Incremental Concept Learning for Bounded Data Mining

Case, John
Department of CIS University of Delaware

Jain, Sanjay
Department of ISCS National University of Singapore

Lange, Steffen
Universität Leipzig Fakultät für Mathematik und Informatik

Zeugmann, Thomas
Department of Informatics Kyushu University

出版情報：DOI Technical Report. 136, 1997-04-16. Department of Informatics, Kyushu University
バージョン：
権利関係：

DOI-TR-136

DOI Technical Report

Incremental Concept Learning for Bounded Data Mining

by

J. Case, S. Jain, S. Lange and T. Zeugmann

April 16, 1997

Department of Informatics
Kyushu University
Fukuoka 812-81, Japan

Email: thomas@i.kyushu-u.ac.jp   Phone: +81-92-642-2688

Incremental Concept Learning for Bounded Data Mining

John Case
Department of CIS
University of Delaware
Newark
DE 19716, USA
case@cis.udel.edu

Sanjay Jain
Department of ISCS
National University of Singapore
Lower Kent Ridge Road
Singapore 119260, Rep. of Singapore
sanjay@iscs.nus.edu.sg

Steffen Lange
Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Mathematik und Informatik
04109 Leipzig, Germany
slange@informatik.uni-leipzig.de

Thomas Zeugmann
Department of Informatics
Kyushu University
Fukuoka 812–81
Japan
thomas@i.kyushu-u.ac.jp

Abstract

Important refinements of concept learning in the limit from positive data *considerably restricting the accessibility of input data* are studied. Let c be any concept; every infinite sequence of elements exhausting c is called *positive presentation* of c. In all learning models considered the learning machine computes a sequence of hypotheses about the target concept from a positive presentation of it. With *iterative* learning, the learning machine, in making a conjecture, has access to its previous conjecture and the latest data item coming in. In *k-bounded example-memory* inference (k is *a priori* fixed) the learner is allowed to access, in making a conjecture, its previous hypothesis, its memory of up to k data items it has already seen, and the next element coming in. In the case of *k-feedback* identification, the learning machine, in making a conjecture, has access to its previous conjecture, the latest data item coming in, *and*, on the basis of this information, it can compute k items and query the database of previous data to find out, for each of the k items, whether or not it is in the database (k is again *a priori* fixed). In all cases, the sequence of conjectures has to converge to a hypothesis correctly describing the target concept.

Our results are manyfold. An infinite hierarchy of more and more powerful feedback learners in dependence on the number k of queries allowed to be asked is established. However, the hierarchy collapses to 1-feedback inference if only indexed families of *infinite* concepts are considered, and moreover, its learning power is then equal to learning in the limit. But it remains infinite for concept classes of only *infinite* r.e. concepts. Both k-feedback inference and k-bounded example-memory identification are more powerful than iterative learning but incomparable to one another. Furthermore, there *are* cases where redundancy in the hypothesis space is shown to be a resource increasing the learning power of iterative learners. Finally, the union of at most k pattern languages is shown to be iteratively inferable.

1. Introduction

The present paper derives its motivation to a certain extent from the rapidly emerging field of knowledge discovery in databases (abbr. KDD). Historically, there is a variety of names including data mining, knowledge extraction, information discovery, data pattern processing, information harvesting, and data archeology all referring to the notion of finding useful information about the data that has not been known before. Throughout this paper we shall use the term *KDD* for the *overall process* of discovering useful knowledge from data and *data mining* to refer to the particular subprocess of applying specific algorithms for learning something useful from the data. Thus, the additional steps such as data presentation, data selection, incorporating prior knowledge, and defining the semantics of the results obtained belong to KDD (cf., e.g., Fayyad *et al.* [14]). Prominent examples of KDD applications in health care and finance include Matheus *et al.* [27] and Kloesgen [22]. The importance of KDD research finds its explanation in the fact that the data collected in various fields such as biology, finance, retail, astronomy, medicine are extremely rapidly growing, while our ability to analyze those data has not kept up proportionally.

KDD mainly combines techniques originating from machine learning, knowledge acquisition and knowledge representation, artificial intelligence, pattern recognition, statistics, data visualization, and databases to automatically extract new interrelations, knowledge, patterns and the like from *huge* collections of data. Usually, the data are available from massive data sets collected, for example, by scientific instruments (cf., e.g.,Fayyad *et al.* [13]), by scientists all over the world (as in the human genome project), or in databases that have been built for other purposes than a current purpose.

We shall be mainly concerned with the extraction of *concepts* in the data mining process. Thereby, we emphasize the aspect of working with *huge* data sets. For example, in Fayyad *et al.* [13] the SKICAT-system is described which operates on 3 terabytes of image data originating from approximately 2 billion sky objects which had to be classified. If huge data sets are around, no learning algorithm can use all the data or even large portions of it simultaneously for computing hypotheses about concepts represented by the data. Different methods have been proposed for overcoming the difficulties caused by huge data sets. For example, *sampling* may be a method of choice. That is, instead of doing the discovery process on all the data, one starts with significantly smaller samples, finds the regularities in it, and uses the different portions of the overall data to verify what one has found. Clearly, a major problem involved concerns the choice of the right sampling size. One way proposed to solve this problem as well as other problems related to huge data sets is *interaction* and *iteration* (cf., e.g., Brachman and Anand [6] and Fayyad *et al.* [14]). That is, the whole data mining process is iterated a few times, thereby allowing human interaction until a satisfactory interpretation of the data is found.

Looking at data mining from the perspective described above, it becomes a true limiting process. That means, the actual result of the data mining algorithm application run on a sample is tested versus (some of) the remaining data. Then, if, for any reason whatever, a current hypothesis is not acceptable, the sample may be enlarged (or replaced) and the algorithm is run again. Since the data set is extremely large, clearly not all data can be validated in a prespecified amount of time. Thus, from a theoretical point of view, it is appropriate to look at the data mining process as an *ongoing, incremental* one.

In the present theoretical study, then, we focus on *important refinements or restrictions of* Gold's [18] model of learning *in the limit* grammars for concepts from positive instances.[1] Gold's [18] model itself makes the unrealistic assumption that the learner has access to samples of increasingly growing size. Therefore, we investigate refinements that *considerably restrict the accessibility of input data.* In particular, we deal with so-called *iterative* learning, *bounded example-memory* inference, and *feedback* identification (cf. Definitions 3, 4, and 5, respectively). Each of these models formalizes a kind of *incremental learning.* In each of these models we imagine a stream of positive data coming in about a concept and that the data that arrived in the past sit in a database which can get very very large. Intuitively, with *iterative* learning, the learning machine, in making a conjecture, has access to its previous conjecture and the latest data item coming in — *period.* In *bounded example-memory* inference, the learning machine, in making a conjecture, has access to its previous conjecture, its *memory* of *up to* k data items it has seen, and a new data item. Hence, a bounded example-memory machine wanting to memorize a *new* data item it's just seen, if it's already remembering k previous data items, must *forget* one of the previous k items in its memory to make room for the new one! In the case of *feedback* identification, the learning machine, in making a conjecture, has access to its previous conjecture, the latest data item coming in, *and*, on the basis of this information, it can compute k items and query the database of previous data to find out, for each of the k items, whether or not it is in the database. For some extremely large databases, a query about whether an item is in there can be very expensive, so, in such cases, feedback identification is interesting when the bound k is small.

Of course the $k = 0$ cases of bounded example-memory inference and feedback identification are just iterative learning.

Next we summarize informally our main results.

Theorems 2 and 3 imply that, for each k, there are concept classes of infinite r.e. languages which can be learned by some feedback machine using no more than $k > 0$ queries of the database, but *no* feedback machine can learn these classes if it's restricted to no more than $k - 1$ queries.[2] Hence, each additional, possibly expensive dip into the database buys more concept learning power. Theorem 2 is a consequence of Theorem 3, and the proof of the latter is non-trivial. However, the feedback hierarchy collapses to its first level if only *indexable classes* of *infinite* concepts are to be learned (cf. Theorem 4).

A bounded example-memory machine can remember *its choice of* k items from the data, and it can *choose* to forget some old items so as to remember some new ones. On the other hand, at each point, the feedback machine can query the database about *its choice of* k things each being or not being in the database. A bounded example-memory machine chooses which k items to *memorize* as being in the database, and the feedback machine can decide which k items to *lookup* to see if they are in the database. There are apparent

[1]The sub-focus on learning *grammars*, or, equivalently, recognizers (cf. Hopcroft and Ullman [19]), for concepts from *positive* instances nicely models the situation where the database flags or contains *examples* of the concept to be learned and doesn't flag or contain the non-examples.

[2]That the concepts in the concept classes witnessing this hierarchy are all *infinite* languages is also interesting and for two reasons: 1. It is arguable that all natural languages are infinite, and 2. many language learning *un*solvability results *depend strongly* on including the finite languages (cf. Gold [18]; Case [11]). Ditto for other results below, namely, Theorems 6 and 7, which are witnessed by concept classes containing only infinite concepts.

similarities between these two kinds of learning machines, yet Theorems 6 and 7 show that in very strong senses, for each of these two models, there are concept class domains where that model is competent and the other is not! Both proofs are non-trivial, and that of Theorem 7 is the most difficult for the present paper.

Theorem 8 shows that, even in fairly concrete contexts, with iterative learning, *redundancy* in the hypothesis space increases learning power.

Angluin's [1] *pattern languages* are learnable from positive data, and they (and finite unions thereof) have been extensively studied and applied to molecular biology and to the learning of interesting special classes of logic programs (see the references in Section 3.4 below). Theorem 9 implies that, for each $k > 0$, the concept class consisting of all unions of at most k pattern languages is learnable (from positive data) by an iterative machine!

2. Preliminaries

Unspecified notation follows Rogers [33]. In addition to or in contrast with Rogers [33] we use the following. By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the set of all natural numbers. We set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. The cardinality of a set S is denoted by $|S|$. Let $\emptyset$, $\in$, $\subset$, $\subseteq$, $\supset$, and $\supseteq$, denote the empty set, element of, proper subset, subset, proper superset, and superset, respectively. Let S_1, S_2 be any sets; then we write $S_1 \triangle S_2$ to denote the symmetric difference of S_1 and S_2, i.e., $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. Additionally, for any sets S_1 and S_2 and $a \in \mathbb{N} \cup \{*\}$ we write $S_1 =^a S_2$ provided $|S_1 \triangle S_2| \leq a$, where $a = *$ means that the symmetric difference is finite. By $\max S$ and $\min S$ we denote the maximum and minimum of a set S, respectively, where, by convention, $\max \emptyset = 0$ and $\min \emptyset = \infty$.

The quantifiers '$\overset{\infty}{\forall}$,' '$\overset{\infty}{\exists}$,' and '$\exists!$' are interpreted as 'for all but finitely many,' 'there exists infinitely many,' and 'there exists a unique,' respectively (cf. [5]).

By $\langle \cdot, \cdot \rangle \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ we denote *Cantor's pairing function.*[3] Moreover, we let π_1 and π_2 denote the corresponding *projection functions* over $\mathbb{N}$ to the first and second components, respectively. That is, $\pi_1(\langle x, y \rangle) = x$ and $\pi_2(\langle x, y \rangle) = y$ for all $x, y \in \mathbb{N}$.

Let φ_0, φ_1, φ_2, ... denote any fixed *acceptable programming system* for all (and only) the partial recursive functions over $\mathbb{N}$ (cf. [33]), and let Φ_0, Φ_1, Φ_2, ... be any associated *complexity measure* (cf. Blum [5]). Then φ_k is the partial recursive function computed by *program* k. Furthermore, let k, $x \in \mathbb{N}$; if $\varphi_k(x)$ is defined (abbr. $\varphi_k(x){\downarrow}$) then we also say that $\varphi_k(x)$ *converges*; otherwise $\varphi_k(x)$ *diverges*.

Any recursively enumerable set $\mathcal{X}$ is called a *learning domain*. By $\wp(\mathcal{X})$ we denote the power set of $\mathcal{X}$. Let $\mathcal{C} \subseteq \wp(\mathcal{X})$, and let $c \in \mathcal{C}$; then we refer to $\mathcal{C}$ and c as a *concept class* and a *concept*, respectively. Let c be a concept, and let $T = x_0, x_1, x_2, \ldots$ an infinite sequence of elements $x_i \in c \cup \{\#\}$ such that range$(T) =_{df} \{x_k \mid x_k \neq \#,\ k \in \mathbb{N}\} = c$. Then T is said to be a *positive presentation* or, synonymously, a *text* for c. By *text*(c) we denote the set of all positive presentations for c. Moreover, let T be a positive presentation, and let y be a number. Then, T_y denotes the initial segment of T of length $y + 1$, and $T_y^+ =_{df} \{x_k \mid x_k \neq \#,\ k \leq y\}$. We refer to T_y^+ as the *content* of T_y. Intuitively, the #'s represent pauses in the positive presentation of the data of a concept c. Furthermore, let $\sigma = x_0, \ldots, x_{n-1}$ be any finite sequence. Then we use $|\sigma|$ to denote the *length* n of σ. Additionally, let T be a text and

[3] This function is easily computable, 1-1, and onto (cf. Rogers [33]).

let τ be a finite sequence; then we use $\sigma \diamond T$ and $\sigma \diamond \tau$ to denote the sequence obtained by *concatenating* σ onto the front of T and τ, respectively. By *SEQ* we denote the set of all finite sequences of elements from $\mathcal{X} \cup \{\#\}$.

As a special case, we often consider the scenario $\mathcal{X} = \mathbb{N}$, and $\mathcal{C} = \mathcal{E}$, where $\mathcal{E}$ denotes the collection of all recursively enumerable sets W_i, $i \in \mathbb{N}$, of natural numbers. These sets W_i can be described as $W_i = \text{domain}(\varphi_i)$. Thus, we also say that W_i is accepted, recognized (or, equivalently, generated) by the φ-program i. Hence, we also refer to the index i of W_i as a *grammar* for W_i.

Furthermore, we sometimes consider the scenario that indexed families of recursive languages have to be learned (cf. Angluin [2]). Let Σ be any finite alphabet of symbols, and let $\mathcal{X}$ be the free monoid over Σ, i.e., $\mathcal{X} = \Sigma^*$. As usual, we refer to subsets $L \subseteq \mathcal{X}$ as to languages. A class of non-empty recursive languages $\mathcal{L}$ is said to be an *indexed family* provided there are an effective enumeration $L_0, L_1, L_2, \ldots$ of all and only the languages in $\mathcal{L}$ and a recursive function f such that for all $j \in \mathbb{N}$ and all strings $x \in \mathcal{X}$ we have

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

Since the paper of Angluin [2] learning of indexed families of languages has attracted much attention (cf., e.g., Zeugmann and Lange [46]). Mainly, this seems due to the fact that most of the established language families such as regular languages, context-free languages, context-sensitive languages, and pattern languages are indexed families.

Essentially from Gold [18] we define an *inductive inference machine* (abbr. *IIM*), or simply a learning machine, to be an algorithmic mapping from *SEQ* to $\mathbb{N} \cup \{?\}$. Intuitively, we interpret the output of a learning machine with respect to a suitably chosen hypothesis space $\mathcal{H}$. The output "?" is uniformly interpreted as "no conjecture." We always take as a hypothesis space a recursively enumerable family $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ of concepts (construed as sets or languages), where the j in h_j is thought of a numerical name for some finite description or computer program for h_j. Moreover, let c be a concept, let h_j be a hypothesis, and let $a \in \mathbb{N} \cup \{*\}$; then we write $c =^a h_j$ iff $|c \triangle h_j| \leq a$. That is, if $a \in \mathbb{N}$, then h_j describes c up to at most a anomalies. The $*$ is used to express any finite number of anomalies. We let M, with or without decorations, range over learning machines.

Let T be a positive presentation, and let $y \in \mathbb{N}$. The sequence $(M(T_y))_{y\in\mathbb{N}}$ is said to *converge* to the number j iff in $(M(T_y))_{y\in\mathbb{N}}$ all but finitely many terms are equal to j.

Now we define some models of learning. We start with Gold's [18] unrestricted learning in the limit (and some variants). Then we will present the definitions of the models which more usefully restrict access to the database.

Definition 1 (*Gold* [18]). *Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N}\cup\{*\}$. An IIM M $TxtEx^a_{\mathcal{H}}$-infers c iff, for every $T \in text(c)$, there exists a $j \in \mathbb{N}$ such that the sequence $(M(T_y))_{y\in\mathbb{N}}$ converges to j and $c =^a h_j$.*

M $TxtEx^a_{\mathcal{H}}$-infers $\mathcal{C}$ iff M $TxtEx^a_{\mathcal{H}}$-infers c, for each $c \in \mathcal{C}$.

Let $TxtEx^a_{\mathcal{H}}$ denote the collection of all concept classes $\mathcal{C}$ for which there is an IIM M such that M $TxtEx^a_{\mathcal{H}}$-infers $\mathcal{C}$.

$TxtEx^a$ denotes the collection of all concept classes $\mathcal{C}$ for which there are an IIM M and a hypothesis space $\mathcal{H}$ such that M $TxtEx^a_{\mathcal{H}}$-infers $\mathcal{C}$.

The a represents the number of mistakes or anomalies allowed in the final conjectures (cf. Case and Smith [12]), with $a = 0$ being Gold's [18] original case where no mistakes are allowed. If $a = 0$, we usually omit the upper index, e.g., we write *TxtEx* instead of $TxtEx^0$. We adopt this convention in the definitions of the learning types below.

Since, by the definition of convergence, only finitely many data about c were seen by the IIM up to the (unknown) point of convergence, whenever an IIM infers the concept c, some form of learning must have taken place. For this reason, hereinafter the terms *infer*, *learn*, and *identify* are used interchangeably.

For $TxtEx^a_{\mathcal{H}}$-inference, a learner has to converge to a *single* description for the target to be inferred. However, it is imaginable that humans do not converge to a single grammar when learning their mother tongue. Instead, we may learn a small number of *equivalent* grammars each of which is easier to apply than the others in quite different situations. This speculation directly suggests the following definition.

Definition 2 (*Case and Smith* [12]). *Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtFex^a_{\mathcal{H}}$-infers c iff, for every $T \in text(c)$, there exists a nonempty finite set D such that $c =^a h_j$, for all $j \in D$ and $M(T_y) \in D$, for all but finitely many y.*

M $TxtFex^a_{\mathcal{H}}$-infers $\mathcal{C}$ iff M $TxtFex^a_{\mathcal{H}}$-infers c, for each $c \in \mathcal{C}$.

Let $TxtFex^a_{\mathcal{H}}$ denote the collection of all concept classes $\mathcal{C}$ for which there is an IIM M such that M $TxtFex^a_{\mathcal{H}}$-infers $\mathcal{C}$.

$TxtFex^a$ denotes the collection of all concept classes $\mathcal{C}$ for which there are an IIM M and a hypothesis space $\mathcal{H}$ such that M $TxtFex^a_{\mathcal{H}}$-infers $\mathcal{C}$.

The following theorem clarifies the relation between Gold's [18] classical learning in the limit and *TxtFex*-inference. The assertion remains true even if the learner is only allowed to *vacillate* between up to 2 descriptions, i.e., in the case $|D| \le 2$ (cf. Case [9, 11]).

Theorem 1 (Osherson *et al.* [31]; *Case* [9, 11]). *$TxtEx^a \subset TxtFex^a$, for all $a \in \mathbb{N} \cup \{*\}$.*

Looking at the above definitions, we see that an IIM M has always access to the whole history of the learning process, i.e., in order to compute its actual guess M is fed all examples seen so far. In contrast to that, next we define *iterative IIMs* and a natural generalization of them called *bounded example-memory IIMs.* An iterative IIM is only allowed to use its last guess and the next element in the positive presentation of the target concept for computing its actual guess. Conceptionally, an iterative IIM M defines a sequence $(M_n)_{n\in N}$ of machines each of which takes as its input the output of its predecessor.

Definition 3 (*Wiehagen* [42]). *Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtItEx^a_{\mathcal{H}}$–infers c iff for every $T = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied:*

(1) *for all $n \in \mathbb{N}$, $M_n(T)$ is defined, where $M_0(T) =_{df} M(x_0)$ and for all $n \ge 0$: $M_{n+1}(T) =_{df} M(M_n(T), x_{n+1})$,*

(2) *the sequence $(M_n(T))_{n\in\mathbb{N}}$ converges to a number j such that $c =^a h_j$.*

Finally, M $TxtItEx^a_{\mathcal{H}}$–infers $\mathcal{C}$ iff, for each $c \in \mathcal{C}$, M $TxtItEx^a_{\mathcal{H}}$–infers c.

The resulting learning types $TxtItEx^a_{\mathcal{H}}$ and $TxtItEx^a$ are analogously defined as above.

In the latter definition $M_n(T)$ denotes the $(n+1)$th hypothesis output by M when successively fed the positive presentation T. Thus, it is justified to make the following convention.

Let $\sigma = x_0, \ldots, x_n$ be any finite sequence of elements over the relevant learning domain. Moreover, let $\mathcal{C}$ be any concept class over $\mathcal{X}$, and let M be any IIM that iteratively learns $\mathcal{C}$. Then we denote by $M_y(\sigma)$ the $(y+1)$th hypothesis output by M when successively fed σ provided $y \leq n$, and there exists a concept $c \in \mathcal{C}$ with $\sigma^+ \subseteq c$. We adopt this convention to the learning types defined below.

Within the following definition we consider a natural relaxation of iterative learning which we call *bounded example-memory* inference.[4] Now, an IIM M is allowed to memorize an *a priori* bounded number of the examples it already has had access to during the learning process. Again, M defines a sequence $(M_n)_{n \in \mathbb{N}}$ of machines each of which takes as input the output of its predecessor. Consequently, a bounded example-memory IIM has to output a hypothesis as well as a subset of the set of examples seen so far.

Definition 4 (*Lange and Zeugmann* [25]). *Let $k \in \mathbb{N}$, let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtBem^kEx^a_{\mathcal{H}}$–infers c iff for every $T = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied:*

(1) *for all $n \in \mathbb{N}$, $M_n(T)$ is defined, where $M_0(T) =_{df} M(x_0) = \langle j_0, S_0\rangle$ such that $S_0 \subseteq T_0^+$ and $|S_0| \leq k$, and for all $n \geq 0$: $M_{n+1}(T) =_{df} M(M_n(T), x_{n+1}) = \langle j_{n+1}, S_{n+1}\rangle$ such that $S_{n+1} \subseteq S_n \cup \{x_{n+1}\}$ and $|S_{n+1}| \leq k$,*

(2) *the j_n in the sequence $(\langle j_n, S_n\rangle)_{n\in\mathbb{N}}$ of M's guesses converges to a $j \in \mathbb{N}$ with $c =^a h_j$.*

Finally, M $TxtBem^kEx^a_{\mathcal{H}}$–infers $\mathcal{C}$ iff, for each $c \in \mathcal{C}$, M $TxtBem^kEx^a_{\mathcal{H}}$–infers c.

For every $k \in \mathbb{N}$, the resulting learning types $TxtBem^kEx^a_{\mathcal{H}}$ and $TxtBem^kEx^a$ are analogously defined as above. Clearly, by definition, $TxtItEx^a = TxtBem^0Ex^a$, for all $a \in \mathbb{N}\cup\{*\}$.

Finally, we define learning by *feedback* IIMs. The idea of feedback learning goes back to Wiehagen [42] who considered it in the setting of inductive inference of recursive functions. Lange and Zeugmann [25] adapted the concept of feedback learning to inference from positive data. Here, we *generalize* this definition. Informally, a feedback IIM M is an iterative IIM that is additionally allowed to make a bounded number of a particular type of query. In each learning Stage $n+1$, M has access to the actual input x_{n+1}, and its previous guess j_n. However, M is additionally allowed to compute queries from x_{n+1} and j_n. Each query concerns the history of the learning process. Let $k \in \mathbb{N}$; then a *k-feedback learner* computes a k-tuple of elements $(y_1, \ldots, y_k) \in \mathcal{X}^k$ and gets a k-tuple of "YES/NO" answers such that the ith component of the answer is 1, if $y_i \in T_n^+$ and it's 0, otherwise. Hence, M can just ask whether or not k particular strings have already been presented in previous learning stages.

Definition 5. *Let $k \in \mathbb{N}$, let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N}\cup\{*\}$. Moreover, let $Q_k\colon \mathbb{N}\times\mathcal{X} \to \mathcal{X}^k$, be a computable total mapping. An IIM M $TxtFb^kEx^a_{\mathcal{H}}$–infers c iff for every positive presentation $T = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied: below $A^n_k\colon \mathcal{X}^k \to \{0,1\}^k$ denotes the answer to the queries (based on whether the corresponding queried elements appear in T_n or not).*

(1) *for all $n \in \mathbb{N}$, $M_n(T)$ is defined, where $M_0(T) =_{df} M(x_0)$ and for all $n \geq 0$:*
$M_{n+1}(T) =_{df} M(M_n(T), A^n_k(Q_k(M_n(T), x_{n+1})), x_{n+1})$,

(2) *the sequence $(M_n(T))_{n\in\mathbb{N}}$ converges to a number j such that $c =^a h_j$ provided that A^n_k truthfully answers the questions computed by Q_k (i.e. the j-th component of $A^n_k(Q_k(M_n(T), x_{n+1}))$ is 1 iff the j-th component of $Q_k(M_n(T), x_{n+1})$ appears in T_n.)*

[4]Our definition is a variant of one found in Osherson, Stob and Weinstein [31] and Fulk *et al.* [17] which will be discussed later.

Finally, M $TxtFb^kEx^a_{\mathcal{H}}$–infers $\mathcal{C}$ iff there is computable mapping Q_k as described above such that, for each $c \in \mathcal{C}$, M $TxtFb^kEx^a_{\mathcal{H}}$–identifies c.

The resulting learning types $TxtFb^kEx^a_{\mathcal{H}}$ and $TxtFb^kEx^a$ are defined analogously as above.

Finally, we extend Definitions 3 through 5 to the *Fex* case analogously to the generalization of $TxtEx^a_{\mathcal{H}}$ to $TxtFex^a_{\mathcal{H}}$ (cf. Definition 1 and 2). The resulting learning types are denoted by $TxtItFex^a_{\mathcal{H}}$, $TxtBem^kFex^a_{\mathcal{H}}$, and $TxtFbEx^a_{\mathcal{H}}$. Moreover, for the sake of notation, we shall use the following convention for learning machines corresponding to Definitions 3 through 5 as well as to $TxtItFex^a_{\mathcal{H}}$, $TxtBem^kFex^a_{\mathcal{H}}$, and $TxtFbEx^a_{\mathcal{H}}$. Let τ be any finite sequence; then we let $M_*(\tau)$ denote $M_{|\tau|-1}(\tau)$.

3. Results

In this section we present our results. In the next subsection, we deal with feedback learning. Our aim is twofold. On the one hand, we investigate the learning power of feedback inference in dependence on k, i.e., the number of strings that may be simultaneously queried. On the other hand, we compare feedback identification with the other learning models introduced, varying the error parameter too (cf. Subsection 3.2). In subsequent subsections we study iterative learning: in Subsection 3.3, the efficacy of redundant hypotheses for iterative learning and, in Subsection 3.4, the iterative learning of finite unions of pattern languages.

3.1. *Feedback Inference*

The next theorem establishes a new infinite hierarchy of successively more powerful feedback learners in dependence on the number k of database queries allowed to be asked simultaneously.[5]

Theorem 2. *$TxtFb^{k-1}Ex \subset TxtFb^kEx$ for all $k \in \mathbb{N}^+$.*

Theorem 3 below not only provides the hierarchy of Theorem 2, but it says that, for suitable concept domains, the feedback learning power of $k+1$ queries of the data base, where a *single, correct* grammar is found in the limit, *beats* the feedback learning power of k queries, even when *finitely many grammars* each with *finitely many anomalies* are allowed in the limit.

Theorem 3. *$TxtFb^{k+1}Ex \setminus TxtFb^kFex^* \neq \emptyset$, for all $k \in \mathbb{N}$. Moreover this separation can be witnessed by a class consisting of only infinite languages.*

Proof. For every $w \in \mathbb{N}$, we define $X_w = \{\langle j, w, i\rangle \mid 1 \le j \le k+2,\ i \in \mathbb{N}\}$, and $X^0_w = \{\langle j, w, 0\rangle \mid 1 \le j \le k+2\}$.

A number e is said to be *nice* iff

(a) $\{x \in \mathbb{N} \mid \langle 0, x, 0\rangle \in W_e\} = \{e\}$; and

(b) $\neg(\exists w)[X^0_w \subseteq W_e]$.

[5]It follows from Fulk *et al.* [17] and Lange and Zeugmann [25] that there is an infinite hierarchy of successively more powerful bounded example-memory learners in dependence on the number k of items that can be memorized.

Finally, we define the desired concept class as follows.

Let $\mathcal{L} = \{L \mid (\exists \text{ nice } e)[|L| = \infty \ \wedge\ [L = W_e \ \vee\ (\exists! w)[L = W_e \cup X_w]]]\}$.

Claim 1. $\mathcal{L} \in \mathit{TxtFb}^{k+1}\mathit{Ex}$.

The idea behind the following proof can be easily explained. Intuitively, from a text for $L \in \mathcal{L}$, a learner can iteratively determine the unique e such that $\langle 0, e, 0\rangle \in L$, and it can remember e in its output using padding. To determine the unique w, if any, such that $L = W_e \cup X_w$, Property (b) in the definition of *nice* as well as $X^0_w \subseteq X_w$ are exploited. That is, the learner tries to verify $X^0_w \subseteq L$ whenever receiving an element of the form $\langle j, w, 0\rangle$ with $1 \le j \le k+2$ by just asking whether the other $k+1$ elements in $X^0_w \setminus \{\langle j, w, 0\rangle\}$ have already appeared in the text. Now, if $L = W_e$, then Property (b) above ensures that the answer is always 'NO,' and the learner just repeats its previous guess. On the other hand, if the answer is 'YES,' then the learner has verified $X^0_w \subseteq L$, and applying Property (b) as well as $X^0_w \subseteq X_w$, it may conclude $L = W_e \cup X_w$. Thus, it remembers w in its output using padding. Moreover, e and w, if any, can be easily used to form a grammar for L along with the relevant padding. We now formally define M behaving as above.

Let pad be a 1–1 recursive function such that, for all i, j, $W_{\mathrm{pad}(0,j)} = \emptyset$, $W_{\mathrm{pad}(i+1,0)} = W_i$, and $W_{\mathrm{pad}(i+1,j+1)} = W_i \cup X_j$. M, and its associated query asking function Q_{k+1}, witnessing that $\mathcal{L} \in \mathit{TxtFb}^{k+1}\mathit{Ex}$ is defined as follows. M's output will be of the form, $\mathrm{pad}(e', w')$. Furthermore, e' and w' are used for "memory" by M. Intuitively, if the input seen so far contains $\langle 0, e, 0\rangle$ then $e' = e+1$; if the input contains X^0_w then $w' = w+1$.

Let $T = s_0, s_1, \ldots$ be a text for some $L \in \mathcal{L}$. Suppose $s_0 = \langle j, z, i\rangle$. If $i = j = 0$, then let $M(s_0) = \mathrm{pad}(z+1, 0)$. Otherwise let $M(s_0) = \mathrm{pad}(0,0)$.

$Q_{k+1}(q, s_{m+1})$ is computed as follows. Suppose $s_{m+1} = \langle j, z, i\rangle$. If $i = 0$ and $1 \le j \le k+2$, then let $y_1, y_2, \ldots, y_{k+1}$ be such that $\{y_1, y_2, \ldots, y_{k+1}\} = \{\langle j', z, 0\rangle \mid 1 \le j' \le k+2,\ j' \ne j\}$. If $i \ne 0$, then let $y_1 = y_2 = \ldots = y_{k+1} = 0$ (we do not need any query in this case).

We now define $M(q, A_{k+1}(Q_{k+1}(q, s_{m+1})), s_{m+1})$ as follows.

```
$M(q, A_{k+1}(Q_{k+1}(q, s_{m+1})), s_{m+1})$
1. Suppose $s_{m+1} = \langle j, z, i\rangle$, and $q = \mathrm{pad}(e', w')$.
2. If $i = 0$, $1 \le j \le k+2$ and $A_{k+1}(Q_{k+1}(q, s_{m+1})) = (1, 1, \ldots, 1)$, then let $w' = z+1$.
3. If $i = 0$, $j = 0$, then let $e' = z+1$.
4. Output $\mathrm{pad}(e', w')$.
End
```

It is easy to verify that M $\mathit{TxtFb}^{k+1}\mathit{Ex}$-infers every language in $\mathcal{L}$. This proves Claim 1.

Claim 2. $\mathcal{L} \notin \mathit{TxtFb}^{k}\mathit{Fex}^*$.

Suppose the converse, i.e., that there are an IIM M and an associated query asking function Q_k such that M witnesses $\mathcal{L} \in \mathit{TxtFb}^{k}\mathit{Fex}^*$. Then by implicit use of the Recursion Theorem (cf. [33]) there exists an e such that W_e may be described as follows. Note that e will be nice.

For any finite sequence $\tau = x_0, x_1, \ldots, x_\ell$, let $M_0(\tau) = M(x_0)$; and for $i < \ell$, let $M_{i+1}(\tau) = M(M_i(\tau), A^i_k(Q_k(M_i(\tau), x_{i+1})), x_{i+1})$, where A^i_k answers questions based on whether the corresponding elements appear in $\{x_j \mid j \le i\}$. Let $\mathrm{ProgSet}(M, \tau) = \{M_*(\sigma) \mid \sigma \subseteq \tau\}$.

Initialization. Enumerate $\langle 0, e, 0\rangle$ in W_e. Let σ_0 be such that content$(\sigma_0) = \{\langle 0, e, 0\rangle\}$. Let W_e^s denote W_e enumerated before Stage s. Go to Stage 0.

```
Stage $s$.
    (* Intuitively, in Stage $s$ we try to search for a suitable sequence $\sigma_{s+1}$ such that the
      condition ProgSet$(M, \sigma_{s+1}) \neq$ ProgSet$(M, \sigma_s)$ holds. Thus, if there are infinitely many
      stages, then $M$ does not $\mathit{TxtFb}^k\mathit{Fex}^*$-identify $\bigcup_s \sigma_s$, which will be a text for $W_e$. In
      case some stage starts but does not end, we will have that a suitable $W_e \cup X_w$ is not
      $\mathit{TxtFb}^k\mathit{Fex}^*$-identified by $M$. *)
1.  Let $S_s =$ ProgSet$(M, \sigma_s)$.
2.  Let $S' = S_s$.
3.  Let Pos = content$(\sigma_s)$; Neg = $\emptyset$. $Y = \emptyset$; $\tau = \sigma_s$
4.  While $M_*(\tau) \in S'$ Do
        (* We will have the following invariant at the beginning of every iteration of the
          while loop:
              If for some *suitable* $\tau'$ extending $\tau$, $M_*(\tau') \notin S'$, then there exists a *suitable* $\gamma$
                extending $\tau'$ such that $M_*(\gamma) \notin$ ProgSet$(M, \sigma_s)$, where by suitable above
                for $\tau'$ and $\gamma$ we mean:
                    (a) content$(\tau') \cap$ Neg $= \emptyset$,
                    (b) Pos $\subseteq$ content$(\tau')$,
                    (c) $(\forall w)[X_w^0 \not\subseteq$ content$(\tau') \cup Y]$,
                    (d) $\{x \mid \langle 0, x, 0\rangle \in ($content$(\tau') \cup Y)\} = \{e\}$,
                    (e) Pos $\cup\, Y \subseteq$ content$(\gamma)$,
                    (f) $(\forall w)[X_w^0 \not\subseteq$ content$(\gamma)]$, and
                    (g) $\{x \mid \langle 0, x, 0\rangle \in$ content$(\gamma)\} = \{e\}$.
        Moreover, $S'$ becomes smaller with each iteration of the while loop. *)
    4.1. Search for $p \in S'$, $y \in \mathbb{N}$ and finite sets Pos$'$, Neg$'$ such that
              $y \notin$ Neg,
              Pos $\subseteq$ Pos$'$,
              Neg $\subseteq$ Neg$'$,
              Pos$' \cap$ Neg$' = \emptyset$,
              $(\forall w)[X_w^0 \not\subseteq$ Pos$' \cup \{y\} \cup Y]$,
              $\{x \mid \langle 0, x, 0\rangle \in ($Pos$' \cup \{y\} \cup Y)\} = \{e\}$, and
              $M(p, A_k(Q_k(p, y)), y)\downarrow \notin S'$, where all the questions asked by $Q_k$ belong to
                Pos$' \cup$ Neg$'$, and $A_k$ answers the question $z$ positively if $z \in$ Pos$'$, and
                negatively if $z \in$ Neg$'$.
    4.2. If and when such $p, y$, Pos$'$, Neg$'$ are found,
              Let $S' = S' - \{p)\}$,
              Neg = Neg$'$,
              Pos = Pos$'$,
              $Y = Y \cup \{y\}$.
              Enumerate Pos in $W_e$.
              Let $\tau$ be an extension of $\sigma_s$ such that content$(\tau) =$ Pos
              (* Note that $y$ may or may not be in Pos or Neg. *)
    Endwhile
5.  Let $\sigma_{s+1}$ extending $\tau$ be such that
```

Pos $\cup Y \cup \{\langle k+3, s, 0\rangle\} \subset \mathrm{content}(\sigma_{s+1})$.
$(\forall w)[X_w^0 \not\subseteq \mathrm{content}(\sigma_{s+1})]$,
$\{x \mid \langle 0, x, 0\rangle \in \mathrm{content}(\sigma_{s+1})\} = \{e\}$, and
$\mathrm{ProgSet}(M, \sigma_{s+1}) \neq \mathrm{ProgSet}(M, \sigma_s)$.

Note that by the invariant above, there exists such a σ_{s+1}.

Enumerate content(σ_{s+1}) in W_e, and go to Stage $s+1$.

End Stage s

Note that the invariant can be easily proved by induction on the number of times the while loop is executed. We now consider two cases.

Case 1. All stages terminate.

In this case clearly, W_e is infinite and e is nice. Thus, we conclude $W_e \in \mathcal{L}$. Also, $T = \bigcup_s \sigma_s$ is a text for W_e. However, M on T outputs infinitely many different programs.

Case 2. Stage s starts but does not terminate.

By construction, if Stage s is not left, then W_e is finite and e is again nice. We show that there is a set X_w such that $W_e \cup X_w \in \mathcal{L}$ but $W_e \cup X_w$ is not $TxtFb^kFex^*$-inferred by M.

Now, let S', Pos, Neg, and τ be as in the last iteration of the while loop that is executed in Step 4 of Stage s. Furthermore, let w be such that

(i) $(\forall p \in S')[X_w \cap W_p = \emptyset \ \vee\ (\overset{\infty}{\exists} w')[X_{w'} \cap W_p \neq \emptyset]]$

(ii) $X_w \cap (\mathrm{Pos} \cup \mathrm{Neg}) = \emptyset$.

Note that there exists such a w, since S', Pos, and Neg are all finite.

Clearly, $W_e \cup X_w \in \mathcal{L}$. We now claim that M (with query asking function Q_k) cannot $TxtFb^kFex^*$-infer $W_e \cup X_w$. Note that, by construction, $W_e \cup X_w$ does not contain any element of Neg. Moreover, none of the programs in S' is a program for a finite variant of $W_e \cup X_w$. For all $p \in S'$, either $X_w \cap W_p = \emptyset$ or W_p intersects infinitely many $X_{w'}$. Note that W_e is finite and $X_w \cap X_{w'} = \emptyset$, for $w \neq w'$.

We claim that for all $\tau' \diamond y$ extending τ such that content$(\tau' \diamond y) \subseteq W_e \cup X_w$, $M_*(\tau' \diamond y) \in S'$. Suppose by way of contradiction the converse. Let $\tau' \diamond y$ be the smallest sequence that violates this condition. Then $M_*(\tau') \in S'$. Let P be the set of questions answered positively, and S be the set of questions answered negatively for the queries $Q_k(M_*(\tau'), y)$. Then $p = M_*(\tau')$, y, $\mathrm{Pos}' = \mathrm{Pos} \cup P$ and $\mathrm{Neg}' = \mathrm{Neg} \cup S$, witness that the search in Step 4.1 will succeed (a contradiction).

Thus we can conclude that M (with associated question asking function Q_k) does not $TxtFb^kFex^*$-identify $(W_e \cup X_w) \in \mathcal{L}$.

From the above claims, the theorem follows. ▮

Theorem 3 above nicely contrasts with the following result actually stating that the feedback hierarchy collapses to its first level provided only indexed families of infinite languages are considered.

THEOREM 4. *Let $\mathcal{L}$ be any* indexed *family consisting of only infinite languages. Then, $\mathcal{L} \in TxtFex$ implies $\mathcal{L} \in TxtFb^1Ex$.*

Proof. Throughout this proof, let $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$ with or without decorations range over indexed families.

The proof is done in three major steps. First, we show that every *TxtFex*-inferable indexed family is *TxtEx*-learnable, too (cf. Lemma 1). Note that this result also nicely contrasts Theorem 1. Next, we point out another peculiarity of indexed families consisting of infinite languages only. That is, we prove them to be *TxtEx*-identifiable by an IIM that never *overgeneralizes* provided the hypothesis space is appropriately chosen (cf. Lemma 2). Finally, we demonstrate the assertion stated in the theorem.

Lemma 1. *Let $\mathcal{L}$ be an indexed family and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for $\mathcal{L}$. Then $\mathcal{L} \in TxtFex_{\mathcal{H}}$ implies $\mathcal{L} \in TxtEx_{\mathcal{H}}$.*

Proof. First, we consider the hypothesis space $\tilde{\mathcal{H}}$ obtained from $\mathcal{H}$ by canonically enumerating all finite intersections of hypotheses from $\mathcal{H}$. Now, let M be any IIM witnessing $\mathcal{L} \in TxtFex_{\mathcal{H}}$. An IIM $\tilde{M}$ that $TxtEx_{\tilde{\mathcal{H}}}$-infers $\mathcal{L}$ can be easily defined as follows. Let $L \in \text{range}(\mathcal{L})$, let $T \in text(L)$, and $x \in \mathbb{N}$.

IIM $\tilde{M}$: "On input T_x do the following: Compute successively $j_y = M(T_y)$ for all $y = 0, \ldots, x$. For every $j_y \neq ?$ test whether or not $T_x^+ \subseteq h_{j_y}$. Let *Cons* be the set of all hypotheses passing this test.

If $Cons = \emptyset$, output ?. Otherwise, output the canonical index in $\tilde{\mathcal{H}}$ for $\bigcap Cons$."

We leave it to the reader to verify that $\tilde{M}$ witnesses $\mathcal{L} \in TxtEx_{\tilde{\mathcal{H}}}$. Finally, the $TxtEx_{\mathcal{H}}$-inferability of $\mathcal{L}$ directly follows from Proposition 1 in Lange and Zeugmann [24], and thus Lemma 1 is proved. ∎

Lemma 2. *Let $\mathcal{L}$ be an indexed family exclusively containing infinite languages such that $\mathcal{L} \in TxtEx$. Then there are a hypothesis space $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ and an IIM M such that*

(1) *M $TxtEx_{\mathcal{H}}$-infers $\mathcal{L}$,*

(2) *for all $L \in \text{range}(\mathcal{L})$, all $T \in text(L)$ and all $y, z \in \mathbb{N}$, if $? \neq M(T_y) \neq M(T_{y+z})$ then $T_{y+z}^+ \not\subseteq h_{M(T_y)}$.*

(3) *for all $L \in \text{range}(\mathcal{L})$, all $T \in text(L)$ and all $y, z \in \mathbb{N}$, if $? \neq M(T_y)$ then $M(T_{y+z}) \neq ?$.*

Proof. Let $\mathcal{L} \in TxtEx$. Without loss of generality, we may assume that there is an IIM M witnessing $\mathcal{L} \in TxtEx_{\mathcal{L}}$ (cf. [24]). By Angluin's [2] characterization of *TxtEx*, there is a uniformly recursively generable family $(\mathcal{T}_j^y)_{j,y\in\mathbb{N}}$ of finite telltale sets such that

(α) for all j, $y \in \mathbb{N}$, $\mathcal{T}_j^y \subseteq \mathcal{T}_j^{y+1} \subseteq L_j$,

(β) for all $j \in \mathbb{N}$, $\mathcal{T}_j = \lim_{y\to\infty}(\mathcal{T}_j^y)$ exists,

(γ) for all j, $k \in \mathbb{N}$, $\mathcal{T}_j \subseteq L_k$ implies $L_k \not\subset L_j$.

Using this family $(\mathcal{T}_j^y)_{j,y\in\mathbb{N}}$, we define the desired hypothesis space $\mathcal{H} = (h_{\langle j,y\rangle})_{j,y\in\mathbb{N}}$ as follows. We specify the languages enumerated in $\mathcal{H}$ via their characteristic functions $f_{h_{\langle j,y\rangle}}$. Let s_0, s_1, s_2, ... be the lexicographically ordered enumeration of all and only the strings from Σ^*. For all j, $y \in \mathbb{N}$, and all $s_z \in \Sigma^*$ we set:

$$f_{h_{\langle j,y\rangle}}(s_z) = \begin{cases} 1, & z \leq y,\ s_z \in L_j, \\ 1, & z > y,\ s_z \in L_j,\ \mathcal{T}_j^z = \mathcal{T}_j^y, \\ 0, & \text{otherwise.} \end{cases}$$

Since $(\mathcal{T}_j^y)_{j,y\in\mathbb{N}}$ is a uniformly recursively generable family of finite sets and since $\mathcal{L}$ is an indexed family, $\mathcal{H}$ is also an indexed family. Furthermore, by construction we directly obtain that for all j, $y \in \mathbb{N}$, $h_{\langle j,y\rangle}$ is either a finite language or $h_{\langle j,y\rangle} = L_j$. Moreover, $h_{\langle j,y\rangle}$ is finite iff the telltale set $\mathcal{T}_j^y \neq \mathcal{T}_j$.

Next, we define the desired IIM M. Let $L \in \text{range}(\mathcal{L})$, $T \in \textit{text}(L)$, and $x \in \mathbb{N}$.

```
IIM M: "On input $T_x$ proceed as follows:
    If $x = 0$ or $M(T_{x-1}) = ?$ then set $h = ?$, and execute Instruction (B); else, goto (A).
    (A) Let $\langle j,y\rangle = M(T_{x-1})$. Check whether or not $T_x^+ \subseteq h_{\langle j,y\rangle}$. In case it is, output
        $\langle j,y\rangle$. Otherwise, set $h = M(T_{x-1})$, and goto (B).
    (B) For all pairs $\langle j,y\rangle \le x$, (ordered by their Cantor numbers) test whether or not
        $\mathcal{T}_j^y \subseteq T_x^+ \subseteq h_{\langle j,y\rangle}$ until the first such pair is found; then output it.
        If all pairs $\langle j,y\rangle \le x$ failed, then output $h$."
```

By definition, M is recursive and fulfills Assertions (2) and (3). It remains to show that M witnesses $\mathcal{L} \in \textit{TxtEx}_{\mathcal{H}}$. Let $L \in \text{range}(\mathcal{L})$ and let $T \in \textit{text}(L)$.

Claim 1. *M converges when fed T.*

Let $j_0 = \min\{j \mid j \in \mathbb{N},\ L = L_j\}$, and let $y_0 = \min\{y \mid y \in \mathbb{N},\ \mathcal{T}_{j_0} = \mathcal{T}_{j_0}^y\}$. Since $T \in \textit{text}(L)$, there must be an $x \ge \langle j_0, y_0\rangle$ such that $\mathcal{T}_{j_0} \subseteq T_x^+$ is fulfilled. Thus past point x, M never outputs ?, and, in Step (B), it never outputs a hypothesis $\langle j,y\rangle > \langle j_0,y_0\rangle$. Moreover, if a guess $\langle j,y\rangle$ has been output and is abandoned later, say on T_z then $T_z^+ \not\subseteq h_{\langle j,y\rangle}$. Thus, it will never be repeated in any subsequent learning step. Finally, at least $\langle j_0,y_0\rangle$ can never be rejected, and thus M has to converge.

Claim 2. *If M converges, say to $\langle j,y\rangle$, then $h_{\langle j,y\rangle} = L$.*

Suppose the converse, i.e., M converges to $\langle j,y\rangle$ but $h_{\langle j,y\rangle} \neq L$. Obviously, $h_{\langle j,y\rangle}$ cannot be a finite languages, since L is infinite, and thus $T_x^+ \subseteq h_{\langle j,y\rangle}$ is eventually contradicted. Consequently, $h_{\langle j,y\rangle}$ describes an infinite language, and hence, by construction of $\mathcal{H}$ we know that $h_{\langle j,y\rangle} = L_j$. Now, since M has converged, it must have verified $\mathcal{T}_j \subseteq T_x^+ \subseteq L$, too. Thus, Condition (γ) immediately implies $L \not\subset L_j = h_{\langle j,y\rangle}$. Taking $L \neq h_{\langle j,y\rangle}$ into account we have $L \setminus h_{\langle j,y\rangle} \neq \emptyset$, contradicting $T_x^+ \subseteq h_{\langle j,y\rangle}$ for all x. Hence, Lemma 2 is proved. ▮

Now, we are ready to prove the theorem, i.e., $\textit{TxtFex} \subseteq \textit{TxtFb}^1\textit{Ex}$ when restricted to indexed families containing only infinite languages.

Let $\mathcal{L} \in \textit{TxtFex}$, and therefore, by Lemmata 1 and 2, we know that there are an IIM and a hypothesis space $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ such that M fulfills (1) through (3) of Lemma 2.

The desired simulation is based on the following idea. The feedback learner M' aims to simulate the machine M. This is done by successively computing a candidate for an initial segment of the lexicographically ordered text of the target language L. If such a candidate has been found, it is fed to the IIM M. If M computes a hypothesis j (referred to as *ordinary hypothesis*), the feedback learner outputs it together with the initial segment used to compute it. Then, M' switches to the so-called test mode, i.e., it maintains this hypothesis as long as it is not contradicted by the data received. Otherwise, the whole process has to be iterated. Now, there are two difficulties we have to overcome. First, we must avoid M' uses the same candidate for an initial segment more then once. This is done by memorizing the misclassified string as well as the old candidate for an initial segment in an *auxiliary* *hypothesis*. Additionally, since M' is only allowed to query one string at a time, auxiliary hypotheses are also used to reflect the results of the queries made until a new sufficiently large initial segment is found. Second, the test phase cannot be exclusively realized by using the actual strings received, since then finitely many strings maybe overlooked. Thus, during the test phase M' has to query one string at a time, too. Obviously, M' cannot use its actual ordinary hypothesis j for computing all the queries needed. Instead, each actual string received is used for computing a query s. If s has been already provided, it is tested whether or not $s \in h_j$. But what if $s \in L$ but did not yet appear in the data provided so far? Clearly, we cannot check $s \notin h_j$, since this would eventually force M' to reject a correct hypothesis, too. Instead, we have to ensure that at least all strings s that are negatively answered are queried again.

The feedback learner M' uses the hypothesis space $\hat{\mathcal{H}} = (\hat{h}_r)_{r\in\mathbb{N}}$ defined as follows. Let $s_0,\ s_1,\ s_2,\ \ldots$ denote the lexicographically ordered enumeration of all strings in Σ^*. Let $F_0,\ F_1,\ F_2,\ \ldots$ be any effective enumeration of all non-empty finite subsets of Σ^*. For every $\ell \in \mathbb{N}$, let $rf(F_\ell)$ be the *repetition free* enumeration of all the elements of F_ℓ in lexicographical order. Let $\hat{h}_{2\langle j,\ell\rangle} = h_j$ for all $j,\ \ell \in \mathbb{N}$, i.e., even indices encode *ordinary hypotheses*. The underlying semantics is as follows: The ordinary hypothesis $2\langle j,\ell\rangle$ represents the fact that the simulated IIM M is outputting the guess j when fed $rf(F_\ell)$. Odd indices are used for *auxiliary hypotheses*. For all $\ell,\ y,\ z \in \mathbb{N}$, we set $\hat{h}_{2\langle \ell,y,z\rangle+1} = F_\ell$. The first component ℓ encodes that all strings belonging to F_ℓ have been already presented. Both y and z are counters that M' uses to compute its queries. For the sake of readability, we introduce the following conventions. When M' outputs an ordinary hypothesis, say $2\langle j,\ell\rangle$, we instead say that M' is guessing the pair (j, F_ℓ). Similarly, if M' is outputting an auxiliary hypothesis, say $2\langle \ell,y,z\rangle + 1$, we say that M' is guessing the triple (F_ℓ, y, z).

Assume any recursive function $g : \Sigma^* \to \mathbb{N}$ such that for each $L \in \text{range}(\mathcal{L})$ and for each $\ell \in \mathbb{N}$, there exist infinitely many $w \in L$ such that $g(w) = \ell$.

Now, we define the desired feedback learner M'. Let $L \in \text{range}(\mathcal{L})$, $T = (w_n)_{n\in\mathbb{N}} \in$ *text*(L), and $n \in \mathbb{N}$. We define M' in stages, where Stage n conceptually describes M'_n.

Stage 0. On input w_0 do the following. Output the triple $(\{w_0\}, 0, 0)$, and goto Stage 1.

Stage n, $n \geq 1$. M' receives as input j_{n-1} and the $(n+1)$st element w_n of T.

CASE A. j_{n-1} is an ordinary hypothesis, say the pair (j, F).

Test whether or not $w_n \in h_j$. If not, goto $(\alpha 3)$. Otherwise, query '$s_{g(w_n)}$.' If the answer is 'NO,' then goto $(\alpha 1)$. If the answer is 'YES,' test whether or not $s_{g(w_n)} \in h_j$. If it is, execute $(\alpha 1)$. Else, goto $(\alpha 2)$.

$(\alpha 1)$ Output the ordinary hypothesis (j, F).

$(\alpha 2)$ Set $F := F \cup \{s_{g(w_n)}\}$, and $z = |F|$. Output the auxiliary hypothesis (F, z, z).

$(\alpha 3)$ Set $F := F \cup \{w_n\}$, and $z = |F|$. Output the auxiliary hypothesis (F, z, z).

CASE B. j_{n-1} is an auxiliary hypothesis, say the triple (F, y, z).

Set $F := F \cup \{w_n\}$ and check whether or not $y \geq 0$. In case it is, goto $(\beta 1)$. Else, execute $(\beta 2)$.

$(\beta 1)$ Query 's_{z-y}.' If the answer is 'YES,' then set $F := F \cup \{s_{z-y}\}$. Output the auxiliary hypothesis $(F, y-1, z)$.

($\beta 2$) Compute $M(rf(F))$ and test whether or not $M(rf(F)) \neq ?$. In case it is, let $j = M(rf(F))$ and output the ordinary hypothesis (j, F).
Otherwise, let $z := z + 1$ and query 's_z.' If the answer is 'YES,' then set $F := F \cup \{s_z\}$. Output the auxiliary hypothesis $(F, -1, z)$.

By definition, M' is a feedback learner. By construction, if M' rejects an ordinary hypothesis then an inconsistency with the data presented has been detected. Moreover, every ordinary hypothesis rejected once by M' will never be output again. It remains to show that M' witnesses $\mathcal{L} \in \mathit{TxtFb}^1\mathit{Ex}_{\hat{\mathcal{H}}}$. Let $L \in \mathrm{range}(\mathcal{L})$, and $T \in \mathit{text}(L)$.

Claim 1. *Let $(F', -1, z')$ be an auxiliary hypothesis output by M', say in Stage z. Then, for all $\ell \leq z'$, $s_\ell \in T_z^+$ implies $s_\ell \in F'$.*

Recall that, by construction, M' outputs in all the Stages $z - z'$, $z - z' + 1$, …, z auxiliary hypotheses, too, and queries s_0, s_1, …, $s_{z'}$, respectively (cf. Case B). Thus, for all $\ell \leq z'$, if $s_\ell \in T^+_{z-z'+\ell}$ the answer to the query must be 'YES', and therefore $s_\ell \in F'$ (cf. Case B). On the other hand, if $s_\ell \in T_z^+ \setminus T^+_{z-z'+\ell}$, then s_ℓ is presented after the query has made for it, and thus it is memorized, too (cf. ($\beta 1$)). This proves Claim 1.

Furthermore, since two successively output auxiliary hypotheses are definitely different, M' cannot converge to an auxiliary hypothesis.

Claim 2. *If M' converges, say to the pair (j, F), then $h_j = L$.*

By construction, $j = M(rf(F))$, and thus it suffices to prove that $L = h_j$. Suppose the converse, i.e., $L \neq h_j$. Let y_0 be the least y such that M' outputs the ordinary hypothesis (j, F) in Stage y. Obviously, $F \subseteq L$ by construction. By Lemma 2, Assertion (2), we know $L \not\subseteq h_j$. Thus, $L \setminus h_j \neq \emptyset$, and hence there must be a string $s_\ell \in L \setminus h_j$. Since $T = (w_n)_{n \in \mathbb{N}} \in \mathit{text}(L)$, there exists a $z \in \mathbb{N}$ with $w_z = s_\ell$. If $z > y_0$, then $s_\ell \notin h_j$ is verified in Stage z (cf. Case A), a contradiction. Now, suppose $z \leq y_0$. Taking into account that $|T^+| = \infty$ and that $g(v) = \ell$ for infinitely many $v \in T^+ = L$, there must be an $r \in \mathbb{N}$ such that $g(w_{y_0+r}) = \ell$. Thus, the query 's_ℓ' is made in Stage $y_0 + r$. But $s_\ell \in T^+_{y_0} \subseteq T^+_{y_0+r}$, and hence the answer to it is 'YES,' and $s_\ell \in h_j$ is tested, too (cf. Case A). Therefore, M' must execute ($\alpha 2$), and cannot converge to (j, F). This proves Claim 2.

Claim 3. *M' outputs an ordinary hypothesis in infinitely many stages.*

Suppose the converse, i.e., there is a least $z \in \mathbb{N}$ such that M' outputs in every Stage $z+n$, $n \in \mathbb{N}$, an auxiliary hypothesis. By Lemma 2, Assertion (1), M learns L from all its texts. Let T^L be L's lexicographically ordered text. Let y be the least η such that $M(T^L_\eta) = j$ and $L = h_j$. Hence, Assertion (2) of Lemma 2 implies $M(T^L_y \diamond \sigma) = j$ for all finite sequences σ satisfying $\sigma^+ \subseteq L$. Let $m_0 = \max\{k \mid k \in \mathbb{N},\ s_k \in T_y^{L,+}\}$, and let x_0 be the least x with $T_y^{L,+} \subseteq T_x^+$.

By construction, there is an $r > \max\{z, x_0, m_0\}$ such that M' in Stage r must output an auxiliary hypothesis of the form $(F', -1, z')$ with $z' \geq m_0$. Hence, $\{s_\ell \mid s_\ell \in T_r^+,\ \ell \leq z'\} \subseteq F'$ by Claim 1. Moreover, $T_y^{L,+} \subseteq T_r^+$ because of $r \geq x_0$ and $T_y^{L,+} \subseteq T^+_{x_0}$, and hence $T_y^{L,+} \subseteq F'$, since $m_0 \leq z'$ and $T_y^{L,+} = \{s_\ell \mid \ell \leq m_0,\ s_\ell \in L\}$. Therefore, M' simulates M on input $rf(F')$ in Stage $r + 1$ (cf. Case B, Instruction ($\beta 2$)). By the choice of T_y^L, and since T_y^L is an initial segment of $rf(F')$ we know that $M(rf(F')) = j$, and thus M' must output an ordinary hypothesis, a contradiction. Thus, Claim 3 follows.

Claim 4. *M' converges.*

Suppose, M' diverges. Then, M' must output infinitely many ordinary hypotheses, since otherwise Claim 3 is contradicted. Let y, m_0, x_0 be as in the proof of Claim 3. Consider the the minimal $r > x_0$ such that M', when successively fed T_r, has already output its m_0-th ordinary hypothesis, say (j', F). Thus, $|F| \geq m_0$ in accordance with the definition of M'. Since M' diverges, the guess (j', F) is abandoned in some subsequent stage, say in Stage ϱ, $\varrho > r$. Thus in Stage ϱ, M' outputs an auxiliary hypothesis, say $(F', |F'|, |F'|)$. Note that $F \subset F'$ (cf. Case A, Instructions $(\alpha 2)$ and $(\alpha 3)$). In all the Stages $\varrho + 1$, $\varrho + 2$, $\ldots$, $\varrho + m_0$, $\ldots$, and $\varrho + |F'| + 1$, M' outputs auxiliary hypotheses, too (cf. Case B, Instruction $(\beta 1)$). Moreover, in Stage $\varrho + |F'| + 1$, M' outputs an auxiliary hypothesis having the form $(F'', -1, |F'|)$. Applying *mutatis mutandis* the same argumentation as in Claim 3, we obtain $T_y^L \subseteq F''$. Therefore in the next stage, M' simulates M when fed a finite sequence τ having the initial segment T_y^L (cf. Case B, Instruction $(\beta 2)$). Again, by Lemma 2, Assertion (2), $M(\tau) = j$ follows, and thus M' outputs the ordinary hypothesis (j, F''). But $h_j = L$ implies that the hypothesis (j, F'') cannot be abandoned, since otherwise an inconsistency to T would be detected. Hence, M' converges, a contradiction. This proves Claim 4. ▮

Hence, in the case of *indexed* families of infinite languages, the hierarchy of Theorem 2 collapses for $k \geq 2$; furthermore, again, for *indexed* families of infinite languages, the *expansion* of Gold's [18] model, which not only has unrestricted access to the data base, but which also allows *finitely many correct grammars* output in the limit, achieves no more learning power than *feedback* identification with only *one* query of the database. Moreover, our proof shows actually a bit more. That is, for indexed families of infinite languages *conservative* learning does not constitute a restriction provided the hypothesis space is appropriately chosen (cf. Lemma 2). As a matter of fact, this result is nicely inherited by our feedback learner defined in the proof above. It also never overgeneralizes; thus what we have actually proved is the equality of *TxtFex* and *conservative feedback* inference with only one query per time.

Next, we compare feedback inference and $TxtFex^a$-identification in dependence on the number of anomalies allowed.

Theorem 5. *$TxtFb^0Ex^{a+1} \setminus TxtFex^a \neq \emptyset$, for all $a \in \mathbb{N}$.*

Hence, for some concept domains, the model of *iterative* learning, where we tolerate $a + 1$ anomalies in the single final grammar, is competent, but the expanded Gold [18] model, where we allow unlimited access to the database and finitely many grammars in the limit each with no more than a anomalies, is not. A little extra anomaly tolerance nicely buys, in such cases, no need to remember any past database history or to query it!

Proof. Let $\mathcal{L} = \{L \mid (\exists! x)[\langle 0, x\rangle \in L] \ \wedge \ [\langle 0, x\rangle \in L \Leftrightarrow W_x =^{a+1} L]\}$. It is easy to see that $\mathcal{L} \in TxtFb^0Ex_0^{a+1}$. However, an easy modification of the proof of $Ex^{a+1} \setminus Ex^a \neq \emptyset$ from [12] shows that $\mathcal{L} \notin TxtFex^a$. ▮

3.2. *Feedback Inference versus Bounded Example-Memory Learning*

As promised in the introductory section, the next two theorems show that, for each of these two models of bounded example-memory inference and feedback identification, there are concept class domains where that model is competent and the other is not! Theorem 6 below says that, for suitable concept domains, the feedback learning power of *one* query of the data base, where a *single, correct* grammar is found in the limit, *beats* the bounded example-memory learning power of memorizing k database items, even where *finitely many grammars* each with *finitely many anomalies* are allowed in the limit.

We start with a technical lemma pointing to combinatorial limitations of bounded example-memory learning.

LEMMA 3. *Suppose M is an k-memory bounded learning machine. Let P be a finite set, let σ be a sequence, and let Z be a set such that $2^{|Z|} > |P| * (|Z| + k)^k$. Then, either*

(a) *there exists a σ' such that* $\mathrm{content}(\sigma') \subseteq Z$ *and* $\pi_1(M_*(\sigma \diamond \sigma')) \notin P$*, or*

(b) *there exist σ', σ'' and $j \in Z$, such that* $\mathrm{content}(\sigma') = Z \setminus \{j\}$, $\mathrm{content}(\sigma'') = Z$ *and* $M_*(\sigma \diamond \sigma') = M_*(\sigma \diamond \sigma'')$.

Proof. Suppose (a) does not hold. Thus, by the pigeonhole principle, there exist τ', τ'' such that

(a) $\mathrm{content}(\tau') \cup \mathrm{content}(\tau'') \subseteq Z$,

(b) $\mathrm{content}(\tau'') \neq \mathrm{content}(\tau')$, and

(c) $M_*(\sigma \diamond \tau') = M_*(\sigma \diamond \tau'')$.

This is so since there are $2^{|Z|}$ possibilities for $\mathrm{content}(\tau)$, but at most $|P| * (|Z| + k)^k$, possibilities for $M_*(\sigma \diamond \tau)$. Let τ', τ'' be such that (a) through (c) are satisfied. Suppose $j \in \mathrm{content}(\tau'') \setminus \mathrm{content}(\tau')$. Now let τ''' be such that $\mathrm{content}(\tau''') = Z \setminus \{j\}$. Taking $\sigma' = \tau' \diamond \tau'''$ and $\sigma'' = \tau'' \diamond \tau'''$ proves the lemma. ∎

Now, we are ready to prove the first of the two theorems announced.

THEOREM 6. *$TxtFb^1Ex \setminus TxtBem^kFex^* \neq \emptyset$, for all $k \in \mathbb{N}$. Moreover this separation can be witnessed by a class consisting of only infinite languages.*

Proof. For any language L, let $C_L^i = \{x \mid \langle i, x\rangle \in L\}$. Intuitively, C_L^i is the i-th cylinder of L. We say that e is *nice* iff

(a) $C_{W_e}^0 = \{e\}$, and

(b) $C_{W_e}^1 \cap C_{W_e}^2 = \emptyset$.

The desired class $\mathcal{L}$ is defined as follows. Let $\mathcal{L}_1 = \{L \mid |L| = \infty \ \wedge\ (\exists \text{ nice } e)[L = W_e]\}$, and let $\mathcal{L}_2 = \{L \mid |L| = \infty \ \wedge\ (\exists e')[C_L^1 \cap C_L^2 = \{e'\} \ \wedge\ L = W_{e'}]\}$. We set $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$.

It is easy to verify that $\mathcal{L} \in TxtFb^1Ex$. We omit the details.

Next, we show that $\mathcal{L} \notin TxtBem^kFex^*$. Suppose the converse, i.e., there is an IIM M that $TxtBem^kFex^*$-identifies $\mathcal{L}$. For a sequence $\sigma = x_0, x_1, \ldots, x_\ell$, let $M_0(\sigma) = M(x_0)$, and for $i < \ell$, let $M_{i+1}(\sigma) = M(M_i(\sigma), x_{i+1})$.

For any finite sequence τ, let $\mathrm{ProgSet}(M, \tau) = \{\pi_1(M_*(\sigma)) \mid \sigma \subseteq \tau\}$, and we define for any text T the set $\mathrm{ProgSet}(M, T)$ similarly.

Then by implicit use of the Operator Recursion Theorem (cf. [8, 10]) there exists a recursive 1–1 increasing function p, such that $W_{p(\cdot)}$ may be defined as follows ($p(0)$ will be nice).

Enumerate $\langle 0, p(0)\rangle$ in $W_{p(0)}$. Let σ_0 be such that $\mathrm{content}(\sigma_0) = \{\langle 0, p(0)\rangle\}$. Let $W_{p(0)}^s$ denote $W_{p(0)}$ enumerated before Stage s. Let $\mathrm{avail}_0 = 1$. Intuitively, avail_s denotes a number such that for all $j \geq \text{avail}_s$, $p(j)$ is available for use in the diagonalization at the beginning of Stage s. Go to Stage 0.

```
Stage $s$.
    (* Intuitively, if infinitely many stages are there, i.e. Step 2 succeeds infinitely often,
      then $W_{p(0)} \in \mathcal{L}_1$ witnesses the diagonalization. If Stage $s$ starts but does not finish,
      then each of $W_{p(j_i)}$, $1 \leq i \leq k$, as defined in Steps 3 and 4, is in $\mathcal{L}_2$, and one of them
      witnesses the diagonalization. *)
1.  Let $P_s = \text{ProgSet}(M, \sigma_s)$.
    Dovetail Steps 2 and 3–4, until Step 2 succeeds. If Step 2 succeeds, then go to Step 5.
2.  Search for a $\sigma$ extending $\sigma_s$ such that
        (a) $C^0_{\text{content}(\sigma)} = \{p(0)\}$,
        (b) $C^1_{\text{content}(\sigma)} \cap C^2_{\text{content}(\sigma)} = \emptyset$,
        (c) $\text{ProgSet}(M, \sigma) \neq P_s$.
3.  Let $m_0 = \text{avail}_s$.
    Let $k = |P_s| + 1$, $\tau_0 = \sigma_s$.
    Search for $m_1, m_2, \ldots, m_k$, $j_1, j_2, \ldots, j_k$, $\tau_1, \tau_2, \ldots, \tau_k$, $\tau'_1, \tau'_2, \ldots, \tau'_k$, such that,
        (a) For $1 \leq i \leq k$, $m_{i-1} < j_i < m_i$,
        (b) For $1 \leq i \leq k$, $\text{content}(\tau_i) = \{\langle 1, p(j)\rangle \mid m_{i-1} \leq j < m_i \ \wedge \ j \neq j_i\}$.
        (c) For $1 \leq i \leq k$, $\text{content}(\tau'_i) = \{\langle 1, p(j)\rangle \mid m_{i-1} \leq j < m_i\}$.
        (d) For $1 \leq i \leq k$, $M_*(\tau_0 \diamond \tau_1 \diamond \ldots \tau_{i-1} \diamond \tau_i) = M_*(\tau_0 \diamond \tau_1 \diamond \ldots \tau_{i-1} \diamond \tau'_i)$
        (e) For $1 \leq i \leq k$, $\pi_1(M_*(\tau_0 \diamond \tau_1 \diamond \ldots \tau_{i-1} \diamond \tau_i)) \in P_s$.
4.  Let $m_1, m_2, \ldots, m_k$, $j_1, j_2, \ldots, j_k$, $\tau_1, \tau_2, \ldots, \tau_k$, $\tau'_1, \tau'_2, \ldots, \tau'_k$, be as found in Step 3.
    Let $Y = \text{content}(\sigma_s) \cup \{\langle 1, p(j)\rangle \mid m_0 \leq j \leq m_k\} \setminus \{\langle 1, p(j_i)\rangle \mid 1 \leq i \leq k\}$.
    For $1 \leq i \leq k$, enumerate $Y \cup \{\langle 1, p(j_i)\rangle, \langle 2, p(j_i)\rangle\}$ in $W_{p(j_i)}$.
    For $x = 0$ to $\infty$ do
        For $1 \leq i \leq k$, enumerate $\langle 3+i, x\rangle$ in $W_{p(j_i)}$.
    Endfor
5.  Enumerate $\text{content}(\sigma) \cup \{\langle 3, s\rangle\}$ in $W_{p(0)}$.
    Let $\sigma_{s+1}$ be an extension of $\sigma$ such that $\text{content}(\sigma_{s+1}) = \text{content}(\sigma) \cup \{\langle 3, s\rangle\}$.
    Let $z = m_k$, if Step 3 succeeded; otherwise $z = 0$.
    Let $\text{avail}_{s+1} = 1 + \text{avail}_s + z + \max\{j \mid p(j) \in C^1_{\text{content}(\sigma_{s+1})} \cup C^2_{\text{content}(\sigma_{s+1})}\}$.
End Stage $s$.
```

We now consider two cases.

Case 1. All stages terminate.

In this case clearly, W_e is nice and infinite, and thus in $\mathcal{L}_1$. Also, $T = \bigcup_s \sigma_s$ is a text for W_e. However, M on T outputs infinitely many programs.

Case 2. Stage s starts but does not terminate.

In this case we first claim that Step 3, must have succeeded. This follows directly from repeated use of Lemma 3. Let $k, m_i, j_i, \tau_i, \tau'_i$ be as in Step 4. Now, for $1 \leq i \leq k$, $W_{p(j_i)} \in \mathcal{L}_2$ and $W_{p(j_i)}$ are pairwise infinitely different. Thus, by the pigeonhole principle, there exists an i, $1 \leq i \leq k$, such that $\mathrm{ProgSet}(M, \sigma_s)$ does not contain a grammar for a finite variant of $W_{p(j_i)}$. Fix one such i.

Let T_i be a text for $W_{p(j_i)} \setminus \{\langle 1, p(j_i)\rangle\}$. Furthermore, let

$$T_i' = \tau_0 \diamond \tau_1 \diamond \ldots \diamond \tau_{i-1} \diamond \tau_i' \diamond \tau_{i+1} \diamond \ldots \diamond \tau_k \diamond T_i$$

$$T_i'' = \tau_0 \diamond \tau_1 \diamond \ldots \diamond \tau_{i-1} \diamond \tau_i \diamond \tau_{i+1} \diamond \ldots \diamond \tau_k \diamond T_i$$

Note that T_i' is a text for $W_{p(j_i)}$. However, we have $\mathrm{ProgSet}(M, T_i') = \mathrm{ProgSet}(M, T_i'') = \mathrm{ProgSet}(M, \sigma_s)$ (the first equality follows from the definition of memory bounded and the choice of τ_i, τ_i' in Step 3; the second equality holds since Step 2 did not succeed in Stage s). Thus, M does not TxtFex^*-identify $W_{p(j_i)}$.

From the above cases we have that $\mathcal{L} \notin \mathit{TxtBem}^k\mathit{Fex}^*$. ∎

Next we show the second theorem announced above. Theorem 7 below says that, for suitable concept domains, the bounded example-memory learning power of memorizing *one* item from the data base history *beats* the feedback learning power of k queries of the database, even where the final grammar is allowed to have *finitely many anomalies.* It is currently open whether or not $\mathit{TxtFb}^k\mathit{Ex}^*$ in Theorem 7 can be replaced by $\mathit{TxtFb}^k\mathit{Fex}^*$.

Theorem 7. *$\mathit{TxtBem}^1\mathit{Ex} \setminus \mathit{TxtFb}^k\mathit{Ex}^* \neq \emptyset$, for all $k \in \mathbb{N}$. Moreover this separation can be witnessed by a class consisting of only infinite languages.*

Proof. For a query asking function Q_k, we denote by $\mathrm{Questions}(Q_k, q, x)$ the questions asked by $Q_k(q, x)$. For all L, let C_L^i denote the set $\{x \mid \langle i, x\rangle \in L\}$. We say that e is *nice* iff $C_{W_e}^0 = \{e\}$, and $C_{W_e}^1 = \emptyset$.

Let $\mathcal{L}_1 = \{L \mid |L| = \infty \ \wedge \ (\exists \text{ nice } e)[L = W_e]\}$. Furthermore, let $\mathcal{L}_2 = \{L \mid |L| = \infty \ \wedge \ (\exists \text{ nice } e)(\exists w, m)[C_L^1 = \{w\} \ \wedge \ \max C_L^2 = m < w \ \wedge \ (L = W_e \cup \{\langle 1, w\rangle\})]\}$, and let $\mathcal{L}_3 = \{L \mid |L| = \infty \ \wedge \ (\exists w, m)[C_L^1 = \{w\} \ \wedge \ \max C_L^2 = m \geq w \ \wedge \ L = W_m]\}$. Finally, we set $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$.

It is easy to show that $\mathcal{L} \in \mathit{TxtBem}^1\mathit{Ex}$. The machine just needs to remember $\max C_L^2$; C_L^0, C_L^1 can be padded onto the output program. From C_L^0, C_L^1, $\max C_L^2$, one can easily find a grammar for L. We omit the details.

Next, we show $\mathcal{L} \notin \mathit{TxtFb}^k\mathit{Ex}^*$. The intuitive idea behind the formal proof below is that no feedback learner can memorize what the maximal m with $m = \langle 2, x\rangle \in L$ is. Suppose by way of contradiction that M (with associated query asking function Q_k) is a k-feedback machine which $\mathit{TxtFb}^k\mathit{Ex}^*$-identifies $\mathcal{L}$. For $\sigma = x_0, x_1, \ldots, x_\ell$, let $M_0(\sigma) = M(x_0)$ and for $i < \ell$ let $M_{i+1}(\sigma) = M(M_i(\sigma), A_k^i(Q_k(M_i(\sigma), x_{i+1})), x_{i+1})$, where A_k^i answers the questions based on whether the corresponding elements appear in $\{x_j \mid j \leq i\}$. By the Operator Recursion Theorem [8, 10] there exists a recursive 1–1 increasing function p such that $W_{p(\cdot)}$ may be defined as follows. Initially enumerate $\langle 0, p(0)\rangle$ in $W_{p(0)}$. Let σ_0 be such that $\mathrm{content}(\sigma_0) = \{\langle 0, p(0)\rangle\}$. Let avail $= 1$. Intuitively, avail denotes a number such that, for all $j \geq$ avail, $p(j)$ is available for use in the diagonalization. Go to Stage 0.

Stage s.

(* Intuitively, if infinitely many stages are there, (i.e. Step 2 succeeds infinitely often) then $W_{p(0)} \in \mathcal{L}_1$ witnesses the diagonalization. If Stage s starts but does not finish,

```
      then let $\ell$ be as in Step 3 of Stage $s$. If there are infinitely many substages in Stage $s$
      (i.e. Step 3.2 succeeds infinitely often in Stage $s$), then $(W_{p(0)} \cup \{\langle 1, \ell \rangle\}) \in \mathcal{L}_2$ witnesses
      the diagonalization. Otherwise, one of $W_{p(j_i)}, W_{p(j_i')}$, $i \leq k$, will be in $\mathcal{L}_3$, and witness
      the diagonalization. *)
1.  Dovetail Steps 2 and 3. If and when Step 2 succeeds, go to Step 4.
2.  Search for a $\sigma$ extending $\sigma_s$ such that
        $C^0_{\mathrm{content}(\sigma)} = \{p(0)\}$,
        $C^1_{\mathrm{content}(\sigma)} = \emptyset$, and
        $M_*(\sigma) \neq M_*(\sigma_s)$.
3.  Let $\ell = 1 + \max C^2_{\mathrm{content}(\sigma_s)}$.
    Let $\tau_0$ be such that content$(\tau_0) = \{\langle 1, \ell \rangle\}$.
    Go to Substage 0.
        Substage $t$.
        3.1    Dovetail Steps 3.2, 3.3, and 3.4 until Step 3.2 succeeds. If and when
               Step 3.2 succeeds, then go to Step 3.5.
        3.2    Search for a $\tau$ such that
                   content$(\tau) \subseteq \{\langle 3, x \rangle \mid x \in \mathbb{N}\}$, and
                   $M_*(\sigma_s \diamond \tau_t \diamond \tau) \neq M_*(\sigma_s \diamond \tau_t)$.
        3.3    Let $q = M_*(\sigma_s \diamond \tau_t)$.
               Let Ques $= \bigcup_{\gamma \diamond y \subseteq \tau_t}$ Questions$(Q_k, M_*(\sigma_s \diamond \gamma), y)$.
               Set avail $= 1 +$ avail $+ \ell + \max \{x \mid \langle 2, p(x) \rangle \in$ Ques$\}$
               (* Note that this implies $p(\mathrm{avail}) > \ell$ and any $p(j)$ such that a question
                  of the form $\langle 2, p(j) \rangle$ was asked by $M$ (using $Q_k$) on $\tau$ such that $\sigma_s \subset$
                  $\tau \subseteq \sigma_s \diamond \tau_t$. *)
               For $i \leq k$, let $j_i =$ avail $+ i$.
               For $i \leq k$, let $j_i' =$ avail $+ k + 1 + i$.
               Let avail $=$ avail $+ 2 * (k + 1)$.
               For $i \leq k$, let $O_i = \{\langle 3, x \rangle \mid (\forall B \mid B \subseteq C^3_{\mathbb{N}})[M(q, A_k(Q_k(q, \langle 3, x \rangle)), \langle 3, x \rangle)\downarrow$,
                  where $A_k$ answers queries by $Q_k$ based on whether the correspond-
                  ing elements appear in content$(\sigma_s \diamond \tau_t) \cup B$ and $\{\langle 2, p(j_i) \rangle, \langle 2, p(j_i') \rangle\} \cap$
                  Questions$(Q_k, q, \langle 3, x \rangle) = \emptyset]\}$.
               For $i \leq k$, let $x_i^0, x_i^1, \ldots$, denote a 1–1 enumeration of elements of $O_i$.
               For $i \leq k$, let $W_{p(j_i)} =$ content$(\sigma_s \diamond \tau_t) \cup \{\langle 2, p(j_i) \rangle\} \cup \{x_i^{2r} \mid |O_i| > 2r\}$.
               For $i \leq k$, let $W_{p(j_i')} =$ content$(\sigma_s \diamond \tau_t) \cup \{\langle 2, p(j_i') \rangle\} \cup \{x_i^{2r+1} \mid |O_i| > 2r+1\}$.
        3.4    For $x = 0$ to $\infty$ do
                   enumerate $\langle 3, x \rangle$ in $W_{p(0)}$.
               EndFor
        3.5    If and when Step 3.2 succeeds, let $\tau$ be as found in Step 3.2.
               Enumerate content$(\tau) \cup \{\langle 3, t \rangle\}$ in $W_{p(0)}$.
               Let $S = ([W_{p(0)}$ enumerated until now $] \cap \{\langle 3, x \rangle \mid x \in \mathbb{N}\}) \cup \{\langle 1, \ell \rangle\}$.
               Let $\tau_{t+1}$ be an extension of $\tau_t \diamond \tau$ such that content$(\tau_{t+1}) = S$.
               Go to Substage $t + 1$.
        End Substage $t$.
4.  If and when Step 2 succeeds, let $\sigma$ be as found in Step 2.
    Enumerate content$(\sigma) \cup \{\langle 3, s \rangle\}$ in $W_{p(0)}$.
    Let $S = W_{p(0)}$ enumerated until now.
```

Let σ_{s+1} be an extension of σ such that content$(\sigma_{s+1}) = S$.
Go to Stage $s+1$.
End Stage s.

We now consider the following cases.

Case 1. All stages terminate.

In this case clearly, $L = W_{p(0)} \in \mathcal{L}_1$. However, on $T = \bigcup_s \sigma_s$, a text for L, M does not converge.

Case 2. Stage s starts but does not terminate.

Let ℓ be defined in Step 3 of Stage s.

Case 2.1. All substages in Stage s terminate.

In this case clearly, $L = (W_{p(0)} \cup \{\langle 1, \ell \rangle\}) \in \mathcal{L}_2$. However, on $T = \bigcup_t \sigma_s \diamond \tau_t$, a text for L, M does not converge.

Case 2.2. Substage t in Stage s starts but does not terminate.

In this case let q, j_i, j'_i, O_i, (for $i \leq k$) be as defined in Step 3.3 of Stage s, Substage t.

Now, for all all τ such that content$(\tau) \subseteq C^3_{\mathbb{N}}$, $M_*(\sigma_s \diamond \tau_t \diamond \tau) = M_*(\sigma_s \diamond \tau_t) = q$. Thus $(\forall B \mid B \subseteq C^3_{\mathbb{N}})[M(q, A_k(Q_k(q, \langle 3, x \rangle)), \langle 3, x \rangle)\downarrow]$, where A_k answers queries from Q_k based on whether the corresponding elements appear in content$(\sigma_s \diamond \tau_t) \cup B$. Moreover, taking into account that $\bigcup_{B \subseteq C^3_{\mathbb{N}}}$ Questions$(Q_k, q, \langle 3, x \rangle)$ can have at most k elements, we have that at least one of O_i's must be infinite. Let i be such that O_i is infinite. It follows that $W_{p(j_i)}$ and $W_{p(j'_i)}$ are both infinite and infinitely different from each other. Now,

(a) $M_*(\sigma_s \diamond \langle 2, p(j_i) \rangle) = M_*(\sigma_s \diamond \langle 2, p(j'_i) \rangle) = M(\sigma_s)$,

(b) $\langle 2, p(j_i) \rangle$ and $\langle 2, p(j'_i) \rangle$ are not in content$(\sigma_s \diamond \tau_t)$,

(c) for all $B \subseteq O_i$, for all texts T for B, $M_*(\sigma_s \diamond \tau_t \diamond T) = q$ and

(d) for all $B \subseteq C^3_{\mathbb{N}}$, for all texts T for B, for any τ, y such that $\sigma_s \subset \tau \diamond y \subseteq \sigma_s \diamond \tau_t \diamond T$, $Q_k(M_*(\tau), y)$ does not ask a question about $\langle 2, p(j_i) \rangle$ or $\langle 2, p(j'_i) \rangle$.

Thus, for $w \in \{j_i, j'_i\}$, for any text T for $W_{p(w)} \setminus$ content$(\sigma_s \diamond \langle 2, p(w) \rangle \diamond \tau_t)$, we have $M_*(\sigma_s \diamond \langle 2, p(w) \rangle \diamond \tau_t \diamond T) = q$. Thus, M fails to *TxtEx**-identify at least one of $W_{p(j_i)}$ and $W_{p(j'_i)}$, both of which are in $\mathcal{L}_3$.

From the above cases we have that $\mathcal{L} \notin$ *TxtFb*^{k}*Ex**. ∎

3.3. *Iterative Learning*

We start this subsection by showing that *redundancy* in the hypothesis space may considerably increase the learning power of iterative learners. Interestingly, it turns out that, redundancy may serve as a tool exploited by the iterative learner allowing it to *overgeneralize* in learning stages before convergence. Here, overgeneralization refers to the situation in which the learner outputs a description for a proper superset of the target concept. Furthermore, this phenomenon can be already observed at the fairly concrete level of indexed families.

Theorem 8. *There are an indexed family $\mathcal{L}$ and a redundant hypothesis space $\mathcal{H}$ for it such that $\mathcal{L} \in TxtItEx_{\mathcal{H}} \setminus TxtItEx_{\mathcal{L}}$*

Proof. The basic idea of the following proof consists in reducing the halting problem to a suitably chosen learning task. The desired indexed family $\mathcal{L} = (L_{\langle k,j\rangle})_{k,j\in\mathbb{N}}$ is defined as follows. Without loss of generality, for all $k \in \mathbb{N}$, assume $\Phi_k(k) > 1$. For all $k \in \mathbb{N}$, we set $L_{\langle k,0\rangle} = \{a^{k+1}\} \cup \{b^n \mid\ n \in \mathbb{N}^+\}$. For all $j \geq 1$, we distinguish the following cases.

Case 1. $\neg\ \Phi_k(k) \leq j$

We set $L_{\langle k,j\rangle} = \{a^{k+1}\}$.

Case 2. $\Phi_k(k) = x \leq j$

Then, we set $L_{\langle k,j\rangle} = \{a^{k+1}\} \cup \{b^m \mid\ 1 \leq m \leq x\}$.

Obviously, $\mathcal{L}$ is an indexed family. The non-learnability of $\mathcal{L}$ in the sense of $TxtItEx$ is due to the following facts. For every $k \in \mathbb{N}$, there is exactly one index for the infinite language $L_{\langle k,0\rangle}$. Moreover, no IIM infers $\mathcal{L}$ iteratively without overgeneralizing at least once with respect to the hypothesis space $\mathcal{L}$. Therefore, it has sometimes to shrink its guess to a finite language. But afterwards, it might receive data forcing it to output the corresponding number of the relevant infinite language. Now, every iterative IIM $TxtItEx_{\mathcal{L}}$-inferring $\mathcal{L}$ is in serious trouble, since the only available hypothesis does not suffice to memorize the fact that the shrunk guess has been provably rejected. We continue with the formal proof.

Claim 1. $\mathcal{L} \notin TxtItEx_{\mathcal{L}}$.

Suppose that there exists any IIM M which $TxtItEx_{\mathcal{L}}$-identifies $\mathcal{L}$. Let us consider M's behavior when fed the text $T = a^{k+1}, b, b^2, \ldots$ for the language $L_{\langle k,0\rangle}$. Since $\mathcal{L}$ cannot be learned without outputting at least once an overgeneralization (cf. Lange and Zeugmann [24]), there have to be indices $k, x \in \mathbb{N}$ such that $\varphi_k(k)$ is defined such that $\Phi_k(k) > x$ and M outputs the hypothesis $\langle k, 0\rangle$ after processing T_x. Obviously, T_x serves as an initial segment of a text for $L_{\langle k,\Phi_k(k)\rangle}$, too. Thus, there has to be a string $s \in L_{\langle k,\Phi_k(k)\rangle}$ such that $M(\langle k, 0\rangle, s) = \langle k, y\rangle$ for some $y \geq \Phi_k(k)$. (Note that $L_{\langle k,y\rangle} = L_{\langle k,\Phi_k(k)\rangle}$, if $y \geq \Phi_k(k)$.) On the other hand, M has, in particular, to infer the infinite language $L_{\langle k,0\rangle}$ from its text $\hat{T} = T_x, s, b, s, b^2, \ldots$ Since the string s appears infinitely many times, M outputs infinitely many times the wrong hypothesis $\langle k, y\rangle$. Thus, M fails to converge to a correct guess on $\hat{T}$, a contradiction.

Claim 2. *There is a redundant hypothesis space $\mathcal{H}$ such that $\mathcal{L} \in TxtItEx_{\mathcal{H}}$.*

We sketch the underlying idea, only. Now, assume the following hypothesis space $\mathcal{H} = (h_{\langle k,n\rangle})_{k,n\in\mathbb{N}}$ that satisfies $h_{\langle k,0\rangle} = L_{\langle k,0\rangle}$ and $h_{\langle k,n+1\rangle} = L_{\langle k,n\rangle}$ for all $k, n \in \mathbb{N}$. Hence, there are two different indices for the infinite language $L_{\langle k,0\rangle}$, namely $j_k = \langle k, 0\rangle$ and $\hat{j}_k = \langle k, 1\rangle$. Applying this *a priori* knowledge about the underlying hypothesis space, an iterative IIM M is able to handle overgeneralization. Thereby, M may use each of both semantically equivalent hypotheses to represent different stages. Clearly, as long as M is exclusively fed a^{k+1}, it outputs a canonical number of that singleton language. Now we describe how M uses the semantical equivalent hypotheses. The index j_k may be used to encode that M's last guess $L_{\langle k,0\rangle}$ is a possibly overgeneralized hypothesis which may be changed in some subsequent step. M outputs this hypothesis as long as it has no knowledge whether or not

$\Phi_k(k)$ is defined, i.e., as long as it has exclusively seen strings b^z such that $\neg\Phi_k(k) \leq z$. On the other hand, if M has been fed a string b^z satisfying $\Phi_k(k) \leq z$ then it knows for sure that $\Phi_k(k)$ is defined. After having gained this knowledge, it never outputs j_k. Instead, it either output an index for the corresponding finite language or, in case enough evidence has been presented, the index $\hat{\jmath}_k$ for $L_{\langle k,0\rangle}$. We omit the details. ∎

3.4. *The Pattern Languages*

The pattern languages (defined two paragraphs below) were formally introduced by Angluin [1] and have been widely investigated (cf., e.g., Salomaa [34, 35], and Shinohara and Arikawa [39] for an overview). Moreover, Angluin [1] proved that the class of all pattern languages is learnable in the limit from positive data. Subsequently, Nix [30] as well as Shinohara and Arikawa [39] outlined interesting applications of pattern inference algorithms. For example, pattern language learning algorithms have been successfully applied for solving problems in molecular biology (cf., e.g. Shimozono *et al.* [36], Shinohara and Arikawa [39]).

Pattern languages and finite unions of pattern languages turn out to be subclasses of Smullyan's [41]) elementary formal systems (EFS). Arikawa *et al.* [3] have shown that EFS can also be treated as a logic programming language over strings. Recently, the techniques for learning finite unions of pattern languages have been extended to show the learnability of various subclasses of EFS (cf. Shinohara [38]). From a theoretical point of view, investigations of the learnability of subclasses of EFS are important because they yield corresponding results about the learnability of subclasses of logic programs. Arimura and Shinohara [4] have used the insight gained from the learnability of EFS subclasses to show that a class of linearly covering logic programs with local variables is identifiable in the limit from only positive data. More recently, using similar techniques, Krishna-Rao [32] has established the learnability from only positive data of an even larger class of logic programs. These results have consequences for Inductive Logic Programming.[6]

Patterns and pattern languages are defined as follows (cf. Angluin [1]). Let $\mathcal{A} = \{0, 1, \dots\}$ be any non-empty finite alphabet containing at least two elements. By $\mathcal{A}^*$ we denote the free monoid over $\mathcal{A}$ (cf. Hopcroft and Ullman [19]). The set of all finite non-null strings of symbols from $\mathcal{A}$ is denoted by $\mathcal{A}^+$, i.e., $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\varepsilon\}$, where ε denotes the empty string. By $|\mathcal{A}|$ we denote the cardinality of $\mathcal{A}$. Furthermore, let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\mathcal{A} \cap X = \emptyset$. *Patterns* are non-empty strings over $\mathcal{A} \cup X$, e.g., 01, $0x_0111$, $1x_0x_00x_1x_2x_0$ are patterns. A pattern π is in *canonical form* provided that if k is the number of different variables in π then the variables occurring in π are precisely $x_0, \dots, x_{k-1}$. Moreover, for every j with $0 \leq j < k-1$, the leftmost occurrence of x_j in π is left to the leftmost occurrence of x_{j+1} in π. The examples given above are patterns in canonical form. In the sequel we assume, without loss of generality, that all patterns are in canonical form. By *Pat* we denote the set of all patterns in canonical form.

The length of a string $s \in \mathcal{A}^*$ and of a pattern π is denoted by $|s|$ and $|\pi|$, respectively. By $\#\mathrm{var}(\pi)$ we denote the number of different variables occurring in π, and by $\#_{x_i}(\pi)$ we denote the number of occurrences of variable x_i in π. If $\#\mathrm{var}(\pi) = k$, then we refer to π as a *k-variable pattern*. Let $k \in \mathbb{N}$, by Pat_k we denote the set of all *k-variable patterns*.

[6]We are grateful to Arun Sharma for bringing to our fuller attention these potential applications to ILP of learning special cases of pattern languages and finite unions of pattern languages.

Furthermore, if $\#_{x_i}(\pi) = 1$ for all variables occurring in pattern π, then we call π a *regular* pattern.

Now let $\pi \in Pat_k$, and let $u_0, \dots, u_{k-1} \in \mathcal{A}^+$. We denote by $\pi[u_0/x_0, \dots, u_{k-1}/x_{k-1}]$ the string $s \in \mathcal{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, \dots, k-1$, in the pattern π. The tuple $(u_0, \dots, u_{k-1})$ is called *substitution.* For every $\pi \in Pat_k$ we define the *language generated by pattern* π by $L(\pi) = \{\pi[u_0/x_0, \dots, u_{k-1}/x_{k-1}] \mid u_0, \dots, u_{k-1} \in \mathcal{A}^+\}$.[7] By PAT_k we denote the set of all *k-variable pattern languages.* Finally, $PAT = \bigcup_{k \in \mathbb{N}} PAT_k$ denotes the set of all *pattern languages* over $\mathcal{A}$.

Furthermore, we let Q range over finite sets of patterns and define $L(Q) = \bigcup_{\pi \in Q} L(\pi)$, i.e., the union of all pattern languages generated by patterns from Q. Moreover, we use $Pat(k)$ and $PAT(k)$ to denote the family of all unions of at most k canonical patterns and the family of all unions of at most k pattern languages, respectively. That is, $Pat(k) = \{Q \mid Q \subseteq Pat,\ |Q| \leq k\}$ and $PAT(k) = \{L \mid (\exists Q \in Pat(k))[L = L(Q)]\}$. Finally, let $L \subseteq \mathcal{A}^+$ be a language, and let $k \in \mathbb{N}^+$; we define $Club(L, k) = \{Q \mid |Q| \leq k,\ L \subseteq L(Q),\ \forall Q'[Q' \subset Q \rightarrow L \not\subseteq L(Q')]\}$. *Club* stands for **c**onsistent **l**east **u**pper **b**ounds.

As already mentioned above, the class PAT is $TxtEx_{Pat}$-learnable from positive data (cf. Angluin [1]). Subsequently, Lange and Wiehagen [23] showed PAT to be $TxtItEx_{Pat}$-inferable. As for unions, the first result goes back to Shinohara [37] who proved the class of all unions of at most two pattern languages to be in $TxtEx_{Pat(2)}$. Wright [44] extended this result to $PAT(k) \in TxtEx_{Pat(k)}$ for all $k \geq 1$. Moreover, Theorem 4.2 in Shinohara and Arimura's [40] together with a lemma from Blum and Blum's [7] shows that $\bigcup_{k \in \mathbb{N}} PAT(k)$ is not $TxtEx_{\mathcal{H}}$-inferable for every hypothesis space $\mathcal{H}$. However, nothing was known previous to the present paper concerning the *incremental* learnability of $PAT(k)$. We resolve this problem by showing the strongest possible result (Theorem 9 below): each $PAT(k)$ is *iteratively* learnable!

PROPOSITION 1.

(1) *For all* $L \subseteq \mathcal{A}^+$, $k \in \mathbb{N}^+$, $Club(L, k)$ *is finite.*

(2) *If* $L \in PAT(k)$, *then* $Club(L, k)$ *is nonempty and contains* Q, *such that* $L(Q) = L$.

Proof. Part (2) is obvious. Part (1) is easy for finite L. For infinite L, it follows from the lemma below.

LEMMA 4. *Let* $k \in \mathbb{N}^+$, *and let* $L \subseteq \mathcal{A}^+$ *be any language. Suppose* $T = s_0, s_1, \dots$ *is a text for* L. *Let* L_n *below denote* $\{s_i \mid i \leq n\}$. *Then,*

(1) $Club(L_0, k)$ *can be effectively obtained from* s_0, *and* $Club(L_{n+1}, k)$ *can be effectively obtained from* $Club(L_n, k)$ *and* s_{n+1} *(* note the iterative nature *).*

(2) *The sequence* $Club(L_0, k)$, $Club(L_1, k)$, ... *converges to* $Club(L, k)$.

Proof. (1): Fix $k \geq 1$, and suppose $T = s_0, s_1 \dots, s_n, s_{n+1}, \dots$ is a text for L. Furthermore, let $\mathcal{S}_0 = \{\{\pi\} \mid s_0 \in L(\pi)\}$. We proceed inductively; for $n \geq 0$, we define $\mathcal{S}'_{n+1} = \{Q \in \mathcal{S}_n \mid s_{n+1} \in L(Q)\} \cup \{Q \cup \{\pi\} \mid Q \in \mathcal{S}_n\ \wedge\ s_{n+1} \notin L(Q)\ \wedge\ |Q| < k\ \wedge\ s_{n+1} \in L(\pi)\}$, and then $\mathcal{S}_{n+1} = \{Q \in \mathcal{S}'_{n+1} \mid (\forall Q' \in \mathcal{S}'_{n+1})[Q' \not\subset Q]\}$.

[7]We study so-called *non-erasing* substitutions. It is also possible to consider *erasing* substitutions where variables may be replaced by empty strings, leading to a different class of languages (cf. Filé [15]).

Note that $\mathcal{S}_0$ can be effectively obtained from s_0, since every pattern π with $s_0 \in L(\pi)$ must satisfy $|\pi| \leq |s_0|$. Thus, there are only finitely many candidate patterns π with $s_0 \in L(\pi)$ which can be effectively constructed. Since membership is uniformly decidable, we are done. Furthermore, using the same argument, $\mathcal{S}_{n+1}$ can be effectively obtained from $\mathcal{S}_n$ and s_{n+1}, too. Also it is easy to verify, by induction on n, that $\mathcal{S}_n = \mathit{Club}(L_n, k)$. Thus, (1) is satisfied.

(2): Consider a tree $\mathcal{T}$ formed mimicking the above construction of $\mathcal{S}_n$ as follows. The nodes of $\mathcal{T}$ will be labeled either "empty" or by a pattern. The root is labeled "empty". The children of any node in the tree (and their labels) are defined as follows. Suppose the node, v, is at distance n from the root. Let Q denote the set of patterns formed by collecting the labels on the path from root to v (ignoring the "empty" labels). Children of v are defined as follows:

(a) If $s_n \in L(Q)$, then v has only one child with label "empty."

(b) If $s_n \notin L(Q)$, and $|Q| = k$, then v has no children.

(c) If $s_n \notin L(Q)$, and $|Q| < k$, then v has children with labels π, where $s_n \in L(\pi)$ (the number of children is equal to the number of patterns π such that $s_n \in L(\pi)$). Suppose $\mathcal{U}_n = \{Q \mid (\exists v$ at a distance $n+1$ from root $)[Q =$ the set of patterns formed by collecting the labels on the path from root to v (ignoring the "empty" labels) $]\}$. Then it is easy to verify using induction that $\mathcal{S}_n = \{Q \in \mathcal{U}_n \mid (\forall Q' \in \mathcal{U}_n)[Q' \not\subset Q]\}$.

Since the number of non-empty labels on any path of the tree is bounded by k, using Köning's Lemma we have that the number of nodes with non empty label must be finite. Thus the sequence $\mathcal{U}_0, \mathcal{U}_1, \ldots$ converges. Hence the sequence $\mathcal{S}_0 = \mathit{Club}(L_0, k), \mathcal{S}_1 = \mathit{Club}(L_1, k), \ldots$ converges, to say $\mathcal{S}$. Now, for all $Q \in \mathcal{S}$, for all n, $L_n \subseteq L(Q)$. Thus, for all $Q \in \mathcal{S}$, $L \subseteq L(Q)$. Also, for all $Q \in \mathcal{S}$ and $Q' \subset Q$, for all but finitely many n, $L_n \not\subseteq L(Q')$. Thus for all $Q \in \mathcal{S}$ and $Q' \subset Q$, $L \not\subseteq L(Q')$. It follows that $\mathcal{S} = \mathit{Club}(L, k)$. Part (2) of Lemma follows. ▮

THEOREM 9. *For all $k \geq 1$, $PAT(k) \in TxtItEx$.*

Proof. Let can$(\cdot)$, be some computable bijection from finite classes of finite sets of patterns onto $\mathbb{N}$. Let pad be a 1–1 padding function such that, for all x, y, $W_{\mathrm{pad}(x,y)} = W_x$. For a finite class $\mathcal{S}$ of sets of patterns, let $g(\mathcal{S})$ denote a grammar obtained, effectively from $\mathcal{S}$, for $\bigcap_{Q \in \mathcal{S}} L(Q)$.

Let $L \in PAT(k)$, and let $T = s_0,\ s_1,\ \ldots$ be a text for L. The desired IIM M is defined as follows. We set $M_0(T) = M(s_0) = \mathrm{pad}(g(\mathit{Club}(\{s_0\}, k)), \mathrm{can}(\mathit{Club}(\{s_0\}, k)))$, and $M_{n+1}(T) = M(M_n(T), s_{n+1}) = \mathrm{pad}(g(\mathit{Club}(\{s_0, s_1, \ldots, s_n\}, k)), \mathrm{can}(\mathit{Club}(\{s_0, s_1, \ldots, s_n\}, k)))$ for all $n > 0$. Using Lemma 4 it is easy to verify that $M_{n+1}(T) = M(M_n(T), s_{n+1})$ can be obtained effectively from $M_n(T)$ and s_{n+1}. Thus, M *TxtItEx*-identifies $PAT(k)$. ▮

3.5. *Further Comparisons*

Finally, we turn our attention to the differences and similarities between Definition 4 and a variant of bounded example-memory inference that has been considered in the literature. The following learning type, called k-memory bounded inference, goes back to Fulk *et al.* [17] and is a slight modification of k-memory limited learning defined in Osherson *et al.* [31], where the learner could just memorize the latest k data items received. It has been thoroughly studied by Fulk *et al.* [17]. The main differences to Definition 4 are easily explained. In Definition 4 the bounded example-memory learner is exclusively allowed to use its last conjecture, the new data item coming in, and up to k data items its has already seen for computing the new hypothesis and the possibly *new* data item to be memorized. In contrast, Definition 7 below allows using the *whole initial segment* provided so far to *decide* whether or not it will store the latest data item received. Moreover, the actual hypothesis computed is allowed to depend on the previous conjecture, the new data item coming in and the *newly* stored elements.

We continue with the formal definition. Subsequently, let ϱ denote the empty sequence.

Definition 6 (Fulk *et al.* [17]). *Let $\mathcal{X}$ be a learning domain, an let $k \in \mathbb{N}$; then*

(a) *$mem\colon SEQ \rightarrow SEQ$ is a k-memory function iff, $mem(\varrho) = \varrho$, and, for all sequences $\sigma \in SEQ$ and all $x \in \mathcal{X}$, $\mathrm{content}(mem(\sigma \diamond x)) \subseteq \mathrm{content}(\sigma \diamond x)$, $|mem(\sigma \diamond x)| = k$ and $\mathrm{content}(mem(\sigma \diamond x)) \subseteq \mathrm{content}(mem(\sigma)) \cup \{x\}$.*

(b) *An IIM M is said to be k-memory bounded iff there is a recursive k-memory function mem such that, $(\forall \sigma, \tau)(\forall x \in \mathcal{X})[[M_{|\sigma|}(\sigma) = M_{|\tau|}(\tau) \ \wedge \ mem(\sigma \diamond x) = mem(\tau \diamond x)] \Rightarrow [M_{|\sigma|+1}(\sigma \diamond x) = M_{|\tau|+1}(\tau \diamond x)]]$.*

Definition 7 (Fulk *et al.* [17]). *Let $k \in \mathbb{N}$; then we set $TxtMb^kEx = \{\mathcal{C} \subseteq \wp(\mathcal{X}) \mid (\exists k$-memory bounded machine M $TxtEx$-inferring $\mathcal{C})\}$.*

Our next theorem shows that, for every k, 1-memory bounded inference may outperform k bounded example-memory identification.

Theorem 10. *$TxtMb^1Ex \setminus TxtBem^kEx \neq \emptyset$ for all $k \in \mathbb{N}$.*

Proof. Assume any $k \in \mathbb{N}$. Let $L_1 = \{\langle i, x\rangle \mid x \in \mathbb{N},\ i \leq k\}$ and for all $m_0, \dots, m_k \in \mathbb{N}$ let $L_k^{m_0,\dots,m_k} = \{\langle 0, x\rangle \mid x < m_0\} \cup \dots \cup \{\langle k, x\rangle \mid x < m_k\} \cup \{\langle k+1, x\rangle \mid x \in \mathbb{N}\}$. Furthermore, let $\mathcal{L}_k$ be the collection of L_1 and all $L_k^{m_0,\dots,m_k}$, $m_0, \dots, m_k \in \mathbb{N}$. Now, one easily shows that $\mathcal{L}_k \notin TxtBem^kEx$ using the same ideas as in [17].

On the other hand, $\mathcal{L}_k \in TxtMb^1Ex$. The crucial point here is that the 1-memory function mem can be applied to encode, if necessary, the appropriate $m_0, \dots, m_k$ by using the elements from $\{\langle k+1, x\rangle \mid x \in \mathbb{N}\}$ that appear in the text.

We proceed formally. Let pad be a 1–1 recursive function such that, for all $m_0, \dots, m_k$, $W_{\mathrm{pad}(0,\dots,0)} = L_1$ and $W_{\mathrm{pad}(m_0+1, m_1, \dots, m_k)} = L_k^{m_0,\dots,m_k}$. Furthermore, assume any recursive function g that satisfies, for all $m_0, \dots, m_k$, $g(x) = \langle m_0, \dots, m_k\rangle$ for infinitely many x.

M, and its associated memory function mem, witnessing that $\mathcal{L} \in TxtMb^1Ex$ is defined as follows. M's output will be of the form, $\mathrm{pad}(m'_0, \dots, m'_k)$. Let $L \in \mathcal{L}$, let $T = (s_j)_{j\in\mathbb{N}} \in text(L)$, and let $z \in \mathbb{N}$.

On input T_z, mem is computed as follows. We set $mem(T_0) = s_0$, and proceed inductively for all $z > 0$. Let $y = mem(T_{z-1})$; if $y = \langle k+1, x\rangle$ for some x and $g(x) = \langle m_0, \dots, m_k\rangle$ with $m_i = \max\{m' \mid \langle i, m'\rangle \in T_z^+\}$ for all $i \leq k$ then $mem(T_z) = y$. Otherwise, let $mem(T_z) = s_z$.

Next, we formally define the desired 1-memory bounded learner M. Suppose $s_0 = \langle j, x\rangle$. If $j \neq k+1$, then let $M(s_0) = \mathrm{pad}(0, \dots, 0)$. Otherwise, let $M(s_0) = \mathrm{pad}(1, 0, \dots, 0)$.

For $z > 0$ we define $M_*(T_z)$ as follows. Let $q = M_*(T_{z-1})$, then we set:

```
$M_*(T_z)$
1. Suppose $s_z = \langle j, x\rangle$, and $q = \mathrm{pad}(m'_0, \dots, m'_k)$.
2. If $m'_0 = 0$ and $j \neq k+1$, then let $m'_0 = \dots = m'_k = 0$.
3. Otherwise, let $\langle j', x'\rangle = mem(T_z)$ and $g(x') = \langle m_0, \dots, m_k\rangle$. Set $m'_0 = m_0 + 1$, $m'_1 = m_1$,
      $\dots$, and $m'_k = m_k$.
4. Output $\mathrm{pad}(m'_0, \dots, m'_k)$.
End
```

Clearly, if the target language L equals L_1, M always outputs a correct hypothesis. Otherwise, L equals $L_k^{m_0,\dots,m_k}$ for some $m_0, \dots, m_k$. Since $|L \cap \{\langle i, x\rangle \mid i \le k,\, x \in \mathbb{N}\}| < \infty$, and by the choice of g, M must receive an element $\langle k+1, x\rangle$ with $g(x) = \langle m_0, \dots, m_k\rangle$ after all $k+1$ elements $\langle i, m_i\rangle$, $i \le k$, appeared in the text T. By definition, M outputs a correct guess in this and every subsequent learning step, and thus M *TxtMb*^{1}*Ex*-infers every language in $\mathcal{L}$. ∎

The latter theorem immediately allows the following corollary.

Corollary 11. *TxtBem*^{k}*Ex* $\subset$ *TxtMb*^{k}*Ex for all* $k \in \mathbb{N}$.

Proof. *TxtBem*^{k}*Ex* $\subseteq$ *TxtMb*^{k}*Ex* for all $k \in \mathbb{N}$, since the memory bounded learner may easily simulate the bounded example memory machine while computing the actual $mem(T_x)$ for every text T and $x \in \mathbb{N}$. Thus, the corollary follows by Theorem 10. ∎

But there is more. The following theorem nicely contrasts Theorem 7 and puts the condition to use $mem(T_z)$ in computing $M_*(T_z)$ in memory bounded identification as defined in [17] into the right perspective.

Theorem 12. *TxtFb*^{1}*Ex* $\subset$ *TxtMb*^{1}*Ex*.

Proof. It suffices to show that *TxtFb*^{1}*Ex* $\subseteq$ *TxtMb*^{1}*Ex*, since *TxtMb*^{1}*Ex* $\setminus$ *TxtFb*^{1}*Ex* $\neq \emptyset$ follows immediately from Theorems 7 and 10.

Let M together with the query asking function Q be witnessing $\mathcal{C} \in$ *TxtFb*^{1}*Ex*. The desired IIM M', and its associated memory function *mem*, witnessing that $\mathcal{C} \in$ *TxtMb*^{1}*Ex* are defined as follows. Let $c \in \mathcal{C}$, let $T = (s_j)_{j \in \mathbb{N}} \in text(c)$, and let $z \in \mathbb{N}$.

On input T_z, *mem* is computed as follows. We set $mem(T_0) = s_0$, and proceed inductively for all $z > 0$. Let $q = M(T_{z-1})$. If $Q(q, s_z) \in \mathrm{content}(T_z)$, then $mem(T_z) = s_z$. Otherwise, $mem(T_z) = \varrho$. Recall that ϱ stands for the empty sequence.

Next, we formally define the desired 1-memory bounded learner M. For $z = 0$, let $M'(s_0) = M(s_0)$.

For $z > 0$ we define $M'(T_z)$ as follows. Let $q = M'(T_{z-1})$; we set:

```
$M(T_z)$
1. If $mem(T_z) = s_z$ then output $M(q, 1, s_z)$.
2. Otherwise, output $M(q, 0, s_z)$.
End
```

Now, one immediately sees that M', when fed T, outputs the same sequences of hypotheses as the feedback learner M would do. Hence, M' learns every $c \in \mathcal{C}$ as required. ▮

Though k-memory bounded inference is more powerful than k-bounded example-memory inference, it has the serious disadvantage that *all* data are needed for computing the sequence to be memorized. This is somehow counterintuitive to the idea of incremental learning. It may be, however, an option provided the computation of the memory function $mem(T_z)$ can be done in roughly the same time as the computation of M on input $M(T_{z-1})$, s_z, and $mem(T_{z-1})$.

A further variation is obtained by modifying Definition 6 as follows. Instead of allowing mem to depend on the whole initial segment T_z, it is only allowed to depend on $mem(T_{z-1})$, s_z, and $M(T_{z-1})$. Then the only remaining difference to Definition 4 is that one can still memorize the order of particular elements in accordance with their presentation. On the one hand, it is currently open whether or not this information may increase the resulting learning power. On the other hand, all relevant theorems remain valid if $TxtBem^kEx^a$ and $TxtBem^kFex^a$ are replaced by the new resulting learning type.

4. Conclusions and Future Directions

We studied refinements of concept learning in the limit from positive data that are considerably restricting the accessibility of input data. Our research derived its motivation from the rapidly emerging field of data mining. Here, huge data sets are around, and any practical learning system has to deal with the limitations of space available. Given this, a systematic study of incremental learning is important for gaining a better understanding of how *different* restrictions to the accessibility of input data do affect the resulting *inference capabilities* of the corresponding learning models. The study undertaken extends previous work done by Osherson *et al.* [31], Fulk *et al.* [17] and Lange and Zeugmann [25] in various directions.

First, the class of all unions of at most k pattern languages has been shown to be iteratively learnable. Moreover, we proved redundancy in the hypothesis space to be a resource extending the learning power of iterative learners in fairly concrete contexts. As a matter of fact, the hypothesis space used in showing Theorem 9 is highly redundant, too. Moreover, we strongly conjecture this redundancy to be necessary, i.e., no *iterative learner* can identify all unions of at most k pattern languages with repsect to a 1–1 hypothesis space. Clearly, once the principal learnability has been established, complexity becomes a central issue. Thus, further research should address the problem of designing *time efficient* iterative learners for $PAT(k)$. This problem is even unsolved for $k = 1$. On the one hand, Lange and Wiehagen [23] designed an iterative pattern learner having *polynomial update time.* Nevertheless, the *expected total learning time*, i.e., the overall time needed until convergence is exponential in the number of different variables occurring in the target pattern for inputs drawn with respect to the uniform distribution (cf. Zeugmann [45]).

Second, we considerably generalized the model of feedback inference introduced in [25] by allowing the feedback learner to ask simultaneously k queries. Though at first glance it may seem that asking simultaneously for k elements and memorizing k carefully selected data items may be traded one to another, we rigorously proved the resulting learning types to be advantageous in very different scenarios (cf. Theorem 6 and 7). Consequently, there is no unique way to design superior incremental learning algorithms. Therefore, the comparison of k-feedback learning and k-bounded example-memory inference deserves special interest, and future research should address the problem under what circumstances which model is preferable. Characterizations may serve as suitable tool for accomplishing this goal (cf., e.g., [2, 7, 47]).

Additionally, feed-back identification and bounded example-memory inference have been considered in the general context of classes of recursively enumerable concepts rather than uniformly recursives ones as done in [25]. As our Theorem 4 shows, there are subtle differences. Furthermore, a closer look at the proof of Theorem 4 directly yields the interesting problem whether or not allowing a learner to ask simultaneously k questions instead of querying one data item per time may speed-up the learning process.

A further generalization can be obtained by allowing a k-feedback learner to ask its queries *sequentially*, i.e., the next query is additionally allowed to depend on the answers to its previous questions. Interstingly, our theorems hold in this case, too. It is, however, currently open whether or not sequentially querying the database does has any advantage at all.

Next, we discuss possible extensions of the incremental learning models considered. A natural relaxation of the constraint to fix k *a priori* can be obtained by using the notion of constructive ordinals as done by Freivalds and Smith [16] for mind changes. Intuitively, the paramenter k is now specified to be a constructive ordinal, and the bounded example-memory learner as well as a feedback machine can change their mind of how many data items to store and to ask for, respectively, in dependence on k. Furthermore, future research should examine a hybrid model which permits *both* memorizing k_1 items from the database *and* k_2 queries of the database, where again, k_1 and k_2 may be specified as constructive ordinals.

Moreover, it would also be interesting to extend this and the topics of the present paper to probabilistic learning machines. This branch of learning theory has recently seen as variety of surprising results (cf., e.g., [20, 28, 29]), and thus, one may expect further interesting insight into the power of probabilism by combining it with incremental learning.

Finally, while the research presented in the present paper clarified what are the strength and limitations of incremental learning, further investigations are necessary dealing with the impact of incremental inference to the complexity of the resulting learner. First results along this line are established in Wiehagen and Zeugmann [43], and we shall see what the future brings concerning this interesting topic.

References

[1] D. Angluin, Finding patterns common to a set of strings, *Journal of Computer and System Science* **21** (1980), 46–62.

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

[3] S. Arikawa, T. Shinohara, and A. Yamamoto, Learning elementary formal systems, *Theoretical Computer Science* **95** (1992), 97–113.

[4] H. Arimura and T. Shinohara, Inductive inference of Prolog programs with linear data dependency from positive data, In *Proceedings Information Modeling and Knowledge Bases V*, pp. 365–375, IOS Press, 1994.

[5] M. Blum M, A machine independent theory of the complexity of recursive functions, *Journal of the ACM* **14** (1967), 322–336.

[6] R. Brachman and T. Anand, The process of knowledge discovery in databases: A human centered approach, In (U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 37–58, Menlo Park, CA, AAAI Press, 1996.

[7] M. Blum and L. Blum, Toward a mathematical theory of inductive inference, *Information and Control* **28** (1975), 125–155.

[8] J. Case, Periodicity in generation of automata, *Mathematical Systems Theory* **8** (1974), 15–32.

[9] J. Case, The power of vacillation, In (D. Haussler and L. Pitt, Eds.), *Proceedings of the Workshop on Computational Learning Theory*, pp. 133–142, Morgan Kaufmann Publishers, Inc., 1988.

[10] J. Case, Infinitary self-reference in learning theory, *Journal of Experimental & Theoretical Artificial Intelligence* **6** (1994), 3–16.

[11] J. Case, The power of vacillation in language learning, Technical Report LP-96-08, Logic, Philosophy and Linguistics Series of the Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands, 1996. To appear revised in *SIAM Journal on Computing.*

[12] J. Case and C.H. Smith, Comparison of identification criteria for machine inductive inference, *Theoretical Computer Science* **25** (1983), 193–220.

[13] U.M. Fayyad, S.G. Djorgovski, and N. Weir, Automating the analysis and cataloging of sky surveys, In (U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 471–494, Menlo Park, CA, AAAI Press, 1996.

[14] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, From data mining to knowledge discovery: An overview, In (U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 1–34, Menlo Park, CA, AAAI Press, 1996.

[15] G. Filé, The relation of two patterns with comparable languages, In (R. Cori and M. Wirsing, Eds.), *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Vol. 294, pp. 184–192, Springer-Verlag, Berlin, 1988.

[16] R. Freivalds and C.H. Smith, On the role of procrastination for machine learning, *Information and Computation* **107** (1993), 237–271.

[17] M. Fulk, S. Jain, and D.N. Osherson, Open problems in systems that learn, *Journal of Computer and System Science* **49** (1994), 589–604.

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

[19] J.E. Hopcroft and J.D. Ullman, "Formal Languages and their Relation to Automata," Addison–Wesley, Reading, MA, 1969.

[20] S. Jain and A. Sharma, On identification by teams and probabilistic machines, (In K. P. Jantke and S. Lange, Eds.), *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence, Vol. 961, pp. 108–145, Springer-Verlag, Berlin, 1995.

[21] T. Jiang, A. Salomaa, K. Salomaa, and S. Yu, Inclusion is undecidable for pattern languages, In (A. Lingas, R. Karlsson, and S. Carlsson, Eds.), *Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 700, pp. 301–312, Springer-Verlag, Berlin, 1993.

[22] W. Kloesgen, Efficient discovery of interesting statements in databases, *Journal of Intelligent Information Systems* **4** (1995), 53–69.

[23] S. Lange and R. Wiehagen, Polynomial-time inference of arbitrary pattern languages, *New Generation Computing* **8** (1991), 361–370.

[24] S. Lange and T. Zeugmann, Learning recursive languages with bounded mind changes, *International Journal of Foundations of Computer Science* **4** (1993), 157–178.

[25] S. Lange and T. Zeugmann, Incremental learning from positive data, *Journal of Computer and System Sciences* **53** (1996) 88–103.

[26] S. Lange and T. Zeugmann, Set-driven and rearrangement-independent learning of recursive languages, *Mathematical Systems Theory* **29** (1996), 599–634.

[27] C.J. Matheus, G. Piatetsky-Shapiro, and D. McNeil, Selecting and reporting what is interesting, In (U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds.), *Advances in Knowledge Discovery and Data Mining* pp. 495–515, Menlo Park, CA, AAAI Press, 1996.

[28] L. Meyer, Probabilistic language learning under monotonicity constraints, In (K.P. Jantke, T. Shinohara and T. Zeugmann, Eds.), *Proceedings of the 6th International Workshop on Algorithmic Learning Theory, ALT'95*, Lecture Notes in Artificial Intelligence, Vol. 997, pp. 169–184, Springer-Verlag, Berlin, 1995.

[29] L. Meyer, Monotonic and dual monotonic probabilistic language learning of indexed families with high probability, In (S. Ben-David, Ed.), *Proceedings of the 3rd European Conference on Computational Learning Theory, EuroColt'97*, Lecture Notes in Artificial Intelligence, Vol. 1208, pp. 66–78, Springer-Verlag, Berlin, 1997.

[30] R.P. Nix, Editing by examples, Technical Report 280, Department of Computer Science, Yale University, New Haven, USA, 1983.

[31] D.N. Osherson, M. Stob, and S. Weinstein, "Systems that Learn, An introduction to Learning Theory for Cognitive and Computer Scientists," MIT Press, Cambridge, MA, 1986.

[32] K. Rao, A class of Prolog programs inferable from positive data, In (S. Arikawa and A.K. Sharma, Eds.), *Proceedings of the 7th International Workshop on Algorithmic Learning Theory, ALT'96*, pp. 272–284. Lecture Notes in Artificial Intelligence, Vol. 1160, Springer-Verlag, 1996.

[33] H. Rogers, "Theory of Recursive Functions and Effective Computability," McGraw Hill, New York, 1967. Reprinted, MIT Press, Cambridge, MA, 1987.

[34] A. Salomaa, Patterns (The Formal Language Theory Column), EATCS Bulletin **54** (1994), 46–62.

[35] A. Salomaa, Return to patterns (The Formal Language Theory Column), EATCS Bulletin **55** (1994), 144–157.

[36] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara, and S. Arikawa, Knowledge acquisition from amino acid sequences by machine learning system BONSAI, *Trans. Information Processing Society of Japan* **35** (1994), 2009–2018.

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

[38] T. Shinohara, Inductive inference of monotonic formal systems from positive data, *New Generation Computing* **8** (1991), 371–384.

[39] T. Shinohara and S. Arikawa, Pattern inference, In (K.P. Jantke and S. Lange, Eds.), *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence, Vol. 961, pp. 259–291, Springer-Verlag, Berlin, 1995.

[40] S. Shinohara and H. Arimura, Inductive inference of unbounded unions of pattern languages from positive data, In (S. Arikawa and A.K. Sharma, Eds.), *Proceedings of the 7th International Workshop on Algorithmic Learning Theory, ALT'96*, Lecture Notes in Artificial Intelligence, Vol. 1160, pp. 256–271, Springer-Verlag, Berlin, 1996.

[41] R. Smullyan, *Theory of Formal Systems, Annals of Mathematical Studies, No. 47.* Princeton, NJ, 1961.

[42] R. Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, *Journal of Information Processing and Cybernetics (EIK)* **12** (1976), 93–99.

[43] R. Wiehagen and T. Zeugmann, Ignoring data may be the only way to learn efficiently, *Journal of Experimental & Theoretical Artificial Intelligence* **6** (1994), 131–144.

[44] K. Wright, Identification of unions of languages drawn from an identifiable class, In (R. Rivest, D. Haussler, and M. Warmuth, Eds.), *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pp. 328–333, Morgan Kaufmann Publishers, Inc., 1989.

[45] T. Zeugmann, Lange and Wiehagen's pattern language learning algorithm: An average-case analysis with respect to its total learning time, *Annals of Mathematics and Artificial Intelligence.* to appear 1997.

[46] T. Zeugmann and S. Lange, A guided tour across the boundaries of learning recursive languages, (In K.P. Jantke and S. Lange, Eds.), *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence, Vol. 961, pp. 190–258, Springer-Verlag, Berlin, 1995.

[47] T. Zeugmann, S. Lange, and S. Kapur, Characterizations of monotonic and dual monotonic language learning, *Information and Computation* **120** (1995), 155–173.