Information and Computation **152**, 74–110 (1999)
Article ID inco.1998.2784, available online at http://www.idealibrary.com on IDEAL®

Incremental Concept Learning for

John Case

Department of CIS, University of Delaware, Newark, Delaware 19716
E-mail: case@cis.udel.edu

Sanjay Jain

Department of ISCS, National University of Singapore, Lower Kent Ridge Road, Singapore 119260, Republic of Singapore
E-mail: sanjay@iscs.nus.edu.sg

Steffen Lange

Universität Leipzig, Fakultät für Mathematik und Informatik, Institut für Informatik, 04109 Leipzig, Germany
E-mail: slange@informatik.uni-leipzig.de

and

Thomas Zeugmann*

Department of Informatics, Kyushu University, Kasuga 816-8580, Japan
E-mail: thomas@i.kyushu-u.ac.jp

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 items 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

* Corresponding author.

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 (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.* (1996a, 1996b)). Prominent examples of KDD applications in health care and finance include Matheus *et al.* (1996) and Kloesgen (1995). 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.* (1996a, 1996b)), 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.* (1996a) the SKICAT-system is described which operates on 3 terabytes of image data originating from approximately two 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, 1996; Fayyad *et al.*, 1996b). 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 (1967) model of learning *in the limit* grammars for concepts from positive instances.[1] Gold's (1967) 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 *k-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 has just seen, if it is 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 *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. For some extremely large databases, a query about whether an item is in there can be very expensive, so, in such cases, k-feedback identification is interesting when the bound k is small.

[1] The subfocus on learning *grammars*, or, equivalently, recognizers (cf. Hopcroft and Ullman, 1969), for concepts from *positive* instances nicely models the situation where the database flags or contains *examples* of the concept to be learned and does not flag or contain the nonexamples.

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 3 and 4 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 is restricted to no more than $k - 1$ queries.[2] Hence, each additional, possibly expensive dip into the database buys more concept learning power. However, the feedback hierarchy collapses to its first level if only *indexable classes* of *infinite* concepts are to be learned (cf. Theorem 5).

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 *look up* to see if they are in the database. There are apparent similarities between these two kinds of learning machines, yet Theorems 7 and 8 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!

Theorem 9 shows that, even in fairly concrete contexts, with iterative learning, *redundancy* in the hypothesis space increases learning power.

Angluin's (1980a) *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 13 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 (1967). In addition to or in contrast with Rogers (1967) we use the following. By $\mathbb{N} = \{0, 1, 2, ...\}$ 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 $|S_1 \triangle S_2| \leqslant a$, where $a = *$ means that the symmetric difference is finite.

[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; (2) many language learning *un*solvability results *depend strongly* on including the finite languages (cf. Gold, 1967; Case, 1996). Ditto for other results below, namely, Theorems 7 and 8, which are witnessed by concept classes containing only infinite concepts.

By max S and min S we denote the maximum and minimum of a set S, respectively, where, by convention, max $\varnothing = 0$ and min $\varnothing = \infty$.

The quantifiers "$\overset{\infty}{\forall}$," "$\overset{\infty}{\exists}$," and "$\exists!$" are interpreted as "for all but finitely many," "there exist infinitely many," and "there exists a unique," respectively (cf. Blum, 1967).

By $\langle \cdot, \cdot \rangle: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ 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 $\varphi_0, \varphi_1, \varphi_2, ...$ denote any fixed *acceptable programming system* (cf. Rogers, 1967) for all (and only) the partial recursive functions over $\mathbb{N}$, and let Φ_0, Φ_1, Φ_2, ... be any associated *complexity measure* (cf. Blum, 1967). 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 ($\varphi_k(x)\downarrow$) then we also say that $\varphi_k(x)$ *converges*; otherwise $\varphi_k(x)$ *diverges.*

In the following two subsections we define the learning models discussed in the Introduction.

2.1. *Defining Gold-Style Concept Learning*

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 $T = (x_j)_{j \in \mathbb{N}}$ be an infinite sequence of elements $x_j \in c \cup \{\#\}$ such that range$(T) =_{df} \{x_j \mid x_j \neq \#, j \in \mathbb{N}\} = c$. 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 $y \in \mathbb{N}$. Then, we set $T_y = x_0, ..., x_y$; i.e., T_y is the initial segment of T of length $y+1$, and $T_y^+ =_{df} \{x_j \mid x_j \neq \#, j \leqslant y\}$. 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 $\sigma = x_0, ..., x_{n-1}$ be any finite sequence. Then we use $|\sigma|$ to denote the *length n* of σ, and let content(σ) and σ^+, respectively, denote the content of σ. Additionally, let T be a text and 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, 1980b). 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 languages. A sequence $\mathcal{L} = (L_j)_{j \in \mathbb{N}}$ is said to be an *indexed*

[3] This function is easily computable, 1–1, and onto (cf. Rogers, 1967).

family provided all the L_j are nonempty and there is 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 (1980b) learning of indexed families of languages has attracted much attention (cf., e.g., Zeugmann and Lange, 1995). 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 (1967) we define an *inductive inference machine* (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 as a numerical name for some finite description or computer program for h_j.

Let M be an IIM, 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 (1967) 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, 1967). 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, 1983), with $a = 0$ being Gold's (1967) original case where no mistakes are allowed. The $a = *$ case goes back to Blum and Blum (1975). If $a = 0$, we usually omit the upper index; i.e., 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, 1983). Let $\mathscr{C}$ be a concept class, let c be a concept, let $\mathscr{H}=(h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a\in\mathbb{N}\cup\{*\}$. *An* IIM M *$TxtFex^a_{\mathscr{H}}$-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 *$TxtFex^a_{\mathscr{H}}$-infers* $\mathscr{C}$ iff M $TxtFex^a_{\mathscr{H}}$-infers c, for each $c\in\mathscr{C}$. Let $TxtFex^a_{\mathscr{H}}$ denote the collection of all concept classes $\mathscr{C}$ for which there is an IIM M such that M $TxtFex^a_{\mathscr{H}}$-infers $\mathscr{C}$. $TxtFex^a$ denotes the collection of all concept classes $\mathscr{C}$ for which there are an IIM M and a hypothesis space $\mathscr{H}$ such that M $TxtFex^a_{\mathscr{H}}$-infers $\mathscr{C}$.

The following theorem clarifies the relation between Gold's (1967) classical learning in the limit and *TxtFex*-inference. The assertion remains true even if the learner is only allowed to *vacillate* between up to two descriptions, i.e., in the case $|D|\leqslant 2$ (cf. Case, 1988, 1996).

Theorem 1 (Osherson et al., 1986; Case, 1988, 1996). *$TxtEx^a\subset TxtFex^a$, for all $a\in\mathbb{N}\cup\{*\}$.*

2.2. Formalizing Incremental Concept Learning

Looking at the above definitions, we see that an IIM M always has 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, we next define *iterative* IIMs and a natural generalization of them called *k-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 $(M_n)_{n\in\mathbb{N}}$ of machines each of which takes as its input the output of its predecessor.

Definition 3 (Wiehagen, 1976). Let $\mathscr{C}$ be a concept class, let c be a concept, let $\mathscr{H}=(h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a\in\mathbb{N}\cup\{*\}$. *An* IIM M *$TxtItEx^a_{\mathscr{H}}$-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 $n\geqslant 0$: $M_{n+1}(T)=_{df}M(M_n(T),x_{n+1})$,

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

Finally, M *$TxtItEx^a_{\mathscr{H}}$-infers* $\mathscr{C}$ iff, for each $c\in\mathscr{C}$, M $TxtItEx^a_{\mathscr{H}}$-infers c.

The resulting learning types $TxtItEx^a_{\mathscr{H}}$ 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 $\mathscr{C}$ be any concept class over $\mathscr{X}$, and let M be any IIM that iteratively learns $\mathscr{C}$. Then we denote by $M_y(\sigma)$ the $(y+1)$th hypothesis output by M when successively fed σ, provided $y\leqslant n$, and there exists a concept $c \in \mathscr{C}$ with $\sigma^+ \subseteq c$. Furthermore, we let $M_*(\sigma)$ denote $M_{|\sigma|-1}(\sigma)$. We adopt these conventions to the learning types defined below.

Within the following definition we consider a natural relaxation of iterative learning which we call *k-bounded example-memory* inference.[4] Now, an IIM M is allowed to memorize at most k of the examples it already has had access to during the learning process, where $k \in \mathbb{N}$ is *a priori* fixed. 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 k-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, 1996a). Let $k \in \mathbb{N}$, let $\mathscr{C}$ be a concept class, let c be a concept, let $\mathscr{H} = (h_j)_{j \in \mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M *$TxtBem^k Ex^a_{\mathscr{H}}$-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 $|S_0| \leqslant k$, and for all $n \geqslant 0$: $M_{n+1}(T) =_{df} M(M_n(T), x_{n+1}) = \langle j_{n+1}, S_{n+1} \rangle$ such that $S_{n+1} \subseteq S_n \cup \{x_{n+1}\}$ and $|S_{n+1}| \leqslant k$,

(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 $TxtBem^k Ex^a_{\mathscr{H}}$-infers $\mathscr{C}$* iff, for each $c \in \mathscr{C}$, M $TxtBem^k Ex^a_{\mathscr{H}}$-infers c.

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

Finally, we define learning by *feedback* IIMs. The idea of feedback learning goes back to Wiehagen (1976) who considered it in the setting of inductive inference of recursive functions. Lange and Zeugmann (1996a) 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 queries. 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 $(y_1, ..., y_k) \in \mathscr{X}^k$ 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 is 0 otherwise. Hence, M can just ask whether or not k particular strings have already been presented in previous learning stages. Below $A_k^n \colon \mathscr{X}^k \to \{0, 1\}^k$ denotes the answer to the queries based on whether the corresponding queried elements appear in T_n or not.

Definition 5. Let $k \in \mathbb{N}$, let $\mathscr{C}$ be a concept class, let c be a concept, let $\mathscr{H} = (h_j)_{j \in \mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. Moreover, let $Q_k \colon \mathbb{N} \times \mathscr{X} \to \mathscr{X}^k$, be a computable total mapping. An IIM M, *with query asking function Q_k, $TxtFb^k Ex^a_{\mathscr{H}}$-infers c* iff for every positive presentation $T = (x_j)_{j \in \mathbb{N}} \in text(c)$ the following conditions are satisfied:

[4] Our definition is a variant of one found in Osherson *et al.* (1986) and Fulk *et al.* (1994) which will be discussed later.

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

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

Finally, M *$TxtFb^k Ex^a_{\mathscr{H}}$-infers* $\mathscr{C}$ iff there is computable mapping Q_k as described above such that, for each $c \in \mathscr{C}$, M, with query asking function Q_k, $TxtFb^k Ex^a_{\mathscr{H}}$-identifies c.

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

Finally, we extend Definitions 3 through 5 to the *Fex* case analogously to the generalization of $TxtEx^a_{\mathscr{H}}$ to $TxtFex^a_{\mathscr{H}}$ (cf. Definitions 1 and 2). The resulting learning types are denoted by $TxtItFex^a_{\mathscr{H}}$, $TxtBem^k Fex^a_{\mathscr{H}}$, and $TxtFbEx^a_{\mathscr{H}}$. Moreover, for the sake of notation, we shall use the convention for learning machines corresponding to Definitions 3 through 5, as well as to $TxtItFex^a_{\mathscr{H}}$, $TxtBem^k Fex^a_{\mathscr{H}}$, and $TxtFbEx^a_{\mathscr{H}}$.

In all of the criteria of inference considered above, the hypothesis space of $(W_j)_{j \in \mathbb{N}}$ is the most general; i.e., if a class of languages is learnable using some hypothesis space $\mathscr{H}$, then it is also learnable using the hypothesis space $(W_j)_{j \in \mathbb{N}}$. For this reason, unless explicitly stated otherwise, we will often assume the hypothesis space to be $(W_j)_{j \in \mathbb{N}}$, without explicitly saying so.

3. RESULTS

At the beginning of this section, we briefly summarize what has been known concerning the pros and cons of incremental concept learning. The first thorough investigation has been provided by Lange and Zeugmann (1996a). In their paper, the important special case of learning indexed families of recursive languages has been analyzed.

When learning indexed families $\mathscr{L}$, it is generally assumed that the hypothesis space $\mathscr{H}$ has to be an indexed family, too. We distinguish *class preserving learning* and *class comprising learning* defined by range($\mathscr{L}$) = range($\mathscr{H}$) and range($\mathscr{L}$) $\subseteq$ range($\mathscr{H}$), respectively. When dealing with class preserving learning, one has the freedom to choose as hypothesis space a possibly *different enumeration* of the target family $\mathscr{L}$. In contrast, when class comprising learning is concerned, the hypothesis space may enumerate, additionally, languages not belonging to range($\mathscr{L}$). Note that, in general, one has to allow class comprising hypothesis spaces to obtain the maximum possible learning power (cf. Lange and Zeugmann, 1993a, 1996b).

Lange and Zeugmann (1996a) studies class comprising incremental learning of indexed families. In particular, it has been proved that all models of incremental learning are less powerful than unrestricted learning in the limit, and that 1-feedback learning and k-bounded example-memory inference are strictly extending iterative learning. Moreover, the learning capabilities of 1-feedback learning and k-bounded example-memory inference are incomparable to one another.

Since the set of admissible hypothesis spaces has been restricted to indexed families, it is conceivable that the separating classes used do not witness the same separations in the general case of unrestricted recursively enumerable hypothesis spaces. However, a closer look at their proofs shows that the nonlearnability in all considered cases is due to purely information-theoretic arguments. Consequently, their results translate into our more general setting and are summarized in the following theorem.

THEOREM 3.1 (Lange and Zeugmann, 1996a). (1) $TxtFb^1 Ex \subset TxtEx$.

(2) $TxtBem^k Ex \subset TxtEx$, *for all* $k \in \mathbb{N}^+$.

(3) $TxtFb^1 Ex \,\#\, TxtBem^k Ex$, *for all* $k \in \mathbb{N}^+$.

Within the remaining part of 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. Finally, we turn our attention to the differences and similarities between Definition 4 and a variant thereof that has been considered in the literature.

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.

THEOREM 3. $TxtFb^{k-1} Ex \subset TxtFb^k Ex$ *for all* $k \in \mathbb{N}^+$.

Theorem 4 below not only provides the hierarchy of Theorem 3, 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 4. $TxtFb^{k+1} Ex \backslash TxtFb^k Fex^* \neq \emptyset$ *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 $X_w = \{\langle j, w, i\rangle \mid 1 \leqslant j \leqslant k+2,\ i \in \mathbb{N}\}$, and $X_w^0 = \{\langle j, w, 0\rangle \mid 1 \leqslant j \leqslant k+2\}$.

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) $\neg(\exists w)[X_w^0 \subseteq W_e]$.

Finally, we define the desired concept class:

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

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

Proof. 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 $X_w^0 \subseteq X_w$ are exploited. That is, the learner tries to verify $X_w^0 \subseteq L$ whenever receiving an element of the form $\langle j, w, 0\rangle$ with $1 \leqslant j \leqslant k+2$ by just asking whether the other $k+1$ elements in $X_w^0 \backslash \{\langle j, w, 0\rangle\}$ 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 $X_w^0 \subseteq L$, and applying property (b) as well as $X_w^0 \subseteq X_w$, 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 the pad be a 1–1 recursive function such that, for all $i, j \in \mathbb{N}$, $W_{\text{pad}(0, j)} = \varnothing$, $W_{\text{pad}(i+1, 0)} = W_i$, and $W_{\text{pad}(i+1, j+1)} = W_i \cup X_j$. M, and its associated query asking function Q_{k+1}, witnessing that $\mathcal{L} \in TxtFb^{k+1}Ex$ are defined as follows. M's output will be of the form, 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_w^0 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 $M(s_0) = \text{pad}(z+1, 0)$. Otherwise, let $M(s_0) = \text{pad}(0, 0)$.

$Q_{k+1}(q, s_{m+1})$ is computed as follows. Suppose $s_{m+1} = \langle j, z, i\rangle$. If $i = 0$ and $1 \leqslant j \leqslant k+2$, then let $y_1, y_2, \ldots, y_{k+1}$ be such that $\{y_1, y_2, \ldots, y_{k+1}\} = \{\langle j', z, 0\rangle \mid 1 \leqslant j' \leqslant k+2, j' \neq j\}$. If $i \neq 0$, then let $y_1 = y_2 = \cdots = y_{k+1} = 0$ (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:

```
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 $q = \text{pad}(e', w')$.
    2. If $i = 0$, $1 \leqslant j \leqslant k+2$ 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 pad$(e', w')$.
End
```

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

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

Proof. Suppose the converse, i.e., that there are an IIM M and an associated query asking function Q_k such that M witnesses $\mathcal{L} \in TxtFb^{k}Fex^*$. Then by implicit use of the recursion theorem (cf. Rogers, 1967) there exists an e such that W_e may be described as follows (note that e will be nice):

For any finite sequence $\tau = x_0, x_1, ..., x_\ell$, let $M_0(\tau) = M(x_0)$; and for $i < \ell$, let $M_{i+1}(\tau) = M(M_i(\tau), A_k^i(Q_k(M_i(\tau), x_{i+1})), x_{i+1})$, where A_k^i answers questions based on whether the corresponding elements appear in $\{x_j \mid j \leqslant i\}$. Let 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 $TxtFb^k Fex^*$-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 $TxtFb^k Fex^*$-identified by $M$. *)
1. Let $S_s =$ ProgSet$(M, \sigma_s)$.
2. Let $S' = S_s$.
3. Let Pos $=$ content$(\sigma_s)$; Neg $= \emptyset$. Let $Y = \emptyset$; $\tau = \sigma_s$.
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 $S' = S' \setminus \{p\}$,
            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 $\cup Y \cup \{\langle k+3, s, 0\rangle\} \subset$ content($\sigma_{s+1}$).
            $(\forall w)[X_w^0 \not\subseteq$ content($\sigma_{s+1}$)],
            $\{x \mid \langle 0, x, 0\rangle \in$ content($\sigma_{s+1}$)$\} = \{e\}$, and
            ProgSet($M, \sigma_{s+1}$) $\neq$ ProgSet($M, \sigma_s$).
       (* Note that by the invariant above, there exists such a $\sigma_{s+1}$. *)
       Enumerate content($\sigma_{s+1}$) 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 $TxtFb^k Fex^*$-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) $X_w \cap (\text{Pos} \cup \text{Neg}) = \emptyset$.

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 $TxtFb^k Fex^*$-infer $W_e \cup X_w$. Note that, by construction, $W_e \cup X_w$ does not contain any element of Neg. Also, W_e is finite and $X_w \cap X_{w'} = \emptyset$ for $w \neq w'$. Furthermore, for all $p \in S'$, either $X_w \cap W_p = \emptyset$ or W_p intersects infinitely many $X_{w'}$. Thus, none of the programs in S' is a program for a finite variant of $W_e \cup X_w$.

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 let S be the set of questions answered negatively for the queries $Q_k(M_*(\tau'), y)$. Then $p = M_*(\tau'), y$, $\text{Pos}' = \text{Pos} \cup P$ and $\text{Neg}' = \text{Neg} \cup S$, 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 $TxtFb^k Fex^*$-identify $(W_e \cup X_w) \in \mathcal{L}$. ▌

From the above claims, the theorem follows. Q.E.D.

Theorem 4 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. Note that it is necessary to require that the target indexed family consists of infinite languages, only. For seeing this, consider the indexed family that contains the infinite language $L = \{a\}^+ \setminus \{a\}$, together with all finite languages $L_k = \{a, ..., a^k\}$, $k \geqslant 1$. This indexed family separates *TxtEx* and $TxtFb^1 Ex$ (cf. Lange and Zeugmann, 1996a, for further indexed families witnessing this separation and a detailed discussion).

THEOREM 5. *Let $\mathcal{L}$ be any indexed family consisting of only infinite languages, and let $\mathcal{H}$ be a class comprising hypothesis space for it. Then, $\mathcal{L} \in TxtFex_{\mathcal{H}}$ implies that there is a class comprising hypothesis space $\hat{\mathcal{H}}$ for $\mathcal{L}$ such that $\mathcal{L} \in TxtFb^1Ex_{\hat{\mathcal{H}}}$.*

Proof. Throughout this proof, let $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$, with or without superscripts, 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 *TxtEx*-identifiable 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 any 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 let $x \in \mathbb{N}$.

IIM $\tilde{M}$. On input T_x do the following: Compute successively $j_y = M(T_y)$ for all $y = 0, ..., x$. 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 (1993b), and thus Lemma 1 is proved. Q.E.D.

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 $T_{y+z}^+ \nsubseteq h_{M(T_y)}$.*

(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. Lange and Zeugmann, 1993b). By Angluin's

(1980b) characterization of *TxtEx*, there is a uniformly recursively generable family $(\mathscr{T}_j^y)_{j,\,y\in\mathbb{N}}$ of finite telltale sets such that

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

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

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

Using this family $(\mathscr{T}_j^y)_{j,\,y\in\mathbb{N}}$, we define the desired hypothesis space $\mathscr{H}=(h_{\langle j,\,y\rangle})_{j,\,y\in\mathbb{N}}$ as follows. We specify the languages enumerated in $\mathscr{H}$ via their characteristic functions $f_{h_{\langle j,\,y\rangle}}$. For all $j,\ y,\ z\in\mathbb{N}$, we set

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

Since $(\mathscr{T}_j^y)_{j,\,y\in\mathbb{N}}$ is a uniformly recursively generable family of finite sets and since $\mathscr{L}$ is an indexed family, $\mathscr{H}$ 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 $\mathscr{T}_j^y\neq\mathscr{T}_j$.

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

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 go to (B).

(B) For all pairs $\langle j,\ y\rangle\leqslant x$, (ordered by their Cantor numbers) test whether or not $\mathscr{T}_j^y\subseteq T_x^+\subseteq h_{\langle j,\,y\rangle}$ until the first such pair is found; then output it. If all pairs $\langle j,\ y\rangle\leqslant x$ failed, then output h.

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

Claim 1. *M converges when fed T.*

Proof. Let $j_0=\min\{j\mid j\in\mathbb{N},\ L=L_j\}$, and let $y_0=\min\{y\mid y\in\mathbb{N},\ \mathscr{T}_{j_0}=\mathscr{T}_{j_0}^y\}$. Since $T\in\textit{text}(L)$, there must be an $x\geqslant\langle j_0,\ y_0\rangle$ such that $\mathscr{T}_{j_0}\subseteq T_x^+$ 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 $h_{\langle j,\,y\rangle}=L$.*

Proof. 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 language, 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 $\mathscr{H}$ we know that $h_{\langle j,\,y\rangle}=L_j$. Now, since M has converged, it must have verified $\mathscr{T}_j \subseteq T_x^+ \subseteq L$, too. Thus, Condition ($\gamma$) immediately implies $L \not\subset L_j = h_{\langle j, y\rangle}$. Taking $L \neq h_{\langle j, y\rangle}$ into account we have $L \backslash h_{\langle j, y\rangle} \neq \emptyset$, contradicting $T_x^+ \subseteq h_{\langle j, y\rangle}$ for all x. ▮

Hence, Lemma 2 is proved. Q.E.D.

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

Proof. Let $\mathscr{L} \in$ *TxtFex* and, therefore, by Lemmata 1 and 2, we know that there are an IIM and a hypothesis space $\mathscr{H} = (h_j)_{j \in \mathbb{N}}$ 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' using the same candidate for an initial segment more than 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 may be 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 class comprising hypothesis space $\hat{\mathscr{H}} = (\hat{h}_r)_{r \in \mathbb{N}}$ defined as follows. Let $F_0, F_1, F_2, ...$ be any effective enumeration of all nonempty finite subsets of $\mathbb{N}$. For every $\ell \in \mathbb{N}$, let $rf(F_\ell)$ be the *repetition-free* enumeration of all the elements of F_ℓ in increasing 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 ease of presentation, assume that $\langle\cdot, \cdot, \cdot\rangle$ is a bijection from $\mathbb{N} \times (\mathbb{N} \cup \{-1\}) \times \mathbb{N}$ onto $\mathbb{N}$. (Note that, we can easily convert it to regular coding of triples by just adding one to the second argument.) 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 such that for each $L \in \text{range}(\mathscr{L})$ 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 $L \in \text{range}(\mathscr{L})$, $T = (w_n)_{n\in\mathbb{N}} \in$ *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 go to Stage 1.

Stage n, $n \geqslant 1$. M' receives as input j_{n-1} and the $(n+1)$th 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, go to (α3). Otherwise, query "$g(w_n)$." If the answer is "NO," then go to (α1). If the answer is "YES," test whether or not $g(w_n) \in h_j$. If it is, execute (α1). Else, go to (α2).

(α1) Output the ordinary hypothesis (j, F).

(α2) Set $F := F \cup \{g(w_n)\}$, and $z = |F|$. Output the auxiliary hypothesis (F, z, z).

(α3) 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 $y \geqslant 0$. In case it is, go to (β1). Else, execute (β2).

(β1) Query "$z - y$." If the answer is "YES," then set $F := F \cup \{z - y\}$. Output the auxiliary hypothesis $(F, y-1, z)$.

(β2) 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 "z." If the answer is "YES," then set $F := F \cup \{z\}$. 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. It remains to show that M' witnesses $\mathscr{L} \in TxtFb^1Ex_{\mathscr{H}}$. Let $L \in \text{range}(\mathscr{L})$ and $T \in$ *text*(L).

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

Proof. Recall that, by construction, M' outputs in all the Stages $z - z'$, $z - z' + 1, \ldots, z$ auxiliary hypotheses, too, and queries $0, 1, \ldots, z'$, respectively (cf. case B). Thus, for all $\ell \leqslant z'$, if $\ell \in T^+_{z-z'+\ell}$ the answer to the query must be "YES," and therefore, $\ell \in F'$ (cf. case B). On the other hand, if $\ell \in T_z^+ \setminus T^+_{z-z'+\ell}$, then ℓ is presented after the query has been made for it, and thus it is memorized, too (cf. case B). 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 to an ordinary hypothesis, say to the pair (j, F), then $h_j = L$.*

Proof. 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. By Lemma 2, assertion (2), we know $L \not\subset h_j$. Thus, $L \backslash h_j \neq \emptyset$, and hence there must be a string $\ell \in L \backslash h_j$. Since $T = (w_n)_{n \in \mathbb{N}} \in text(L)$, there exists a $z \in \mathbb{N}$ with $w_z = \ell$. If $z > y_0$, then $\ell \notin h_j$ is verified in Stage z (cf. case A), a contradiction. Now, suppose $z \leqslant y_0$. 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 "ℓ" is made in Stage $y_0 + r$. But $\ell \in T^+_{y_0} \subseteq T^+_{y_0+r}$, and hence the answer to it is "YES," and $\ell \in h_j$ is tested, too (cf. case A). Therefore, M' must execute (α2), and cannot converge to (j, F). This proves Claim 2. ▮

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

Proof. 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 $m_0 = \max\{k \mid k \in \mathbb{N}, k \in T^{L,+}_y\}$, and let x_0 be the least x with $T^{L,+}_y \subseteq T^+_x$.

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 $z' \geqslant m_0$. Hence, $\{\ell \mid \ell \in T^+_r, \ell \leqslant z'\} \subseteq F'$ by Claim 1. Moreover, $T^{L,+}_y \subseteq T^+_r$ because of $r \geqslant x_0$ and $T^{L,+}_y \subseteq T^+_{x_0}$, and hence, $T^{L,+}_y \subseteq F'$, since $m_0 \leqslant z'$ and $T^{L,+}_y = \{\ell \mid \ell \leqslant m_0, \ell \in L\}$. Therefore, M' simulates M on input $rf(F')$ in Stage $r + 1$ (cf. case B, instruction (β2)). By the choice of T^L_y, and since T^L_y 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.*

Proof. Suppose, M' diverges. Then, M' must output infinitely often an ordinary hypothesis, since otherwise Claim 3 is contradicted. Let j, y, m_0, x_0 be as in the proof of Claim 3. Consider the minimal $r > x_0$ such that M', when successively fed T_r, has already output its m_0th ordinary hypothesis, say (j', F). Thus, $|F| \geqslant m_0$ 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 (α2), (α3)). In all the Stages $\varrho + 1$, $\varrho + 2$, ..., $\varrho + m_0$, ..., and $\varrho + |F'| + 1$, M' outputs auxiliary hypotheses, too (cf. case B, instruction (β1)). Moreover, in Stage $\varrho + |F'| + 1$, M' outputs an auxiliary hypothesis having the form $(F'', -1, |F'|)$. Applying *mutatis mutandis* the same argument as in Claim 3, we obtain $T^L_y \subseteq F''$. Therefore in the next stage, M' simulates M when fed a finite sequence τ having the initial segment T^L_y (cf. case B, instruction (β2)). 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. ▌ Q.E.D.

Hence, in the case of *indexed* families of infinite languages, the hierarchy of Theorem 3 collapses for $k \geqslant 2$; furthermore, again, for *indexed* families of infinite languages, the *expansion* of Gold's (1967) 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*[5] 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. Here overgeneralization[6] means that the learner outputs a description for a proper superset of the target concept. 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 6. *$TxtFb^0Ex^{a+1} \setminus TxtFex^a \neq \emptyset$, for all $a \in \mathbb{N}$.*

Proof. Let $\mathscr{L}$ be any indexed family such that exactly $L=\{b\}^+$ and all $L' \subseteq L$ with $|L \setminus L'| \leqslant a+1$ belong to range($\mathscr{L}$). Obviously, $\mathscr{L} \in TxtFb^0Ex^{a+1}$, since it suffices to output always an index for L. Now, suppose to the contrary that there is an IIM M' that $TxtFex^a$-identifies $\mathscr{L}$. Then, one can easily show that there is some text for L on which M' outputs infinitely many different hypotheses. We omit the details. Q.E.D.

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 (1967) 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!

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 k-bounded example-memory inference and feedback identification, there are concept class domains where that model is competent and the other is not! Theorem 7 below says that, for suitable concept domains, the feedback learning power of *one* query of the data base, where a *single*, *correct* grammar

[5] Conservativeness is nothing else than condition (2) in Lemma 2.

[6] Note that in the setting of indexed families conservative inference and learning without overgeneralization are essentially equivalent (cf. Lange and Zeugmann, 1993c), while, in general, they are not (cf. Jain and Sharma, 1994).

is found in the limit, *beats* the k-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 k-bounded example-memory learning.

LEMMA 3. *Suppose M is a k-bounded example-memory 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* $\text{content}(\sigma') \subseteq Z$ *and* $\pi_1(M_*(\sigma \diamond \sigma')) \notin P$, *or*

(b) *there exist σ', σ'' and $j \in Z$, such that* $\text{content}(\sigma') = Z \backslash \{j\}$, $\text{content}(\sigma'') = Z$, *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) $\text{content}(\tau') \cup \text{content}(\tau'') \subseteq Z$,

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

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

This is so since there are $2^{|Z|}$ possibilities for $\text{content}(\tau)$, 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 $j \in \text{content}(\tau'') \backslash \text{content}(\tau')$. Now let τ''' be such that $\text{content}(\tau''') = Z \backslash \{j\}$. Taking $\sigma' = \tau' \diamond \tau'''$ and $\sigma'' = \tau'' \diamond \tau'''$ proves the lemma. Q.E.D.

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

THEOREM 7. *$TxtFb^1 Ex \backslash TxtBem^k Fex^* \neq \emptyset$, 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 $C^i_L = \{x \mid \langle i, x \rangle \in L\}$. We say that e is *nice* iff

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

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

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

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

Next, we show that $\mathscr{L} \notin TxtBem^k Fex^*$. Suppose the converse; i.e., there is an IIM M that $TxtBem^k Fex^*$-identifies $\mathscr{L}$. For a sequence $\sigma = x_0, x_1, ..., x_\ell$, 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 $\text{ProgSet}(M, \tau) = \{\pi_1(M_*(\sigma)) \mid \sigma \subseteq \tau\}$, and we define for any text T the set $\text{ProgSet}(M, T)$ similarly.

Then by implicit use of the operator recursion theorem (cf. Case, 1974, 1994) 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 $\text{content}(\sigma_0) = \{\langle 0, p(0) \rangle\}$. Let $W^s_{p(0)}$ denote $W_{p(0)}$ enumerated before Stage s. Let $\text{avail}_0 = 1$. Intuitively, avail_s

denotes a number such that for all $j \geqslant \text{avail}_s$, $p(j)$ 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 $W_{p(0)} \in \mathscr{L}_1$ witnesses the diagonalization. If Stage s starts but does not finish, then each of $W_{p(j_i)}$, $1 \leqslant i \leqslant \ell$, as defined in steps 3 and 4, is in $\mathscr{L}_2$, 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) $C^1_{\text{content}(\sigma)} \cap C^2_{\text{content}(\sigma)} = \varnothing$,
      (c) $\text{ProgSet}(M, \sigma) \neq P_s$.
3. Let $m_0 = \text{avail}_s$.
   Let $\ell = |P_s| + 1$, $\tau_0 = \sigma_s$.
   Search for $m_1, m_2, ..., m_\ell, j_1, j_2, ..., j_\ell, \tau_1, \tau_2, ..., \tau_\ell, \tau'_1, \tau'_2, ..., \tau'_\ell$, such that,
      (a) For $1 \leqslant i \leqslant \ell$, $m_{i-1} < j_i < m_i$,
      (b) For $1 \leqslant i \leqslant \ell$, $\text{content}(\tau_i) = \{\langle 1, p(j)\rangle \mid m_{i-1} \leqslant j < m_i \wedge j \neq j_i\}$.
      (c) For $1 \leqslant i \leqslant \ell$, $\text{content}(\tau'_i) = \{\langle 1, p(j)\rangle \mid m_{i-1} \leqslant j < m_i\}$.
      (d) For $1 \leqslant i \leqslant \ell$, $M_*(\tau_0 \diamond \tau_1 \diamond \cdots \tau_{i-1} \diamond \tau_i) = M_*(\tau_0 \diamond \tau_1 \diamond \cdots \tau_{i-1} \diamond \tau'_i)$
      (e) For $1 \leqslant i \leqslant \ell$, $\pi_1(M_*(\tau_0 \diamond \tau_1 \diamond \cdots \tau_{i-1} \diamond \tau_i)) \in P_s$.
4. Let $m_1, m_2, ..., m_\ell, j_1, j_2, ..., j_\ell, \tau_1, \tau_2, ..., \tau_\ell, \tau'_1, \tau'_2, ..., \tau'_\ell$, be as found in step 3.
   Let $Y = \text{content}(\sigma_s) \cup \{\langle 1, p(j)\rangle \mid m_0 \leqslant j \leqslant m_\ell\} \setminus \{\langle 1, p(j_i)\rangle \mid 1 \leqslant i \leqslant \ell\}$.
   For $1 \leqslant i \leqslant \ell$, 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 $1 \leqslant i \leqslant \ell$, 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 $z = m_\ell$, 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 W_e belongs to $\mathscr{L}_1$. 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 $\ell, m_i, j_i, \tau_i, \tau'_i$ be as in step 4. Now, for $1 \leqslant i \leqslant \ell$, $W_{p(j_i)} \in \mathscr{L}_2$ and $W_{p(j_i)}$ are pairwise infinitely different. Thus, by the pigeonhole principle, there exists an i, $1 \leqslant i \leqslant \ell$, such that 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 $W_{p(j_i)} \backslash \{\langle 1, p(j_i)\rangle\}$. Furthermore, let

$$T'_i = \tau_0 \diamond \tau_1 \diamond \cdots \diamond \tau_{i-1} \diamond \tau'_i \diamond \tau_{i+1} \diamond \cdots \diamond \tau_\ell \diamond T_i$$

$$T''_i = \tau_0 \diamond \tau_1 \diamond \cdots \diamond \tau_{i-1} \diamond \tau_i \diamond \tau_{i+1} \diamond \cdots \diamond \tau_\ell \diamond T_i.$$

Note that T'_i is a text for $W_{p(j_i)}$. However, we have ProgSet$(M, T'_i) =$ ProgSet(M, T''_i) $=$ ProgSet(M, σ_s) (the first equality follows from the definition of k-bounded example-memory inference (cf. Definition 4) 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 $\mathscr{L} \notin TxtBem^k Fex^*$. Q.E.D.

Next we show the second theorem announced above. Theorem 8 below says that, for suitable concept domains, the k-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 $TxtFb^k Ex^*$ in Theorem 8 can be replaced by $TxtFb^k Fex^*$.

THEOREM 8. *$TxtBem^1 Ex \backslash TxtFb^k Ex^* \neq \emptyset$, 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 Questions(Q_k, q, x) the questions asked by $Q_k(q, x)$. For all L, let C^i_L denote the set $\{x \mid \langle i, x\rangle \in L\}$. We say that e is *nice* iff $C^0_{W_e} = \{e\}$, and $C^1_{W_e} = \emptyset$.

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

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

Next, we show $\mathscr{L} \notin TxtFb^k Ex^*$. 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 $TxtFb^k Ex^*$-identifies $\mathscr{L}$. For $\sigma = x_0, x_1, ..., x_\ell$, let $M_0(\sigma) = M(x_0)$ and for $i < \ell$ let $M_{i+1}(\sigma) = M(M_i(\sigma), A^i_k(Q_k(M_i(\sigma), x_{i+1})), x_{i+1})$, where A^i_k answers the questions based on whether the corresponding elements appear in $\{x_j \mid j \leqslant i\}$. By the operator recursion theorem (cf. Case, 1974, 1994) 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 content$(\sigma_0) = \{\langle 0, p(0)\rangle\}$. Let avail $= 1$. Intuitively, avail denotes a number such that, for all $j \geqslant$ 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 $W_{p(j_i)}$, $W_{p(j'_i)}$, $i \leqslant k$, 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
        $C^0_{\mathrm{content}(\sigma)} = \{p(0)\}$,
        $C^1_{\mathrm{content}(\sigma)} = \varnothing$, and
        $M_*(\sigma) \neq M_*(\sigma_s)$.
3. Let $\ell = 1 + \max C^2_{\mathrm{content}(\sigma_s)}$.
   Let $\tau_0$ be such that $\mathrm{content}(\tau_0) = \{\langle 1, \ell \rangle\}$.
   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
              $\mathrm{content}(\tau) \subseteq \{\langle 3, x \rangle \mid x \in \mathbb{N}\}$, 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 $= \bigcup_{\gamma \diamond y \subseteq \tau_t} \mathrm{Questions}(Q_k, M_*(\sigma_s \diamond \gamma), y)$.
          Set avail $= 1 + \mathrm{avail} + \ell + \max\{x \mid \langle 2, p(x) \rangle \in \mathrm{Ques}\}$
          (* Note that this implies $p(\mathrm{avail}) > \ell$ and any $p(j)$ such that a ques-
             tion 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 $i \leqslant k$, let $j_i = \mathrm{avail} + i$.
          For $i \leqslant k$, let $j'_i = \mathrm{avail} + k + 1 + i$.
          Let avail $= \mathrm{avail} + 2 * (k + 1)$.
          For $i \leqslant k$, 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 corresponding elements appear in $\mathrm{content}(\sigma_s \diamond \tau_t) \cup B$, and
            $\{\langle 2, p(j_i) \rangle, \langle 2, p(j'_i) \rangle\} \cap \mathrm{Questions}(Q_k, q, \langle 3, x \rangle) = \varnothing]\}$.
          For $i \leqslant k$, let $x^0_i, x^1_i, \ldots$, denote a 1–1 enumeration of elements of $O_i$.
          For $i \leqslant k$, let $W_{p(j_i)} = \mathrm{content}(\sigma_s \diamond \tau_t) \cup \{\langle 2, p(j_i) \rangle\} \cup \{x^{2r}_i \mid |O_i| > 2r\}$.
          For $i \leqslant k$, let $W_{p(j'_i)} = \mathrm{content}(\sigma_s \diamond \tau_t) \cup \{\langle 2, p(j'_i) \rangle\} \cup \{x^{2r+1}_i \mid |O_i|$
          $> 2r + 1\}$.
      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 $\mathrm{content}(\tau) \cup \{\langle 3, t \rangle\}$ 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 $\sigma_{s+1}$ be an extension of $\sigma$ 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, $L = W_{p(0)} \in \mathscr{L}_1$. 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, $L = (W_{p(0)} \cup \{\langle 1, \ell\rangle\}) \in \mathscr{L}_2$. 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 $i \leqslant k$) 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 the 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 one another. 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 $B \subseteq C^3_{\mathbb{N}}$, for all texts T for B, $M_*(\sigma_s \diamond \tau_t \diamond T) = q$, and

(d) for all $B \subseteq O_i$, 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 $\mathscr{L}_3$.

From the above cases we have that $\mathscr{L} \notin TxtFb^k Ex^*$. Q.E.D.

3.3. *Iterative Learning*

In this subsection we show that *redundancy* in the hypothesis space may considerably increase the learning power of iterative learners. Intuitively, *redundancy* means that the hypothesis space $\mathscr{H}$ is larger than necessary; i.e., there is at least one hypothesis in $\mathscr{H}$ not describing any concept from the target class or one concept possesses at least two different descriptions in $\mathscr{H}$. Thus, *nonredundant* hypothesis spaces are as small as possible.

Formally, a hypothesis space $\mathscr{H} = (h_j)_{j \in \mathbb{N}}$ is *nonredundant* for some target concept class $\mathscr{L}$ iff range$(\mathscr{H})$ = range$(\mathscr{L})$ and $h_i \neq h_j$ for all $i, j \in \mathbb{N}$ with $i \neq j$. Otherwise, $\mathscr{H}$ is a *redundant* hypothesis space for $\mathscr{L}$.

Lange and Zeugmann (1996a) point out that redundancy may serve as a resource for iterative learners allowing them to *overgeneralize* in learning stages before convergence. Their proof uses an argument based on the noncomputability of the halting problem. Next, we show the weakness of nonredundant hypothesis spaces by applying a purely information-theoretic argument, again on the level of indexed families.

THEOREM 9. *There is an indexed family $\mathscr{L}$ such that*

(1) *$\mathscr{L} \in TxtItEx_{\mathscr{H}}$ for a class preserving redundant hypothesis space $\mathscr{H}$, and*

(2) *$\mathscr{L} \notin TxtItEx_{\hat{\mathscr{H}}}$ for every nonredundant hypothesis space $\hat{\mathscr{H}}$.*

Proof. Let $\mathscr{L}_{\text{red}}$ be the canonical enumeration of all languages $L \subseteq \{b\}^+$ with $|L| = 2$ or $|L| = 3$. We show that $\mathscr{L}_{\text{red}}$ satisfies assertions (1) and (2). For proving (1), we define $\mathscr{H} = (h_{\langle i, j, k \rangle})_{i, j, k \in \mathbb{N}}$. The semantics is as follows. If $i, j \in \mathbb{N}^+$ and $i \neq j$ then $h_{\langle i, j, k \rangle} = \{b^i, b^j, b^k\}$ in case that $k \neq 0$, and $h_{\langle i, j, k \rangle} = \{b^i, b^j\}$, if $k = 0$. Furthermore, we set $h_{\langle i, i, k \rangle} = \{b^i, b^{i+1}\}$ for $i > 0$, and $h_{\langle i, j, k \rangle} = \{b, b^2\}$ otherwise. Obviously, $\mathscr{H}$ is class preserving. Since it contains for every L with $|L| = 2$ at least two descriptions, it is redundant, too.

We define $M(b^i) = \langle i, i, 0 \rangle$, and

$$M(\langle i, j, k \rangle, b^z) = \begin{cases} \langle i, j, k \rangle, & \text{if} \quad b^z \in h_{\langle i, j, k \rangle}, \\ \langle i, z, 0 \rangle & \text{if} \quad b^z \notin h_{\langle i, j, k \rangle}, \quad i = j, \\ \langle i, j, z \rangle, & \text{otherwise.} \end{cases}$$

One easily verifies that M $TxtItEx_{\mathscr{H}}$-learns $\mathscr{L}_{\text{red}}$. We omit the details.

Next, we show assertion (2). Suppose that there are a nonredundant hypothesis space $\hat{\mathscr{H}} = (\hat{h}_j)_{j \in \mathbb{N}}$ for $\mathscr{L}_{\text{red}}$ and an IIM M which $TxtItEx_{\hat{\mathscr{H}}}$-identifies $\mathscr{L}_{\text{red}}$. We define a text T for some L and a text T' for some L' such that M fails to learn at least one of them.

Let $j_0 = M(b)$; then we must have $b \in \hat{h}_{j_0}$. For seeing this suppose that $b \notin \hat{h}_{j_0}$, and let $\hat{h}_{j_0} = \{b^\ell, b^m\} \cup \{b^n\}$ with $\ell \neq m$. By assumption, M has to infer $\hat{h}_{j_0}$, and therefore it converges on text $\hat{T} = b^\ell, b^m, b^n, b^\ell, b^m, b^n, \ldots$ to j_0, since $\hat{\mathscr{H}}$ is nonredunant. Hence, $M(j_0, b^\ell) = j_0$ and M, when fed the text $\tilde{T} = b, b^\ell, b^\ell, \ldots$, converges to j_0, but $b \notin \hat{h}_{j_0}$.

Now, let $\hat{h}_{j_0} = \{b, b^m\} \cup \{b^n\}$. Applying *mutatis mutandis* the same argument as above, we have $M(j_0, b^m) = j_0$. We distinguish the following cases. First, if $|\hat{h}_{j_0}| = 3$ then M fails to learn $L = \{b, b^m\}$ from text $T = b, b^m, b^m, \ldots$.

Finally, let $|\hat{h}_{j_0}| = 2$. Hence, $\hat{h}_{j_0} = \{b, b^m\}$. As above, one easily verifies that $M(j_0, b) = j_0$, too. Now, select any $z > 1$ with $z \neq m$. Set $L = \{b, b^m, b^z\}$, $L' = \{b, b^z\}$, and $T = b, b^m, b^z, b^z, \ldots$, as well as $T' = b, b, b^z, b^z, \ldots$. Since $M_*(T_1) = M_*(T'_1)$, M

converges, if ever, on both texts T and T' to the same hypothesis, and thus fails to learn L or L'. Since $L, L' \in \mathscr{L}_{\text{red}}$, this proves assertion (2). Q.E.D.

A closer look at the latter proof shows that we have exploited two properties any IIM M must possess provided it $TxtItEx_{\mathscr{H}}$-learns the concept class $\mathscr{L}_{\text{red}}$ with respect to some nonredundant hypothesis space $\mathscr{H}$, i.e., *conservativeness* and *consistency*.[7] Thus, it is natural to ask whether or not these conditions have to be fulfilled in general, too. The answer is yes and no, that is, conservativeness is inevitable (cf. Theorem 10), while consistency is not.

THEOREM 10. *Let $\mathscr{C}$ be any concept class, and let $\mathscr{H} = (h_j)_{j \in \mathbb{N}}$ be any nonredundant hypothesis space for $\mathscr{C}$. Then, every IIM M that $TxtItEx_{\mathscr{H}}$-infers $\mathscr{C}$ is conservative.*

Proof. Suppose the converse, i.e., there are a concept $c \in \mathscr{C}$, a text $T = (x_j)_{j \in \mathbb{N}} \in text(c)$, and a $y \in \mathbb{N}$ such that, for $j = M_*(T_y)$ and $k = M_*(T_{y+1}) = M(j, x_{y+1})$, both $j \neq k$ and $T^+_{y+1} \subseteq h_j$ are satisfied. The latter implies $x_{y+1} \in h_j$, and thus we may consider the following text $\tilde{T} \in text(h_j)$. Let $\hat{T} = (\hat{x}_j)_{j \in \mathbb{N}}$ be any text for h_j and let $\tilde{T} = \hat{x}_0, x_{y+1}, \hat{x}_1, x_{y+1}, \hat{x}_2, \ldots$. Since M has to learn h_j from $\tilde{T}$ there must be a $z \in \mathbb{N}$ such that $M_*(\tilde{T}_{z+r}) = j$ for all $r \geqslant 0$. But $M_*(\tilde{T}_{2z+1}) = M(j, x_{y+1}) = k$, a contradiction. Q.E.D.

Consider the set $\mathscr{L}_s$ of all singleton languages over $\{b\}^+$ and any nonredundant hypothesis space $\mathscr{H}$ for it. Just defining an IIM M by $M(b^z) = 0$ and behaving otherwise consistently shows that consistency may be violated. Clearly, $\mathscr{L}_s$ can also be learned iteratively and consistently with respect to $\mathscr{H}$. Naturally, the question arises whether this simple example is hiding some general insight; i.e., if some indexed family can be iteratively learned with respect to some nonredundant hypothesis space then there is also an iterative and consistent learner doing the same job. This is not the case! As we shall see, there are prominent indexed families, e.g., the pattern languages (cf. Subsection 3.4 below), that can be iteratively learned with respect to some nonredundant hypothesis space, but every iterative IIM doing so has inevitably to output *inconsistent* intermediate hypotheses.

The final theorem in this subsection sheds some light on the limitations of iterative IIM's that are supposed to learn consistently with respect to nonredundant hypothesis spaces. Additionally, it is an essential tool in achieving the nonlearnability result for pattern languages announced above.

Let $\mathscr{L}$ be any indexed family. $\mathscr{L}$ meets the *superset condition* if, for all $L, L' \in$ range$(\mathscr{L})$, there is some $\hat{L} \in$ range$(\mathscr{L})$ being a superset of both L and L'.

THEOREM 11. *Let $\mathscr{L}$ be any index family meeting the superset condition, and let $\mathscr{H} = (h_j)_{j \in \mathbb{N}}$ be any nonredundant hypothesis space for $\mathscr{L}$. Then, every consistent IIM M that $TxtItEx_{\mathscr{H}}$-infers $\mathscr{L}$ may be used to decide the inclusion problem for $\mathscr{H}$.*

Proof. Let Σ be the underlying alphabet, and let $(w_j)_{j \in \mathbb{N}}$ be an effective enumeration of all strings in Σ^*. Then, for every $i \in \mathbb{N}$, $T^i = (x^i_j)_{j \in \mathbb{N}}$ is the following computable text for h_i. Let z be the least index such that $w_z \in h_i$. Recall that, by

[7] An IIM M is said to be *consistent* iff $T^+_x \subseteq h_{M(T_x)}$ for all $x \in \mathbb{N}$ and every text T for every concept c in the target class $\mathscr{C}$.

definition, $h_i \neq \emptyset$, since $\mathscr{H}$ is an indexed family, and thus w_z must exist. Then, for all $j \in \mathbb{N}$, we set $x^i_j = w_j$, if $w_j \in h_i$, and $x^i_j = w_z$, otherwise.

We claim that the following algorithm *Inc* decides, for all $i, k \in \mathbb{N}$, whether or not $h_i \subseteq h_k$.

Algorithm *Inc*. On input $i, k \in \mathbb{N}$ do the following: Determine the least $y \in \mathbb{N}$ with $i = M_*(T^i_y)$. Check whether or not $T^{i,+}_y \subseteq h_k$. In case it is, output "Yes," and stop. Otherwise, output "No," and stop.

Clearly, since $\mathscr{H}$ is an indexed family and T^i is a computable text, *Inc* is an algorithm. Moreover, M learns h_i on every text for it, and $\mathscr{H}$ is a nonredundant hypothesis space. Hence, M has to converge on text T^i to i, and therefore *Inc* has to terminate.

It remains to verify the correctness of *Inc*. Let $i, k \in \mathbb{N}$. Clearly, if *Inc* outputs "No," a string $s \in h_i \backslash h_k$ has been found, and $h_i \nsubseteq h_k$ follows.

Next, consider the case that *Inc* outputs "Yes." Suppose to the contrary that $h_i \nsubseteq h_k$. Then, there is some string $s \in h_i \backslash h_k$. Now, consider M when fed the text $T = T^i_y \diamond T^k$. Since $T^{i,+}_y \subseteq h_k$, T is a text for h_k. Since M learns h_k, there is some $r \in \mathbb{N}$ such that $k = M_*(T^i_y \diamond T^k_r)$. By assumption, there are some $\hat{L} \in \text{range}(\mathscr{L})$ with $h_i \cup h_k \subseteq \hat{L}$, and some text $\hat{T}$ for $\hat{L}$ having the initial segment $T^i_y \diamond s \diamond T^k_r$. By Theorem 10, M is conservative. Since $s \in h_i$ and $i = M_*(\hat{T}_y)$, we obtain $M_*(\hat{T}_{y+1}) = M(i, s) = i$. Consequently, $M_*(T^i_y \diamond s \diamond T^k_r) = M_*(T^i_y \diamond T^k_r)$. Finally, since $s \in \hat{T}^+_{y+r+2}$, $k = M_*(T^i_y \diamond T^k_r)$, and $s \notin h_k$, M fails to consistently learn $\hat{L}$ from text $\hat{T}$, a contradiction. This proves the theorem. Q.E.D.

3.4. The Pattern Languages

The pattern languages (defined two paragraphs below) were formally introduced by Angluin (1980a) and have been widely investigated (cf., e.g., Salomaa, 1994a, 1994b, and Shinohara and Arikawa, 1995, for an overview). Moreover, Angluin (1980a) proved that the class of all pattern languages is learnable in the limit from positive data. Subsequently, Nix (1983), as well as Shinohara and Arikawa (1995) 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.*, 1994; Shinohara and Arikawa, 1995).

Pattern languages and finite unions of pattern languages turn out to be subclasses of Smullyan's (1961) elementary formal systems (EFS). Arikawa *et al.* (1992) 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, 1991). 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 (1994) 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 (1996) has established the learnability from only positive data of an even larger class of logic programs. These results have consequences for inductive logic programming.[8]

Patterns and pattern languages are defined as follows (cf. Angluin, 1980a). Let $\mathscr{A} = \{0, 1, ...\}$ be any nonempty finite alphabet containing at least two elements, and let $\mathscr{A}^*$ be the free monoid over $\mathscr{A}$. The set of all finite nonnull strings of symbols from $\mathscr{A}$ is denoted by $\mathscr{A}^+$, i.e., $\mathscr{A}^+ = \mathscr{A}^* \setminus \{\varepsilon\}$, where ε denotes the empty string. By $|\mathscr{A}|$ we denote the cardinality of $\mathscr{A}$. Furthermore, let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\mathscr{A} \cap X = \emptyset$. *Patterns* are nonempty strings over $\mathscr{A} \cup X$, 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 $x_0, ..., x_{k-1}$. Moreover, for every j with $0 \leqslant j < k-1$, 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 $s \in \mathscr{A}^*$ 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. 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*.

Now let $\pi \in Pat_k$, and let $u_0, ..., u_{k-1} \in \mathscr{A}^+$. We denote by $\pi[u_0/x_0, ..., u_{k-1}/x_{k-1}]$ the string $s \in \mathscr{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, ..., k-1$, in the pattern π. The tuple $(u_0, ..., u_{k-1})$ is called *substitution*. For every $\pi \in Pat_k$ we define the *language generated by pattern* π by $L(\pi) = \{\pi[u_0/x_0, ..., u_{k-1}/x_{k-1}] \mid u_0, ..., u_{k-1} \in \mathscr{A}^+\}$.[9] 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 $\mathscr{A}$.

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, $Pat(k) = \{Q \mid Q \subseteq Pat,\ |Q| \leqslant k\}$ and $PAT(k) = \{L \mid (\exists Q \in Pat(k))[L = L(Q)]\}$. Finally, let $L \subseteq \mathscr{A}^+$ be a language, and let $k \in \mathbb{N}^+$; we define $Club(L, k) = \{Q \mid |Q| \leqslant k,\ L \subseteq L(Q),\ (\forall Q')[Q' \subset Q \Rightarrow L \nsubseteq L(Q')]\}$. *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, 1980a). Subsequently, Lange and Wiehagen (1991) showed PAT to be $TxtItEx_{Pat}$-inferable. Their algorithm is allowed to output inconsistent intermediate hypotheses. Next, we argue that inconsistency cannot be avoided when iteratively learning PAT with respect to Pat. Note that Pat is a nonredundant hypothesis space. PAT also meets the superset condition, since $L(x_0) = \mathscr{A}^+$.

[8] 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.

[9] We study so-called *nonerasing* 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, 1988).

Moreover, the inclusion problem for *Pat* is undecidable (cf. Jiang *et al.*, 1993). Therefore, by Theorem 11, we immediately arrive at the following corollary.

COROLLARY 12. *There is no consistent IIM M that $TxtItEx_{Pat}$-learns PAT.*

As a matter of fact, the latter corollary generalizes to *all* nonredundant hypothesis spaces for *PAT*. All the ingredients to prove this can be found in Zeugmann *et al.* (1995). Consequently, if unions of pattern languages can be iteratively learned at all, then either redundant hypothesis spaces or inconsistent learners cannot be avoided.

As for unions, the first result goes back to Shinohara (1983) who proved the class of all unions of at most two pattern languages to be in $TxtEx_{Pat(2)}$. Wright (1989) extended this result to $PAT(k) \in TxtEx_{Pat(k)}$ for all $k \geqslant 1$. Moreover, Theorem 4.2 in Shinohara and Arimura's (1996), together with a lemma from Blum and Blum (1975) shows that $\bigcup_{k \in \mathbb{N}} PAT(k)$ is not $TxtEx_{\mathscr{H}}$-inferable for every hypothesis space $\mathscr{H}$. 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 13 below); each $PAT(k)$ is *iteratively* learnable! Moreover, the learner presented in the proof is consistent, too. Thus, the hypothesis space used had to be designed to be *redundant*.

PROPOSITION 1. (1) *For all $L \subseteq \mathscr{A}^+$ and all $k \in \mathbb{N}^+$, $Club(L, k)$ 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 $k \in \mathbb{N}^+$, let $L \subseteq \mathscr{A}^+$ be any language, and suppose $T = (s_j)_{j \in \mathbb{N}} \in text(L)$. Then,*

(1) *$Club(T_0^+, k)$ can be effectively obtained from s_0, and $Club(T_{n+1}^+, k)$ can be effectively obtained from $Club(T_n^+, k)$ and s_{n+1} (* note the iterative nature *).*

(2) *The sequence $Club(T_0^+, k)$, $Club(T_1^+, k)$, ... converges to $Club(L, k)$.*

Proof. For proving assertion (1), fix any $k \geqslant 1$ and suppose $T = s_0, s_1 ..., s_n, s_{n+1}, ...$ to be a text for L. Furthermore, let $\mathscr{S}_0 = \{\{\pi\} \mid s_0 \in L(\pi)\}$. We proceed inductively; for $n \geqslant 0$, we define $\mathscr{S}'_{n+1} = \{Q \in \mathscr{S}_n \mid s_{n+1} \in L(Q)\} \cup \{Q \cup \{\pi\} \mid Q \in \mathscr{S}_n, s_{n+1} \notin L(Q), |Q| < k, s_{n+1} \in L(\pi)\}$, and then $\mathscr{S}_{n+1} = \{Q \in \mathscr{S}'_{n+1} \mid (\forall Q' \in \mathscr{S}'_{n+1}) [Q' \not\subset Q]\}$.

Note that $\mathscr{S}_0$ can be effectively obtained from s_0, since every pattern π with $s_0 \in L(\pi)$ must satisfy $|\pi| \leqslant |s_0|$. 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, $\mathscr{S}_{n+1}$ can be effectively obtained from $\mathscr{S}_n$ and s_{n+1}, too. Also it is easy to verify, by induction on n, that $\mathscr{S}_n = Club(T_n^+, k)$. Thus, (1) is satisfied.

Next, we show assertion (2). Consider a tree $\mathscr{T}$ formed mimicking the above construction of $\mathscr{S}_n$ as follows. The nodes of $\mathscr{T}$ 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 $s_n \in L(\pi)$.

Note that the number of children is equal to the number of patterns π such that $s_n \in L(\pi)$.

Suppose $\mathscr{U}_n = \{Q \mid (\exists v$ 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 $\mathscr{S}_n = \{Q \in \mathscr{U}_n \mid (\forall Q' \in \mathscr{U}_n)[Q' \not\subset Q]\}$.

Since the number of nonempty labels on any path of the tree is bounded by k, using König's lemma we have that the number of nodes with nonempty label must be finite. Thus the sequence $\mathscr{U}_0, \mathscr{U}_1, \ldots$ converges. Hence, the sequence $\mathscr{S}_0 = Club(T_0^+, k)$, $\mathscr{S}_1 = Club(T_1^+, k)$, ... converges, to say $\mathscr{S}$. Now, for all $Q \in \mathscr{S}$, for all n, $T_n^+ \subseteq L(Q)$. Therefore, for all $Q \in \mathscr{S}$, $L \subseteq L(Q)$. Also, for all $Q \in \mathscr{S}$ and $Q' \subset Q$, for all but finitely many n, $T_n^+ \not\subseteq L(Q')$. Thus for all $Q \in \mathscr{S}$ and $Q' \subset Q$, $L \not\subseteq L(Q')$. It follows that $\mathscr{S} = Club(L, k)$, and hence, assertion (2) of Lemma 4 is proved.

Q.E.D.

THEOREM 13. *For all* $k \geqslant 1$, $PAT(k) \in TxtItEx$.

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 $x, y \in \mathbb{N}$, $W_{\text{pad}(x,\, y)} = W_x$. For a finite class $\mathscr{S}$ of sets of patterns, let $g(\mathscr{S})$ denote a grammar obtained, effectively from $\mathscr{S}$, for $\bigcap_{Q \in \mathscr{S}} L(Q)$.

Let $L \in PAT(k)$, and let $T = (s_j)_{j \in \mathbb{N}} \in text(L)$. The desired IIM M is defined as follows: We set $M_0(T) = M(s_0) = \text{pad}(g(Club(T_0^+, k)), \text{can}(Club(T_0^+, k))$, and for all $n > 0$, let

$$\begin{aligned} M_{n+1}(T) &= M(M_n(T), s_{n+1}) \\ &= \text{pad}(g(Club(T_{n+1}^+, k)), \text{can}(Club(T_{n+1}^+, k))). \end{aligned}$$

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 $PAT(k)$.

Q.E.D.

3.5. *Further Comparisons*

Finally, we turn our attention to the differences and similarities between Definition 4 and a variant of k-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.* (1994) and is a slight modification of k-memory limited learning defined in Osherson *et al.* (1986), where the learner could just memorize the latest k data items received. It has been thoroughly studied by Fulk *et al.* (1994). The main differences to Definition 4 are easily explained. In Definition 4 the k-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 λ denote the empty sequence.

DEFINITION 6 (Fulk *et al.*, 1994). Let $\mathscr{X}$ be a learning domain, and let $k \in \mathbb{N}$; then

(a) *mem*: *SEQ* $\to$ *SEQ is a k-memory function* iff, $mem(\lambda)=\lambda$, and, for all sequences $\sigma \in SEQ$ and all $x \in \mathscr{X}$, $\text{content}(mem(\sigma \diamond x)) \subseteq \text{content}(\sigma \diamond x)$, $|mem(\sigma \diamond x)| \leqslant k$ and $\text{content}(mem(\sigma \diamond x)) \subseteq \text{content}(mem(\sigma)) \cup \{x\}$.

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

DEFINITION 7 (Fulk *et al.*, 1994). Let $k \in \mathbb{N}$; then we set $TxtMb^k Ex=\{\mathscr{C} \subseteq \wp(\mathscr{X}) \mid \text{there exists a } k\text{-memory bounded machine } M\ TxtEx\text{-inferring } \mathscr{C})\}$.

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

THEOREM 14. *$TxtMb^1 Ex \setminus TxtBem^k Ex \neq \emptyset$ for all $k \in \mathbb{N}$.*

Proof. Assume any $k \in \mathbb{N}$. Let $L_1=\{\langle i, x\rangle \mid x \in \mathbb{N},\ i \leqslant k\}$ and for all $m_0, ..., m_k \in \mathbb{N}$ let $L_k^{m_0, ..., m_k}=\{\langle 0, x\rangle \mid x<m_0\} \cup \cdots \cup \{\langle k, x\rangle \mid x<m_k\} \cup \{\langle k+1, x\rangle \mid x \in \mathbb{N}\}$. Furthermore, let $\mathscr{L}_k$ be the collection of L_1 and all $L_k^{m_0, ..., m_k}$, $m_0, ..., m_k \in \mathbb{N}$. Now, one easily shows that $\mathscr{L}_k \notin TxtBem^k Ex$ using the same ideas as in Fulk *et al.* (1994).

On the other hand, $\mathscr{L}_k \in TxtMb^1 Ex$. The crucial point here is that the 1-memory function *mem* can be applied to encode, if necessary, the appropriate $m_0, ..., m_k$ 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 $m_0, ..., m_k$, $W_{\text{pad}(0, ..., 0)}=L_1$ and $W_{(m_0+1, m_1, ..., m_k)}=L_k^{m_0, ..., m_k}$. Furthermore, assume any recursive function g that satisfies, for all $m_0, ..., m_k$, $g(x)=\langle m_0, ..., m_k\rangle$ for infinitely many x.

M, and its associated memory function *mem*, witnessing that $\mathscr{L} \in TxtMb^1 Ex$ is defined as follows. M's output will be of the form, $\text{pad}(m_0', ..., m_k')$. Let $L \in \mathscr{L}$, 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 $g(x)=\langle m_0, ..., m_k\rangle$ with $m_i=\max\{m' \mid \langle i, m'\rangle \in T_z^+\}$ for all $i \leqslant k$ 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 $M(s_0) = \text{pad}(0, ..., 0)$. Otherwise, let $M(s_0) = \text{pad}(1, 0, ..., 0)$.

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

End

Clearly, if the target language L equals L_1, M always outputs a correct hypothesis. Otherwise, L equals $L_k^{m_0, ..., m_k}$ for some $m_0, ..., m_k$. Since $|L \cap \{\langle i, x \rangle \mid i \leqslant k,\ x \in \mathbb{N}\}| < \infty$, and by the choice of g, M must receive an element $\langle k+1, x \rangle$ with $g(x) = \langle m_0, ..., m_k \rangle$ after all $k+1$ elements $\langle i, m_i \rangle$, $i \leqslant k$, 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 $\mathscr{L}$. Q.E.D.

The latter theorem immediately allows the following corollary.

Corollary 15. *$TxtBem^k Ex \subset TxtMb^k Ex$ for all $k \in \mathbb{N}$.*

Proof. $TxtBem^k Ex \subseteq TxtMb^k Ex$ for all $k \in \mathbb{N}$, since the k-memory bounded learner may easily simulate the k-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 14. Q.E.D.

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 k-memory bounded identification as defined in Fulk *et al.* (1994) into the right perspective.

Theorem 16. *$TxtFb^1 Ex \subset TxtMb^1 Ex$.*

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 Theorem 8 and Corollary 15.

Let M, together with the query asking function Q, witness that $\mathscr{C} \in TxtFb^1 Ex$. The desired IIM M', and its associated memory function *mem*, witnessing that $\mathscr{C} \in TxtMb^1 Ex$ are defined as follows: Let $c \in \mathscr{C}$, 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 $Q(q, s_z) \in \text{content}(T_z)$, then $mem(T_z) = s_z$. Otherwise, $mem(T_z) = \lambda$. Recall that λ 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 $c \in \mathscr{C}$ as required. Q.E.D

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 $TxtBem^k Ex^a$ and $TxtBem^k Fex^a$ 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.* (1986), Fulk *et al.* (1994), and Lange and Zeugmann (1996a) in various directions.

First, the class of all unions of at most k pattern languages has been shown to be simultaneously both iteratively and consistently 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 13 is highly redundant, too. Moreover, we proved this redundancy to be necessary; i.e., no *iterative and consistent learner* can identify all unions of at most k pattern languages with respect to a 1–1 hypothesis space. It remains, however, open whether or not there exists an inconsistent iterative learner inferring $PAT(k)$ with respect to a nonredundant 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 (1991) 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 a large class of probability distributions (cf. Zeugmann, 1995, 1998; and Rossmanith and Zeugmann, 1998).

Second, we considerably generalized the model of feedback inference introduced in Lange and Zeugmann (1996a) 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. Theorems 7 and 8). 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 a suitable tool for accomplishing this goal (cf., e.g., Angluin, 1980b; Blum and Blum, 1975; Zeugmann *et al.*, 1995).

Additionally, feed-back identification and k-bounded example-memory inference have been considered in the general context of classes of recursively enumerable concepts rather than uniformly recursive ones as done in Lange and Zeugmann (1996a). As our Theorem 5 shows, there are subtle differences. Furthermore, a closer look at the proof of Theorem 5 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. Interestingly, our theorems hold in this case, too. It is, however, currently open whether or not sequentially querying the database does have 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 (1993) for mind changes. Intuitively, the parameter k is now specified to be a constructive ordinal, and the k-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 a variety of surprising results (cf., e.g., Jain and Sharma, 1995; Meyer, 1995, 1997), 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 the strength and limitations of incremental learning are, further investigations are necessary to deal with the impact of incremental inference to the complexity of the resulting learner. First results along this line are established in Wiehagen and Zeugmann (1994), and we shall see what the future brings concerning this interesting topic.

ACKNOWLEDGMENT

We heartily thank the anonymous referees for their careful reading and comments which improved the paper considerably.

Received October 8, 1997; final manuscript received November 6, 1998

REFERENCES

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

Angluin, D. (1980b), Inductive inference of formal languages from positive data, *Inform. and Control* **45**, 117–135.

Arikawa, S., Shinohara, T., and Yamamoto, A. (1992), Learning elementary formal systems, *Theoret. Comput. Sci.* **95**, 97–113.

Arimura, H., and Shinohara, T. (1994), 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, Burke, VA.

Blum, M. (1967), A machine independent theory of the complexity of recursive functions, *J. Assoc. Comput. Mach.* **14**, 322–336.

Blum, L., and Blum, M. (1975), Toward a mathematical theory of inductive inference, *Inform. and Control* **28**, 122–155.

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

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

Case, J. (1988), The power of vacillation, *in* "Proceedings of the 1st Workshop on Computational Learning Theory" (D. Haussler and L. Pitt, Eds.), pp. 196–205, Morgan Kaufmann, San Mateo, CA.

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

Case, J. (1996), "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.

Case, J., and Smith, C. H. (1983), Comparison of identification criteria for machine inductive inference, *Theoret. Comput. Sci.* **25**, 193–220.

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

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

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

Freivalds, R., and Smith, C. H. (1993), On the role of procrastination for machine learning, *Inform. and Comput.* **107**, 237–271.

Fulk, M., Jain, S., and Osherson, D. N. (1994), Open problems in systems that learn, *J. Comput. System Sci.* **49**, 589–604.

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

Hopcroft, J. E., and Ullman, J. D. (1969), "Formal Languages and their Relation to Automata," Addison-Wesley, Reading, MA.

Jain, S., and Sharma, A. (1994), On monotonic strategies for learning r.e. languages, *in* "Proceedings of the 5th International Workshop on Algorithmic Learning Theory" (K. P. Jantke and S. Arikawa, Eds.), Lecture Notes in Artificial Intelligence, Vol. 872, pp. 349–364, Springer-Verlag, Berlin.

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

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

Kloesgen, W. (1995), Efficient discovery of interesting statements in databases, *J. Intell. Inform. Systems* **4**, 53–69.

Krishna Rao, M. R. K. (1996), A class of Prolog programs inferable from positive data, *in* "Proceedings of the 7th International Workshop on Algorithmic Learning Theory" (S. Arikawa and A. K. Sharma, Eds.), Lecture Notes in Artificial Intelligence, Vol. 1160, pp. 272–284, Springer-Verlag, Berlin.

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

Lange, S., and Zeugmann, T. (1993a), Language learning in dependence on the space of hypotheses, *in* "Proceedings of the 6th Annual ACM Conference on Computational Learning Theory" (L. Pitt, Ed.), pp. 127–136, ACM Press, New York.

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

Lange, S., and Zeugmann, T. (1993c), Monotonic versus non-monotonic language learning, *in* "Proceedings 2nd International Workshop on Nonmonotonic and Inductive Logic" (G. Brewka, K. P. Jantke, and P. H. Schmitt, Eds.), Lecture Notes in Artificial Intelligence, Vol. 659, pp. 254–269, Springer-Verlag, Berlin.

Lange, S., and Zeugmann, T. (1996a), Incremental learning from positive data, *J. Comput. System Sci.* **53**, 88–103.

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

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

Meyer, L. (1995), Probabilistic language learning under monotonicity constraints, *in* "Proceedings 6th International Workshop on Algorithmic Learning Theory" (K. P. Jantke, T. Shinohara, and T. Zeugmann, Eds.), Lecture Notes in Artificial Intelligence, Vol. 997, pp. 169–184, Springer-Verlag, Berlin.

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

Nix, R. P. (1983), "Editing by Examples," Yale University, Dept. Computer Science, Technical Report 280.

Osherson, D. N., Stob, M., and Weinstein, S. (1986), "Systems that Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists," MIT Press, Cambridge, MA.

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

Rossmanith, P., and Zeugmann, T. (1998), Learning *k*-variable pattern languages efficiently stochastically finite on average from positive data, *in* “Proceedings 4th International Colloquium on Grammatical Inference - ICGI’98” (V. Honavar and G. Slutzki, Eds.), Lecture Notes in Artificial Intelligence, Vol. 1433, pp. 13–24, Springer-Verlag, Berlin.

Salomaa, A. (1994a), Patterns (The formal language theory column), *EATCS Bull.* **54**, 46–62.

Salomaa, A. (1994b), Return to patterns (The formal language theory column), *EATCS Bull.* **55**, 144–157.

Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S., and Arikawa, S. (1994), Knowledge acquisition from amino acid sequences by machine learning system BONSAI, *Trans. Inform. Process. Soc. Jpn.* **35**, 2009–2018.

Shinohara, T. (1983), Inferring unions of two pattern languages, *Bull. Inform. Cybern.* **20**, 83–88.

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

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

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

Smullyan, R. (1961), “Theory of Formal Systems,” Annals of Mathematical Studies, Vol. 47, Princeton Univ. Press, Princeton, NJ.

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

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

Wright, K. (1989), Identification of unions of languages drawn from an identifiable class, *in* “Proceedings of the 2nd Workshop on Computational Learning Theory” (R. Rivest, D. Haussler, and M. Warmuth, Eds.), pp. 328–333, Morgan Kaufmann, San Mateo, CA.

Zeugmann, T. (1995), “Lange and Wiehagen’s Pattern Language Learning Algorithm: An Average-Case Analysis with Respect to its Total Learning Time,” RIFIS Technical Report RIFIS-TR-CS-111, RIFIS, Kyushu University.

Zeugmann, T. (1998), Lange and Wiehagen’s pattern language learning algorithm: An average-case analysis with respect to its total learning time, *Annals of Mathematics and Artificial Intelligence* **23**, 117–145.

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

Zeugmann, T., Lange, S., and Kapur, S. (1995), Characterizations of monotonic and dual monotonic language learning, *Inform. and Comput.* **120**, 155–173.