Advanced elementary formal systems

Steffen Lange\(^a\),* \quad \text{Gunter Grieser}^b \quad \text{Klaus P. Jantke}^a

\(^a\text{Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany}
\(^b\text{FB Informatik, Technische Universität Darmstadt, Alexanderstraße 10, 64283 Darmstadt, Germany}

Abstract

An elementary formal system (EFS) is a logic program such as a Prolog program, for instance, that directly manipulates strings. Arikawa and his co-workers proposed elementary formal systems as a unifying framework for formal language learning.

In the present paper, we introduce advanced elementary formal systems (AEFSs), i.e., elementary formal systems which allow for the use of a certain kind of negation, which is nonmonotonic, in essence, and which is conceptually close to negation as failure.

We study the expressiveness of this approach by comparing certain AEFS definable language classes to the levels in the Chomsky hierarchy and to the language classes that are definable by EFSs that meet the same syntactical constraints.

Moreover, we investigate the learnability of the corresponding AEFS definable language classes in two major learning paradigms, namely in Gold’s model of learning in the limit and Valiant’s model of probably approximately correct learning. In particular, we show which learnability results achieved for EFSs extend to AEFSs and which do not.

\(©\) 2002 Elsevier Science B.V. All rights reserved.

**Keywords:** Formal language theory; Logic programming; Computational learning theory; Machine learning

1. Introduction and motivation

Elementary formal systems (EFSs) have been introduced by Smullyan [20] to develop his theory of recursive functions over strings. In Arikawa [2] and in a series of subsequent publications like [3–5,13,19,25,26], for example, Arikawa and

\(^*\text{Corresponding author.}
\)

\text{E-mail addresses:} \text{ Lange@dfki.de (S. Lange), grieser@informatik.tu-darmstadt.de (G. Grieser), jantke@dfki.de (K.P. Jantke).}

0304-3975/03/$-see front matter © 2002 Elsevier Science B.V. All rights reserved.

\text{PII: S}0304-3975(02)00418-8
his co-workers proposed elementary formal systems as a unifying framework for formal language learning.

EFSs are a kind of logic programs such as Prolog programs, for instance. EFSs directly manipulate non-empty strings over some underlying alphabet and can be used to describe formal languages. For instance, the EFS depicted in Fig. 1 describes the language that contains all non-empty strings of form $a^n b^n$. More formally speaking, if a ground atom $p(w)$ can be derived from the given rules, then the string $w$ has to be of form $a^n b^n$.

Arikawa and his co-workers (cf. e.g. [3,4]) used EFSs as a uniform framework to define acceptors for formal languages. In this context, they discussed the relation of certain EFS definable language classes to the standard levels in the classical Chomsky hierarchy. In addition, they have studied the learnability/non-learnability of EFS definable language classes in different learning paradigms, including Gold’s [7] model of learning in the limit as well as Valiant’s [24] model of probably approximately correct learning (cf. [3,4,13,19,26]). For instance, the results in [18,19] impressively show that EFSs provide an appropriate framework to prove that rich language classes are Gold-style learnable from only positive examples.

In the present paper, we follow the line of research of Arikawa and his co-workers. But in generalizing ordinary EFSs, we introduce so-called advanced elementary formal systems (AEFSs). In contrast to an EFS, an AEFS may additionally contain rules of the form $A \leftarrow \text{not } B_1$, where $A$ and $B_1$ are atoms and not stands for a certain kind of negation, which is non-monotonic, in essence, and which is conceptually close to negation as failure. Even this rather limited approach to use negation has its benefits in that it may seriously simplify the definition of formal languages. For instance, the rules in Fig. 2 define the language of all square-free strings. Formally speaking, a ground atom $p(w)$ can be derived only in case that the string $w$ is square-free.

---

**Fig. 1. An example EFS.**

1. $p(xy) \leftarrow q(x,y)$.
2. $q(a,b)$.
3. $q(ax,by) \leftarrow q(x,y)$.

**Fig. 2. An example AEFS.**

1. $p(x) \leftarrow \text{not } q(x)$.
2. $q(xx)$.
3. $q(xy) \leftarrow q(x)$.
4. $q(xy) \leftarrow q(y)$.

---

1 As usual, a string $w$ is square-free if it does not contain a non-empty substring of form $vv$. 
The work reported in the present paper mainly draws its motivation from ongoing research related to knowledge discovery and information extraction (IE) in the World Wide Web. Documents prepared for the Internet in HTML, in XML or in any other syntax have to be interpreted by browsers sitting anywhere in the World Wide Web. For this purpose, the documents do need to contain syntactic expressions which are controlling their interpretation, their visual appearance, and their interactive behaviour. While the document’s content is embedded into those syntactic expressions which are usually hidden from the user and which are obviously apart from the user’s interest, the user is typically interested in the information itself. Accordingly, the user deals exclusively with the desired contents, whereas a system for IE should deal with the syntax.

In a characteristic scenario of system-supported IE, the user is taking a source document and is highlighting representative pieces of information that are of interest. Now, it is left to the system to understand how the target information is wrapped into syntactic expressions and to learn a procedure (henceforth called wrapper) that allows for an extraction of this information (cf. e.g. [8,11,21,22]).

AEFSs seem to provide an appropriate framework to describe those extraction procedures that naturally comprises the approaches proposed in the IE community (cf. e.g. [11,23]).

For illustration, consider the table in Fig. 3 and its LATEX source in Fig. 4 which contains details about the first half-dozen of workshops on Algorithmic Learning Theory (ALT). The aim of the IE task is to extract all pairs \((y, c)\) that refer to the year \(y\) and the corresponding conference site \(c\) of a workshop in the ALT series that has proceedings co-edited by Arikawa. So, the pairs \((1990,\text{Tokyo})\) and \((1994,\text{Reinhardsbrunn})\) may serve as illustrating examples.

An AEFS that describes how the required information is wrapped into the LATEX source in Fig. 4 is depicted in Fig. 5:

The first rule can be interpreted as follows: A year \(y\) and the conference site \(c\) can be extracted from a LATEX source document \(d\) in case that (i) \(d\) matches the pattern

\[
\begin{array}{cccc}
  y & x_1 & x_2 & c \\
  \hline
  x_3 & & & \backslash x_3
\end{array}
\]

and (ii) the instantiations of the variables \(y, x_1, x_2,\) and \(c\) meet certain constraints. For example, the constraint \(h(x_1)\) states that the variable \(x_1\) can only be replaced by some string that contains the substring Arikawa. Further constraints like \(p(y)\) explicitly state which text segments are suited to be substituted for the variable \(y\), for instance. In this particular case, text segments that do not contain the substring \& are allowed. If a document \(d\) matches the pattern

\[
\begin{array}{cccc}
  y & x_1 & x_2 & c \\
  \hline
  x_3 & & & \backslash x_3
\end{array}
\]

and if all specified constraints are fulfilled, then the instantiations of the variables \(y\) and \(c\) yield the information required.

As the above example shows, the explicit use of logical negation seems to be quite useful, since it may help to describe wrappers in a natural way. In this particular case, the predicate \(p\) is used to guarantee that the specified wrapper does not allow for the extraction of pairs \((y, c)\) such that \(y\) and \(c\) belong to different rows in the table depicted in Fig. 3.

The focus of the present paper is twofold. On the one hand, we study the expressiveness of the proposed extension of EFSs by comparing certain AEFS definable language classes to the levels in the Chomsky hierarchy as well as to the language classes that
are definable by EFSs that meet the same syntactical constraints. This may help to better understand the strength of the proposed framework.

In the long term, we are interested in IE systems that automatically infer wrappers from examples. With respect to the illustrating example above, we are targeting at learning systems that are able to infer, for instance, the wrapper of Fig. 5 from the source document of Fig. 4 together with the two samples (1990,Tokyo) and (1994,Reinhardsbrunn). Therefore, on the other hand, we investigate the learnability of the corresponding AEFS definable language classes in Gold’s [8] model of learning in the limit and Valiant’s [24] model of probably approximately correct learning. In this context, we systematically discuss the question which learnability results achieved for EFSs lift to AEFSs and which do not.

2. Advanced elementary formal systems

AEFSs generalize Smullyan’s [20] elementary formal systems which he introduced to develop his theory of recursive functions over strings.
2.1. Preliminaries

By $\Sigma$ we denote any fixed finite alphabet. Let $\Sigma^+$ be the set of all non-empty words over $\Sigma$. Moreover, we let $\Sigma^n$ denote the set of all words in $\Sigma^+$ having length less than or equal to $n$, i.e., $\Sigma^n = \{ w \mid w \in \Sigma^+, |w| \leq n \}$. Let $a \in \Sigma$. Then, for all $n \geq 1$, $a^{n+1} = aa^n$, where, by convention, $a^1 = a$.

Any subset $L \subseteq \Sigma^+$ is called a language. By $\bar{L}$ we denote the complement of $L$, i.e., $\bar{L} = \Sigma^+ \setminus L$. Furthermore, let $\mathcal{L}$ be a language class. Then, we let $\mathcal{L}^n = \{ L \cap \Sigma^n \mid L \in \mathcal{L} \}$.

By $\mathcal{L}_{\text{reg}}$, $\mathcal{L}_{\text{cf}}$, $\mathcal{L}_{\text{cs}}$, and $\mathcal{L}_{\text{re}}$ we denote the class of all regular, context free, context sensitive, and recursively enumerable languages, respectively. These are the standard levels in the well-known Chomsky hierarchy (cf. e.g. [9]).

The following lemmata provide standard knowledge about context free languages (cf. e.g. [9]) that is helpful in proving Theorem 8.

**Lemma 1.** Let $L \subseteq \{ a \}^+$. Then, $L \in \mathcal{L}_{\text{cf}}$ if $L \in \mathcal{L}_{\text{reg}}$.

**Lemma 2.** Let $L \subseteq \Sigma^+$ be a context free language and let $\Sigma_0 \subseteq \Sigma$. Then, $L' = L \cap \Sigma_0^+$ constitutes a context free language.

### 2.2. Elementary formal systems

Next, we provide notions and notations that allow for a formal definition of ordinary EFSs.

Assume three mutually disjoint sets—a finite set $\Sigma$ of characters, a finite set $\Pi$ of predicate symbols, and an enumerable set $X$ of variables. We call every element in $(\Sigma \cup X)^+$ a pattern and every string in $\Sigma^+$ a ground pattern. For a pattern $\pi$, we let $v(\pi)$ be the set of variables in $\pi$. Furthermore, a pattern $\pi$ is said to be regular iff every variable occurs at most once in $\pi$.

Let $p \in \Pi$ be a predicate symbol of arity $n$ and let $\pi_1, \ldots, \pi_n$ be patterns. Let $A = p(\pi_1, \ldots, \pi_n)$. Then, $A$ is said to be an atomic formula (an atom, for short). $A$ is ground, if all the patterns $\pi_i$ are ground. Moreover, $v(A)$ denotes the set of all variables in $A$.

Let $A$ and $B_1, \ldots, B_n$ be atoms. Then, $r = A \leftarrow B_1, \ldots, B_n$ is a rule, $A$ is the head of $r$, and all the $B_i$ form the body of $r$. If all atoms in $r$ are ground, then $r$ is a ground rule. Moreover, if $n = 0$, then $r$ is called a fact. Sometimes, we write $A$ instead of $A \leftarrow$.

---

Fig. 5. Sample wrapper represented as hereditary AEFS.
Let $\sigma$ be a non-erasing substitution, i.e., a mapping from $X$ to $(\Sigma \cup X)^+$ such that, for almost all $x \in X$, $\sigma(x) = x$. For any pattern $\pi$, $\pi\sigma$ is the pattern which one obtains when applying $\sigma$ to $\pi$. Let $C = p(\pi_1, \ldots, \pi_n)$ be an atom and let $r = A \leftarrow B_1, \ldots, B_n$ be a rule. Then, we set $C\sigma = p(\pi_1\sigma, \ldots, \pi_n\sigma)$ and $r\sigma = A\sigma \leftarrow B_1\sigma, \ldots, B_n\sigma$. If $r\sigma$ is ground, then it is said to be a ground instance of $r$.

**Definition 1** (Arikawa et al. [5]). Let $\Sigma$, $\Pi$, and $X$ be fixed, and let $\Gamma$ be a finite set of rules over $\Sigma$, $\Pi$, and $X$. Then, $S = (\Sigma, \Pi, \Gamma)$ is said to be an EFS.

EFSs can be considered as particular logic programs without negation. There are two major differences: (i) patterns play the role of terms and (ii) unification has to be realized modulo the equational theory $E = \{\circ(x, \circ(y, z)) = \circ(\circ(x, y), z)\}$, where $\circ$ is interpreted as concatenation of patterns.

As for logic programs (cf. e.g. [12]), the semantics of an ordinary EFS $S$, denoted by $\text{Sem}_o(S)$, can be defined via the operator $T_S$ (see below). In the corresponding definition, we use the following notations. For any EFS $S = (\Sigma, \Pi, \Gamma)$, we let $B(S)$ denote the set of all well-formed ground atoms over $\Sigma$ and $\Pi$. Moreover, we let $G(S)$ denote the set of all ground instances of rules in $\Pi$.

**Definition 2.** Let $S$ be an EFS. Moreover, let and let $I \subseteq B(S)$. Then, we let $T_S(I) = I \cup \{A \leftarrow B_1, \ldots, B_n \mid B_i \in G(S) \text{ for some } B_1 \in I, \ldots, B_n \in I\}$.

Note that, by definition, the operator $T_S$ is embedding (i.e., $I \subseteq T_S(I)$ for all $I \subseteq B(S)$) and monotonic (i.e., $I \subseteq I'$ implies $T_S(I) \subseteq T_S(I')$ for all $I, I' \subseteq B(S)$).

As usual, we let $T_S^{n+1}(I) = T_S(T_S^n(I))$, where $T_S^0(I) = I$, by convention.

**Definition 3.** Let $S$ be an EFS. Then, we let $\text{Sem}_o(S) = \bigcup_{n \in \mathbb{N}} T_S^n(\emptyset)$.

In general, $\text{Sem}_o(S)$ is semi-decidable, but not decidable. However, as we will see below, $\text{Sem}_o(S)$ turns out to be decidable in case where $S$ meets several natural syntactical constraints.

Finally, by $\mathcal{EFS}$ we denote the collection of all EFSs.

### 2.3. Beyond elementary formal systems

Informally speaking, an AEFS is an EFS that may additionally contain rules of the form $A \leftarrow \text{not } B_1$, where $A$ and $B_1$ are atoms and $\text{not}$ stands for a certain kind of negation, which is nonmonotonic, in essence, and which is conceptually close to negation as failure. The underlying meaning is as follows. If, for instance, $A = p(x_1, \ldots, x_n)$ and $B_1 = q(x_1, \ldots, x_n)$, then the predicate $p$ succeeds iff the predicate $q$ fails.

However, taking the conceptual difficulties into consideration that occur when defining the semantics of logic programs with negation as failure (cf. e.g. [12]), AEFSs are constrained to meet several additional syntactic requirements (cf. Definition 4).
The requirements posed guarantee that, similarly to stratified logic programs (cf. e.g. [12]), the semantics of AEFSs can easily be described. Moreover, as a side-effect, it is guaranteed that AEFSs inherit some of the convenient properties of EFSs.

Before formally defining how AEFSs look like, we need some more notations. Let \( \Gamma \) be a set of rules (including rules of the form \( A \leftarrow \text{not} \ B_1 \)). Then, \( hp(\Gamma) \) denotes the set of predicate symbols that appear in the head of any rule in \( \Gamma \).

**Definition 4.** AEFSs and their semantics are inductively defined as follows.

1. An EFS \( S' \) is also an AEFS and its semantics is \( \text{Sem}(S')=\text{Sem}_o(S') \).
2. If \( S_1=(\Sigma, \Pi_1, I_1) \) and \( S_2=(\Sigma, \Pi_2, I_2) \) are AEFSs such that \( \Pi_1 \cap \Pi_2=\emptyset \), then \( S=(\Sigma, \Pi_1 \cup \Pi_2, I_1 \cup I_2) \) is an AEFS and its semantics is \( \text{Sem}(S)=\text{Sem}(S_1) \cup \text{Sem}(S_2) \).
3. If \( S_1=(\Sigma, \Pi_1, I_1) \) is an AEFS and \( p \notin \Pi_1 \) and \( q \in \Pi_1 \) are \( n \)-ary predicate symbols, then \( S=(\Sigma, \Pi_1 \cup \{p\}, I_1 \cup \{p(x_1,\ldots,x_n) \leftarrow \text{not} \ q(x_1,\ldots,x_n)\}) \) is an AEFS and its semantics is \( \text{Sem}(S)=\text{Sem}(S_1) \cup \{p(s_1,\ldots,s_n) \in B(S), q(s_1,\ldots,s_n) \notin \text{Sem}(S_1)\} \).
4. If \( S_1=(\Sigma, \Pi_1, I_1) \) is an AEFS and \( S'=(\Sigma, \Pi', I') \) is an EFS such that \( hp(\Gamma') \cap \Pi_1 =\emptyset \), then \( S=(\Sigma, \Pi' \cup \Pi_1, I' \cup I_1) \) is an AEFS and its semantics is \( \text{Sem}(S)=\bigcup_{n \in \mathbb{N}} T^n_S(\text{Sem}(S_1)) \).

Finally, by \( \mathcal{AEFS} \) we denote the collection of all AEFSs.

According to Definition 4, the same AEFS may be constructed either via (2) or (4). Since \( T_S \) is both embedding and monotonic, the semantics is the same in both cases. To see this, let \( S_1=(\Sigma, \Pi_1, I_1) \) be an EFS and let \( S_2=(\Sigma, \Pi_2, I_2) \) be an AEFS such that \( \Pi_1 \cap \Pi_2=\emptyset \). Then, (2) and (4), respectively, allows for the definition of the AEFS \( S=(\Sigma, \Pi_1 \cup \Pi_2, I_1 \cup I_2) \). By (2), \( \text{Sem}(S)=\text{Sem}(S_1) \cup \text{Sem}(S_2) \), while, by (4), \( \text{Sem}(S)=\bigcup_{n \in \mathbb{N}} T^n_S(\text{Sem}(S_2)) \). By definition, \( T^n_S(\text{Sem}(S_2))=\text{Sem}(S_2) \). Since \( \Pi_1 \cap \Pi_2 =\emptyset \), we directly obtain \( T^n_S(\text{Sem}(S_2))=T^n_S(\emptyset) \cup \text{Sem}(S_2) \) for all \( n \in \mathbb{N} \). Therefore, we may conclude that \( \text{Sem}(S_1) \cup \text{Sem}(S_2)=\bigcup_{n \in \mathbb{N}} T^n_S(\text{Sem}(S_2)) \).

### 2.4. Using AEFS for defining formal languages

In the following, we show how AEFSs can be used to describe formal languages and relate the resulting language classes to the language classes of the classical Chomsky hierarchy (cf. [9]).

**Definition 5.** Let \( S=(\Sigma, \Pi, \Gamma) \) be an AEFS and let \( p \in \Pi \) be a unary predicate symbol. Then, we let \( L(S, p)=\{s \mid (\exists s) p(s) \subseteq \text{Sem}(S)\} \).

Furthermore, a language \( L \subseteq \Sigma^+ \) is said to be AEFS definable iff there are a super-set \( \Sigma_0 \) of \( \Sigma \), an AEFS \( S=(\Sigma_0, \Pi, \Gamma) \), and a unary predicate symbol \( p \in \Pi \) such that \( L=L(S, p) \).

Intuitively speaking, \( L(S, p) \) is the language which the AEFS \( S \) defines via the unary predicate symbol \( p \).
Definition 6. Let \( \mathcal{M} \subseteq \mathcal{AEFS} \) and let \( k \in \mathbb{N} \). Then, \( \mathcal{L}(\mathcal{M}) \) is the set of all languages that are definable by AEFSs in \( \mathcal{M} \). Moreover, \( \mathcal{L}(\mathcal{M}(k)) \) is the set of all languages that are definable by AEFSs in \( \mathcal{M} \) that have at most \( k \) rules.

For example, \( \mathcal{L}(\mathcal{AEFS}(2)) \) is the class of all languages that are definable by unconstrained AEFSs that consist of at most 2 rules.

Our first result puts the expressive power of AEFSs into the right perspective.

Theorem 1. \( \mathcal{L}_{re} \subseteq \mathcal{L}(\mathcal{AEFS}) \).

Proof. Since, by definition, \( \mathcal{L}(\mathcal{EFS}) \subseteq \mathcal{L}(\mathcal{AEFS}) \), and \( \mathcal{L}_{re} \subseteq \mathcal{L}(\mathcal{EFS}) \) (cf. e.g. [5]), we get \( \mathcal{L}_{re} \subseteq \mathcal{L}(\mathcal{AEFS}) \). Since there are languages \( L \in \mathcal{L}_{re} \) that have a complement which is not recursively enumerable (cf. [17]), \( \mathcal{L}(\mathcal{AEFS}) \setminus \mathcal{L}_{re} \neq \emptyset \) is an immediate consequence of Theorem 2. \( \square \)

Moreover, the following closedness properties can be shown.

Theorem 2. \( \mathcal{L}(\mathcal{AEFS}) \) is closed under the operations union, intersection, and complement.

Proof. Let \( L_1, L_2 \in \mathcal{L}(\mathcal{AEFS}) \) be given. Hence, there are AEFSs \( S_1 = (\Sigma, \Pi_1, \Gamma_1) \) and \( S_2 = (\Sigma, \Pi_2, \Gamma_2) \) as well as unary predicate symbols \( p_1 \in \Pi_1 \) and \( p_2 \in \Pi_2 \) such that \( L(S_1, p_1) = L_1 \) and \( L(S_2, p_2) = L_2 \). Without loss of generality, we may assume that \( \Pi_1 \cap \Pi_2 = \emptyset \).

First, we show that \( L_1 \in \mathcal{L}(\mathcal{AEFS}) \). Let \( q \notin \Pi_1 \) be any unary predicate symbol and let \( S = (\Sigma, \Pi, \Gamma) \) with \( \Pi = \Pi_1 \cup \{ q \} \) and \( \Gamma = \Gamma_1 \cup \{ q(x) \leftarrow \neg p_1(x) \} \). By Definition 4, \( S \) is an AEFS that meets \( L(S, q) = L(S_1, p_1) = L_1 \).

Next, we show that \( L_1 \cup L_2 \in \mathcal{L}(\mathcal{AEFS}) \). By Definition 4, \( S' = (\Sigma, \Pi', \Gamma') \) with \( \Pi' = \Pi_1 \cup \Pi_2 \) and \( \Gamma' = \Gamma_1 \cup \Gamma_2 \) is an AEFS. Moreover, we have \( L(S', p_1) = L(S_1, p_1) \) as well as \( L(S', p_2) = L(S_2, p_2) \). Now, let \( q \notin \Pi' \) and let \( S = (\Sigma, \Pi, \Gamma) \) with \( \Pi = \Pi' \cup \{ q \} \) and \( \Gamma = \Gamma' \cup \{ q(x) \leftarrow p_1(x), q(x) \leftarrow p_2(x) \} \). According to Definition 4, \( S \) is an AEFS that meets \( L(S, q) = L(S', p_1) \cup L(S', p_2) = L_1 \cup L_2 \).

Finally, since \( L_1 \cap L_2 = L_1 \cup L_2 \), we may conclude that \( L_1 \cap L_2 \in \mathcal{L}(\mathcal{AEFS}) \). \( \square \)

To elaborate a more accurate picture, similarly to Arikawa et al. [5], we next introduce several constraints on the structure of the rules an AEFS may contain.

Let \( r \) be a rule of form \( A \leftarrow B_1, \ldots, B_n \). Then, \( r \) is said to be variable-bounded iff, for all \( i \leq n, v(B_i) \subseteq v(A) \). Moreover, \( r \) is said to be length-bounded iff, for all substitutions \( \sigma, |A\sigma| \geq |B_1\sigma| + \cdots + |B_n\sigma| \). Clearly, if \( r \) is length-bounded, then \( r \) is also variable-bounded. Note that, in general, the opposite does not hold.

Moreover, let \( r \) be a rule of form \( p(\pi) \leftarrow q_1(x_1), \ldots, q_n(x_n) \), where \( x_1, \ldots, x_n \) are mutually distinct variables and \( \pi \) is a regular pattern which contains exactly the variables \( x_1, \ldots, x_n \). Then, \( r \) is said to be regular.

In addition, every rule of form \( p(x_1, \ldots, x_n) \leftarrow \neg q(x_1, \ldots, x_n) \) is both variable-bounded and length-bounded. Moreover, every rule of form \( p(x) \leftarrow \neg q(x) \) is regular.
Definition 7. Let $S = (\Sigma, \Pi, \Gamma)$ be an AEFS. Then, $S$ is said to be
(1) variable-bounded iff all $r \in \Gamma$ are variable-bounded,
(2) length-bounded iff all $r \in \Gamma$ are length-bounded, and
(3) regular iff all $r \in \Gamma$ are regular.

By $\text{vb-AEFS}$ (vb-EFS), $\text{lb-AEFS}$ (lb-EFS), and $\text{reg-AEFS}$ (reg-EFS) we denote the collection of all AEFSs (EFSs) that are variable-bounded, length-bounded, and regular, respectively.

The following three theorems illuminate the expressive power of ordinary EFSs.

Theorem 3 (Arikawa et al. [5]).
(1) $L(\text{vb-EFS}) \subseteq L_{\text{re}}$.
(2) For any $L \in L_{\text{re}}$, there is an $L' \in L(\text{vb-EFS})$ such that $L = L' \cap \Sigma^+$.

If $\Sigma$ contains at least two symbols, assertion (2) rewrites to $L_{\text{re}} \subseteq L(\text{vb-EFS})$ (cf. [5]).

Theorem 4 (Arikawa et al. [5]).
(1) $L(\text{lb-EFS}) \subseteq L_{\text{cs}}$.
(2) For any $L \in L_{\text{cs}}$, there is an $L' \in L(\text{lb-EFS})$ such that $L = L' \cap \Sigma^+$.

Theorem 5 (Arikawa et al. [5]). $L(\text{reg-EFS}) = L_{\text{cf}}$.

Concerning AEFSs the situation changes slightly. This is mainly caused by the fact that variable-bounded, length-bounded, and regular AEFSs are closed under intersection.

Theorem 6. $L(\text{vb-AEFS}), L(\text{lb-AEFS})$, and $L(\text{reg-AEFS})$ are closed under the operations union, intersection, and complement.

Proof. The same argumentation as in the demonstration of Theorem 2 justifies the theorem on hand. To see this, note that all predicate symbols that have been used are unary ones. Moreover, all rules that have to be added to the original AEFS are variable-bounded, length-bounded, and regular. □

For AEFSs, Theorems 3 and 4 rewrites as follows.

Theorem 7. (1) $L_{\text{re}} \subseteq L(\text{vb-AEFS})$.
(2) $L(\text{lb-AEFS}) = L_{\text{cs}}$.

Proof. First, we show (1). Applying Theorem 6, one sees that assertion (2) of Theorem 3 rewrites to $L_{\text{re}} \subseteq L(\text{vb-AEFS})$. Next, $L(\text{vb-AEFS}) \setminus L_{\text{re}} \neq \emptyset$ can be shown by applying the same arguments as in the demonstration of Theorem 1.

Second, we verify (2). Again, applying Theorem 6, one directly sees that assertion (2) of Theorem 4 rewrites to $L_{\text{cs}} \subseteq L(\text{lb-AEFS})$. Moreover, by definition, for any $L \in L(\text{lb-AEFS})$, there are languages $L_0, \ldots, L_n \in L(\text{lb-EFS})$ such that $L$ can be defined by applying the operations union and intersection to these languages. Since
$L(lb-EFS) \subseteq L_{cs}$ and since $L_{cs}$ is closed with respect to the operations union and intersection (cf. e.g. [9]), we may conclude that $L(lb-AEFS) \subseteq L_{cs}$. □

In our opinion, assertion (2) of Theorem 7 witnesses the naturalness of our approach to extend EFSs to AEFSs. In contrast to assertion (2) of Theorem 4, there is no need to use auxiliary characters in the terminal alphabet.

**Theorem 8.** $L_{cf} \subset L(\text{reg-AEFS}) \subset L_{cs}$.

**Proof.** First, $L_{cf} \subset L(\text{reg-AEFS}) \subset L_{cs}$ follows immediately from Theorems 5 and 7.

Second, $L_{cf} \subset L(\text{reg-AEFS})$ follows from the fact that $L(\text{reg-AEFS})$ is closed under intersection (cf. Theorem 6), while $L_{cf}$ is not (cf. e.g. [9]).

Third, we show that $L_{cs} \setminus L(\text{reg-AEFS}) \neq \emptyset$. Let $L \subseteq \{a\}^+$ with $L \in L_{cs} \setminus L_{cf}$ (cf. e.g. [9], for some illustrating examples). We claim that $L \notin L(\text{reg-AEFS})$. Suppose the contrary, i.e., $L \in L(\text{reg-AEFS})$. By definition, there are languages $L_0, \ldots, L_n \in L(\text{reg-AEFS})$ such that $L$ can be defined by applying the operations union and intersection to these languages. Let $i \leq n$. By Theorem 5, $L_i \in L_{cf}$. Moreover, let $L'_i = L_i \cap \{a\}^+$. By Lemma 2, $L'_i \in L_{cf}$, and thus, by Lemma 1, $L'_i \in L_{reg}$. Finally, one easily sees that $L$ can also be defined by applying the operations union and intersection to the languages $L'_0, \ldots, L'_n$. Finally, since $L_{reg}$ is closed with respect to the operations union and intersection, we may therefore conclude that $L \in L_{reg}$ which in turn yields $L \in L_{cf}$, a contradiction. □

### 3. Learning of AEFSs

#### 3.1. Notions and notations

First, we briefly review the necessary basic concepts concerning Gold’s [7] model of learning in the limit. We refer the reader to the survey papers Angluin and Smith [1] and Zeugmann and Lange [27] as well as to the textbooks Osherson et al. [16] and Jain et al. [19] which contain all missing details.

There are several ways to present information about formal languages to be learned. The basic approaches are defined via the key concept text and informant, respectively. Let $L$ be the target language. A **text for** $L$ is just any sequence of words labelled ‘+’ that exhausts $L$. An **informant for** $L$ is any sequence of words labelled alternatively either by ‘+’ or ‘−’ such that all the words labelled by ‘+’ form a text for $L$, while the remaining words labelled by ‘−’ constitute a text for $L$. Sometimes, labelled words are called **examples**.

As in Gold [7], we define an **inductive inference machine** (abbr. IIM) to be an algorithmic device working as follows: The IIM takes as its input larger and larger initial segments of a text (an informant). After processing an initial segment $\sigma$, the IIM outputs a hypothesis $M(\sigma)$, i.e., a number encoding a certain computer program. More formally, an IIM maps finite sequences of elements from $\Sigma^+ \times \{+,−\}$ into numbers in $\mathbb{N}$.
The numbers output by an IIM are interpreted with respect to a suitably chosen hypothesis space $H=(h_j)_{j \in \mathbb{N}}$. When an IIM outputs some number $j$, we interpret it to mean that the machine is hypothesizing $h_j$.

Now, let $L$ be a language class, let $L$ be a language, and let $H=(h_j)_{j \in \mathbb{N}}$ be a hypothesis space. An IIM $M$ \textit{LinTtxt$_H$} ($\text{LimInf}_H$)-learns $L$ iff, for every text $t$ for $L$ (for every informant $i$ for $L$), there exists a $j \in \mathbb{N}$ such that $h_j=L$, and moreover $M$ almost always outputs the hypothesis $j$ when fed the text $t$ (the informant $i$). Furthermore, an IIM $M$ \textit{LinTtxt$_H$} ($\text{LimInf}_H$)-learns $L$ iff, for every $L \in L'$, $M$ \textit{LinTtxt$_H$} ($\text{LimInf}_H$)-learns $L$. In addition, we write $L \in \text{LimTtxt}$ ($L \in \text{LimInf}$) provided there are a hypothesis space $H$ and an IIM $M$ that \textit{LinTtxt$_H$} ($\text{LimInf}_H$)-learns $L$.

Next, we focus our attention on Valiant’s [24] model of probably approximately correct learning (PAC model, for short; see also the textbook Natarajan [15] for further details). In contrast to Gold’s [7] model, the focus is now on learning algorithms that, based on randomly chosen positive and negative examples, find, fast and with high probability, a sufficiently good approximation of the target language.

To give a precise definition of the PAC model, we need the following notions and notations.

We use a finite alphabet $A$ for representing languages. A \textit{representation} for a language class $L$ is a function $R: L \rightarrow \varphi(A^*)$ such that, for all distinct languages $L, L' \in L$, $R(L) \neq \emptyset$ and $R(L) \cap R(L') = \emptyset$. Let $L \in L$. Then, $R(L)$ is the set of representations for $L$ and $\ell_{\min}(L, R)$ is the length of the shortest string in $R(L)$. Moreover, let $T$ be a set of examples. As usual, a language $L$ is said to be \textit{consistent} with $T$ iff, for all $(x,+), x \in L$ and, for all $(x,-), x \notin L$. Then, $\ell_{\min}(T, R)$ is the length of a shortest string in $\bigcup_{L \in L'} R(L)$ where $L'$ is the subset of all languages in $L$ that are consistent with $T$.

**Definition 8** (Valiant [24]). A language class $L$ is polynomial-time PAC learnable in a representation $R$ iff there exists a learning algorithm $\mathcal{A}$ such that

1. $\mathcal{A}$ takes a sequence of examples as input and runs in polynomial time with respect to the length of the input and
2. there exists a polynomial $q(\cdot, \cdot, \cdot, \cdot)$ such that, for any $L \in L$, any $n \in \mathbb{N}$, any $s \geq 1$, any reals $e,d$ with $0 < e,d < 1$, and any probability distribution $Pr$ on $\Sigma^n$, if $\mathcal{A}$ takes $q(1/e, 1/d, n, s)$ examples, which are generated randomly according to $Pr$, then $\mathcal{A}$ outputs, with probability at least $1-d$, a hypothesis $h \in R$ with $Pr(w \in ((L \setminus h) \cup (h \setminus L))) < e$, when $\ell_{\min}(L, R) \leq s$ is satisfied.

We complete this section by providing some more notions and notations that are of relevance when proving some of the learnability/non-learnability results presented below.

**Definition 9.** A pair $(S, p)$ consisting of an AEFS $S=(\Sigma, \Pi, \Gamma)$ and a unary predicate symbol $p \in \Pi$ is said to be reduced with respect to a set $T$ of examples iff $L(S, p)$ is consistent with $T$ and, for any $S'=(\Sigma, \Pi, \Gamma')$ with $\Gamma' \subset \Gamma$, $L(S', p)$ is not consistent with $T$. 
The following notion adopts one of the key concepts in [18], where it has been shown that, for classes of elementary formal systems, bounded finite thickness implies that the corresponding language class is learnable in the limit from only positive examples.

Definition 10 (Shinohara [18]). Let $M \subseteq \mathcal{EFS}$. $M$ is said to have bounded finite thickness iff, for all $w \in \Sigma^+$, there are at most finitely many EFS $S = (\Sigma, \Pi, \Gamma)$ in $M$ such that, for some unary predicate $p \in \Pi$, $(S, p)$ is reduced with respect to $T = \{(w, +)\}$.

Finally, we define the notion polynomial dimension which is one of the key notions when studying the learnability of formal languages in the PAC model.

Definition 11 (Natarajan [14]). Let $L$ be a language class. $L$ has polynomial dimension iff there is a polynomial $d(\cdot)$ such that, for all $n \in \mathbb{N}$, $\log_2 |L_n| \leq d(n)$.

3.2. Gold-style learning

The following theorem summarizes the known learnability results for EFSs. Recall that, by definition, $L(lb-\mathcal{EFS}(k))$ is the collection of all languages that are definable by length-bounded EFSs that consist of at most $k$ rules.

Theorem 9 (Gold [8] and Shinohara [19]). (1) $L(lb-\mathcal{EFS}) \in \text{LimInf}$.  
(2) $L(lb-\mathcal{EFS}) \notin \text{LimTxt}$.  
(3) For all $k \in \mathbb{N}$, $L(lb-\mathcal{EFS}(k)) \in \text{LimTxt}$.

Having in mind that $L(lb-\mathcal{EFS}) = L(lb-\mathcal{AEFS})$, we may directly conclude:

Corollary 1. (1) $L(lb-\mathcal{AEFS}) \in \text{LimInf}$.  
(2) $L(lb-\mathcal{AEFS}) \notin \text{LimTxt}$.

The next theorem points to a major difference concerning the learnability of EFSs and AEFSs, respectively.

Theorem 10. (1) $L(lb-\mathcal{AEFS}(1)) \in \text{LimTxt}$.  
(2) For all $k \geq 2$, $L(lb-\mathcal{AEFS}(k)) \notin \text{LimTxt}$.

Proof. By definition, $L(lb-\mathcal{AEFS}(1)) = L(lb-\mathcal{EFS}(1))$, and thus (1) follows from Theorem 9.

Next, let $k = 2$. Let $\Sigma = \{a\}$ and consider the family $L_{sf} = \{L_i\}_{i \in \mathbb{N}}$ such that $L_0 = \{a^n | 1 \leq n \}$ and $L_{i+1} = \{a^n | 1 \leq n \leq i+1 \}$. $L_{sf}$ can be defined via the family of regular AEFSs $(S_i = (\Sigma, \Pi, \Gamma_i))_{i \in \mathbb{N}}$ with $\Pi = \{p, q\}$, $\Gamma_0 = \{p(a), p(ax) \rightarrow p(x)\}$, $\Gamma_i = \{q(a^n), p(x) \rightarrow \text{not } q(x)\}$ for all $i \geq 1$. Obviously, for every $i \in \mathbb{N}$, $L(S_i, p) = L_i$. On the other hand, it is well-known that $L_{sf} \notin \text{LimTxt}$ (cf. e.g. [27]), and therefore we are done. □
3.3. Probably approximately correct learning

In [3,13], the polynomial-time PAC learnability of several language classes that are definable by EFSs has been studied. It has been shown that even quite simple classes are not polynomial-time PAC learnable—for instance, the class of all regular pattern languages. However, if one bounds the number of variables that may occur in the defining patterns, regular pattern languages become polynomial-time PAC learnable. Moreover, by putting further constraints on the rules that can be used to define EFSs, positive results for even larger EFS definable language classes have been achieved (cf. [3,13]). The relevant technicalities are as follows.

A rule of form \( p(\pi_1, \ldots, \pi_n) \leftarrow p_1(\tau_{1_1}, \ldots, \tau_{1_{t_1}}), \ldots, p_m(\tau_{m_{i_m-1}}, \ldots, \tau_{m_{i_m}}) \) is said to be hereditary iff, for every \( j = 1, \ldots, t_m \), the pattern \( \tau_j \) is a subword of some pattern \( \pi_i \). Moreover, any rule of form \( p(x_1, \ldots, x_n) \leftarrow \text{not } q(x_1, \ldots, x_n) \) is a hereditary one, since it obviously meets the syntactical constraints stated above. Note that, by definition, every hereditary rule is variable-bounded.

**Definition 12.** Let \( S = (\Sigma, \Pi, \Gamma) \) be an AEFS. Then, \( S \) is said to be hereditary iff all \( r \in \Gamma \) are hereditary. By \( h\text{-AEFS} \) (h-EFS) we denote the collection of all hereditary AEFSs (EFSs).

In contrast to the general case (cf. Definition 5), hereditary AEFS have the following nice feature. Let \( L \subseteq \Sigma^+ \) with \( L \in \mathcal{L}(\text{reg-EFS}) \). Then, there is a hereditary AEFS for \( L \) consisting only of rules that uses exclusively characters from \( \Sigma \).

**Definition 13.** Let \( m, k, t, r \in \mathbb{N} \). By \( h\text{-AEFS}(m, k, t, r) \) (h-EFS \((m, k, t, r)\)) we denote the collection of all hereditary AEFSs (EFSs) \( S \) that satisfy (1)–(4), where

1. \( S \) contains at most \( m \) rules.
2. the number of variable occurrences in the head of every rule in \( S \) is at most \( k \).
3. the number of atoms in the body of every rule in \( S \) is at most \( t \).
4. the arity of each predicate symbol in \( S \) is at most \( r \).

Taking into consideration that \( \mathcal{L} \text{(reg-EFS)} = \mathcal{L}_{ct} \) (cf. Theorem 5), one easily sees that \( \mathcal{L} \text{(reg-EFS)} \subseteq \bigcup_{m \in \mathbb{N}} \mathcal{L} \text{(h-EFS}(m, 2, 1, 2)) \) (cf. [3]). Similarly, it can be verified that \( \mathcal{L} \text{(reg-AEFS)} \subseteq \bigcup_{m \in \mathbb{N}} \mathcal{L} \text{(h-AEFS}(m, 2, 1, 2)) \). Hence, hereditary EFSs resp. AEFSs are much more expressive than it might seem.

For hereditary EFSs, the following learnability result is known.

**Theorem 11** (Miyano et al. [13]). Let \( m, k, t, r \in \mathbb{N} \). Then, the class \( \mathcal{L} \text{(h-EFS}(m, k, t, r)) \) is polynomial-time PAC learnable.

As the results in [13] impressively show, it is inevitable to \textit{a priori} bound all the defining parameters. In other words, none of the resulting language classes is

\(^2\)That is, the class of all languages that are definable by an EFS that consists of exactly one rule of form \( p(\pi) \), where \( \pi \) is a regular pattern.
polynomial-time PAC learnable, if at least one of the parameters involved may arbitrarily grow.

Next, we turn our attention to study the learnability of language classes that are definable by hereditary AEFSs.

Our first result demonstrates that hereditary AEFSs are more expressive than hereditary EFSs.

**Theorem 12.** \( L(h\text{-AEFS}(2,1,1,1)) \setminus \bigcup_{m,k,t,r \in \mathbb{N}} L(h\text{-EFS}(m,k,t,r)) \neq \emptyset \).

**Proof.** Consider the language family \( \mathcal{L}_{sf} = \{ L_i \}_{i \in \mathbb{N}} \) such that \( L_0 = \{ a^n \mid 1 \leq n \} \) and \( L_{i+1} = \{ a^n \mid 1 \leq n \leq i+1 \} \). Having a closer look at the demonstration of Theorem 10, one directly sees that \( \mathcal{L}_{sf} \in L(h\text{-AEFS}(2,1,1,1)) \).

We claim that \( \mathcal{L}_{sf} \) witnesses the stated separation. Suppose to the contrary that there are \( m,k,t,r \in \mathbb{N} \) such that \( \mathcal{L}_{sf} \in L(h\text{-EFS}(m,k,t,r)) \). Since \( \mathcal{L}_{sf} \in \text{LimTxt} \) (cf. e.g. [27]), this directly implies \( L(h\text{-EFS}(m,k,t,r)) \notin \text{LimTxt} \). However, by combining results from Shinohara [18] and Miyano et al. [13], it can be shown that \( L(h\text{-EFS}(m,k,t,r)) \in \text{LimTxt} \), a contradiction. The relevant details are as follows: It has been shown that, for every \( m,k,t,r \in \mathbb{N} \), \( L(h\text{-EFS}(m,k,t,r)) \) has polynomial dimension (cf. [13]; see also Lemma 4 in the demonstration of Theorem 13 below).

Moreover, every EFS definable language class with polynomial dimension has bounded finite thickness which in turn implies that this language class is \( \text{LimTxt} \)-identifiable (cf. [18]).

Surprisingly, Theorem 11 remains valid in case that one considers hereditary AEFSs instead of EFSs. This nicely contrasts the fact that, in Gold’s [7] model, AEFS definable language classes may become harder to learn than EFS definable ones, although they are supposed to meet the same syntactical constraints (cf. Theorems 9 and 10). Moreover, having Theorem 12 in mind, the next theorem establishes the polynomial-time PAC learnability of a language class that properly comprises the class in [13].

**Theorem 13.** Let \( m,k,t,r \in \mathbb{N} \). Then, the class \( L(h\text{-AEFS}(m,k,t,r)) \) is polynomial-time PAC learnable.

**Proof.** Let \( m,k,t,r \in \mathbb{N} \), let \( \mathcal{L} = L(h\text{-AEFS}(m,k,t,r)) \), and let \( R \) be a mapping that assigns AEFSs in \( h\text{-AEFS}(m,k,t,r) \) to languages in \( L \). Applying results from Blumer et al. [6] and Natarajan [14], it suffices to show:

1. \( \mathcal{L} \) is of polynomial dimension.
2. There is a polynomial-time finder for \( R \), i.e., there exists a polynomial-time algorithm that, given a finite set \( T \) of examples for any \( L \in \mathcal{L} \), computes an AEFS \( S \in h\text{-AEFS}(m,k,t,r) \) that is consistent with \( T \).

---

\(^3\)Note that, for AEFS definable language classes, an analogue implication does not hold. This is caused by the fact that the entailment relation for AEFSs does not meet the monotonicity principle of classical logics.
The following series of lemmata proves that (1) and (2) are indeed fulfilled. Lemma 3 is needed to show (1), while Lemma 3 is used in order to verify (2). □

**Lemma 3.** Let $T$ be a set of examples over $\Sigma$. Furthermore, let $(S, p)$ be a pair consisting of a hereditary AEFS $S = (\Sigma, \Pi, \Gamma)$ and a unary predicate symbol $p \in \Pi$. If $(S, p)$ is reduced with respect to $T$, then for each rule $q_0(\pi_1^0, \ldots, \pi_n^0) \leftarrow q_1(\pi_1^1, \ldots, \pi_n^1), \ldots, q_r(\pi_j^r, \ldots, \pi_n^r)$ in $\Gamma$ there exists a substitution $\sigma$ such that all the $\pi_i^j \sigma$ are subwords of some labelled word from $T$.

**Proof.** Assume the contrary. Let $T$ be a set of examples over $\Sigma$, let $(S, p)$ be a pair consisting of a hereditary AEFS $S = (\Sigma, \Pi, \Gamma)$ and a unary predicate symbol $p \in \Pi$ such that $(S, p)$ is reduced with respect to $T$. Moreover, let $r = q_0(\pi_1^0, \ldots, \pi_n^0) \leftarrow q_1(\pi_1^1, \ldots, \pi_n^1), \ldots, q_r(\pi_j^r, \ldots, \pi_n^r)$ be a rule in $\Gamma$ that violates the assertions stated in Lemma 3.

We claim that $L(S', p)$ with $S' = (\Sigma, \Pi, \Gamma')$ is also consistent with $T$, where $\Gamma' = \Gamma \setminus \{r\}$. To see this, assume the contrary.

*Case 1:* There is a word $w$ such that $(w, +) \in T$ and $w \notin L(S', p)$.

Hence, during the derivation of $p(w)$, a ground instance $r\sigma$ of rule $r$ has to be used. Since $S$ is hereditary, each $\pi_j^0 \sigma, \ldots, \pi_n^0 \sigma$ is a subword of $w$. Consequently, this implies that all $\pi_j^0 \sigma$ are subwords of $w$, contradicting our assumption.

*Case 2:* There is a word $w$ such that $(w, -) \in T$ and $w \notin L(S', p)$.

Hence, there must be an atom $p'(w_1, \ldots, w_{r'})$ that is used when deriving $p(w)$ such that (i) $p'(w_1, \ldots, w_{r'}) \in \text{Sem}(S')$, (ii) $p'(w_1, \ldots, w_{r'}) \notin \text{Sem}(S)$, and (iii) there is a rule $p'(x_1, \ldots, x_{r'}) \leftarrow \text{not } q'(x_1, \ldots, x_{r'})$ in $\Gamma'$ such that $q'(w_1, \ldots, w_{r'}) \in \text{Sem}(S)$ and $q'(w_1, \ldots, w_{r'}) \notin \text{Sem}(S')$. Since $S$ is hereditary, all $w_1, \ldots, w_{r'}$ are subwords of $w$. Now, analogously to Case 1, during the derivation of $q'(w_1, \ldots, w_{r'})$ according to the rules in $S$, a ground instance $r\sigma$ of rule $r$ has to be used. As argued above, all the $\pi_j^0 \sigma$ are subwords of the words $w_1, \ldots, w_{r'}$, and therefore they are subwords of $w$, too. Since $(w, -) \in T$, this contradicts our assumption.

Summing up, $L(S', p)$ must be consistent with $T$. Hence, $S$ is not reduced with respect to $T$, a contradiction, and thus Lemma 3 follows. □

**Lemma 4.** For any $m, k, t, r \in \mathbb{N}$, the class $L(h-AEFS(m, k, t, r))$ has polynomial dimension.

**Proof.** Let $m, k, t, r \in \mathbb{N}$ be fixed. We estimate the cardinality of the language class $L(h-AEFS(m, k, t, r))^n$ in dependence on $n$.

Let $(S, p)$ be a pair of a hereditary AEFS $S = (\Sigma, \Pi, \Gamma) \in h-AEFS(m, k, t, r)$ and a unary predicate symbol $p \in \Pi$. Since $\Gamma$ contains at most $m$ rules, we may assume that $|\Pi| \leq m$. Furthermore, we may assume that $(S, p)$ is reduced with respect to some finite set of examples $T \subseteq \Sigma^n \times \{+, -\}$.

By definition, each rule in $\Gamma$ is either of form (i) $A \leftarrow B_1, \ldots, B_n$ or of form (ii) $A' \leftarrow \text{not } B'_1$, where $A' = p'(x_1, \ldots, x_j)$ and $B'_1 = q'(x_1, \ldots, x_j)$ for some $p', q' \in \Pi$ and

---

4 We abstain from formally defining the term derivation, since an intuitive understanding shall suffice. For the missing details, the interested reader is referred to Arikawa et al. [5], for instance.
variables $x_1, \ldots, x_j$. Because of Lemma 3, the same counting arguments as in [13] can be invoked to show that there at most $O(2^{an})$ rules of form (i). Moreover, as a simple calculation shows, there are $O((2mr^2)^2)$ rules of form (ii) (which does not depend on $n$). Consequently, there are at most $O(2^{an})$ rules that can be used when defining an AEFS in $h-\text{AEFS}(m,k,t,r)$, and thus there are at most $O(2^{2n})$ hereditary AEFS with at most $m$ rules that have to be considered when estimating the cardinality of the class $\mathcal{L}(h-\text{AEFS}(m,k,t,r))$. Hence, the class $\mathcal{L}(h-\text{AEFS}(m,k,t,r))$ has polynomial dimension, and thus Lemma 4 follows.

Lemma 5. For any $m,k,t,r \in \mathbb{N}$, any $S \in h-\text{AEFS}(m,k,t,r)$, and any $w \in \Sigma^+$, it can be decided in polynomial-time whether or not $w$ belongs to the language defined by $S$.

Proof. Let $m,k,t,r \in \mathbb{N}$, $S = (\Sigma, \Pi, \Gamma) \in h-\text{AEFS}(m,k,t,r)$, $w \in \Sigma^+$, and a unary predicate symbol $p \in \Pi$ be given.

Let $G(w)$ be the set of all ground facts $q'(w_1, \ldots, w_r)$ with $q' \in \Pi$ and subwords $w_1, \ldots, w_r$ of $w$. In a first step, we define a polynomial-time algorithm $A$ that, given $w$, outputs the set $A(w) = \text{Sem}(S) \cap G(w)$. In order to decide whether or not $w \in L(S, p)$, it suffices to check whether or not $p(w) \in A(w)$. Since there are at most $O(m|w|2r)$ elements in $G(w)$, the second step can easily be performed in polynomial time.

In order to define the required algorithm $A$, we distinguish the following cases.

Case 1: $S$ is defined according to item (1) of Definition 4.

Hence, $S \in h-\text{AEFS}(m,k,t,r)$. In [13], it has been shown that there is a polynomial-time decision procedure that, given any $w' \in \Sigma^+$, decides whether or not $w' \in L(S, p)$. Again, since there are at most $O(m|w|2r)$ elements in $G(w)$, it is not hard to define the required algorithm $A$ based on the polynomial-time decision procedure from Miyano et al. [13].

Case 2: $S$ is defined according to item (2) of Definition 4.

Let $S_1 = (\Sigma, \Pi_1, \Gamma_1)$ and $S_2 = (\Sigma, \Pi_2, \Gamma_2)$ be the AEFSs used to define $S$ according to Definition 4, item (2). Assume that there are corresponding algorithms $A_1$ and $A_2$ for $S_1$ and $S_2$, respectively. Since $\Pi_1 \cap \Pi_2 = \emptyset$, it suffices to define a polynomial-time algorithm $A$ that meets $A(w) = A_1(w) \cup A_2(w)$. This can easily be done, since $A_1$ and $A_2$ are given.

Case 3: $S$ is defined according to item (3) of Definition 4.

Let $S_1 = (\Sigma, \Pi_1, \Gamma_1)$ be the AEFS and $p', q'$ be the predicate symbols that have been used to define $S$ according to Definition 4, item (3). Hence, $S$ contains the additional rule $p'(x_1, \ldots, x_j) \leftarrow \text{not } q'(x_1, \ldots, x_j)$. Assume that there is a corresponding algorithm $A_1$ for $S_1$. It suffices to define a polynomial-time algorithm $A$ such that $A(w) = A_1(w) \cup \{ p'(w_1, \ldots, w_j) \mid p'(w_1, \ldots, w_j) \in G(w), q'(w_1, \ldots, w_j) \in A_1(w) \}$. Since $A_1$ is given and since there are at most $O(m|w|2r)$ elements in $G(w)$, the required polynomial-time algorithm $A$ can easily be defined.

Case 4: $S$ is defined according to item (4) of Definition 4.

Let $S_1 = (\Sigma, \Pi_1, \Gamma_1)$ be the AEFS and $S' = (\Sigma, \Pi', \Gamma')$ be the EFS used to define $S$ according to Definition 4, item (4). Hence, $hp(\Gamma') \cap \Pi_1 = \emptyset$. Assume that there is
a corresponding algorithm $A_1$ for $S_1$. Now, set $\tilde{\Pi} = \Pi \cup \Pi'$, $\tilde{\Gamma} = \Gamma' \cup A_1(w)$, and $\tilde{S} = (\Sigma, \tilde{\Pi}, \tilde{\Gamma})$. Clearly, $\tilde{S}$ is a hereditary AEFS. Moreover, since $S$ is a hereditary AEFS, we know that $G(w) \cap \text{Sem}(S) = G(w) \cap \text{Sem}_w(\tilde{S})$. Now, let $m = |\Gamma|$. Analogously to Case 1, based on the results in [13], one can define an algorithm $A$ that, on input $w$, outputs the set $G(w) \cap \text{Sem}_w(\tilde{S})$. Since the involved polynomial-time decision procedure from Miyano et al. [13] runs in time $O(m^2 |w|^{4k+1}r)$ and since $m \leq |G(w)| + m \leq c \cdot |w|^{2r}$ for some sufficiently large $c \in \mathbb{N}$, $A$ is the polynomial-time algorithm we are interested in.

As a careful analysis of the cases considered shows, the required decision procedure runs in polynomial-time with respect to $|w|$. This completes the proof of Lemma 5.

\begin{lemma}
For any $m, k, t, r \in \mathbb{N}$, there is a polynomial-time finder for $R$.
\end{lemma}

\textbf{Proof.} Let $m, k, t, r \in \mathbb{N}$ and let $T$ be a finite set of examples for some language $L \in \mathcal{L}(h$-\(\mathcal{AEFS}(m, k, t, r))$. Assume that $T \neq \emptyset$.

We let $\Pi = \{p, p_1, \ldots, p_{m-1}\}$, where only the arity of $p$ is a priori fixed, namely $p$ is a unary predicate symbol. Furthermore, we let $P(k, T)$ be the set of all patterns $\pi$ such that (i) $\pi(\pi) \subseteq \{x_1, \ldots, x_k\}$ and (ii) there is a substitution $\sigma$ such that $\pi \sigma$ is a subword of some labelled word from $T$. Now, the set $\mathcal{G}(m, k, t, r, T)$ of all candidate AEFSs is defined to be the set of all hereditary AEFSs $S = (\Sigma, \Pi, \Gamma)$ in $h$-\(\mathcal{AEFS}(m, k, t, r)$ such that each pattern $\pi$ in each atom of each rule in $\Gamma'$ belongs to $P(k, T)$.

First, we verify that $\mathcal{G}(m, k, t, r, T)$ contains an AEFS $S$ that is consistent with $T$. To see this, let $S'$ be any AEFS in $h$-\(\mathcal{AEFS}(m, k, t, r)$ such that $L(S', p)$ is consistent with $T$. Without loss of generality, we may assume that (a) $(S', p)$ is reduced with respect to $T$, (b) $S'$ contains only predicate symbols from $\{p, p_1, \ldots, p_{m-1}\}$ and (c) all variables in $S'$ are from $\{x_1, \ldots, x_k\}$. Because of (a), by Lemma 3, we know that, given any rule $r$ in $S'$, there is a substitution $\sigma$ such that, for each pattern $\pi$ in $r$, $\pi \sigma$ is a subword of some labelled word from $T$. Hence, the rules in $S'$ exclusively contain patterns from $P(k, T)$, and thus we obtain $S' \in \mathcal{G}(m, k, t, r, T)$.

Next, we show that there are at most polynomially many hereditary AEFSs in $\mathcal{G}(m, k, t, r, T)$. The relevant details are as follows. Let $n = \max\{|w| | (w, b) \in T\}$. Applying the same counting arguments as in [13], there are $O(|T|^{n^{2k+2}r!})$ patterns $\pi$ such that $\pi$ contains at most $k$ occurrences of variables from $\{x_1, \ldots, x_k\}$ and there is a substitution $\sigma$ such that $\pi \sigma$ is a subword of some labelled word from $T$. Hence, there are at most $\mathcal{G}(m, k, t, r, T)$ possible heads for rules for AEFS in $\mathcal{G}(m, k, t, r, T)$. Moreover, the number of possible atoms in the body of such a rule is at most $O(m(m(n-1)/2)^r)$, since, in hereditary rules, every pattern in the body must be a subword of some pattern in the head. Hence, there are at most $O((m^2 |T|^{n^{2k+2}+2mr^k!})$ rules without negation. Since there are at most $O((2mr^k)^2)$ rules with negation (cf. the verification of Lemma 4) and since every AEFS in $\mathcal{G}(m, k, t, r, T)$ consists of at most $m$ rules, the overall number of AEFSs in $\mathcal{G}(m, k, t, r, T)$ is polynomially bounded in $|T|$ and $n$.

Combining these insights with Lemma 5, one immediately sees that the following algorithm $F$ serves as a polynomial-time finder for the representation $R$: 
Algorithm $F$: On input $m, k, t, r, T$ proceed as follows:

Enumerate $\mathcal{G}(m, k, t, r, T)$. For each $S \in \mathcal{G}(m, k, t, r, T)$, test whether or not $L(S, p)$ is consistent with $T$. If some $S \in \mathcal{G}(m, k, t, r, T)$ consistent with $T$ is found, output the first one.

By Lemma 5, one easily sees that $F$ runs in time polynomial in $\sum_{(w, b) \in T} |w|$.

This completes the proof of Lemma 6. $\square$

Hence, (1) and (2) are fulfilled, and thus the theorem follows.

4. Conclusions

Motivated by research related to knowledge discovery and information extraction in the World Wide Web, we introduced advanced elementary formal systems (AEFSs)—a kind of logic programs to manipulate strings.

The authors are currently applying the approach presented here within a joint research and development project named LExIKON on information extraction from the Internet. This project is supported by the German Federal Ministry for Economics and Technology.

Advanced elementary formal systems generalize elementary formal systems (EFSs) in that they allow for the use of a certain kind of negation, which is non-monotonic, in essence, and which is conceptually close to negation as failure. In our approach, we syntactically constrained the use of negation. This guarantees that AEFSs inherit some of the convenient properties of EFSs—for instance, their clear and easy to capture semantics.

Negation as failure allows one to describe formal languages in a more natural and compact manner. Moreover, as Theorems 7 and 8 show, AEFSs are more expressive than EFSs. Naturally, this leads to the question of whether or not the known learnability results for EFS definable language classes remain valid if one considers the more general framework of AEFSs. Interestingly, the answer to this question heavily depends on the underlying learning paradigm.

As we have shown, certain AEFS definable language classes are not Gold-style learnable from only positive data, although the corresponding language classes that are definable by EFSs are known to be learnable (cf. Theorem 10). Surprisingly, in the PAC model, differences of this type cannot be observed (cf. Theorems 11 and 13).

Although the considered classes of AEFS definable languages properly comprise the corresponding classes of EFS definable languages—which are the largest classes of EFS definable languages formerly known to be polynomial-time PAC learnable—both language classes are polynomial-time PAC learnable.

Acknowledgements

This work has been partially supported by the German Ministry of Economics and Technology (BMWi) within the joint project LExIKON under grants 01 MD 948 and 01 MD 949.
The authors thank the anonymous referees for carefully reading the submitted version and for their valuable comments.

References
