

Hypothesis Spaces for Learning

Sanjay Jain^{a,1}

^a*Department of Computer Science, National University of Singapore, Singapore 117417,
Republic of Singapore. Email: sanjay@comp.nus.edu.sg*

Abstract

In this paper we survey some results in inductive inference showing how learnability of a class of languages may depend on the hypothesis space chosen. Additionally, optimal hypothesis spaces, usable for every learnable class, are considered. We also discuss results which consider how learnability is effected if one requires learning using every suitable hypothesis space.

1. Introduction

A learning scenario can be described as follows. Consider a learner (a computable device) receiving data, one piece at a time, about some target concept (which is from a class of possible concepts). As the learner is receiving its data, it conjectures a possible description of the target concept. One may consider the learner to be successful if its sequence of conjectures converges to a correct description of the target concept.

In this paper we will be mostly concerned with language learning. A language is some recursively enumerable (r.e.) subset of a universal set. By appropriate coding, one may take the universal set to be the set of natural numbers, $\mathbb{N} = \{0, 1, 2, \dots\}$.

For learning languages, the data provided to the learner is usually the set of elements (positive data) of the language, one element at a time, where all the elements are eventually provided and no non-elements of the language are provided. This form of data presentation is called a text for the language. Another model of data presentation is called informant, where the learner is presented with all the elements of the universal set, one element at a time, appropriately labeled as positive or negative with respect to the target language. The model for presenting only positive data to the learner originates from the observation that in many natural situations the learner gets essentially only positive data. In some scientific studies such as in astronomy [18], one can only observe events that happen, and do not explicitly know what cannot happen (except inferring it by absence). Furthermore, with respect to child learning a language, Gold [20] pointed out that the psycholinguistic literature indicates

¹Supported in part by NUS grant number R252-000-308-112.

that children are rarely informed of grammatical errors. Though some of these claims are open to doubt, see for example [9, 14].

The conjectures of the learner take the form of a grammar from some hypothesis space. We always assume that the hypothesis space is an r.e. indexing of r.e. languages; in some cases we additionally assume that membership question for a hypothesis i is decidable algorithmically in i — in these cases the hypothesis space is an indexed family of recursive languages.

A criterion of success, as considered above, is for the sequence of grammars to converge to a grammar for the target language. This is essentially the criterion of learning first considered by Gold [20], and commonly called explanatory learning (abbreviated **TextEx**-learning, for explanatory learning from text, and **InfEx**-learning, for explanatory learning from informant). Note that the learner is expected to succeed on all possible orders of presentation of the elements of the target language. Learning of one language is usually not much interesting, as some learner, which always outputs the grammar for this language, will succeed. What is more interesting is whether some learner can learn all the languages from a class \mathcal{L} of languages.

We now formally define the criterion described above. A *sequence* is a mapping from \mathbb{N} or an initial segment of \mathbb{N} to $\mathbb{N} \cup \{\#\}$. Content of a sequence σ , denoted $\text{content}(\sigma)$, is $\text{range}(\sigma) - \{\#\}$. $\#$ is considered as a pause symbol, representing no data; this is useful, for example, when one considers presenting data for the empty language. We let $\text{length}(\sigma)$ denote the length of the sequence σ . Let SEQ denote the set of all finite sequences. An infinite sequence T is a *text* for a language L iff $\text{content}(T) = L$. We let T (with or without subscripts/superscripts) range over texts. $T[n]$ denotes the finite sequence consisting of the first n elements of the sequence T .

A *learning machine* M (also called a learner) is an algorithmic mapping (possibly partial) from SEQ to $\mathbb{N} \cup \{?\}$. One can view $M(T[0])$, $M(T[1])$, $M(T[2])$, \dots as the indices output (in that order) by M on T . A learner M converges on a text T to $i \in \mathbb{N}$ (denoted $M(T)\downarrow = i$) iff for all but finitely many n , $M(T[n]) = i$.

Here one interprets the output of the learner as an index for some language in a hypothesis space. Thus output i would represent the conjecture H_i , where $(H_j)_{j \in \mathbb{N}}$ is the hypothesis space used by the learner. The output of $?$ denotes that the learner is not making a formal conjecture (this is useful when one considers some special cases, such as bounding the number of mind changes made by the learner). We will always assume that the hypothesis space is a r.e. family: that is, $\{\langle i, x \rangle : x \in H_i\}$ is recursively enumerable. Here the pairing function, $\langle \cdot, \cdot \rangle$, is some computable, 1-1 and onto mapping from $\mathbb{N} \times \mathbb{N}$ to \mathbb{N} . In some cases we even consider uniformly recursive family (indexed family) as a hypothesis space. In this case, $\{\langle i, x \rangle : x \in H_i\}$ is recursive.

We now give the formal definition for **TextEx**-identification.

Definition 1. [20] Fix a hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$.

(a) A learner M **TextEx** $_{\mathcal{H}}$ -identifies a language L (written: $L \in \mathbf{TextEx}_{\mathcal{H}}(M)$) iff for all texts T for L , (i) $M(T[n])$ is defined for all n , and (ii) there exists an

i such that $M(T)\downarrow = i$ and $H_i = L$.

(b) A learner M **TxtEx** $_{\mathcal{H}}$ -identifies a class \mathcal{L} of languages (written: $\mathcal{L} \subseteq \mathbf{TxtEx}_{\mathcal{H}}(M)$) iff it **TxtEx** $_{\mathcal{H}}$ -identifies all languages in the class \mathcal{L} .

(c) $\mathbf{TxtEx}_{\mathcal{H}} = \{\mathcal{L} : (\exists \text{ learner } M)[M \text{ **TxtEx**}_{\mathcal{H}}\text{-identifies } \mathcal{L}]\}$.

(d) \mathcal{L} is **TxtEx**-learnable (written $\mathcal{L} \in \mathbf{TxtEx}$) if some learner **TxtEx** $_{\mathcal{V}}$ -learns \mathcal{L} using some hypothesis space $\mathcal{V} = (V_i)_{i \in \mathbb{N}}$.

One can similarly define **InfEx**-identification, where one replaces texts in the above definition by informants.

As the learner has seen only finitely many inputs before it converges to its final hypothesis, some form of learning must have taken place. We use the terms identify, learn, infer as synonyms for this reason.

Let $\min(S)$ denote the minimum of the set S , where we take $\min(\emptyset) = \infty$. Let $\max(S)$ denote the maximum of the set S , where we take $\max(\emptyset) = 0$. Some examples of **TxtEx**-learnable classes (using a suitable hypothesis space) include the class of all finite sets: $\{D : D \text{ is finite}\}$, the class of all graphs of primitive recursive functions and the class of self-describing sets, $\{L : L \neq \emptyset \text{ and } W_{\min(L)} = L\}$. Some non-learnable classes are the class of all graphs of all computable functions, and the class of all finite sets plus the set \mathbb{N} (see [20]). The class of all finite sets plus the set \mathbb{N} becomes learnable under the criterion **InfEx**, however the class of all graphs of all computable functions remains unlearnable even under the criterion **InfEx** (see [20]).

Note that in the above model of learning, the learner need not know when it has converged to its final hypothesis. If one additionally requires this kind of ability from the learner, then the learning criterion is equivalent to finite learning, where the learner is allowed to make only one conjecture.

Definition 2. [20] Fix a hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$.

(a) A learner M **TxtFin** $_{\mathcal{H}}$ -identifies a language L (written: $L \in \mathbf{TxtFin}_{\mathcal{H}}(M)$) iff for all texts T for L , there exist i, n such that (i) $H_i = L$, (ii) $(\forall m < n)[M(T[m]) = ?]$, and (iii) $(\forall m \geq n)[M(T[m]) = i]$.

(b) A learner M **TxtFin** $_{\mathcal{H}}$ -identifies a class \mathcal{L} of languages (written: $\mathcal{L} \subseteq \mathbf{TxtFin}_{\mathcal{H}}(M)$) iff it **TxtFin** $_{\mathcal{H}}$ -identifies all languages in the class \mathcal{L} .

(c) $\mathbf{TxtFin}_{\mathcal{H}} = \{\mathcal{L} : (\exists \text{ learner } M)[M \text{ **TxtFin**}_{\mathcal{H}}\text{-identifies } \mathcal{L}]\}$.

(d) \mathcal{L} is **TxtFin**-learnable (written: $\mathcal{L} \in \mathbf{TxtFin}$) if some learner **TxtFin** $_{\mathcal{V}}$ -learns \mathcal{L} using some hypothesis space $\mathcal{V} = (V_i)_{i \in \mathbb{N}}$.

One can similarly define **InfFin**-identification. Let $E = \{2x : x \in \mathbb{N}\}$. The class, $\{L : \text{card}(L \cap E) = 1 \text{ and } [2x \in L \Rightarrow W_x = L]\}$ is **TxtFin**-learnable using a suitable hypothesis space. However the class of all finite sets is not **TxtFin**-learnable using any hypothesis space.

Since Gold [20], various other criteria of learning have been explored in the literature, especially those which require some additional properties on the conjectures of the learner. We will consider some of these in Section 2. We refer the reader to the textbook [22] and the papers [3, 7, 13, 48, 50, 53] for some literature on the topic.

We now define some of the possible hypothesis spaces that we will be considering. A *programming system (numbering)* for recursively enumerable languages is a recursively enumerable sequence $\mathcal{V} = (V_i)_{i \in \mathbb{N}}$ of recursively enumerable sets (that is, $\{\langle i, x \rangle : x \in V_i\}$ is recursively enumerable). Here, i is also called a grammar or index for V_i (in \mathcal{V} programming system). Thus, a hypothesis space is a programming system in the above sense. An *universal programming system (or universal numbering)* is a programming system $(V_i)_{i \in \mathbb{N}}$ such that, $\{V_i : i \in \mathbb{N}\}$ contains every recursively enumerable set. A universal programming system $(V_i)_{i \in \mathbb{N}}$ is called a *Friedberg programming system* (Friedberg numbering) (see [17]) if $V_i \neq V_j$ for $i \neq j$. A programming system $(V_i)_{i \in \mathbb{N}}$ is called an *acceptable programming system* (acceptable numbering), if for any programming system $(U_i)_{i \in \mathbb{N}}$, there exists a computable function r such that, for all i , $V_{r(i)} = U_i$. Thus, acceptable programming systems are maximal in the sense that grammars from other programming systems can be algorithmically converted to grammars in an acceptable programming system. Most common programming languages are acceptable programming systems. We fix a standard acceptable programming system W_0, W_1, \dots for the rest of the paper. A programming system $(V_i)_{i \in \mathbb{N}}$ is said to be *K-acceptable* [11, 29] iff for any programming system $(U_i)_{i \in \mathbb{N}}$, there exists a limiting computable function f such that for all i , $V_{f(i)} = U_i$. Here a function f is said to be limiting computable iff there exists a computable function g of two arguments such that, for all i , $f(i) = \lim_{t \rightarrow \infty} g(i, t)$. A universal programming system $(V_i)_{i \in \mathbb{N}}$, is called a **Ke-programming system** [24] if the set $\{\langle i, j \rangle : V_i = V_j\}$ is limiting decidable, that is, if the grammar equivalence problem in $(V_i)_{i \in \mathbb{N}}$ is limiting decidable.

Note that the hypothesis space chosen for interpreting the conjectures of the learner may play a crucial role in whether the learner is successful in identifying the language. A commonly used hypothesis space is an acceptable programming system. Every class which is **TextEx**-identifiable using some hypothesis space is also **TextEx**-identifiable using every acceptable programming system as a hypothesis space; hence the set of **TextEx**-learnable classes does not depend on the exact acceptable programming system chosen. This is the main reason why researchers often consider some fixed acceptable programming system as a hypothesis space.

However, acceptable programming systems have their own problems, such as difficulty of checking semantic equivalence of hypotheses, determining membership and so on. Thus for certain applications, it is advantageous to consider special programming systems for which such questions might be easier to solve. For example in Friedberg programming systems [17], one can trivially check whether two hypotheses are equivalent.

On the other hand, when one considers non-acceptable programming systems such as Friedberg programming systems [17] as hypothesis spaces or requires some other properties about hypothesis spaces (such as membership question being algorithmically decidable, or the hypothesis space not being allowed to contain languages other than those in the class of languages being learnt), then it may affect the classes which are learnable. This paper surveys some of the recent results which show how learnability depends on the type of hypothesis

spaces allowed.

In Section 2 we define some commonly used criteria of learning. In Section 3 we consider the special case of learning indexed families, where often the hypothesis space allowed depends on the class of languages being learnt. In Section 4 we consider hypothesis spaces being restricted programming systems, such as Friedberg programming systems. In Section 5 we consider optimal hypothesis spaces in the sense that if learning a class is possible using some hypothesis space, then the class can be learnt using the given hypothesis space. In Section 6 we consider whether learning is at all possible if one requires learning in all (reasonable) possible hypothesis spaces. In Section 7 we briefly discuss how presence or absence of certain control structures in hypothesis spaces effects learnability of certain classes.

In the rest of the paper, for ease of notation, we will often omit the hypothesis space from the subscript of learning criterion, and it will be implicit (the allowed hypothesis space may be constrained in some cases due to conventions of the section). We will provide some sample proofs, and refer the reader to the original papers for the other proofs.

2. Some Further Criteria of Learning

Below we consider the criteria mainly for learning from texts. Similar definitions can be made for learning from informants also. Below, let $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ be the hypothesis space used by the learner.

We first consider two generalizations of explanatory learning. The following generalization considers semantic convergence rather than syntactic convergence to the correct hypothesis by the learner. A learner M is said to *behaviourally correctly* learn (abbreviated: **TxtBc**-learn) [6, 12, 39] a language L iff for all texts T for L , for all but finitely many n , $H_{M(T[n])} = L$. One can similarly define **TxtBc**-learning of a class, and the set **TxtBc** of all behaviourally correctly learnable classes. It can be shown that **TxtBc** is a strict generalization of **TxtEx**-learning if one allows arbitrary hypothesis spaces [6, 12, 13, 39].

The following criterion is somewhere between explanatory and behaviourally correct learning. It allows the learner to eventually vacillate between finitely many correct hypotheses. A learner M is said to *vacillatorily* learn (abbreviated: **TxtFex**-learn; *Fex* stands for finite explanatory learning) [10] a language L iff it **TxtBc**-learns the language L and on all texts T for L , it outputs at most finitely many distinct grammars (in other words, the learner eventually vacillates between finitely many correct grammars for the language). One can similarly define **TxtFex**-learning of a class, and the set **TxtFex** of all vacillatorily learnable classes.

We now consider some natural requirements on the learner and its hypotheses. A learner M is said to be *conservative* [2] on L if for all texts T for L , for all $n > m$, if $M(T[n]) \neq M(T[m])$, then $\text{content}(T[n]) \not\subseteq H_{M(T[m])}$. That is, M changes its hypothesis only if it finds evidence of inconsistency of its earlier conjecture. Learner M conservatively learns (**Conserv**-identifies) L if

it **TextEx**-identifies L and is conservative on L . **Conserv**-identification of a class of languages and the class **Conserv** can be defined similarly. When using acceptable programming systems as hypothesis spaces, requiring learners to be conservative is a restriction on the learning capabilities of the machines [2].

A learner M is *consistent* [1, 5, 7] on L if for all texts T for L , for all n , $\text{content}(T[n]) \subseteq H_{M(T[n])}$. Consistency seems like a natural requirement, as if the hypothesis is not consistent, then it is obviously wrong. However, when using general hypothesis spaces such as acceptable programming systems, it can be shown that requiring consistency restricts learning capabilities of the machines [5, 7]. A learner M is *confident* [38] if it converges on every text, even if the text is for a language outside the class of languages being learnt. Confidence is restrictive: it can be shown that even simple classes, such as the class of all finite languages, cannot be learnt confidently. One can define the corresponding learning criteria for learners satisfying consistency and confidence properties (for **I**-learning) similarly. These criteria are called respectively **ConsI** and **ConfI**.

In AI, often one requires that the conjectures grow monotonically. For example, one often requires that more data imply the “set of truths” derivable from the hypothesis/axioms formed should increase. In inductive inference three forms of monotonicity have been studied. A learner M is said to be *strongly monotonic* [28] on L if for all texts T for L , for all $n > m$, $H_{M(T[m])} \subseteq H_{M(T[n])}$. Wiehagen considered a relaxation of the strong monotonicity requirement, by requiring monotonicity of conjectures only within the language being learnt. A learner M is said to be *monotonic* [49] on L if for all texts T for L , for all $n > m$, $H_{M(T[m])} \cap L \subseteq H_{M(T[n])} \cap L$. Finally, in the third version one requires monotonicity only as long as the hypothesis is consistent with the data seen so far. A learner M is said to be *weakly monotonic* [28] on L if for all texts T for L , for all $n > m$, if $\text{content}(T[n]) \subseteq H_{M(T[m])}$, then $H_{M(T[m])} \subseteq H_{M(T[n])}$ (that is the learner behaves strongly monotonically as long as the input data does not contradict the hypothesis conjectured). The criteria of learning corresponding to the above properties being satisfied by the learner (on texts for the language being learnt), in addition to **TextEx**-learning the target language, are respectively called **SMon**, **Mon** and **WMon**. The class of all finite sets can be learnt strongly monotonically. The class $\mathcal{L} = \{L_0, L_1, \dots\}$, where $L_0 = \{2x : x \in \mathbb{N}\}$, and $L_{n+1} = \{2x : x \leq n\} \cup \{2n+1\}$ is monotonically learnable but not strongly monotonically learnable. The class of cosingletons, $\{L : \text{card}(\mathbb{N} - L) = 1\}$, is weakly monotonically but not monotonically learnable.

Let \bar{L} denote $\mathbb{N} - L$, the complement of L . [34] considered the dual of above monotonic requirements, where for *dual strongly monotonic* learning of L by M one requires that for all texts T for L , for all $n > m$, $\overline{H_{M(T[m])}} \subseteq \overline{H_{M(T[n])}}$. Similarly, for *dual monotonic* learning of L one requires that for all texts T for L , for all $n > m$, $\overline{H_{M(T[m])}} \cap \bar{L} \subseteq \overline{H_{M(T[n])}} \cap \bar{L}$, and for *dual weakly monotonic* learning of L one requires that for all texts T for L , for all $n > m$, if $\text{content}(T[n]) \subseteq H_{M(T[m])}$, then $\overline{H_{M(T[m])}} \subseteq \overline{H_{M(T[n])}}$. The criteria of learning corresponding to the above properties being satisfied by the learner, in addition to **TextEx**-learning the target language, are respectively called **DSMon**, **DMon**

and **DWMon**.

[34] explore the relationship between the above (dual) monotonic criteria of learning.

Here note that the restrictions considered above are *class* versions, that is, the restrictions (such as consistency, monotonicity etc) considered are required by the learner only for the texts of languages in the class to be learnt — the learner need not satisfy the restrictions for texts of languages outside the class to be learnt.

We next consider two restrictions on how data is used by the learner. A learner M is *set-driven* [37, 46] if $\text{content}(\sigma) = \text{content}(\tau)$ implies $M(\sigma) = M(\tau)$. That is the output of the learner depends only on the content of the input, and not on its length or order. When using acceptable programming systems as hypothesis spaces, it can be shown that set drivenness restricts the learning capabilities of machines [44]. A learner M is *rearrangement-independent* [7, 19, 44] if $\text{content}(\sigma) = \text{content}(\tau)$ and $\text{length}(\sigma) = \text{length}(\tau)$ implies $M(\sigma) = M(\tau)$. That is the output of the learner depends only on the content and length of the input, and not on the order of the elements in it. Unlike most other requirements considered, rearrangement independence is not restrictive for explanatory learning [19, 44], when one considers acceptable programming systems as hypothesis spaces. One can define the corresponding learning criteria for learners satisfying set drivenness and rearrangement independence (for **I**-learning) similarly. These criteria are called *s-I* and *r-I*.

For any of the learning criteria **I** discussed in this paper, one can give a recursive enumeration M_0, M_1, \dots of learning machines such that, if $\mathcal{L} \in \mathbf{I}$, then there exists an i such that M_i witnesses that $\mathcal{L} \in \mathbf{I}$.

3. Learning Indexed Families

Angluin [2] considered learnability of indexed families of recursive languages. A class \mathcal{L} of languages consisting of languages L_0, L_1, \dots (with the corresponding indexing) is said to be an indexed family iff there exists a computable function f such that $f(i, x) = 1$ iff $x \in L_i$. Many of the commonly studied classes of languages, such as the class of regular languages or context-free languages, are indexed families.

For learning indexed families, the hypothesis space is usually considered to be an indexed family also. Additionally, one often considers the following requirements on the hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ (see [29, 31]):

- (a) the hypothesis space is the class being learnt (with the corresponding indexing) itself; this is called *exact* learning;
- (b) the hypothesis space is *class-preserving*, that is $\{H_0, H_1, \dots\} = \{L_0, L_1, \dots\}$; this is called class-preserving learning;
- (c) the hypothesis space is *class-comprising*, that is $\{H_0, H_1, \dots\} \supseteq \{L_0, L_1, \dots\}$; this is called class-comprising learning.

Note that in (b) and (c), there are several possible hypothesis spaces that might be used — if learning can be successfully done using at least one such hypothesis space, then one considers the class to be learnable according to the corresponding criterion.

We prefix E , ϵ or C (where ϵ denotes empty string) to the names of the criteria of learning to denote whether we are considering exact, class-preserving or class-comprising learning. This convention on criteria names is for this section only.

Lange and Zeugmann [29, 31] showed that $\mathbf{ETxtEx} = \mathbf{TxtEx} = \mathbf{CTxtEx}$ and $\mathbf{ETxtFin} = \mathbf{TxtFin} = \mathbf{CTxtFin}$. Thus, for explanatory and finite learning, choosing an appropriate hypothesis space (in the sense of exact, class-preserving or class-comprising) is not so crucial.

However, for monotonic learning, the choice of different kind of hypothesis spaces makes a critical difference. As the following theorem shows, for all (dual) monotonic criteria of learning, except for strong dual monotonic learning, we get a strict hierarchy from exact, to class-preserving to class-comprising learning. For strong dual monotonic learning, exact and class-preserving learning are the same (and equal to finite learning). Class-comprising dual strong monotonic learning is strictly more general than class-preserving dual strong monotonic learning.

Theorem 3. (a) [29] $\mathbf{ESMon} \subset \mathbf{SMon} \subset \mathbf{CSMon}$.

(b) [29] $\mathbf{EWMon} \subset \mathbf{WMon} \subset \mathbf{CWMon}$.

(c) [29] $\mathbf{EDWMon} \subset \mathbf{DWMon} \subset \mathbf{CDWMon}$.

(d) [29] $\mathbf{EDSMon} = \mathbf{DSMon} \subset \mathbf{CDSMon}$.

(e) [30, 34] $\mathbf{EMon} \subset \mathbf{Mon} \subset \mathbf{CMon}$.

(f) $\mathbf{EDMon} \subset \mathbf{DMon} \subset \mathbf{CDMon}$.

We refer the reader to [29, 34] for the proof of parts (a) to (e) above. As an idea of the techniques involved, we give the classes witnessing the separations in part (a) above.

Let $\varphi_0, \varphi_1, \dots$ denote a fixed acceptable programming system for partial recursive functions [41]. Let Φ denote a Blum complexity measure for the φ system [7].

Let

$$A_{i,j} = \begin{cases} \{\langle i, x \rangle : x \in \mathbb{N}\}, & \text{if } j < \Phi_i(i) \\ \{\langle i, x \rangle : x \leq \Phi_i(i)\}, & \text{otherwise.} \end{cases}$$

Let $\mathcal{A} = \{A_{i,j} : i, j \in \mathbb{N}\}$. Then \mathcal{A} is in $\mathbf{SMon} - \mathbf{EMon}$.

Let

$$B_{i,j} = \begin{cases} \{\langle i, x \rangle : x \in \mathbb{N}\}, & \text{if } j < \Phi_i(i) \\ \{\langle i, x \rangle : x \leq \Phi_i(i) \text{ or } x > j\}, & \text{otherwise.} \end{cases}$$

Let $\mathcal{B} = \{B_{i,j} : i, j \in \mathbb{N}\}$. Then, \mathcal{B} is in $\mathbf{CSMon} - \mathbf{SMon}$.

The proof of Theorem 4 in [30] shows that $\mathbf{EDMon} \subset \mathbf{DMon}$. We do not know if anyone has explicitly shown that $\mathbf{DMon} \subset \mathbf{CDMon}$, but it can be shown as follows. As every hypothesis space can be translated algorithmically

to any acceptable programming system, for ease of presentation, in the following we consider the hypothesis space of the learner (for the diagonalization against **DMon**-learnability) to be an acceptable programming system, where the learner only conjectures grammars for the languages in the class being learnt. One can extend the pairing function to coding of multiple arguments by taking $\langle x_1, x_2, \dots, x_n \rangle = \langle x_1, \langle x_2, \dots, x_n \rangle \rangle$.

Let $L_i = \{\langle i, 0, x \rangle : x \in \mathbb{N}\}$, $L_{i,j} = \{\langle i, 0, x \rangle : x \leq j\}$, $X_{i,j,k} = L_{i,j} \cup \{\langle i, 1, x \rangle : x \geq k\}$, and $X_{i,j,k,k'} = L_{i,j} \cup \{\langle i, 1, x \rangle : k \leq x \leq k'\} \cup \{\langle i, 2, k \rangle\}$.

Let M_0, M_1, \dots denote a recursive enumeration of all learning machines. Let T_i be the canonical text for L_i given by $T_i(j) = \langle i, 0, j \rangle$. Let $s_i > 0$ denote the first s found, if any in some standard algorithmic search, such that $\text{content}(T_i[s]) \cup \{\langle i, 0, s \rangle\} \subseteq W_{M_i(T_i[s])}$. Note that $\langle i, 0, s \rangle \notin \text{content}(T_i[s])$. Let r_i denote the time needed to find s_i , if defined (where we assume without loss of generality that $r_i \geq s_i$).

Let $\mathcal{L} = \{L_i : i \in \mathbb{N} \text{ and for all } s, \text{content}(T_i[s]) \cup \{\langle i, 0, s \rangle\} \not\subseteq W_{M_i(T_i[s])}\} \cup \{L_{i,r_i} : i \in \mathbb{N} \text{ and } s_i \text{ is defined}\} \cup \{X_{i,s_i-1,r_i} : i \in \mathbb{N} \text{ and } s_i \text{ is defined}\} \cup \{X_{i,s_i-1,r_i,k'} : i \in \mathbb{N} \text{ and } s_i \text{ is defined, } k' \geq r_i\}$.

It is easy to verify that \mathcal{L} is an indexed family. Note that if s_i is defined then L_{i,r_i} is the only language in \mathcal{L} which contains $\langle i, s_i \rangle$.

We first show that \mathcal{L} cannot be **DMon**-identified. If s_i is not defined, then M_i does not **TextEx**-identify $L_i \in \mathcal{L}$. If s_i is defined then M_i on input $T_i[s_i]$ outputs a conjecture which contains $\langle i, s_i \rangle$, and thus this conjecture must be for L_{i,r_i} . Thus, M_i on $T_i[s_i]$ has already conjectured a language which omitted every element in $\{\langle i, 1, x \rangle : x \in \mathbb{N}\}$. Now consider a text T for X_{i,s_i-1,r_i} which extends $T_i[s_i]$. Let $t > s_i$ be such that $M_i(T[t])$ is a grammar for X_{i,s_i-1,r_i} . Let k' be such that $k' \geq \max(\{r_i\} \cup \{k : \langle i, 1, k \rangle \in \text{content}(T_i[t])\})$. Then, let T' , extending $T[t]$, be a text for $X_{i,s_i-1,r_i,k'}$. Then, M_i converges on T' to a grammar for $X_{i,s_i-1,r_i,k'}$. But then M_i is not dual monotonic on T' , as it changes its conjecture from L_{i,r_i} to X_{i,s_i-1,r_i} and then to $X_{i,s_i-1,r_i,k'}$, but L_{i,r_i} and $X_{i,s_i-1,r_i,k'}$ do not contain $\langle i, 1, k' + 1 \rangle$, whereas X_{i,s_i-1,r_i} does.

On the other hand, with class-comprising hypothesis space, the following learner M can dual monotonically learn \mathcal{L} .

If $\text{content}(\sigma) = \emptyset$, then $M(\sigma) = ?$. Otherwise, let i be such that $\text{content}(\sigma) \subseteq \langle i, \cdot, \cdot \rangle$. If the learner cannot verify within $\text{length}(\sigma)$ steps that s_i is defined, then $M(\sigma)$ is a (canonical) grammar for $L_i \cup (X_{i,s_i-1,r_i} \cup \{\langle i, 2, r_i \rangle\})$ (where $(X_{i,s_i-1,r_i} \cup \{\langle i, 2, r_i \rangle\})$ is taken to be empty set, if s_i does not get defined).

If the learner can verify within $\text{length}(\sigma)$ steps that s_i is defined then, if $\langle i, 0, s_i \rangle \in \text{content}(\sigma)$, then $M(\sigma)$ is a (canonical) grammar for L_{i,r_i} ; otherwise, if $\langle i, 1, r_i \rangle \in \text{content}(\sigma)$, but $\langle i, 2, r_i \rangle \notin \text{content}(\sigma)$, then $M(\sigma)$ is a (canonical) grammar for L_{i,s_i-1,r_i} ; otherwise if $\{\langle i, 1, r_i \rangle, \langle i, 2, r_i \rangle\} \subseteq \text{content}(\sigma)$, then $M(\sigma)$ is a (canonical) grammar for $L_{i,s_i-1,r_i,k'}$, where k' is the largest number such that $\langle i, 1, k' \rangle \in \text{content}(\sigma)$; otherwise, (that is neither $\langle i, 1, r_i \rangle$ nor $\langle i, 2, r_i \rangle$ is in $\text{content}(\sigma)$) the learner repeats its previous hypothesis.

It is easy to verify that M would **CDMon**-identify \mathcal{L} . This completes the proof for **DMon** \subset **CDMon**.

[52] gave some interesting characterization of classes which are (strong, weak) monotonically learnable in dependence of hypothesis space used.

Even though **TxtEx** does not depend on whether one uses class-preserving, exact or class-comprising hypothesis space, if one considers restricting the number of mind changes to a non-zero value, then learnability does depend on what kind of hypothesis space one chooses. Let \mathbf{TxtEx}_m (see [13]) denote the criterion of learning where the learner is allowed at most m mind changes (here a change from ? to a proper conjecture (member of \mathbb{N}) is not counted as a mind change).

Theorem 4. *Suppose $m \geq 1$.*

- (a) [31] $\mathbf{ETxtEx}_m \subset \mathbf{TxtEx}_m$.
- (b) $\mathbf{TxtEx}_m \subset \mathbf{CTxtEx}_m$.

We do not know if anyone has explicitly shown that $\mathbf{TxtEx}_m \subset \mathbf{CTxtEx}_m$, but it can be shown as follows.

Consider the class $\mathcal{L} = \{D : \text{card}(D) = 2 \text{ or } (\text{card}(D) = 1 \text{ and } D \subseteq K')\}$, where K is the halting problem and K' is the halting problem relative to K . Note that \mathcal{L} is an indexed family. To see this, first note that there is a two-place computable function g with $x \in K'$ iff $g(x, y) = 1$ for almost all y and $x \notin K'$ iff $g(x, y) = 0$ for infinitely many y . Now let

$$L_{2\langle x, y \rangle} = \{x, x + y + 1\} \text{ and}$$

$$L_{2\langle x, y \rangle + 1} = \begin{cases} \{x, x + z + 1\}, & \text{if } z \text{ is the least number with} \\ & z > y \text{ and } g(x, z) \neq 1; \\ \{x\}, & \text{if } g(x, z) = 1 \text{ for all } z > y. \end{cases}$$

It is easy to verify that $\{L_0, L_1, \dots\} = \mathcal{L}$, and witnesses that \mathcal{L} is an indexed family.

Also, \mathcal{L} can easily be \mathbf{TxtEx}_1 -learnt using a hypothesis space consisting of all the sets with cardinality 1 or 2.

Now suppose by way of contradiction that some learner M \mathbf{TxtEx}_m -identifies \mathcal{L} using a class preserving hypothesis space \mathcal{H} . Then one can define the K -recursive function f with $f(x)$ being the hypothesis to which M converges on the text x^∞ (for the purposes of the proof, if M converges on x^∞ to ?, then we take $M(x^\infty) = 0$). If $x \in K'$ then $H_{f(x)} = \{x\}$ as M learns this set. If $x \notin K'$ then $H_{f(x)} \neq \{x\}$ as no hypothesis in \mathcal{H} equals $\{x\}$. The test whether $H_{f(x)} = \{x\}$ is also K -recursive. This would give a contradiction to $K' \not\leq_T K$. Thus there is no class-preserving \mathbf{TxtEx}_m -learner for \mathcal{L} .

[34] also studied how the structure of relationship between various versions of (dual) monotonicity changes if one considers class-comprising hypothesis space as opposed to class-preserving/exact hypothesis spaces. For example, $\mathbf{CTxtFin} \subset \mathbf{CDMon}$, though $\mathbf{TxtFin} = \mathbf{DMon}$. Similarly, $\mathbf{CDWMon} = \mathbf{CTxtEx}$, though $\mathbf{DWMon} \subset \mathbf{TxtEx}$. Furthermore, $\mathbf{DSMon} \subset \mathbf{SMon}$, though \mathbf{CDSMon} and \mathbf{CSMon} are incomparable. Thus, not only do the classes learnable (under a learning criterion) depend on the kind of hypothesis spaces

that are allowed, but even the relationship among the learning criteria depend on what kind of hypothesis spaces are allowed.

If one considers iterative learning (where the learner's hypotheses depend only on its last hypothesis and current datum, rather than all the data it has seen so far) [32, 47], then [32] showed that for certain classes and particular class-comprising hypothesis spaces iterative learning may outperform conservative learning, though in general iterative learning is contained in conservative learning (when arbitrary class-comprising hypothesis spaces are allowed).

In [33] the authors study set driven and rearrangement independent learning in dependence of hypothesis space for indexed families (where hypothesis spaces are indexed families too). They showed that for set driven and rearrangement independent learning, the classes that can be **TxtFin**-learnt do not depend on the type of hypothesis spaces allowed (among the types, exact, class-preserving and class-comprising).

Theorem 5. [33] $r\text{-ETxtFin} = s\text{-ETxtFin} = \text{ETxtFin} = \text{TtxtFin} = \text{CTxtFin}$.

For explanatory learning, set driven learning forms a hierarchy depending on the type of hypothesis space allowed, whereas for rearrangement independent learning, it does not depend on the type of hypothesis space allowed.

Theorem 6. [33]

- (a) $s\text{-ETxtEx} \subset s\text{-TtxtEx} \subset s\text{-CTxtEx} \subset \text{ETxtEx} = \text{TtxtEx} = \text{CTxtEx}$.
- (b) $r\text{-ETxtEx} = r\text{-TtxtEx} = r\text{-CTxtEx} = \text{ETxtEx} = \text{TtxtEx} = \text{CTxtEx}$.

For monotonic learning (all three types) we get a proper hierarchy for both set driven as well as rearrangement independent learning.

Theorem 7. [33]

- (a) $s\text{-ESMon} \subset s\text{-SMon} \subset s\text{-CSMon}$.
- (b) $s\text{-EMon} \subset s\text{-Mon} \subset s\text{-CMon}$.
- (c) $s\text{-EWMon} \subset s\text{-WMon} \subset s\text{-CWMon}$.
- (d) $r\text{-ESMon} \subset r\text{-SMon} \subset r\text{-CSMon}$.
- (e) $r\text{-EMon} \subset r\text{-Mon} \subset r\text{-CMon}$.
- (f) $r\text{-EWMon} \subset r\text{-WMon} \subset r\text{-CWMon}$.

We refer the reader to [51] for several other results and characterizations for learning indexed families in dependence on hypothesis spaces.

[35] considered the situation where the learner may make queries regarding certain kind of relationship between a potential hypothesis and the input language. The queries allowed are subset, superset or disjointness queries. The learner, after making a finite number of such queries, outputs a single hypothesis which must be correct for languages in the class being learnt. They showed that the learnability of a class depends very much on whether the hypothesis space (query space) chosen is an indexed family, recursively enumerable (r.e.) family or a limiting r.e. family. [21] extended above work to learning r.e. classes of r.e. languages.

4. Special Hypothesis Spaces

In this section we revert back to learning recursively enumerable languages using some fixed hypothesis spaces. Criteria **I** (such as **TxtEx**, **TxtBc**, **TxtFin**, or **TxtFex**) without a specified hypothesis space refers to using acceptable programming system as a hypothesis space.

Freivalds, Kinber and Wiehagen [15] considered learning of functions using Friedberg programming systems as hypothesis spaces. Later Jain and Stephan [24] considered learning using Friedberg programming systems or **Ke**-programming systems as hypothesis spaces. For a criterion **I** of learning, let **FrI** (**KeI**) denote the class of languages which can be learnt under the criterion **I** using some Friedberg programming system (some **Ke**-programming system) as a hypothesis space. [11, 24] showed that every **TxtEx**-learnable class can be learnt using some Friedberg programming system as a hypothesis space. However, no single Friedberg programming system is enough to be used as a hypothesis space for all **TxtEx**-learnable classes. On the other hand, for **TxtFin**-learning, there are classes of languages which can be **TxtFin**-learnt (using acceptable programming system as a hypothesis spaces), but which cannot be learnt using any Friedberg programming system as a hypothesis space. In contrast, every **TxtFin**-learnable class can be learnt using some **Ke**-programming system as a hypothesis space.

Theorem 8. (a) [11, 24] $\mathbf{FrTtxtEx} = \mathbf{KeTtxtEx} = \mathbf{TtxtEx}$.
 (b) [24] $\mathbf{FrTtxtFin} \subset \mathbf{TtxtFin}$.
 (c) [24] $\mathbf{KeTtxtFin} = \mathbf{TtxtFin}$.

Proof. We give the proof for part (b) from [24], and refer the reader to the above paper for the proof of parts (a) and (c).

Let $\mathcal{L} = \{L : (\forall x \in L)[W_x = L]\}$. Clearly, $\mathcal{L} \in \mathbf{TtxtFin}$. Suppose by way of contradiction that some learner M **TtxtFin**-identifies \mathcal{L} using a Friedberg programming system \mathcal{H} as a hypothesis space. Without loss of generality assume that M does not output more than one distinct conjecture on any text. Then, by Smullyan's double recursion theorem [41], there exist distinct e_1, e_2 such that W_{e_1}, W_{e_2} may be defined as follows.

Let $W_{e_1} = \{e_1, e_2\}$ and $W_{e_2} = \{e_1, e_2\}$, if there exist finite sequences $\tau_1, \tau_2 \in SEQ$ such that $\text{content}(\tau_i) \subseteq \{e_i\}$ for $i \in \{1, 2\}$, $M(\tau_1) \downarrow \neq ?$, $M(\tau_2) \downarrow \neq ?$ and $M(\tau_1) \downarrow \neq M(\tau_2) \downarrow$; otherwise, let $W_{e_1} = \{e_1\}$ and $W_{e_2} = \{e_2\}$. It is easy to verify that both W_{e_1} and W_{e_2} are members of \mathcal{L} . Now suppose, for some p , M outputs either $?$ or p , on all sequences $\tau_1, \tau_2 \in SEQ$ such that $\text{content}(\tau_1) \subseteq \{e_1\}$ and $\text{content}(\tau_2) \subseteq \{e_2\}$. Then clearly $W_{e_1} \neq W_{e_2}$ and thus M does not **TtxtFin** $_{\mathcal{H}}$ -identify \mathcal{L} — as M outputs only p or $?$ on the texts for both W_{e_1} and W_{e_2} . On the other hand, suppose there exist τ_1, τ_2 such that $\text{content}(\tau_1) \subseteq \{e_1\}$, $\text{content}(\tau_2) \subseteq \{e_2\}$, $M(\tau_1) \downarrow \neq ?$, $M(\tau_2) \downarrow \neq ?$ and $M(\tau_1) \downarrow \neq M(\tau_2) \downarrow$. Then $W_{e_1} = W_{e_2}$ and M does not **TtxtFin** $_{\mathcal{H}}$ -identify \mathcal{L} — as M outputs at least two distinct conjectures on (different) texts for W_{e_1} , but at most one of them can be a grammar for W_{e_1} (as \mathcal{H} is a Friedberg programming system). Thus, in either

case, M does not **TxtFin** $_{\mathcal{H}}$ -identify \mathcal{L} . This completes the proof for part (b). \square

Even though every **TxtEx**-learnable class can be learnt using some Friedberg programming system as a hypothesis space, these hypothesis spaces very much depend on the class being learnt — the classes $\mathcal{L}_1 = \{L : L \neq \emptyset \text{ and } W_{\min(L)} = L\}$ and $\mathcal{L}_2 = \{L : \text{card}(L) \geq 2 \text{ and } W_{\min(L - \{\min(L)\})} = L\}$ can be **FrTtxtEx**-learnt, but cannot be **FrTtxtEx**-learnt using the *same* Friedberg programming system as a hypothesis space! (We refer the reader to [24] for a proof).

An interesting result shown by [24] is that a recursively enumerable class can be **TxtFin**-learnt using some Friedberg programming system as a hypothesis space iff it is 1–1 recursively enumerable and **TxtFin**-learnable.

The situation changes for vacillatory and behaviourally correct learning. For vacillatory learning, there are vacillatorily learnable classes which cannot be vacillatorily learnt in any **Ke**-programming system. In particular, every class which can be vacillatorily learnt in some **Ke**-programming system is explanatorily learnable!

Theorem 9. [24] $\text{FrTtxtFex} = \text{KeTtxtFex} = \text{TtxtEx} \subset \text{TtxtFex}$.

Here $\text{TtxtEx} \subset \text{TtxtFex}$ was shown by [10].

Similarly, there exist behaviourally correctly learnable classes which cannot be behaviourally correctly learnt in any Friedberg programming system.

Theorem 10. [24] $\text{FrTtxtBc} \subset \text{KeTtxtBc}$.

It is open at this point whether every behaviourally correctly learnable class is behaviourally correctly learnable in some **Ke**-programming system.

Even though every **TxtEx**-learnable class is learnable using some Friedberg programming system, the learner may not satisfy some desirable properties. For example, consider prudent learning. Prudent learning requires that a learner only outputs hypotheses describing languages it is able to learn [38]. Every **TtxtEx**-learnable class can also be learnt prudently [19]. However, even simple classes, such as the class of all finite sets, cannot be prudently learnt using any Friedberg programming system as a hypothesis space. This is so, since otherwise, one would get a Σ_1 procedure for enumerating all infinite r.e. sets, a contradiction to a well-known result [41]. On the other hand, one can do prudent learning of every **TtxtEx**-learnable class using some **Ke**-programming system as a hypothesis space.

Similar results regarding learning the class of finite sets can also be shown for non-U-shaped learning, conservative learning and monotonic learning. Here a learner is non-U-shaped if it never abandons a correct hypothesis [4]. *U-shaped learning* is a learning behaviour in which the learner first learns the correct behaviour, then abandons the correct behaviour and finally returns to the correct behaviour once again. This pattern of learning behaviour has been observed by cognitive and developmental psychologists in a variety of child development phenomena, for example language learning [8, 36, 45]. Note that every **TtxtEx**-learnable class can also be learnt in a non-U-shaped way using some acceptable programming system as a hypothesis space [4].

In contrast to the above result, for consistent learning [1, 5] one can use some Friedberg programming system as a hypothesis space for every class of languages which can be consistently learnt using some hypothesis space.

Even though usage of Friedberg programming systems in general as a hypothesis space, for **TextEx**-learning, is quite powerful, there are some Friedberg programming systems which make learning almost impossible: only **TextEx**-learnable classes which contain finitely many infinite languages could be (explanatorily, behaviourally correctly, or vacillatorily) learnt using such Friedberg programming systems as a hypothesis space. Similarly, there exist Friedberg programming systems, using which as a hypothesis space, only inclusion free finite classes of languages can be **TextFin**-learnt (a class is inclusion free if no language in the class is included in another language in the class). We refer the reader to [24] for further results on learning using Friedberg programming systems or **Ke**-programming systems as hypothesis spaces.

5. Optimal Hypothesis Spaces

As we have seen, chosen hypothesis spaces play a crucial role in whether a learner is able to learn the target class of languages. Thus it is interesting to explore *optimal* hypothesis spaces \mathcal{H} in the sense that any class learnable using some hypothesis space is also learnable using the hypothesis space \mathcal{H} . This would allow one to use the same hypothesis space for learning various classes of languages. Optimality may (and does) of course depend on the learning criterion under investigation. Furthermore, we consider whether a hypothesis space being optimal for a particular learning criterion implies it being optimal for some other learning criterion. Such studies were done by [25].

For a criterion of learning **I**, a hypothesis space \mathcal{H} is said to be *optimal* if any class \mathcal{L} , **I**-learnable using some hypothesis space, is also **I**-learnable using \mathcal{H} as the hypothesis space. The hypothesis space \mathcal{H} is said to be *algorithmically optimal* for a criterion **I** if given any learner M using a hypothesis space \mathcal{H}' , one can algorithmically find a learner M' using \mathcal{H} as a hypothesis space (for the class of languages which was **I**-learnt by M using \mathcal{H}' as a hypothesis space).

Clearly, all acceptable programming systems are optimal for **TextEx**, **TextFin**, **TextBc**, **TextFex**.
. Are there other optimal programming systems?

Definition 11. [25] *A programming system A_0, A_1, A_2, \dots is called nearly acceptable iff there is a computable function f such that $A_{f(d,e)} = W_e$ whenever $d \in W_e$.*

The nearly acceptable programming systems are algorithmically optimal for explanatory, vacillatory and behaviourally correct learning. They are also optimal for **TextFin**-learning, but not necessarily algorithmically optimal for **TextFin**-learning. Note that one can easily construct nearly acceptable programming systems which are not acceptable.

The algorithmically optimal programming systems for finite, explanatory and vacillatory learning are easy to characterize.

Theorem 12. [25] A hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ of all r.e. sets is
 (a) algorithmically optimal for **TxtFin**-learning iff \mathcal{H} is acceptable;
 (b) algorithmically optimal for **TxtEx**-learning iff \mathcal{H} is K -acceptable;
 (c) algorithmically optimal for **TxtFex**-learning iff there is a limiting-computable function g such that, for all d , there is an $e \leq g(d)$ with $H_e = W_d$.

[11] also showed part (b) above.

The following theorem gives the relation between optimal programming systems for finite, explanatory, behaviourally correct and vacillatory learning.

Theorem 13. [25]

- (a) For each $\mathbf{I} \in \{\mathbf{TxtEx}, \mathbf{TxtFin}, \mathbf{TxtBc}, \mathbf{TxtFex}\}$, there are programming systems which are optimal but not algorithmically optimal for \mathbf{I} .
 (b) For any two distinct \mathbf{I} and \mathbf{J} in $\{\mathbf{TxtEx}, \mathbf{TxtFin}, \mathbf{TxtBc}, \mathbf{TxtFex}\}$, there is a programming system which is optimal for \mathbf{I} but not optimal for \mathbf{J} .

In (b) above, if $\mathbf{I} \neq \mathbf{TxtFin}$ and ($\mathbf{I} \neq \mathbf{TxtEx}$ or $\mathbf{J} \neq \mathbf{TxtFex}$), then we can even take the corresponding programming system to be *algorithmically* optimal for \mathbf{I} .

[11] had also shown that there are programming systems which are optimal but not algorithmically optimal for **TxtEx** and that there are K -acceptable, but not acceptable, programming systems which are optimal for **TxtFin**.

Another interesting result is that every (algorithmically) optimal programming system for **TxtEx** is also (algorithmically) optimal for consistent learning. On the other hand, there are programming systems which are algorithmically optimal for consistent learning but not optimal for finite, explanatory, vacillatory or behaviourally correct learning.

In learning with additional information, in addition to a text for the language, a learner is also provided with an upper bound on a (code for) grammar (in the hypothesis space) for the target language [16, 23]. It was shown in [25] that the **Ke**-programming systems are exactly those hypothesis spaces which are optimal for learning with additional information.

6. Prescribed Learning

Until now we have been mostly concentrating on learning using some *suitable* hypothesis space, perhaps with some constraints such as being class-preserving, class-comprising or being a Friedberg programming system. What if one requires that learning has to happen using *every* suitable hypothesis space? This kind of situation is useful if one expects that the seller provides a learner which works based on the programming system used by any potential buyer, rather than only with the programming system used by the seller. Here one may distinguish between two cases, one where there exists a learner for each of the suitable hypothesis spaces and another one where one expects the *same* learner (with hypothesis space being a parameter) to work for *all* hypothesis spaces. The issue here is of being able to algorithmically generate a learner given a description

of the suitable hypothesis space. Jain, Stephan and Ye [26, 27] considered the above situation.

We say that a class \mathcal{L} is *prescribed I-learnable*, if for every hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$, such that $\{H_i : i \in \mathbb{N}\} \supseteq \mathcal{L}$, \mathcal{L} can be learnt according to the criterion **I** using \mathcal{H} as a hypothesis space.

We say that a class \mathcal{L} is *class-preserving-prescribed I-learnable*, if for every class-preserving hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ (that is hypothesis space satisfying $\{H_i : i \in \mathbb{N}\} = \mathcal{L}$), \mathcal{L} can be learnt according to criterion **I** using \mathcal{H} as a hypothesis space.

We say that \mathcal{L} is *uniformly I-learnable*, if there exists an algorithmic listing M_0, M_1, M_2, \dots of learners such that given a program i describing the hypothesis space $\mathcal{H} = (H_j)_{j \in \mathbb{N}}$ such that $\{H_j : j \in \mathbb{N}\} \supseteq \mathcal{L}$, \mathcal{L} can be learnt by M_i according to criterion **I** using \mathcal{H} as the hypothesis space. Here we say that the program i describes the hypothesis space $\mathcal{H} = (H_j)_{j \in \mathbb{N}}$ if $\varphi_i(j, x) = 1$ iff $x \in H_j$ (where, when considering indexed family as a hypothesis space, we require φ_i to be total). In above, φ_i denotes the function computed by the i -th program in some standard acceptable programming system for partial recursive functions.

One can define *uniformly class-preserving learning* similarly.

For general learnability of r.e. languages, where hypothesis spaces are r.e. classes (rather than indexed families) prescribed learning is quite weak as in some Friedberg programming systems only restricted classes can be learnt. Thus, for r.e. languages one normally considers class-preservingly-prescribed (uniformly class-preserving) learning only. Note that the concept classes being considered here would be r.e. classes of r.e. languages.

For finite and explanatory learning, uniform learning can very much be done.

Theorem 14. [27] *Every **TxtFin**-learnable r.e. class of languages is also uniformly class-preservingly **TxtFin**-learnable.*

Proof. We give a proof from [27].

Suppose M is a **TxtFin**-learner for $\mathcal{L} = \{L_0, L_1, \dots\}$ (using some hypothesis space). Let e be an index for a hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$, that is, $W_e = \{\langle j, x \rangle : x \in H_j\}$. Further suppose $\{H_i : i \in \mathbb{N}\} = \mathcal{L}$. We give the learner M_e which **TxtFin**-learns \mathcal{L} using the hypothesis space \mathcal{H} as follows. Let $H_{j,n} = \{x : x < n \text{ and } x \text{ is enumerated in } H_j \text{ within } < n \text{ steps}\}$. $M_e(T[n])$ is defined as follows.

- (1) If there exists an $m < n$ with $M_e(T[m]) \neq ?$, then $M_e(T[n]) = M_e(T[m])$ for the least such m ;
- (2) Else If there exist $m \leq n$ and $j \leq n$ with $M_e(T[m]) \neq ?$ and $\text{content}(T[m]) \subseteq H_{j,n}$, then $M_e(T[n]) = j$ for the least such j ;
- (3) Otherwise $M_e(T[n]) = ?$.

By step (1) above, M_e outputs at most one hypothesis (besides ?) on any text T . Also, it follows from the definition of **TxtFin**-learning that $H_j = H_{j'}$ whenever $M_e(T[m]) \neq ?$, $\text{content}(T[m]) \subseteq H_j$ and $\text{content}(T[m]) \subseteq H_{j'}$. Hence

the j chosen in step (2) is a correct hypothesis, whenever the condition in step (2) holds. Furthermore, for all texts for languages in \mathcal{L} , the condition in step (2) would eventually hold, as M **TextFin**-learns \mathcal{L} . Thus we have that \mathcal{L} is uniformly **TextFin**-learnable. \square

Theorem 15. [27] *Every **TextEx**-learnable r.e. class of languages is also uniformly class-preservingly **TextEx**-learnable.*

For confident learning, there are classes which are class-comprisingly confidently learnable but not class-preservingly confidently learnable. So we have a restricted version of the above theorems for confident learning.

Theorem 16. [27] *Every class-preservingly confidently **TextEx**-learnable r.e. class of languages is also uniformly class-preservingly confidently **TextEx**-learnable.*

[27] also consider behaviourally correct learning and vacillatory learning. Though these criteria are similar to explanatory learning (in semantic sense), it was shown that there are classes which are behaviourally correctly (vacillatorially) learnable using class-preserving hypothesis spaces but not class-preservingly-prescribed behaviourally correctly (vacillatorially) learnable. It is open at this point whether uniform and non-uniform prescribed version of class-preserving learning for behaviourally correct learning are same. Similar question for vacillatory learning is also open.

On the other hand, for conservative learning, prescribed and uniform learning are a restriction and are separated from each other.

Theorem 17. [27]

- (a) *The class $\{D : |D| \leq 1\}$ is class-preservingly-prescribed conservatively but not uniformly class-preservingly conservatively **TextEx**-learnable.*
- (b) *The class $\{D : |D| < \infty\}$ is class-preservingly conservatively but not class-preservingly-prescribed conservatively **TextEx**-learnable.*
- (c) *The class $\{D : |D| = 2 \vee (|D| = 1 \wedge D \subseteq K')\}$ is class-comprisingly conservatively but not class-preservingly conservatively **TextEx**-learnable.*

We now turn our attention to prescribed learning of indexed families. The rest of the section considers learning of indexed families only. Thus, as in Section 3, the hypothesis spaces are assumed to be indexed families.

For **TextFin**-learning, prescribed and uniform learning are very restricted. However, uniform class-preserving learning can be done for all **TextFin**-learnable indexed families.

Definition 18. *Define $S = \cup_{n=0,1,2,\dots} J_n$, where J_n contains for each $e < n$ the first element (found in some algorithmic search), if any, of W_e enumerated from $I_n = \{2^n - 1, 2^n, 2^n + 1, \dots, 2^{n+1} - 2\}$. Then: S is recursively enumerable; S intersects with every infinite recursively enumerable set; for every n there is an m in I_n which is not in S . In other words, S is a simple set [?]. Let S_t be the set of elements enumerated into S within t steps via some standard algorithmic procedure. Here we take $S_0 = \emptyset$.*

In the following, a canonical text T_i for a language L_i (from a class $\mathcal{L} = \{L_0, L_1, \dots\}$ of languages) is some fixed standard text for L_i which can be obtained algorithmically from i .

Theorem 19. [26] *Suppose \mathcal{L} is an indexed family.*

- (a) *If $\mathcal{L} \neq \emptyset$, then \mathcal{L} is not uniformly **TxtFin**-learnable.*
- (b) *\mathcal{L} is uniformly class-preservingly **TxtFin**-learnable iff \mathcal{L} is **TxtFin**-learnable.*
- (c) *\mathcal{L} is prescribed **TxtFin**-learnable iff \mathcal{L} is finite and inclusion free.*

Proof. We give a proof from [26].

Suppose $\mathcal{L} = \{L_0, L_1, \dots\}$, where one can algorithmically decide (in i) membership questions for L_i , and $L_i \neq L_j$ for $i \neq j$.

(a) Let $G_e = L_e$ if $e < |\mathcal{L}|$; otherwise let G_e be some recursive set outside \mathcal{L} . Note that the programming system G_0, G_1, \dots is introduced in order to handle finite and infinite classes uniformly (for infinite \mathcal{L} , note that $G_i = L_i$). Suppose \mathcal{L} is uniformly **TxtFin**-learnable as witnessed by the recursive enumeration of learners M_0, M_1, M_2, \dots . Let F be a recursive set such that no finite variant of F is in \mathcal{L} . By Kleene's recursion theorem [41], there exists an e such that for every $d \in \mathbb{N}$ and $c \in \{0, 1\}$,

$$\varphi_e(2d + c, x) = \begin{cases} F(x), & \text{if } M_e \text{ outputs } 2d + c \text{ as first grammar on the} \\ & \text{canonical text } T_d \text{ for } G_d \text{ within } x \text{ steps;} \\ G_d(x), & \text{otherwise.} \end{cases}$$

For this e , φ_e defines an indexed family hypothesis space \mathcal{H} which is a superclass of \mathcal{L} . By construction, M_e does not **TxtFin**-learn any language in \mathcal{L} using the given hypothesis space \mathcal{H} .

(b) Follows from Theorem 14.

(c) If $\mathcal{L} = \{L_0, \dots, L_n\}$ for some $n \in \mathbb{N}$ and $L_i \not\subseteq L_j$ for all $i, j < n + 1$ with $i \neq j$, then \mathcal{L} is prescribed **TxtFin**-learnable as follows. Given a hypothesis space \mathcal{H} , let i_0, \dots, i_n be indices for L_0, \dots, L_n in \mathcal{H} respectively. Let $x_{k,l}$ be an element in $L_k - L_l$ for all $k, l \leq n$ with $k \neq l$. On input $T[t]$, search for the least k such that $x_{k,l} \in \text{content}(T[t])$ for all $l \leq n$ with $l \neq k$. If such a k is found, output i_k and stop; otherwise output ?. It is easy to verify that the above learner **TxtFin**-learns \mathcal{L} using hypothesis space \mathcal{H} .

Suppose $\mathcal{L} = \{L_0, L_1, \dots\}$ is prescribed **TxtFin**-learnable but infinite and $L_i \neq L_j$ whenever $i \neq j$. Let S, I_n be as in Definition 18. Let $\mathcal{H} = (H_m)_{m \in \mathbb{N}}$, where H_m is defined as follows. For each $m \in \mathbb{N}$, let $n \in \mathbb{N}$ be the number such that $m \in I_n$, then

$$H_m(x) = \begin{cases} 1 - L_{x-m-t}(x), & \text{if } x \geq m + t \text{ and } m \in S_{t+1} - S_t \text{ for some} \\ & t \in \{0, 1, \dots, x\}; \\ L_n(x), & \text{otherwise.} \end{cases}$$

Note that $\{H_i : i \in \mathbb{N}\} \supseteq \mathcal{L}$ and \mathcal{H} is an indexed family. Let M be a **TxtFin**-learner for \mathcal{L} using hypothesis space \mathcal{H} . For each $i \in \mathbb{N}$, let $f(i)$ be the first

index which M outputs on the canonical text T_i for L_i . Thus, $H_{f(i)} = L_i$. Consequently, $f(i) \notin S$ and $f(i) \in I_i$. Hence $f(i) \neq f(j)$ for distinct i, j . Thus $f(0), f(1), f(2), \dots$ is an infinite r.e. subset of \bar{S} , a contradiction. Hence, any prescribed **TextFin**-learnable class \mathcal{L} must be finite. In addition, there do not exist i, j with $i \neq j$ and $L_i \subset L_j$ for any **TextFin**-learnable class $\mathcal{L} = \{L_0, L_1, \dots\}$ — otherwise, a σ such that, (i) $\text{content}(\sigma) \subseteq L_i$ and (ii) the learner on σ outputs a hypothesis for L_i , can be extended to a text for L_j ; thus the learner fails to **TextFin**-learn L_j . \square

For conservative learning, there are infinite classes which can be uniformly conservatively learnt. One such example is the class $\mathcal{L} = \{L_a : a \in \mathbb{N}\}$, where $L_a = \mathbb{N} - \{a\}$. Note that all the languages in this class are cofinite. In fact this is unavoidable as for conservative learning, uniform and prescribed learning imply that (almost) all the languages in the class are cofinite.

Theorem 20. [26]

- (a) If \mathcal{L} is uniformly conservatively **TextEx**-learnable, then every $L \in \mathcal{L}$ is cofinite.
- (b) If \mathcal{L} is prescribed conservatively **TextEx**-learnable, then all but finitely many languages $L \in \mathcal{L}$ are cofinite.

Proof. We only give the proof of part (a) from [26]. We refer the reader to [26] for the proof of part (b).

Let S be as in Definition 18. Suppose $\mathcal{L} = \{L_0, L_1, \dots\}$, where $L_i \neq L_j$ for $i \neq j$. Furthermore, let $G_a = L_a$ if $a < |\mathcal{L}|$ and $G_a = \mathbb{N}$ otherwise.

We define a sequence of hypothesis spaces $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \dots$, where for each $n \in \mathbb{N}$ the hypothesis space $\mathcal{H}_n = (H_m^n)_{m \in \mathbb{N}}$ is defined as follows:

$$H_{\langle i, j \rangle}^n = \begin{cases} G_i, & \text{if } j \notin S \text{ and } j > n; \\ G_i \cup \{t+1, t+2, t+3, \dots\}, & \text{if } j \in S_{t+1} - S_t \text{ and } j > n; \\ \mathbb{N}, & \text{if } j \leq n. \end{cases}$$

Note that the case distinction covers all cases as $S_0 = \emptyset$. Furthermore, $\mathcal{H}_0, \mathcal{H}_1, \dots$ is a recursive enumeration of indexed families. Since \mathcal{L} is uniformly conservatively learnable, there exists a recursive enumeration of learners M_0, M_1, M_2, \dots such that for all n , M_n conservatively learns \mathcal{L} using hypothesis space \mathcal{H}_n .

For all $a < |\mathcal{L}|$ and $n \in \mathbb{N}$, let $e = \langle v(a, n), w(a, n) \rangle$ be the first number found (in some algorithmic search) such that M_n outputs e on the canonical text T_a of L_a and one of the following conditions hold:

- (a) $w(a, n) \in S$ and $L_a \subseteq H_{\langle v(a, n), w(a, n) \rangle}^n$ (note that this can be verified by finding a t with $w(a, n) \in S_t$ and checking $L_a(x) \subseteq H_{\langle v(a, n), w(a, n) \rangle}^n(x)$ for all $x \leq t$);
- (b) $v(a, n) = a$;
- (c) $w(a, n) \leq n$.

Note that such $e = \langle v(a, n), w(a, n) \rangle$ exists for all n .

Now note that, for every $a < |\mathcal{L}|$, there is an n such that either $w(a, n) \leq n$ or $w(a, n) \in S$: otherwise the set $\{w(a, n) : n \in \mathbb{N}\}$ would be an infinite r.e. set disjoint to S , a contradiction as S is simple.

Now for each $a < |\mathcal{L}|$, let n_a be such that $w(a, n_a) \leq n$ or $w(a, n_a) \in S$. Then, it follows that $L_a \subseteq H_{\langle v(a, n_a), w(a, n_a) \rangle}^{n_a}$ (by definition of \mathcal{H}_n and the definition of $v(a, n_a), w(a, n_a)$), and $H_{\langle v(a, n_a), w(a, n_a) \rangle}$ is cofinite (by definition of \mathcal{H}_n). Thus, as M_n is conservative and learns L_a , it follows that $L_a = H_{\langle v(a, n_a), w(a, n_a) \rangle}^n$ and thus L_a is cofinite. This completes the proof of part (a). \square

Furthermore, uniformly class-preserving conservative learning and prescribed conservative learning are incomparable.

Theorem 21. [26]

- (a) *There exists a class \mathcal{L} which is uniformly class-preservingly conservatively **TextEx**-learnable, but not prescribed conservatively **TextEx**-learnable.*
- (b) *There exists a class \mathcal{L} which is prescribed conservatively **TextEx**-learnable but not uniformly class-preservingly conservatively **TextEx**-learnable.*

We now consider the effect of prescribing the hypothesis space for monotonic learning.

Theorem 22. [26]

- (a) *\mathcal{L} is prescribed strongly monotonically **TextEx**-learnable iff \mathcal{L} is finite.*
- (b) *Any non-empty \mathcal{L} is not uniformly strongly monotonically **TextEx**-learnable.*

Proof. We give the proof of the theorem from [26].

(a) If \mathcal{L} is finite, then it is easily seen to be prescribed strongly monotonically learnable.

Now assume that \mathcal{L} is infinite. Let $odd(x) = 1$ for odd x and $odd(x) = 0$ for even x . Furthermore $even(x) = 1 - odd(x)$. Let M_0, M_1, M_2, \dots be a recursive enumeration of all learners. Suppose $\mathcal{L} = \{L_0, L_1, \dots\}$ is an infinite indexed family, where $L_i \neq L_j$ for $i \neq j$. We define an indexed family hypothesis space $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ such that \mathcal{L} is not strongly monotonically learnable using hypothesis space \mathcal{H} . Let F be a recursive set such that F differs from each set in \mathcal{L} on infinitely many even and infinitely many odd inputs. Let T_i denote the canonical text for L_i , obtained algorithmically from i . Let

$$H_{\langle i, j \rangle}(x) = \begin{cases} \max(\{even(x), F(x)\}), & \text{if } \langle i, j \rangle \text{ is the first index, if any, with} \\ & \text{the first component being } i, \text{ output by} \\ & M_i \text{ on } T_i \text{ within } x \text{ steps;} \\ \min(\{odd(x), F(x)\}), & \text{if } \langle i, j \rangle \text{ is the second distinct index, if} \\ & \text{any, with the first component being } i, \\ & \text{output by } M_i \text{ on } T_i \text{ within } x \text{ steps;} \\ L_i(x), & \text{otherwise.} \end{cases}$$

Clearly, \mathcal{H} is an indexed family hypothesis space which contains \mathcal{L} . Furthermore, note that either $H_{\langle i, j \rangle} = L_i$, or $H_{\langle i, j \rangle} \notin \mathcal{L}$ (as, either $H_{\langle i, j \rangle}$ never follows the first (second) clause, or it follows it for all but finitely many x).

For $i \in \mathbb{N}$, consider the behaviour of M_i on the canonical text T_i for L_i :

1. If M_i does not output an index of the form $\langle i, j \rangle$, then M_i fails to learn L_i because from the definition of \mathcal{H} , only indices of such form can be indices for L_i .
2. If M_i outputs only one such index, then from the definition of \mathcal{H} , the index is not for any $L \in \mathcal{L}$, thus not for L_i .
3. If M_i outputs at least two different such indices, say $\langle i, j_1 \rangle$ and $\langle i, j_2 \rangle$ being the first and second one respectively, then from the definition of \mathcal{H} , $H_{\langle i, j_1 \rangle} \not\subseteq H_{\langle i, j_2 \rangle}$, because $H_{\langle i, j_1 \rangle}$ contains all even numbers larger than x while $H_{\langle i, j_2 \rangle}$ does not, where x is the number of steps needed for M_i to output $\langle i, j_2 \rangle$.

Hence, M_i fails to learn L_i strongly monotonically from T_i . Thus, no learner learns \mathcal{L} strongly monotonically using hypothesis space \mathcal{H} , a contradiction. Hence, \mathcal{L} must be finite.

(b) Suppose $\mathcal{L} = \{L_0, L_1, \dots\}$, where $L_i \neq L_j$ for $i \neq j$. Let $G_e = L_e$ if $e < |\mathcal{L}|$ and let G_e be some recursive set outside \mathcal{L} otherwise. To see that \mathcal{L} is not uniformly strongly monotonically learnable, suppose by way of contradiction that there exists a recursive enumeration of learners M_0, M_1, \dots such that whenever φ_i defines a hypothesis space \mathcal{H} which contains \mathcal{L} , then M_i learns \mathcal{L} strongly monotonically using hypothesis space \mathcal{H} . Let F be a recursive set such that F differs from each set in \mathcal{L} on infinitely many even and infinitely many odd inputs. Let T_i denote a standard text for G_i , obtained algorithmically from i . By Kleene's recursion theorem [41], there exists an e such that:

$$\varphi_e(\langle i, j \rangle, x) = \begin{cases} \max(\{\text{even}(x), F(x)\}), & \text{if } \langle i, j \rangle \text{ is the first index, if any,} \\ & \text{with the first component being } i, \\ & \text{output by } M_e \text{ on } T_i \text{ within } x \text{ steps;} \\ \min(\{\text{odd}(x), F(x)\}), & \text{if } \langle i, j \rangle \text{ is the second distinct index,} \\ & \text{if any, with the first component} \\ & \text{being } i, \text{ output by } M_e \text{ on } T_i \\ & \text{within } x \text{ steps;} \\ G_i(x), & \text{otherwise.} \end{cases}$$

It can be verified that M_e does not strongly monotonically learn \mathcal{L} using hypothesis space defined by φ_e in a way similar to part (a). \square

On the other hand, the class $\mathcal{L} = \{L_i : i \in \mathbb{N}\}$, where $L_i = \{i\}$, is uniformly monotonically learnable.

In contrast to conservative learning, for monotonic learning, uniform learnability implies that the languages in the class are finite.

Theorem 23. [26]

(a) If \mathcal{L} is uniformly monotonically **TextEx**-learnable, then \mathcal{L} contains only finite sets.

(b) If \mathcal{L} is prescribed monotonically **TextEx**-learnable, then \mathcal{L} contains only finitely many infinite sets.

The following theorem gives relationship between uniformly class-preserving and prescribed (strong) monotonic learning.

Theorem 24. [26] (a) *There exists a class \mathcal{L} which is uniformly class-preservingly strongly monotonically **TxtEx**-learnable but not prescribed monotonically **TxtEx**-learnable.*

(b) *There exists a class \mathcal{L} which is prescribed monotonically **TxtEx**-learnable but not uniformly class-preservingly monotonically **TxtEx**-learnable.*

(c) *Every prescribed strongly monotonically **TxtEx**-learnable class is also uniformly class-preservingly strongly monotonically **TxtEx**-learnable.*

7. Control Structures in Hypothesis Spaces

Case, Jain and Suraj [11] considered whether presence or absence of some control structures (see [40, 42, 43]) in hypothesis spaces is needed for being able to learn certain classes. In this section we will only be concerned with universal programming systems.

An *extensional* (denotational) control structure [40, 42, 43] is given by (Θ, m, n) , where Θ is an enumeration operator [41], which maps m sets and n natural numbers to a set. For example, the control structure *union* takes as input two sets A and B , and gives as output $A \cup B$. Here are few more control structures, some of which take both sets and natural numbers as input, whereas some take only natural numbers as input.

- $\mathbf{fin}(x) = D_x$, where D_x is the x -th finite set in some 1–1 computable listing of all finite sets;
- $\mathbf{coinit}(x) = \{y : y \geq x\}$;
- $\mathbf{cosingle}(x) = \mathbb{N} - \{x\}$;
- $\mathbf{proj}(S, j) = \{k : \langle j, k \rangle \in S\}$.

An implementation of an extensional control structure (Θ, m, n) in a universal programming system $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ is given by a function f such that, for all $i_1, i_2, \dots, i_m, x_1, x_2, \dots, x_n$,

$$H_{f(i_1, i_2, \dots, i_m, x_1, x_2, \dots, x_n)} = \Theta(H_{i_1}, H_{i_2}, \dots, H_{i_m}, x_1, \dots, x_n).$$

This function f may or may not be computable or even limiting computable depending on the universal programming system \mathcal{H} chosen. It can be shown that acceptable programming systems support all extensional control structures via a computable function f .

For a control structure C , we say that $\mathcal{H} \vdash C$, if there is a computable function f which implements C in \mathcal{H} . We say that $\mathcal{H} \vdash \mathit{lim}\text{-}C$, if there is a limiting computable function f which implements C in \mathcal{H} .

[11] showed that the class of finite sets is **TxtEx**-learnable using a universal programming system $\mathcal{H} = (H_i)_{i \in \mathbb{N}}$ as a hypothesis space iff $\mathcal{H} \vdash \mathbf{fin}$. Similarly, $\{\{y : y \geq x\} : x \in \mathbb{N}\}$ is **TxtEx**-learnable using \mathcal{H} as a hypothesis space iff $\mathcal{H} \vdash \mathit{lim}\text{-}\mathbf{coinit}$, and $\{\mathbb{N} - \{x\} : x \in \mathbb{N}\}$ is **TxtEx**-learnable using \mathcal{H} as a hypothesis space iff $\mathcal{H} \vdash \mathit{lim}\text{-}\mathbf{cosingle}$.

Note that the class of finite sets is **TextEx**-learnable using every universal programming system \mathcal{H} as a hypothesis space. Thus, for every universal programming system \mathcal{H} , $\mathcal{H} \vdash \text{lim-fin}$. However, there are universal programming systems $\mathcal{H}, \mathcal{H}'$ such that $\mathcal{H} \not\vdash \text{lim-coinit}$ and $\mathcal{H}' \not\vdash \text{lim-cosingle}$. Thus the class of cosingletons and the class of coinital segments of \mathbb{N} are not **TextEx**-learnable using some universal programming system.

[11] gave the following characterizations of acceptable and K -acceptable programming systems:

Theorem 25. [11] *A universal programming system \mathcal{H} is an acceptable programming system iff $[\mathbf{TextFin}_{\mathcal{H}} = \mathbf{TextFin}$ and $\mathcal{H} \vdash \mathbf{proj}]$.*

Theorem 26. [11] *A universal programming system \mathcal{H} is an K -acceptable programming system iff $[\mathbf{TextEx}_{\mathcal{H}} = \mathbf{TextEx}$ and $\mathcal{H} \vdash \text{lim-proj}]$.*

Acknowledgements: We thank Frank Stephan for several helpful discussions and comments on the paper. We also thank the anonymous referees for several helpful comments.

References

- [1] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21(1):46–62, 1980.
- [2] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
- [3] D. Angluin and C. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–289, 1983.
- [4] G. Baliga, J. Case, W. Merkle, F. Stephan, and R. Wiehagen. When unlearning helps. *Information and Computation*, 206(5):694–709, 2008.
- [5] J. Bārzdiņš. Inductive inference of automata, functions and programs. In *Proceedings of the 20th International Congress of Mathematicians, Vancouver*, pages 455–460, 1974. In Russian. English translation in American Mathematical Society Translations: Series 2, 109:107–112, 1977.
- [6] J. Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.
- [7] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28(2):125–155, 1975.
- [8] M. Bowerman. Starting to talk worse: Clues to language acquisition from children’s late speech errors. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, New York, 1982.

- [9] R. Brown and U. Bellugi. Three processes in the child’s acquisition of syntax. *Harvard Educational Review*, 34:133–151, 1964.
- [10] J. Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999.
- [11] J. Case, S. Jain, and M. Suraj. Control structures in hypothesis spaces: The influence on learning. *Theoretical Computer Science*, 270(1–2):287–308, 2002.
- [12] J. Case and C. Lynes. Machine inductive inference and language identification. In M. Nielsen and E. M. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 107–115. Springer-Verlag, 1982.
- [13] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [14] P. Dale. *Language Development, Structure and Function*. Holt, Reinhart, and Winston, New York, 1976.
- [15] R. Freivalds, E. Kinber, and R. Wiehagen. Inductive inference and computable one-one numberings. *Zeitschr. f. math. Logik und Grundlagen d. Math. Bd.*, 28:463–479, 1982.
- [16] R. Freivalds and R. Wiehagen. Inductive inference with additional information. *Journal of Information Processing and Cybernetics (EIK)*, 15:179–195, 1979.
- [17] R. Friedberg. Three theorems on recursive enumeration. *Journal of Symbolic Logic*, 23(3):309–316, 1958.
- [18] M. Fulk. *A Study of Inductive Inference Machines*. PhD thesis, SUNY/Buffalo, 1985.
- [19] M. Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85(1):1–11, 1990.
- [20] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
- [21] S. Jain, S. Lange, and S. Zilles. A general comparison of language learning from examples and from queries. *Theoretical Computer Science A*, 387(1):51–66, 2007. Special Issue on Algorithmic Learning Theory, 2005.
- [22] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Mass., second edition, 1999.

- [23] S. Jain and A. Sharma. Learning with the knowledge of an upper bound on program size. *Information and Computation*, 102:118–166, 1993.
- [24] S. Jain and F. Stephan. Learning in Friedberg numberings. *Information and Computation*, 206(6):776–790, 2008.
- [25] S. Jain and F. Stephan. Numberings optimal for learning. In Y. Freund, L. Györfi, G. Turán, and T. Zeugmann, editors, *Algorithmic Learning Theory: 19th International Conference (ALT' 2008)*, volume 5254 of *Lecture Notes in Artificial Intelligence*, pages 434–448. Springer-Verlag, 2008.
- [26] S. Jain, F. Stephan, and N. Ye. Prescribed learning of indexed families. *Fundamenta Informaticae*, 83(1–2):159–175, 2008.
- [27] S. Jain, F. Stephan, and N. Ye. Prescribed learning of r.e. classes. *Theoretical Computer Science*, 410:1796–1806, 2009.
- [28] K. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.
- [29] S. Lange and T. Zeugmann. Language learning in dependence on the space of hypotheses. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 127–136. ACM Press, 1993.
- [30] S. Lange and T. Zeugmann. The learnability of recursive languages in dependence on the space of hypotheses. Technical Report 20/93, GOSLER-Report, FB Mathematik und Informatik, TH Leipzig, 1993.
- [31] S. Lange and T. Zeugmann. Learning recursive languages with bounded mind changes. *International Journal of Foundations of Computer Science*, 4:157–178, 1993.
- [32] S. Lange and T. Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53(1):88–103, 1996.
- [33] S. Lange and T. Zeugmann. Set-driven and rearrangement-independent learning of recursive languages. *Mathematical Systems Theory*, 29:599–634, 1996.
- [34] S. Lange, T. Zeugmann, and S. Kapur. Monotonic and dual monotonic language learning. *Theoretical Computer Science A*, 155:365–410, 1996.
- [35] S. Lange and S. Zilles. Comparison of query learning and Gold-style learning in dependence of the hypothesis space. In Shai Ben-David, John Case, and Akira Maruoka, editors, *Algorithmic Learning Theory: 15th International Conference (ALT' 2004)*, volume 3244 of *Lecture Notes in Artificial Intelligence*, pages 99–113. Springer-Verlag, 2004.

- [36] G. Marcus, S. Pinker, M. Ullman, M. Hollander, T. Rosen, and F. Xu. *Overregularization in Language Acquisition*. Monographs of the Society for Research in Child Development, vol. 57, no. 4. University of Chicago Press, 1992. Includes commentary by Harold Clahsen.
- [37] D. Osherson, M. Stob, and S. Weinstein. Learning strategies. *Information and Control*, 53:32–51, 1982.
- [38] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, 1986.
- [39] D. Osherson and S. Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.
- [40] G. Riccardi. The independence of control structures in abstract programming systems. *Journal of Computer and System Sciences*, 22:107–143, 1981.
- [41] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.
- [42] J. Royer. *A Connotational Theory of Program Structure*, volume 273 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.
- [43] J. Royer and J. Case. *Subrecursive programming systems: Complexity & succinctness*. Birkhäuser, 1994.
- [44] G. Schäfer-Richter. *Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien*. PhD thesis, RWTH Aachen, 1984.
- [45] S. Strauss and R. Stavy. *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, New York, 1982.
- [46] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition*. MIT Press, 1980.
- [47] R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)*, 12(1–2):93–99, 1976.
- [48] R. Wiehagen. *Zur Theorie der Algorithmischen Erkennung*. PhD thesis, Humboldt University of Berlin, 1978.
- [49] R. Wiehagen. A thesis in inductive inference. In J. Dix, K. Jantke, and P. Schmitt, editors, *Nonmonotonic and Inductive Logic, 1st International Workshop*, volume 543 of *Lecture Notes in Artificial Intelligence*, pages 184–207. Springer-Verlag, 1990.
- [50] R. Wiehagen and T. Zeugmann. Learning and consistency. In K. P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems, (GOSLER) Final Report*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 1–24. Springer-Verlag, 1995.

- [51] T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In K. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer-Verlag, 1995.
- [52] T. Zeugmann, S. Lange, and S. Kapur. Characterizations of monotonic and dual monotonic language learning. *Information and Computation*, 120:155–173, 1995.
- [53] T. Zeugmann and S. Zilles. Learning recursive functions: A survey. *Theoretical Computer Science A*, 397(1–3):4–56, 2008. Special Issue on Forty Years of Inductive Inference. Dedicated to the 60th Birthday of Rolf Wiehagen.