some } \pi_j (1 \leq j \leq r) \end{array} \right\}.$$

Then, the following series of lemmas are necessary to prove the learnability of $\text{HEFS}(*, k, t, r)$.

Lemma 27 Let $\mathcal{S}$ be a signature, H an HEFS over $\mathcal{S}$ and C a clause over $\mathcal{S}$. Then, for every atom A in a proof-DAG for $H \models C$, it holds that $A \in Atom_{\mathcal{S}}(hd(C))$.

Lemma 28 Let $\mathcal{S}$ be a signature (Σ, Π) and A an atom over $\mathcal{S}$. Then, it holds that $\#Atom_{\mathcal{S}}(A) \leq q_1(p, n) = pn^{2r}$, where $p = \#\Pi$, $n = ||A||$ and $r = \text{arity}(\Pi)$.

Lemma 29 For every integer $k \geq 0$ and atom A, there are at most $||A||^{2k}k^k$ atoms B with variable-occurrence no more than k that subsumes A, i.e., $o(B) \leq k$ and $B \sqsupseteq A$.

Procedure LEARN_HEFS_BY_CBA
/* A learning algorithm for HEFS$(*, k, t, r)$ with EQs and PMQs */
/* $\mathcal{S}$: a fixed signature */
1 $H := \emptyset$;
2 **while** EQ$(H) =$"no" **do begin** /* $L(H, p_0) \neq L(H_*, p_0)$ */
3 $\quad E :=$ a counterexample returned by the EQ; /* E is an atom. */
4 $\quad$ **if** $H \models E$ **then** /* E is negative, i.e., $H \models E$ and $H_* \not\models E$*/
5 $\quad\quad T :=$ a proof-DAG for $H \models E$;
6 $\quad\quad A := root(T)$;
7 $\quad\quad$ **while** PMQ$(B) =$"no" for some $B \in Succ(A)$ of A **do**
8 $\quad\quad\quad A := B$;
9 $\quad\quad \{B_1, \ldots, B_{t'}\} := Succ(A)$ $(t' \geq 0)$;
10 $\quad\quad C :=$ a clause in H that subsumes $A \leftarrow B_1, \ldots, B_{t'}$;
11 $\quad\quad H := H - \{C\}$;
12 $\quad$ **else** /* E is positive, i.e., $H \not\models E$ and $H_* \models E$ */
13 $\quad\quad H := H \cup Cand(E, k, t, r)$;
14 **end** /* while */
15 **return** H;

Fig. 2. A polynomial-time learning algorithm for HEFS$(*, k, t, r)$ with EQs and PMQs, based on the contradiction backtracing algorithm [35,34] (Lines 5 to 10).

Let $\mathcal{S}$ be a signature. For integers $k, t, r \geq 0$ and an atom A over $\mathcal{S}$, by $Cand(E, k, t, r)$, we denote the set of all hereditary clauses in HEFS$(*, k, t, r)$ over $\mathcal{S}$ of the form $B \leftarrow B_1, \ldots, B_{t'}$ such that $B \sqsupseteq hd(E)$, $o(B) \leq k$ and $B_i \in Atom_{\mathcal{S}}(B)$, where $0 \leq i \leq t'$ and $0 \leq t' \leq t$. Then, we can see that if $H_* \models E$ then any clause used to construct a proof-DAG for E by H_* is a member of $Cand(E, k, t, r)$. The following lemma immediately follows from Lemma 28 and Lemma 29.

Lemma 30 $\#Cand(E, k, t, r)$ is bounded by $q_2(p, n) = O(p^t n^{2k+2rt} k^k)$, where $p = \#\Pi$ and $n = ||E||$. (k^k reflects that the same variable may occur more than once.)

Theorem 31 Let $\mathcal{S} = (\Sigma, \Pi)$ be a signature. The class HEFS$(*, k, t, r)$ is polynomial-time exact learnable with $O(p^t m\, n^{2k+2rt} k^k)$ equivalence queries and $O(p^{t+1} m\, n^{2k+2r(t+1)} k^k)$ predicate membership queries, where $p = \#\Pi$, m is the cardinality of a target HEFS, and n is the size of the longest counterexample received so far.

PROOF. Fig. 2 shows our learning algorithm LEARN_BY_CBA for the class HEFS$(*, k, t, r)$, which is an extension of the algorithm given by Sakakibara [34]. We will only state the difference between Sakakibara's algorithm and ours in the proof.

Starting with $H = \emptyset$, the algorithm executes the while loop at line 2 until

EQ(H) returns "yes." If a negative counterexample E is returned at line 3, then hypothesis H is *too strong*, i.e., $H \models E$. In this case, the algorithm tries to detect an incorrect clause $C \in H$ such that $H_* \not\models C$ by searching the proof-DAG T for E by H from lines 5 to line 10 with a *contradiction backtracing algorithm* (CBA) [35]. Initially, the root is false in the model $M(H_*)$. Starting from the root, the algorithm goes downward by following any false child of the current node. Eventually, the algorithm reaches a false node A none of whose children is false in $M(H_*)$. Then, we know that there exists some clause $C \in H$ that subsumes $(A \leftarrow B_1, \ldots, B_{t'})$ which is false in $M(H_*)$ and should be removed from H. By the similar discussion as [34] and by Lemma 27, we can show that the CBA still correctly works for any subclass of variable-bounded EFSs and runs in polynomial time in p and n making at most $q_1(p, n)$ PMQs.

On the other hand, if a positive counterexample E is returned, then hypothesis H is *too weak*, i.e., $H \not\models E$. In this case, the algorithm tries to find all candidate clauses used to construct a proof-DAG for E by H_*. By Lemma 7, there exists some hereditary clause C such that $hd(C)\theta = hd(E)$ for some substitution θ. Therefore, by an execution of the step of line 12, we can add at least one clause in H_*. This step may add some false clauses to H, but they will be eventually removed by the CBA steps. By Lemma 30, the cardinality of the candidate set $Cand(E, k, t, r)$ is bounded by $q_2(p, n)$, and the time complexity to construct $Cand(E, k, t, r)$ is also at most $q_2(p, n)$. Finally, we can show that the execution from lines 5 to line 10 and at line 12 are iterated at most $O(m + m\, q_2(p, n))$ and m times, respectively. Hence, the number of EQs and PMQs is bounded by $O(m + m\, q_2(p, n)) = O(mp^t n^{2k+2rt} k^k)$, and $O(m\, q_1(p, n)\, q_2(p, n)) = O(mp^{t+1} n^{2k+2r(t+1)} k^k)$ respectively. $\square$

3.2 *The learnability of a subclass of terminating HEFSs*

In this subsection, we present the learning algorithm LEARN_BY_GEN for THEFS($*, k, *, r$) with EntEQs, EntMQs and DQs as Fig. 3.

In the following, we denote by H_* the target hypothesis and we assume that a fixed signature $\mathcal{S}$ is given to the learner before a learning session. The algorithm starts with the most specific hypothesis $H = \emptyset$ and searches hypothesis space THEFS($*, k, *, r$) from specific to general with respect to the subsumption lattice based on $\sqsupseteq$. For each positive counterexample E returned by EntEQ, the algorithm constructs another positive example D that is subsumed by some clause in H_*. Then, the algorithm generalizes hypothesis H by carefully merging the obtained example D with some clause in H so that only positive counterexamples are provided.

```
Procedure: LEARN_BY_GEN
/* A learning algorithm for THEFS(*, k, *, r) with EntEQs, EntMQs,
and DQs. S: a fixed signature */
H := ∅;
while EntEQ(H) = "no" do begin /* Ent(H) ≠ Ent(H_*) */
    E := the counterexample returned by the EntEQ;
    D := Saturate(E, H, S); /* Compute the saturant by H */
    D := Rewind(D, S); /* Compute the prime counterexample */
    for each C ∈ H do begin
        if EntMQ(F)="yes" for some F ∈ MCS(C, D, S, k) then
            H := (H − {C}) ∪ {F} and goto FOUND;
    end /* for */
    H := H ∪ {D};
    FOUND:
end /* main loop */
return H;
```

Fig. 3. A polynomial-time learning algorithm for THEFS$(*, k, *, r)$ with EntEQs, EntMQs and DQs, based on saturation, rewind and minimal common subsumer.

3.2.1 The Saturation and the Rewind procedures

The first task of the algorithm is, given a positive example E, to constructs another positive example D that is subsumed by some clause in H_*. From the subsumption theorem (Lemma 7), we know that there are three cases for the clause E, (i) E is a tautology, (ii) E is directly subsumed by some clause in H_*, and (iii) there is a non-trivial proof-DAG for E by H_*. The first case (i) is impossible since E is a counterexample for H. If the second case (ii) holds then the task is already done. Therefore, we will deal with the third case (iii) by using the saturation and the rewind procedures, which invert the proof steps by which positive examples are derived from clauses in H_*.

For a clause C, the saturation is an operation to add to the body of C all atoms derivable from the body of C and H. More formally, for a clause C and an EFS H, $Closure_{\mathcal{S},H}(bd(C))$ is the set of all atoms $B \in Atom_{\mathcal{S}}(hd(C))$ such that $H \models \forall(B \leftarrow bd(C))$. Then, the *saturant* of C by H, denoted by $Saturant(C, H, \mathcal{S})$, is the clause $hd(C) \leftarrow Closure_{\mathcal{S},H}(bd(C))$.

Lemma 32 For every fixed $k, r \geq 0$, the saturant of any clause C by any HEFS $H \in$ HEFS$(*, k, *, r)$ is unique up to renaming, of polynomial size in $||C||$, and polynomial-time computable in $||C||$ and $||H||$.

Lemma 33 If a clause C is a positive counterexample of H w.r.t. H_*, then the saturant of C by H is also a positive counterexample of H w.r.t. H_*.

PROOF. By definition, C subsumes its saturant $D = Saturant(C, H, \mathcal{S})$. Therefore, $H_* \models C$ implies $H_* \models D$. Conversely, the saturant D is obtained from C by adding to the body of C only the atoms entailed by H. We have $H \models \forall(bd(C) \rightarrow bd(D))$, and it follows that $H \models D$ implies $H \models C$. $\square$

A positive example $C \in Ent(H_*)$ for H_* is called *prime w.r.t.* H_* if all proof-DAGs for C by H_* are trivial, and called *composite* otherwise.

Lemma 34 If a positive counterexample C is prime then C is subsumed by some clause in H_*.

PROOF. C is neither a tautology nor a clause with some non-trivial proof-DAG by H_*. Thus, the result immediately follows from Lemma 7. $\square$

The converse of the above lemma does not hold in general.

Lemma 35 Let H_* and H be EFSs in THEFS$(*, k, *, r)$. Given any saturated positive counterexample C for H_* w.r.t. H, the algorithm Rewind in Fig. 4 finds a prime positive counterexample for H_* w.r.t. H in polynomial time by using $O(pn^{2r})$ EntMQ and $O(pn^{2r})$ DQ, where $n = ||hd(C)||$, $p = \#\Pi$ and $r = \text{arity}(\Pi)$.

PROOF. Let $C = (A \leftarrow Body)$ be any saturated positive counterexample for H_* w.r.t. H. Let $A_0 = A, A_1, \dots, A_i, \dots$ $(i \geq 0)$ be the sequence of the values of the atom A at line 2 of the algorithm Rewind in Fig. 4, where A_i is the value at the i-th execution of the for-loop (the i-th stage). For every $i \geq 0$, let C_i be the clause $(A_i \leftarrow bd(C))$. By assumption, $C_0 = C$ is a saturated positive counterexample for H_* w.r.t. H. Then, we show the following claim for every $i \geq 0$.

(Claim 1) If C_i is a saturated positive counterexample for H_* w.r.t. H, and furthermore C_i is not prime, then there exists some atom $B = A_{i+1} \in Atom_{\mathcal{S}}(A) - bd(C_i)$ such that $DQ(A_{i+1}, B) =$"yes" and $EntMQ(B \leftarrow bd(C_i)) =$"yes".

(*Proof for the claim*) If C_i is not prime then there is a non-trivial proof-DAG T for C_i by H_*. Such a non-trivial proof-DAG T contains some node B that does not appear in C_i. By definition, B is neither the root nor an atom in $bd(C_i)$. Since C_i is saturated by H, we have $B \in bd(C_i)$ iff $H \models \forall(B \leftarrow bd(C_i))$. Therefore, if $B \notin bd(C_i)$ then we have that $H \not\models \forall(B \leftarrow bd(C_i))$. On the other hand, for any node B in a proof-DAG T for C_i by H_*, $H_* \models \forall(B \leftarrow bd(C_i))$ holds. Thus, we have that $EntMQ(B \leftarrow bd(C_i)) =$"yes". By construction, B is a descendant of the root A_{i+1}. Thus, we also have $DQ(A_{i+1}, B) =$"yes".

Procedure *Saturate*$(D, H, \mathcal{S})$
1 $Body := \emptyset$; $Head := hd(D)$;
2 **for** each $B \in Atom_{\mathcal{S}}(Head)$ **do**
3   Let σ be the Skolem substitution for $(B \leftarrow bd(D))$ w.r.t. H;
4   **if** $(H \cup bd(D\sigma) \models B\sigma)$ **then**
5         $Body := Body \cup \{B\}$;
6 **return** $(Head \leftarrow Body)$;

Procedure *Rewind*$(C, \mathcal{S})$
1 $A := hd(C)$; $Body := bd(C)$; $S := Atom_{\mathcal{S}}(A) - Body$;
2 **while** (DQ(A, B) and EntMQ$(B \leftarrow Body)$ return "yes" for $\exists B \in S$) **do**
3     $A := B$;
4 **return** $(A \leftarrow Body)$; /* prime w.r.t. H_* */

Fig. 4. The procedure Saturate to compute a saturated positive counterexample and the procedure Rewind to compute a prime positive counterexample.

Furthermore, we know that $C_{i+1} = (B \leftarrow bd(C_i))$ is a positive counterexample for H_* w.r.t. H. (*End of the proof for the claim*)

By the above claim, we know that if the while-loop at line 2 terminates then the clause C_i must be prime w.r.t. H_*. Also, C_i is a positive counterexample. On the other hand, the sequence of generated atoms form the decreasing sequence $A_0 = A >_{H_*} A_1 >_{H_*} \cdots >_{H_*} A_i >_{H_*} \cdots$ w.r.t. the dependency relation $>_{H_*}$ for H_*. If H_* is an HEFS, all A_i are members of $Atom_{\mathcal{S}}(A)$ and since H_* is terminating then all $A_0, A_1, \cdots$ must be mutually distinct. Thus, it follows from Lemma 28 that the length of the decreasing sequence is bounded above by $\#Atom_{\mathcal{S}}(A) = O(pn^{2r})$, where $n = ||A||$. Hence, the time and the query complexities immediately follow. □

From Lemma 33, Lemma 34 and Lemma 35, we know that the procedures Saturate and Rewind finds a prime positive counterexample D from a given positive counterexaple E at line 3 to line 5 of the algorithm LEARN_BY_GEN in Fig. 3.

3.2.2 Maximal common subsumers

Once a prime positive counterexample D is found, the remaining task in LEARN_BY_GEN is to generalize the current hypothesis H by merging D with H. This is possibly done by taking the least upper bound of D and some clause $C \in H$ w.r.t. the subsumption relation $\sqsupseteq$ [10,15,20,32]. Unfortunately, no unique upper bound w.r.t. $\sqsupseteq$ exists for patterns or hereditary

Procedure $MCS(D_1, D_2, \mathcal{S}, k)$

$S := \{ (A, \theta_1, \theta_2) \mid A \in Atom_{\mathcal{S}}, o(A) \leq k, A\theta_1 = hd(D_1), A\theta_1 = hd(D_2) \}$;

$CS := \emptyset$;

for each $(A, \theta_1, \theta_2) \in S$ **do**

$$Body := \left\{ B \in Atom_{\mathcal{S}}(A) \;\middle|\; \begin{array}{l} \text{DQ}(A, B) \text{ returns “yes,”} \\ B\theta_1 \in bd(D_1) \text{ and } B\theta_2 \in bd(D_2) \end{array} \right\};$$

$CS := CS \cup \{(A \leftarrow Body)\}$;

return CS;

Fig. 5. The procedure to compute minimal common subsumer.

clauses. Hence, we introduce the maximal common subsumers.

Definition 36 Let $\mathcal{S}$ be a signature, $\mathcal{C}$ a subclass of $Clause_{\mathcal{S}}$, and D_i a clause over $\mathcal{S}$ $(i = 1, 2)$. A *common subsumer* of D_1 and D_2 within $\mathcal{C}$ is a clause $C \in \mathcal{C}$ such that $C \sqsupseteq D_1$ and $C \sqsupseteq D_2$. A common subsumer C of D_1 and D_2 within $\mathcal{C}$ is *maximal* if there is no common subsumer D of D_1 and D_2 in $\mathcal{C}$ such that $bd(C) \subset bd(D)$.

Let $\mathcal{S}$ be a signature (Σ, Π). Then, we denote by $\text{MCS}(D_1, D_2, \mathcal{S}, k)$ the set of all maximal common subsumers of D_1 and D_2 in hereditary clauses over $\mathcal{S}$ of which variable-occurrence is at most k. There are more than exponentially many common subsumers for given C and D. However, there are only polynomially many maximal ones.

Lemma 37 Let $\mathcal{S}$ be a signature (Σ, Π), D_i a clause over $\mathcal{S}$ $(i = 1, 2)$ and $k \geq 0$ an integer. Then, the set $\text{MCS}(D_1, D_2, \mathcal{S}, k)$ is of cardinality $q_3(n) = n^{4k}k^k$, of polynomial size, and polynomial-time computable in $p = \#\Pi$ and $n = ||D_1|| + ||D_2||$.

PROOF. Consider the procedure in Fig. 5 that computes $\text{MCS}(D_1, D_2, \mathcal{S}, k)$ using DQ. It is not hard to see that this procedure works correctly. Furthermore, we can show that $\#S \leq n^{4k}k^k$ and $\#Body \leq pn^{2r}$ by Lemma 28 and Lemma 29. $\square$

3.2.3 The correctness and the time complexity

Now, we prove the correctness of the learning algorithm LEARN_BY_GEN in Fig. 3. In the following, let $H_0, H_1, \ldots, H_n, \ldots$ and $E_0, E_1, \ldots, E_n, \ldots$ $(n \geq 0)$ be the sequence of hypotheses and counterexamples, respectively, where H_0 is the initial hypothesis $\emptyset$, and at each stage $i \geq 1$, LEARN_BY_GEN makes the entailment equivalence query $\text{EntEQ}(H_{i-1})$, receives a counterexample E_i to the query, and produces a new hypothesis H_i from E_i and H_{i-1}. A clause is *missing* if it is subsumed by some clause in H_* but not entailed by the present hypothesis H.

Lemma 38 Suppose that a positive example C subsumes another positive example D, i.e., $C \sqsupseteq D$. If D is prime w.r.t. H_*, then so is C.

PROOF. Since $C \sqsupseteq D$, there exists a substitution θ such that $C\theta \subseteq D$. If C is composite w.r.t. H_*, then we can transform a proof-DAG T_C for $H_* \models C$ to a proof-DAG for $H_* \models D$, by applying θ to all atoms in T_C. Since D is not composite, this is a contradiction. $\square$

Lemma 39 For every $n \geq 0$, $H_* \sqsupseteq H_n$ and E_n is a positive counterexample. Furthermore, H_n is a conservative refinement of H_*.

PROOF. We show by induction on $n \geq 0$ that $H_* \sqsupseteq H_n$ and that H_n consists of just prime clauses w.r.t H_*. If $n = 0$, then $H_0 = \emptyset$ and the claim trivially holds. Next, suppose $n > 0$. By induction hypothesis, $H_* \sqsupseteq H_{n-1}$ and thus the next counterexample $E = E_n$ at line 4 is positive. Let D be the clause obtained after executing lines 4 to line 8. Combining Lemma 33, Lemma 32 and Lemma 35, we can show that D is still saturated and $>$-minimal w.r.t. H_* by H and $D \in Ent(H_*) - Ent(H_{n-1})$. By Lemma 35 D is prime. Thus, by Lemma 34, D is subsumed by some missing clause in H_*. Suppose first that there exists some $C \in H_{n-1}$ and some $F \in \mathrm{MCS}(C, D, \mathcal{S}, k)$ such that EntMQ(F) returns "yes." Then, $H_n = (H_{n-1} - \{C\}) \cup \{F\}$. By induction hypothesis, C as well as D is prime. By Lemma 38, F is also prime, so it follows from Lemma 34 that F is subsumed by some clause in H_*. Since $H_* \sqsupseteq H_{n-1}$, this implies that $H_* \sqsupseteq H_n$. Next suppose that there is no such $C \in H_{n-1}$, and then $H_n = H_{n-1} \cup \{D\}$. Since D is prime, it follows from Lemma 34 that $H_* \sqsupseteq H_n$. A new clause F is added to H_n at line 12 only if there exists no maximal common subsumer of D and C subsumed by H_* for all clauses $C \in H_n$. Hence, the refinement H_n of H_* is always conservative. $\square$

Corollary 40 $H_* \sqsupset \cdots \sqsupset H_n \sqsupset \cdots \sqsupset H_1 \sqsupset H_0$ $(n \geq 0)$.

Lemma 41 For HEFS$(*, k, *, r)$, there exists no increasing sequence $\cdots \sqsupset C_1 \sqsupset C_0$. Furthermore, its length is always bounded by $O(pn^{2r+1})$, where $p = \#\Pi$ and $n = |hd(C_0)|$.

PROOF. By using the discussion in [9], we can show that the length of the sequence $\cdots \sqsupset A_1 \sqsupset A_0$ of atoms is bounded by $||A_0|| = O(n)$ independent from k. For a given head A, the maximum size of the body is bounded by

$\#Atom_{\mathcal{S}}(A) = O(pn^{2r})$. Hence, we have the upper bound of the length of the sequence as $O(pn^{2r+1})$. □

Theorem 42 Let $\mathcal{S} = (\Sigma, \Pi)$ be a signature. For every $k, r \geq 0$, the class THEFS$(>, *, k, *, r)$ is polynomial-time exact learnable with $O(pmn^{2r+1})$ EntEQ, $O(p^2m^2n^{4k+4r+1}k^k)$ EntMQ, and $O(p^2mn^{4k+4r+1}k^k)$ DQ, where m is the cardinality of a target THEFS, $p = \#\Pi$ and n is the size of the longest counterexample received so far.

PROOF. Since the algorithm LEARN_BY_GEN terminates only if the EQ returns "yes," it is sufficient to show the termination in polynomial time. By Corollary 40, the sequence of hypotheses is of the form $H_* \sqsupset \cdots \sqsupset H_n \sqsupset \cdots \sqsupset H_1 \sqsupset H_0$ $(n \geq 0)$ (1). By Lemma 39, each H_n is a conservative refinement of H_*, so $\#H_n \leq \#H_* = m$.

Fix an enumeration $H_* = (C_1^*, \ldots, C_m^*)$. For every $n \geq 0$, we can order H_n as the m-tuple $(C_1^n, \ldots, C_m^n) \in Clause_{\mathcal{S}}^m$ such that, for each i, C_i^n is the unique member of H_n satisfying $C_i^* \sqsupseteq C_i^n$ if it exists and $C_i^n = \perp$ otherwise, where $\perp$ is a special symbol denoting that $C \sqsupseteq \perp$ for every $C \in Clause_{\mathcal{S}}$.

It follows from Lemma 41 that, for every $1 \leq i \leq m$, the length of the longest subsequence such that $\cdots \sqsupseteq C_i^2 \sqsupset C_i^1$ is bounded by $O(pn^{2r+1})$. Thus, both the lengths of the sequence (1) and the number of EntEQs are bounded by $q_4(p, m, n) = O(pmn^{2r+1})$. By Lemma 32, Lemma 35 and Lemma 37, the number of EntMQs is bounded by $q_5 = O(pmn^{4k+2r}k^k)$ and the running time in each iteration of the while-loop is bounded by a polynomial in p, m and n. Hence, the total number of EntMQs is $q_4(p, m, n)q_5(p, m, n) = O(p^2m^2n^{4k+4r+1}k^k)$ and the running time is polynomial in p, m and n. □

Since any counterexample in the language semantics $(Atom_{\mathcal{S}}, L_{\mathcal{S}}(\cdot, p_0))$ is also a counterexample in the entailment semantics $(Clause_{\mathcal{S}}, Ent_{\mathcal{S}}(\cdot))$, we can replace each EntEQ in Theorem 42 with EQ.

Corollary 43 For every $k, r \geq 0$, the class THEFS$(*, k, *, r)$ is polynomial-time exact learnable with EQ, EntMQ, and DQ.

Suppose that we have an efficiently decidable, well-founded transitive relation $>$ over $Atom_{\mathcal{S}}$. In this case, we can eliminate DQ to learn a subclass THEFS$(>, *, k, *, r)$ consisting of the programs uniformly bounded by $>$. The class of *reducing programs* [43] is an example of such uniformly terminating EFS.

Corollary 44 Let $>$ be any well-founded transitive relation over $Atom_{\mathcal{S}}$ that is polynomial time decidable. For every $k, r \geq 0$, the class THEFS$(>, *, k, *, r)$ is polynomial-time exact learnable with EQ and EntMQ.

3.2.4 A lowerbound result

By Theorem 31 and Theorem 42, note that the number $O(pmn^{2r+1})$ of EQ made by LEARN_BY_GEN is significantly smaller than $O(p^t m\, n^{2k+2rt} k^k)$ EQ by LEARN_BY_CBA for large $k, t \geq 1$. In this subsection, we analyze the query complexity of the class THEFS$(m, k, *, r)$, and obtain the lower bound result, which indicates that the query complexity is almost optimal in terms of m and n for EQ.

Theorem 45 Let $\mathcal{S}$ be any signature with at least two letters. For every integers $k, r \geq 0$ such that $k \geq 4r$, any algorithm that exactly identifies all hypotheses in THEFS$(m, k, *, r)$ with EntEQ and EntMQ must make $\Omega(mn^{r/2})$ queries in the worst case, where m is the cardinality of a target THEFS and n is the size of the longest counterexample received so far.

PROOF. We say that a concept class $\mathcal{C}$ *shatters* a set $U \subseteq \Sigma^*$ if $\{ U \cap c \mid c \in \mathcal{C} \} = 2^U$ holds. The *VC-dimension* of $\mathcal{C}$, denoted by $VC(\mathcal{C})$, is the cardinality of the largest set $U \subseteq \Sigma^*$ that is shattered by $\mathcal{C}$. From arguments in Maass and Turán [23], it is sufficient to show that $VC(\text{THEFS}(m, k, *, r)) = \Omega(mn^{r/2})$.

Let $p, q, r, len, bit \in \Pi$ be predicate symbols of arity $r+1, 2r, r, 2, 1$, respectively. Let $x_i, y_i, z_i, v_i \in X$ be variables for $1 \leq i \leq r$. For an integer $n \geq 0$, $[n]$ denotes the set $\{1, \ldots, n\}$. Then, we encode an integer $i \in [n]$ by the bit vector $\psi(i) = 0^{i-1}10^{n-i} \in \{0,1\}^n$ and an r-vector $(i_1, \ldots, i_r) \in [n]^r$ by an atom $p(\psi(i_1), \ldots, \psi(i_r), 0^n) \in Base_{\mathcal{S}}$. Let $S_{r,n}$ be the set

$$\{ p(\psi(i_1), \ldots, \psi(i_r), 0^n) \mid (i_1, \ldots, i_r) \in [n]^r \}$$

of ground atoms of length $(r+1)n$ corresponding to all n^k r-vectors in $[n]^k$. For any subset $T \subseteq S_{r,n}$, we define

$$S_{r,n}(T) = \{ p(\psi(i_1), \ldots, \psi(i_r), 0^n) \mid (i_1, \ldots, i_r) \in T \}.$$

Then, we define the EFS H_T that represents the set $S_{r,n}(T)$ as follows, where $\overline{T} = [n]^k - T$.

$$p(x_1, \ldots, x_r, 0^n) \leftarrow \bigwedge_{(i_1, \ldots, i_r) \in \overline{T}} \left[q(x_1, \ldots, x_r; 0^{i_1}, \ldots, 0^{i_r}) \right].$$
$$q(x_1 y_1 z_1, \ldots, x_r y_r z_r; v_1, \ldots, v_r) \leftarrow$$
$$\bigwedge_{1 \leq j \leq r} \left[len(x_j y_j, v_j) \wedge bit(y_j) \right] \wedge\ r(y_1, \ldots, y_r).$$
$$r(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_r) \leftarrow, \qquad \text{for all } 1 \leq i \leq r.$$
$$len(ax, 0y) \leftarrow len(x, y),$$
$$len(a, 0) \leftarrow,$$
$$bit(a) \leftarrow, \qquad \text{for all } a \in \{0,1\}.$$

Note that H_T is terminating and hereditary.

Let $w \in \{0,1\}^r$ be a bit vector of length r. Then, it holds that, for every $\alpha \in \{0,1\}^*$ and $i \in [n]$, $H_T \models len(\alpha, 0^i)$ iff $|\alpha| = i$. Also, for every $i \in [n]$ and every string $w = \alpha\beta\gamma$ ($\alpha, \beta, \gamma \in \{0,1\}^*$), if $H_T \models len(\alpha\beta, 0^i) \wedge bit(\beta)$, then β is the i-th bit of w. Furthermore, it holds that, for every $b_1 \cdots b_r \in \{0,1\}^r$, $H_T \models r(b_1, \ldots, b_r)$ iff $b_1 \cdots b_r \neq 1^r$, and $H_T \models q(\psi(i_1), \ldots, \psi(i_r), 0^{j_1}, \ldots, 0^{j_r})$ iff $(i_1, \ldots, i_r) \neq (j_1, \ldots, j_r)$. Hence, it is not hard to see that, for every $(i_1, \ldots, i_r) \in [n]^r$, $H_T \models p(\psi(i_1), \ldots, \psi(i_r), 0^n)$ iff $(i_1, \ldots, i_r) \notin \overline{T}$. Since each H_T belongs to HEFS$(r+8, 4r, *, 2r)$, the class HEFS$(r+8, 4r, *, 2r)$ shatters the set $S_{r,n}$ of the cardinality n^r.

Similarly, we can show that the class HEFS$(m+r+7, 4r, *, 2r)$ shatters the direct sum $S_{m,r,n} = S^{(1)}_{r,n} \cup \cdots \cup S^{(m)}_{r,n}$ of cardinality mn^r obtained by making the m copies of the predicate P. Hence, it immediately follows that $VC(\text{HEFS}(m,k,*,r)) = \Omega((m-r-7)\hat{n}^{r/2}/2r^r) = \Omega(m\hat{n}^{r/2})$ in m and n when $k \geq 4r$, where the maximum length of the examples is $\hat{n} = (r+1)n$. $\square$

4 Hardness Results for Learning HEFSs

In this section, we present several *representation-independent* hardness results of predicting the subclasses of HEFSs, which claim the necessity of both the types of queries and the bounds on the parameters are necessary for their efficient learning mentioned in the previous section.

We fix f, g and h to an instance mapping, a concept mapping, and a membership query mapping. Also the parameters n and s denote the bounds of examples and representations, respectively. For simplicity, we assume that the length of examples of Boolean concepts is always fixed to the upper bound n. Furthermore, a signature is always fixed and a semantics is the language semantics.

4.1 Regular pattern languages revisited

We denote by $\mathcal{RP}$, $\cup_m\mathcal{RP}$ and $\cup\mathcal{RP}$ regular pattern languages, at most m unions of regular pattern languages, and unbounded unions of regular pattern languages, respectively (*cf.* [12,25,26,36,37,39]). Since each regular pattern language $L(\pi)$ is definable by the HEFS $\{p(\pi) \leftarrow\}$, we can easily observe that $\mathcal{RP}$, $\cup_m\mathcal{RP}$ and $\cup\mathcal{RP}$ are corresponding to HEFS$(1,*,0,1)$, HEFS$(m,*,0,1)$ and HEFS$(*,*,0,1)$, respectively. It is known that $\mathcal{RP}$ and $\cup_m\mathcal{RP}$ are not polynomial-time PAC-learnable unless $\mathbf{NP} = \mathbf{RP}$ [25,26], where these are *representation-dependent* hardness results.

Theorem 46 The class $\mathcal{RP}$ is not polynomial-time predictable, if the class $\mathcal{DNF}$ is not polynomial-time predictable.

PROOF. It is sufficient to show that $\mathcal{DNF}_n \trianglelefteq \mathcal{RP}$ for all $n \geq 0$. Let $d = t_1 \vee \cdots \vee t_m$ be a DNF formula over the set $\{x_1, \ldots, x_n\}$ of Boolean variables. For each vector $e = e_1 \cdots e_n \in \{0,1\}^n$, let $\tilde{e} = 1e_1 1e_2 1 \cdots 1e_n 1$ and let $\alpha = (01)^{3(2n+1)}$. Then, construct f and g as follows:

$$f(n,s,e) = e' = (\mathtt{A}\tilde{e}\mathtt{A}\alpha)^{m-1} \cdot \mathtt{A}\tilde{e}\mathtt{A},$$

$$g(n,s,d) = P = \mathtt{A}P_1\mathtt{A}P_2\mathtt{A}\cdots\mathtt{A}P_m\mathtt{A}, \text{ where } \mathtt{A} \text{ is a new symbol.}$$

Here, $P_j = *p_1^j * p_2^j * \cdots * p_n^j *$, where all $*$ are mutually distinct variables in X and $p_i^j = 1$ if t_j contains x_i, $p_i^j = 0$ if t_j contains $\overline{x_i}$, and x_i^j otherwise.

We show that, if e satisfies d, then $e' \in L(P)$. The following statements hold: (a) e satisfies d iff there exists an index j ($1 \leq j \leq m$) such that $\tilde{e} \in L(P_j)$, because $|\tilde{e}| = |P_j| = 2n+1$. (b) For each P_j ($1 \leq j \leq m$), α is of the form $\alpha_1\alpha_2\alpha_3$ such that $|\alpha_1|, |\alpha_2|, |\alpha_3| > 0$ and $\alpha_2 \in L(P_j)$. (c) For each P_j ($1 \leq j \leq m$), it holds that both $\tilde{e}\mathtt{A}\alpha, \alpha\mathtt{A}\tilde{e} \in L(P_j)$ because of (b). From the (a) and (c), it holds that $e' \in \mathtt{A}L(P_1)\mathtt{A}\cdots\mathtt{A}L(P_i)\mathtt{A}\cdots\mathtt{A}L(P_m)\mathtt{A}$. Hence, $e' \in L(P)$.

Conversely, suppose that e does not satisfy d. From the (a), it holds that (d) $\tilde{e} \notin L(P_j)$ for every j ($1 \leq j \leq m$). Furthermore, (e) $\tilde{e} \notin L(P')$ for any substring P' of P containing an $\mathtt{A}$, because e contains no $\mathtt{A}$. From the conditions (d) and (e), if $e' \in L(P)$, then at least one of the two $\mathtt{A}$'s for each occurrence $\mathtt{A}\tilde{e}\mathtt{A}$ in e' must be substituted to a variable of a P_j in P. Since the number of $\mathtt{A}$'s in e' is $2m$, the remained $\mathtt{A}$'s in e' to match with all $\mathtt{A}$ in P are at most m. However, P contains only $m+1$ $\mathtt{A}$'s, so it is impossible that $e' \in L(P)$. Hence, $e' \notin L(P)$ and we can conclude that $\mathcal{DNF}_n \trianglelefteq \mathcal{RP}$. □

The $\mathcal{RP}$ is learnable in polynomial-time with membership and equivalence queires [24], however, the learnability of $\cup\mathcal{RP}$ with the queries is not known. We show that, in case of binary alphabet, learning $\cup\mathcal{RP}$ is no easier than learning $\mathcal{DNF}$.

Theorem 47 The class $\cup\mathcal{RP}$ over two-letter alphabet is not polynomial-time predictable with membership queries, if $\mathcal{DNF}$ is not polynomial-time predictable with membership queries.

PROOF. It is sufficient to show that $\mathcal{DNF}_n \trianglelefteq_{\rm pwm} \cup\mathcal{RP}$ for all $n \geq 0$. For a DNF formula $d = t_1 \vee \cdots \vee t_m$, let π_i $(1 \leq i \leq m)$ and π be regular patterns $p_1^j \cdots p_n^j$ and $x_1 \cdots x_n x_{n+1}$, respectively. Here, p_i^j $(1 \leq i \leq n, 1 \leq j \leq m)$ is defined as similar as the proof of Theorem 46. Then, construct f, g and h as follows:

$$f(n,s,e) = e,$$

$$g(n,s,d) = \{\pi_1, \ldots, \pi_m, \pi\},$$

$$h(n,s,e') = \begin{cases} e' \text{ if } |e'| = n, \\ \bot \text{ if } |e'| < n, \\ \top \text{ if } |e'| > n. \end{cases}$$

For each $e' \in \{0,1\}^*$, we can check the properties of h in Definition 23 as follows. Since $L(\pi) = \{w \in \{0,1\}^* \mid |w| \geq n+1\}$, if $h(n,s,e') = \top$, then $e' \in L(g(n,s,d))(= L(\pi_1) \cup \cdots \cup L(\pi_m) \cup L(\pi))$. On the other hand, since $|\pi_j| = n$ $(1 \leq j \leq m)$ and $|\pi| = n+1$, $L(g(n,s,d))$ contains no strings of length $< n$. So, if $h(n,s,e') = \bot$, then $e' \notin L(g(n,s,d))$. If $h(n,s,e') = e'$, then $e' \notin L(\pi)$ because $|e'| = n$. Thus, $e' \in L(\pi_1) \cup \cdots \cup L(\pi_m)$ and there exists an index i $(1 \leq i \leq m)$ such that $e' \in L(\pi_i)$ iff e' is obtained by replacing the variables in π_i with 0 or 1, which is corresponding to a truth assignment satisfying t_i. Hence, $e' \in L(g(n,s,d))$ iff e' satisfies d.

Furthermore, for each $e \in \{0,1\}^n$, e satisfies d iff $f(n,s,e) \in L(g(n,s,d))$. Hence, it holds that $\mathcal{DNF}_n \trianglelefteq_{\rm pwm} \cup\mathcal{RP}$. $\square$

On the other hand, by using the corresponding DFA to a regular pattern, we can obtain the following theorem:

Theorem 48 (Hirata & Sakamoto [17]) For each $m \geq 0$, the class $\cup_m\mathcal{RP}$ is polynomial-time predictable with membership queries.

4.2 Other hardness results

By Theorem 47 in Section 4.1, we can conclude that HEFS$(*,*,t,r)$ $(t \geq 0,\ r \geq 1)$ is not polynomial-time predictable with membership queries, if neither are DNF formulas. In this subsection, we discuss the subclasses of HEFS$^-(*,k,t,r)$, which are restricted that all *facts* contain no variable.

From the learnability of k-bounded ESEFSs by Sakakibara [34] and the learnability of HEFS$(*,k,t,r)$ by Theorem 31, it arises a natural question whether we can replace the predicate membership queries with the ordinal membership queries. The next theorem claims that it is impossible preserving efficient learnability.

Theorem 49 For every $k, t, r \geq 1$, $\mathrm{HEFS}^-(*, k, t, r)$ is not polynomial-time predictable with membership queries under the cryptographic assumptions.

PROOF. It is sufficient to show that $\cup\mathcal{D}FA \trianglelefteq_{\mathrm{pwm}} \mathrm{HEFS}^-(*, 1, 1, 1)$ by Theorem 24 and 25. Let $M_1, \ldots, M_r$ be DFAs over the same alphabet Σ. Suppose that $c \notin \Sigma$. For each $M_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ $(1 \leq i \leq r)$, construct $H_1(n, s, M_i) \in \mathrm{HEFS}^-(*, 1, 1, 1)$ as follows:

(1) $q(ax) \leftarrow r(x) \in H_1(n, s, M_i)$ if $\delta_i(q, a) = r$ for each $q, r \in Q_i$ and $a \in \Sigma$;
(2) $q(c) \leftarrow \in H_1(n, s, M_i)$ for each final state $q \in F_i$.
(3) $p(x) \leftarrow q_0^i(x) \in H_1(n, s, M_i)$ for each $q_0^i \in Q_i$, where $p \notin Q_1 \cup \cdots \cup Q_r$.

Then, construct f, g and h as follows:

$$f(n, s, w) = wc,$$
$$g(n, s, \{M_1, \ldots, M_r\}) = H_1(n, s, M_1) \cup \cdots \cup H_1(n, s, M_r),$$
$$h(n, s, w') = \begin{cases} w \text{ if } w' = wc, \\ \perp \text{ otherwise.} \end{cases}$$

The size of $g(n, s, \{M_1, \ldots, M_r\})$ is bounded by a polynomial in the size of all M_i's $(1 \leq i \leq r)$. Furthermore, it holds that (1) $w \in L(M_1) \cup \cdots \cup L(M_r)$ iff $f(n, s, w) \in L(g(n, s, d), p)$ for each $w \in \Sigma^{[n]}$, (2) if $h(n, s, w') = \perp$, then $w' \notin L(g(n, s, d), p)$, and (3) if $h(n, s, w') = w$, then it holds that $w' \in L(g(n, s, d), p)$ iff $w \in L(M_1) \cup \cdots \cup L(M_r)$. Hence, it holds that $\cup\mathcal{D}FA \trianglelefteq_{\mathrm{pwm}} \mathrm{HEFS}^-(*, 1, 1, 1)$. □

Recall that every k-bounded ESEFSs are contained in $\mathrm{HEFS}^-(*, k, k, 1)$. The following theorem claims that, if neither the variable-occurrence nor the number of atoms in the body are bounded, then HEFSs are not polynomial-time predictable even with predicate membership queries.

Theorem 50 For every $r \geq 1$, $\mathrm{HEFS}^-(*, *, *, r)$ is not polynomial-time predictable with predicate membership queries, if $\mathcal{D}NF$ is not polynomial-time predictable with membership queries.

PROOF. First, we show that $\mathcal{D}NF_n \trianglelefteq_{\mathrm{pwm}} \mathrm{HEFS}^-(*, *, *, 1)$ for all $n \geq 0$. Let $d = t_1 \vee \cdots \vee t_m$ be a DNF formula. Then, construct the following EFS

$H_2(n, s, d)$:

$$H_2(n,s,d) = \left\{ \begin{array}{l} q(0) \leftarrow \\ q(1) \leftarrow \\ p(p_1^1 \cdots p_n^1) \leftarrow q(p_1^1), \ldots, q(p_n^1) \\ \cdots \\ p(p_1^m \ldots p_n^m) \leftarrow q(p_m^1), \ldots, q(p_n^m) \end{array} \right\},$$

where p_i^j $(1 \leq i \leq n, 1 \leq j \leq m)$ is defined as similar as the proof of Theorem 46. Furthermore, let $H_2'(n, s, d)$ be an HEFS obtained by deleting all atoms $q(0)$ and $q(1)$ from the body of each clause in $H_2(n, s, d)$. Then, construct f, g and h as follows:

$$\begin{aligned} f(n,s,e) &= e, \\ g(n,s,d) &= H_2'(n,s,d), \\ h(n,s,e') &= \begin{cases} e' \text{ if } e' \in \{0,1\}^n, \\ \bot \text{ otherwise.} \end{cases} \end{aligned}$$

Since $L(g(n, s, d), p) \subseteq \{0, 1\}^n$, it is easy to see that (1) e satisfies d iff $f(n, s, e) \in L(g(n, s, d), p)$ for each $e \in \{0, 1\}^n$, and (2) $e' \in L(g(n, s, d), p)$ iff $h(n, s, e')$ satisfies d for each $e' \in \{0, 1\}^n$. Hence, it holds that $\mathcal{DNF}_n \trianglelefteq_{\text{pwm}}$ HEFS$^-(*, *, *, 1)$.

Finally, we consider whether the same result holds even if the membership queries are replaced with the predicate membership queries. Although we can extend pwm-reducibility to prediction-preserving reducibility with predicate membership queries according to Definition 23, we only discuss the case HEFS$^-(*, *, *, 1)$. Concerned with the above pwm-reduction $\mathcal{DNF}_n \trianglelefteq_{\text{pwm}}$ HEFS$^-(*, *, *, 1)$, the difference between MQs and PMQs is just to ask whether $H_2'(n, s, d) \models q(w) \leftarrow$ for $w \in \{0, 1\}^*$. Note that the predicate symbol q in $H_2'(n, s, d)$ denotes the value substituted to a Boolean variable x_i in d, so can generate just 0 and 1. Then, we can extend a membership query mapping h to a predicate membership query mapping h' as $h'(n, s, p(w)) = h(w)$; $h'(n, s, q(w)) = \top$ if $|w| = 1$; $h'(n, s, q(w)) = \bot$ if $|w| > 1$. Hence, the statement holds. $\square$

5 Conclusion

We investigated the efficient learnability of a hierarchy HEFS(m, k, t, r) of the HEFSs with the equivalence and other queries, where m is the maximum number of clauses, k is the maximum variable-occurrences in the head, t is the maximum number of atoms in the body, and r is the maximum arity of predicate symbols.

We showed three positive results for the learnability of HEFS(m, k, t, r). First, the class HEFS$(*, k, t, r)$ is polynomial-time learnable with equivalence and predicate membership queries. This is an extension of Sakakibara's result [34] for the class ESEFSs. Second, the more general class is effectively learnable if more powerful queries are allowed and the termination relation over the predicate symbols is assumed, that is the class THEFS$(*, k, *, r)$ of terminating HEFSs with additional information on the termination is learnable in polynomial time with equivalence and entailment membership queries. Third, we showed that the number of queries used in the presented learning algorithm for THEFS$(*, k, *, r)$ is nearly optimal.

The negative results for the learnability of subclasses of EFSs were proved by the prediction-preserving reduction (with membership queries). The class HEFS$(*, k, t, r)$ was shown to be learnable using the above types of queries but the predicate membership query can not be replaced by the membership query under the cryptographic assumptions.

Moreover, the class $\mathcal{R}P$ is not polynomial-time predictable if the class of DNF formulas is not polynomial-time predictable, and the class $\cup\mathcal{R}P$ is not polynomial-time predictable with membership queries, if the class of DNF formulas is not polynomial-time predictable with membership queries. On the other hand, the class $\cup_m\mathcal{R}P$ of bounded union of regular pattern languages is polynomial-time predictable with membership queries [17]. It is a strong evidence for the efficient learnability of the class.

Fig. 1 summarizes the results obtained in this paper. It is a future problem to study the learnability of the class THEFS$(*, k, *, r)$ with equivalence and predicate or entailment membership queries but without dependency queries. Khardon [20] has recently shown that function-free k-variable Horn sentences of arity r are polynomial-time learnable in various active learning models without using termination information. Thus, it would be interesting to apply his method to the classes of HEFSs.

Acknowledgment

First of all, the authors would like to thank Prof. Thomas Zeugmann, Prof. Carl Smith and Prof. Takeshi Shinohara for giving an opportunity for writing this paper, and to thank Akihiro Yamamoto, Ayumi Shinohara, Roni Khardon and Eric Martin for fruitful discussions on learning of logic programs. The second author also would like to thank Noriko Sugimoto, Shinichi Shimozono and Takashi Toyoshima for the fruitful discussions on this issue in the joint work [41], which partially gives a motivation of this paper. Finally, We are grateful to the anonymous referees for valuable comments which greately improve the correctness and the quality of this paper.

References

[1] D. Angluin, Finding patterns common to a set of strings, J. Comput. System Sci. 21 (1980) 46–62.

[2] D. Angluin, Learning regular sets from queries and counterexamples, Inform. Comput. 75 (1987) 87–106.

[3] D. Angluin, Learning k-bounded context-free grammars, Technical Report YALEU/DCS/RR-557, Yale University, 1987.

[4] D. Angluin, Queries and concept learning, Mach. Learn. 2(4) (1988) 319–342.

[5] D. Angluin, M. Kharitonov, When won't membership queries help?, J. Comput. System Sci. 50(2) (1995) 336–355.

[6] K. Apt, M. Bezem, Acyclic programs, in: Proc. 7th Internat. Conf. on Logic Programming (The MIT Press, 1990) 617–633.

[7] S. Arikawa, Elementary formal systems and formal languages – Simple formal systems, Memories of Faculty of Science, Kyushu University, Series A., Mathematics 24 (1970) 47–75.

[8] S. Arikawa, S. Miyano, A. Shinohara, T. Shinohara, A. Yamamoto, Algorithmic learning theory with elementary formal systems, IEICE Trans. Inf. Sys. E75-D(4) (1992) 405–414.

[9] S. Arikawa, T. Shinohara, A. Yamamoto, Learning elementary formal systems, Theoret. Comput. Sci. 95(1) (1992) 97–113.

[10] H. Arimura, Learning acyclic first-order Horn sentences from entailment, in: Proc. 8th Internal. Workshop on Algorithmic Learning Theory, LNAI 1316 (Springer-Verlag, 1997) 432–445.

[11] H. Arimura, H. Ishizaka, T. Shinohara, Learning unions of tree patterns using queries, Theoret. Comput. Sci. 185(1) (1997) 47–62.

[12] H. Arimura, T. Shinohara, S. Otsuki, Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data, in: Proc. 11st Symp. of Theoretical Aspects for Computer Science, LNCS 775 (Springer-Verlag, 1994) 649–660.

[13] L. Cavedon, Continuity, consistency, and completeness properties for logic programs, in: Proc. 6th Internat. Conf. on Logic Programming (The MIT Press, 1989), 571–584.

[14] H.- D. Ebbinghaus, J. Flum, W. Thomas, Mathematical logic (2nd Ed.) (Springer-Verlag, 1994).

[15] M. Frazier, L. Pitt, Learning from entailment: An application to propositional Horn sentences, in: Proc. 10th Internat. Conf. on Machine Learning (Morgan Kaufmann, 1993) 120–127.

[16] M. Frazier, L. Pitt, Classic learning, Mach. Learn. 25(2-3) (1996) 151–193.

[17] K. Hirata, H. Sakamoto, Prediction-preserving reducibility with membership queries on formal languages, in: Proc. 13th International Symposium on Fundamentals of Computation Theory, LNCS (Springer-Verlag, 2001) (to appear).

[18] D. Ikeda, H. Arimura, On the complexity of languages definable by hereditary EFS, in: Proc. 3rd Internat. Conf. on Development in Language Theory, Aristotle University of Thessaloniki, (1997), 223–235.

[19] S. Jain, A. Sharma, Elementary formal systems, intrinsic complexity, and procrastination, Inform. Comput. 132(1) (1997) 65–84.

[20] R. Khardon, Learning function-free Horn expressions, Mach. Learn. 35(1) (1999) 241–275.

[21] S. Kobayashi, Iterated transductions and efficient learning from positive data: A unifying view, in: Proc. 5th Internat. Colloq. on Grammatical Inference, LNAI 1891 (Springer-Verlag, 2000) 157–170.

[22] S. Kobayashi, T. Yokomori, On approximately identifying concept classes in the limit, in: Proc. 6th Internat. Workshop on Algorithmic Learning Theory, LNAI 997 (Springer-Verlag, 1995) 298–312.

[23] W. Maass, G. Turán, Lower bound methods and separation results for on-line learning models, Mach. Learn. 9 (1992) 107–145.

[24] S. Matsumoto, A. Shinohara, Learning pattern languages using queries, in: Proc. 3rd Euro. Conf. on Computational Learning Theory, LNAI 1208 (Springer-Verlag, 1997) 185–197.

[25] S. Miyano, A. Shinohara, T. Shinohara, Which classes of elementary formal systems are polynomial-time learnable?, in: Proc. 1st Workshop on Algorithmic Learning Theory (Ohmsha, 1991) 139–150.

[26] S. Miyano, A. Shinohara, T. Shinohara, Polynomial-time learning of elementary formal systems, New Gener. Computing 18(3) (2000) 217–42.

[27] T. Moriyama, M. Sato, Properties of language classes with finite elasticity, IEICE Trans. Inf. Sys. E78-D(5) (1995) 532–538.

[28] Y. Mukouchi, Inductive inference of an approximate concept from positive data, in: Proc. 4th Internat. Conf. on Algorithmic Learning Theory, LNAI 872 (Springer-Verlag, 1994) 484–499.

[29] Y. Mukouchi, S. Arikawa, Towards a mathematical theory of machine discovery from facts, Theoret. Comput. Sci. 137(1) (1995) 53–84.

[30] S.-H. Nienhuys-Cheng, R. De Wolf, Foundations of inductive logic programming, LNAI 1228 (Springer-Verlag, 1997).

[31] C H. Papadimitriou, Computational complexity, (Addison-Wesley, 1994).

[32] C. D. Page Jr., A. M. Frisch, Generalization and learnability: A study of constrained atoms, in: S. Muggleton (ed.), Inductive logic programming (Academic Press, 1992) l29–161.

[33] L. Pitt, M. K. Warmuth, Prediction-preserving reducibility, J. Comput. System Sci. 41(3) (1990) 430–467.

[34] Y. Sakakibara, On learning Smullyan's elementary formal systems: Towards an efficient learning method for context-sensitive languages, Advances in Soft. Sci. Tech. 2 (JSSST, 1990) 79–101.

[35] E. Y. Shapiro, Algorithmic program debugging (The MIT Press, 1982).

[36] T. Shinohara, Polynomial time inference of extended regular pattern languages, in: Proc. RIMS Symposia on Software Science and Engineering, LNCS 147 (Springer-Verlag, 1982) 191-209.

[37] T. Shinohara, Studies on inductive inference from positive data, Doctoral Thesis, Kyushu University, (1986).

[38] T. Shinohara, Rich classes inferable from positive data: Length-bounded elementary formal systems, Inform. Comput. 108(2) (1994) 175–186.

[39] T. Shinohara, H. Arimura, Inductive inference of unbounded unions of pattern languages from positive data, Theoret. Comput. Sci. 241 (2000) 191–209.

[40] R. M. Smullyan, Theory of formal systems (Princeton University Press, 1961).

[41] N. Sugimoto, T. Toyoshima, S. Shimozono, K. Hirata, Constructive learning of context-free languages with a subpansive tree, in: Proc. 5th Internat. Colloq. on Grammatical Inference, LNAI 1891 (Springer-Verlag, 2000) 270–283.

[42] L. Valiant, A theory of learnable, Comm. ACM 27(11) (1984) 1134–1142.

[43] A. Yamamoto, Procedural semantics and negative information of elementary formal system, J. Logic Program. 13(1) (1992) 89–97.

[44] K. Wright, Identification of unions of languages drawn from an identifiable class. in: Proc. 2nd Ann. Workshop on Computational Learning Theory (ACM, 1989) 328–333.