The Synthesis of Language Learners

Ganesh R. Baliga
Computer Science Department
Rowan College of New Jersey
Mullica Hill, NJ 08024, USA
(Email: baliga@gboro.rowan.edu)
John Case
Department of CIS
University of Delaware
Newark, DE 19716, USA
(Email: case@cis.udel.edu)
Sanjay Jain
Department of ISCS
National University of Singapore
Singapore 119260
(Email: sanjay@iscs.nus.edu.sg)

Abstract

An *index for an r.e. class of languages* (by definition) is a procedure which generates a sequence of grammars defining the class. An *index for an indexed family of languages* (by definition) is a procedure which generates a sequence of decision procedures defining the family.

Studied is the metaproblem of synthesizing *from indices for r.e. classes and for indexed families of languages* various kinds of language-learners for the corresponding classes or families indexed.

Many positive results, as well as some negative results, are presented regarding the existence of such synthesizers. The negative results essentially provide lower bounds for the positive results. The proofs of some of the positive results yield, as pleasant corollaries, subset-principle or tell-tale style characterizations for the learnability of the corresponding classes or families indexed.

For example, the indexed families of recursive languages that can be behaviorally correctly identified from positive data are surprisingly characterized by Angluin's (1980b) Condition 2 (the *subset principle* for circumventing overgeneralization).

1 Introduction

Ex-*learners*, when successful on an object input, (by definition) find a final correct program for that object after at most finitely many trial and error attempts (cf. e.g. Gold (1967), Blum and Blum (1975), Case and Smith (1983), Case and Lynes (1982)).[1]

For function learning, there is a learner-synthesizer algorithm **lsyn** so that, if **lsyn** is fed any procedure that lists programs for some (possibly infinite) class $\mathcal{S}$ of *total* functions, then **lsyn** outputs an **Ex**-learner successful on $\mathcal{S}$ (Gold (1967)). The learners so synthesized are called *enumeration techniques* (cf. e.g. Gold (1967), Blum and Blum (1975), Fulk (1990b)). These enumeration techniques yield many positive learnability results, for example, that the class of all functions computable in time polynomial in the length of input is

[1] **Ex** is short for *explanatory.*

Ex-learnable. The reader is referred to Jantke (1979) for a discussion of synthesizing learners for classes of recursive functions that are not necessarily recursively enumerable.

Ex language learning *from positive data and with learners outputting grammars* is called **TxtEx**-learning. Osherson, Stob and Weinstein (1988) provide an amazingly negative result: there is *no* learner-synthesizer algorithm **lsyn** so that, if **lsyn** is fed a *pair* of grammars g_1, g_2 for a language class $\mathcal{L} = \{L_1, L_2\}$, then **lsyn** outputs an **TxtEx**-learner successful on $\mathcal{L}$.[2] Of course, it follows from this negative result that there is also no synthesizer algorithm **lsyn** so that, if **lsyn** is fed instead *a procedure listing* a pair of grammars g_1, g_2 for a language class $\mathcal{L} = \{L_1, L_2\}$, then **lsyn** outputs an **TxtEx**-learner successful on $\mathcal{L}$.

In the present paper we show how to circumvent some of the sting of this negative result by resorting to more general learners than **TxtEx**. Example more general learners are: **TxtBc**-*learners*, which, take positive data about a language, and, when successful on that language (by definition) find a final (possibly infinite) *sequence* of correct grammars for that object after at most finitely many trial and error attempts (cf. e.g. Bārzdiņš (1974), Case and Smith (1983), Case and Lynes (1982), Osherson and Weinstein (1982a)).[3]

If a suitable learner-synthesizer algorithm **lsyn** is fed procedures for listing *decision procedures* (instead of mere grammars), one also has more success at synthesizing learners. An *indexed family* is a language class defined by an r.e. listing of decision procedures for the languages in the class, and an *index* for an indexed family is a procedure for listing decision procedures defining it. Even for synthesis from indices for indexed families, one has negative results. For example, Kapur (1991) shows that one cannot algorithmically find an **TxtEx**-learning machine for an arbitrary **TxtEx**-*learnable* indexed family of recursive languages from an index of that family. This is a bit weaker than a closely related negative result below (Theorem 31 in Section 3.2 below).

The computational learning theory community has shown considerable interest (spanning from Gold (1967) to Zeugmann and Lange (1995) to the present) in indexed families (sometimes called *uniformly decidable classes*). As is essentially pointed out in Angluin (1980b), all of the formal language style example classes *are* indexed families.[4]

One of our main results (Theorem 30 in Section 3.2 below) implies: there *is* a learner-synthesizer algorithm **lsyn** so that, if **lsyn** is fed any index for any indexed family $\mathcal{L}$ of languages *which can be* **TxtBc**-*learned*, then **lsyn** outputs a **TxtBc**-learner successful on $\mathcal{L}$.

The proof of this positive result yields the surprising characterization (Corollary 25 in Section 3.2 below): if $\mathcal{L}$ is an indexed family, then: $\mathcal{L}$ can be **TxtBc**-learned iff

$$(\forall L \in \mathcal{L})(\exists S \subseteq L \mid S \text{ is finite})(\forall L' \in \mathcal{L} \mid S \subseteq L')[L' \not\subset L].$$

Nicely, whether or not the just above displayed condition holds for an indexed family $\mathcal{L}$ *is easily checkable*! Furthermore, this condition turns out to be Angluin's important Condition 2 from Angluin (1980b), and it

[2]The problem is not that correct grammars for finite classes of languages can't be learned in the limit; they can (Osherson, Stob and Weinstein (1986a)), and by an obvious enumeration technique. The problem is how to pass algorithmically from a list of grammars to a machine which so learns the corresponding languages.

A study of the proof of the result shows that, intuitively, the difficulty, given a pair of grammars g_1, g_2 for a language class $\mathcal{L} = \{L_1, L_2\}$, to synthesize a **TxtEx**-learner successful on $\mathcal{L}$ is in deciding from g_1, g_2 whether or not $L_1 = L_2$. This equivalence problem is well-known to be algorithmically unsolvable (Rogers (1967)).

[3]**Bc** is short for *behaviorally correct*.

[4]Many example indexed families are known to be learnable (cf. e.g. Angluin (1980a), Angluin (1980b), Angluin (1982); Shinohara (1983), Wright (1989)). Particularly influential have been *pattern languages* (Angluin (1980a)) and finite unions thereof (Shinohara (1983), Wright (1989)). Nix (1983) as well as Shinohara and Arikawa (1995) outline interesting applications of pattern inference algorithms. For example, pattern language learning algorithms have been successfully applied for solving problems in molecular biology (see Shimozono, Shinohara, Shinohara, Miyano, Kuhara and Arikawa (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 (EFSs). Arikawa, Shinohara and Yamamoto (1992) show that the EFSs can also be treated as a logic programming language over strings. The techniques for learning finite unions of pattern languages have been extended to show the learnability of various subclasses of EFSs (Shinohara (1991)). Investigations of the learnability of subclasses of EFSs are important because they yield corresponding results about the learnability of subclasses of logic programs. Arimura and Shinohara (1994) use the insight gained from the learnability of EFSs subclasses to show that a class of linearly covering logic programs with local variables is **TxtEx**-learnable. These results have consequences for Inductive Logic Programming (cf. e.g. Muggleton and Raedt (1994), Lavarač and Džeroski (1994)).

is referred to as the *subset principle*, in general a necessary condition for preventing overgeneralization in learning from positive data (cf. e.g Angluin (1980b), Berwick (1985), Zeugmann, Lange and Kapur (1995), Kapur and Bilardi (1992), Case (1998)). Angluin's Condition 1, also from Angluin (1980b), is a constructive version of her Condition 2 additionally requiring that a sufficient collection of finite sets S for the displayed condition above is defined by an r.e. set of grammars. She shows that there are indexed families satisfying her Condition 2 but not her Condition 1. She also shows that her Condition 1 characterizes the indexed families in **TxtEx**! Hence, we have that an indexed family is **TxtBc**-learnable but not **TxtEx**-learnable $\Leftrightarrow$ it satisfies Angluin's Condition 2 but not her Condition 1! Discussion following the proof of Theorem 31 below clarifies the connection between our learning machine synthesizing algorithm from the proof of Theorem 30 and one implicit in Angluin's proof (Angluin (1980b)) of her characterization theorem.

Suppose a is a non-negative integer or a $*$. $\mathbf{TxtBc}^a$*-learning* (Case and Lynes (1982), Case and Smith (1983)) is a variant of **TxtBc**-learning in which the final grammars are each allowed to be incorrect on no more than a words.[5] Theorem 30 below shows more generally that, for each a, there is a learner-synthesizer algorithm **lsyn** so that, if **lsyn** is fed any index for any indexed family $\mathcal{L}$ of languages *which can be* $\mathbf{TxtBc}^a$*-learned*, then **lsyn** outputs a $\mathbf{TxtBc}^a$-learner successful on $\mathcal{L}$. Corollary 26 in Section 3.2 below characterizes the $\mathbf{TxtBc}^a$-learnable indexed families as exactly those satisfying

$$(\forall L \in \mathcal{L})(\exists S \subseteq L \mid S \text{ is finite})(\forall L' \mid S \subseteq L' \in \mathcal{L} \\ \wedge\ L' \subseteq L)[\mathrm{card}(L - L') \leq 2a],$$

another easily checkable condition.

We outline next the principle additional results of the present paper.

Let card(S) denote the cardinality of a set S. We show (Theorem 8 in Section 3.1 below) that there is an algorithm for translating any listing procedure for a *finite* set P of grammars into a learning procedure $\mathbf{M}_P$ which, given any listing of a language L generated by grammars in P, eventually converges to outputting over and over no more than card(P) grammars each of which is correct for L. The requirement for successful learning, in this case, is loosened *from* requiring that $\mathbf{M}_P$ **TxtEx**-learn L *to* merely requiring $\mathbf{M}_P$ to output eventually $\leq$ card(P) grammars correct for L.[6] Furthermore, $\mathbf{M}_P$ does involve an enumeration technique, a procedure which does matching and elimination based on the grammars in P. Interestingly, too, $\mathbf{M}_P$, in this case, outputs conjectures from the "hypothesis space" P itself (Lange and Zeugmann (1993)).

Suppose x is a procedure for listing an r.e. (possibly infinite) set of grammars P. Let $\mathcal{C}_x$ be the set of languages generated by the grammars in P. In Section 3.1 we explore the problem of synthesizing learning machines for *learnable* $\mathcal{C}_x$'s from the corresponding x's.

One shot language identification (called $\mathbf{TxtEx}_0$-identification below[7]) is just the important case of **TxtEx**-identification where the learning procedure makes but one conjecture (which must, then, be correct). The proof of Theorem 12 below (in Section 3.1) presents an algorithm for transforming any x such that $\mathcal{C}_x$ is $\mathbf{TxtEx}_0$-identifiable into a **TxtBc**-learner for $\mathcal{C}_x$.

For this, as well as for our other results providing the synthesis of a learning machine, each synthesized learning machine can be construed as implementing a (perhaps complicated) enumeration technique; however, of necessity, in most cases the conjectures of the synthesized machines go beyond the original hypothesis space (Lange and Zeugmann (1993)).

Regarding the positive results about $\mathcal{C}_x$'s, we also present corresponding lower bound results showing, in many cases, the positive results to be best possible. For example, Theorem 14 below shows the necessity of the cost, from Theorem 12, of passing from *no* mind changes for the input classes to *infinitely* many in the *synthesized* learning machines.

[5]This is where, *no more than $*$ words* means: at most finitely many words. Also, by convention $2* = *$.

[6]The criterion requiring a machine, on positive data for a language L, to output eventually no more than n distinct grammars each of which is correct is called $\mathbf{TxtFex}_n$*-learning* (Case (1998)). $\mathbf{TxtFex}_1$-learning is just **TxtEx**-learning, but one can learn strictly larger classes of languages with $\mathbf{TxtFex}_{n+1}$-learners than with $\mathbf{TxtFex}_n$-learners (Case (1998)). One can learn larger classes of languages with **TxtBc**-learners than with $\mathbf{TxtFex}_n$-learners for any n (Case (1998)); of course this is, then, at a cost of outputting infinitely many distinct grammars in the limit.

[7]The 0 in '$\mathbf{TxtEx}_0$' has a totally different meaning from the n in '$\mathbf{TxtFex}_n$'; the former is a bound on mind changes for convergence to a single final program, the latter is a bound on the number of different programs an associated machine eventually vacillates between in the limit.

One might hope to obtain synthesized learning machines with better mind change complexity if one provided indices for listing *decision procedures* in place of grammars for the languages to be learned from positive data. In Section 3.2 below, we see that this is indeed the case. For example, the proof of Theorem 21 below (in Section 3.2) presents an algorithm for transforming any index of any indexed family for a class of recursive languages $\mathcal{L}$ that is $\mathbf{TxtEx}_0$-identifiable into a learning procedure which $\mathbf{TxtEx}$-identifies $\mathcal{L}$. Theorem 24 shows the necessity of the cost, from Theorem 21, of passing from *no* mind changes for the input classes to *finitely* many in the *synthesized* learning machines. However, the last theorem of Section 3.2 (Theorem 31) shows that the cost of passing from even *one* mind change in the input indexed family to *infinitely* many in the *synthesized* learning machines is necessary.

2 Preliminaries

2.1 Notation

N is the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$. Unless otherwise specified, $e, i, j, k, m, n, p, s, w, x, y, z$, with or without decorations (decorations are subscripts, superscripts and the like), range over N. $*$ is a non-member of N and is assumed to satisfy $(\forall n)[n < * < \infty]$. Furthermore, $2* \stackrel{\text{def}}{=} *$. a, b and c with or without decorations, ranges over $N \cup \{*\}$. By $\emptyset, \in, \subseteq, \subset, \supseteq, \supset$ we mean the empty set, element of, subset, proper subset, superset and proper superset, respectively. P and S, with or without decorations, range over sets. We sometimes write $\text{card}(S) \leq *$ to mean S is finite. We use $S_1 \mathbf{\Delta} S_2$ to denote the symmetric difference of the sets S_1 and S_2. $S_1 =^a S_2$ means that $\text{card}(\{x \mid x \in S_1 \mathbf{\Delta} S_2\}) \leq a$. $\mathbf{ODD} = \{2x + 1 \mid x \in N\}$, and $\mathbf{EVEN} = \{2x \mid x \in N\}$.

$\max(\cdot), \min(\cdot)$ denote the maximum and minimum of a set, respectively, where $\max(\emptyset) = 0$ and $\min(\emptyset)$ is undefined. Fix a recursive *canonical indexing* of the finite sets (Rogers (1967)). The $\min(\cdot)$ of a collection of finite sets is, then, the finite set in the collection with minimal canonical index. Also, when we compare finite sets by $<$ we are comparing their corresponding canonical indices.

We use the symbol $\downarrow$ to mean that a computation converges. f, g and h with or without decorations range over *total* functions with arguments and values from N. $\langle \cdot, \cdot \rangle$ stands for an arbitrary, computable, one-to-one encoding of all pairs of natural numbers onto N (Rogers (1967)).

We fix an *acceptable* programming system φ for the partial computable functions: $N \to N$ (cf. e.g. Rogers (1958), Machtey and Young (1978), Royer (1987)). φ_i is the partial computable function computed by program i in the φ-system. $\mathcal{R}$ represents the class of all (total) recursive functions of one variable. $\mathcal{R}_{0,1}$ denotes the class of all (total) recursive 0-1 valued functions.

W_i is $\text{domain}(\varphi_i)$. W_i is, then, the r.e. set/language ($\subseteq N$) accepted (or equivalently, generated (Hopcroft and Ullman (1979))) by the φ-program i. $W_i^s \stackrel{\text{def}}{=} \{x \leq s \mid x \text{ appears in } W_i \text{ in } \leq s \text{ steps}\}$. For a language $L \subseteq N$, $L[x]$ is $\{w \leq x \mid x \in L\}$, and we use χ_L to denote the characteristic function of L; $\overline{L}$ is the complement of L. $\mathcal{L}$, with or without decorations, ranges over set of subsets of the r.e. sets.

We sometimes consider partial recursive functions with two arguments in the φ system. In such cases we implicitly assume that $\langle \cdot, \cdot \rangle$ is used to code the arguments, so, for example, $\varphi_i(x, y)$ stands for $\varphi_i(\langle x, y \rangle)$.

The quantifiers '$\forall^\infty$', and '$\exists^\infty$' (essentially from Blum (1967)), mean 'for all but finitely many' and 'there exist infinitely many', respectively.

$f : N \to N$ is *limiting recursive* $\stackrel{\text{def}}{\Leftrightarrow}$ $(\exists \text{ recursive } g : (N \times N) \to N)(\forall x)[f(x) = \lim_{t \to \infty} g(x, t)]$. Intuitively, $g(x, t)$ is the output at discrete time t of a mind changing algorithm for f (acting on input x); hence, for f limiting recursive as just above, for all x, for all but finitely many times t, the output of the mind changing algorithm on input x is $f(x)$. It is easy to show that there is a limiting recursive function h such that $(\forall \text{ recursive } g)(\forall^\infty x)[h(x) > g(x)]$. Hence, the limiting recursive functions go way beyond the recursive ones; in fact, they have been known since Post (Shapiro (1971)) to characterize the functions recursive in an oracle for the halting problem. The set of all (total) limiting recursive functions of one variable is $\mathcal{LR}$.

We sometimes use the symbol '$\neg$' for *negation.*

2.2 Learning Machines

We now consider language learning machines. We first introduce a notion that facilitates discussion about elements of a language being fed to a learning machine. A *sequence* σ is a mapping from an initial segment of N into $(N \cup \{\#\})$. The *content* of a sequence σ, denoted content(σ), is the set of natural numbers in the range of σ. The *length* of σ, written $|\sigma|$, is the number of elements in σ. Λ denotes the empty sequence. Intuitively, #'s represent pauses in the presentation of data. We let σ and τ, with or without decorations, range over finite sequences. SEQ is the set of all finite sequences. The set of all finite sequences of natural numbers and #'s, SEQ, can be coded onto N. This latter fact will be used implicitly in some of our proofs.

A *language learning machine* is an algorithmic device which computes a mapping from SEQ into $N \cup \{?\}$. Intuitively the outputted ?s represent the machine not yet committing to an output *program*. The reason we want the ?s is so we can avoid biasing the number of *program mind changes* before a learning machine converges: if we allow initial outputs of ?s before, if ever, the first program is output, then we can learn more things within n mind changes than if we had to begin with a program (numerical) output. In this paper we assume, without loss of generality, that for all $\sigma \subseteq \tau$, $[\mathbf{M}(\sigma) \neq ?] \Rightarrow [\mathbf{M}(\tau) \neq ?]$.

$\mathbf{M}$ ranges over language learning machines.

2.3 Fundamental Language Identification Paradigms

A *text* T for a language L is a mapping from N into $(N \cup \{\#\})$ such that L is the set of natural numbers in the range of T. The *content* of a text T, denoted content(T), is the set of natural numbers in the range of T.

Intuitively, a text for a language is an enumeration or sequential presentation of all the objects in the language with the #'s representing pauses in the listing or presentation of such objects. For example, the only text for the empty language is just the infinite sequence of #'s.

We let T, with or without superscripts, range over texts. $T[n]$ is the finite initial sequence of T with length n. Hence, domain$(T[n]) = \{x \mid x < n\}$.

2.3.1 Explanatory Language Identification

Suppose $\mathbf{M}$ is a learning machine and T is a text. $\mathbf{M}(T)\!\downarrow$ (read: $\mathbf{M}(T)$ *converges*) $\stackrel{\text{def}}{\Leftrightarrow}$ $(\exists i)(\forall^\infty n)\ [\mathbf{M}(T[n]) = i]$. If $\mathbf{M}(T)\!\downarrow$, then $\mathbf{M}(T)$ is defined = the unique i such that $(\forall^\infty n)[\mathbf{M}(T[n]) = i]$.

We now introduce criteria for a learning machine to be considered *successful* on languages.

Definition 1 Recall that a and b range over $N \cup \{*\}$.

(1) $\mathbf{M}$ $\mathbf{TxtEx}^a$-*identifies* L (written: $L \in \mathbf{TxtEx}^a(\mathbf{M})$) $\stackrel{\text{def}}{\Leftrightarrow}$ $(\forall$ texts T for $L)(\exists i \mid W_i =^a L)[\mathbf{M}(T)\!\downarrow = i]$.

(2) $\mathbf{M}$ $\mathbf{TxtEx}^a_b$-*identifies* L (written:$L \in \mathbf{TxtEx}^a_b(\mathbf{M})$) $\stackrel{\text{def}}{\Leftrightarrow}$ $[L \in \mathbf{TxtEx}^a(\mathbf{M})\ \wedge\ (\forall$ texts T for $L)[\text{card}(\{x \mid ? \neq \mathbf{M}(T[x])\ \wedge\ \mathbf{M}(T[x]) \neq \mathbf{M}(T[x+1])\}) \leq b]]$.

(3) $\mathbf{TxtEx}^a_b = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtEx}^a_b(\mathbf{M})]\}$.

Gold (1967) introduced the criteria we call $\mathbf{TxtEx}^0_*$. The generalization to the $a > 0$ cases in Definition 1 is motivated by the observation that humans rarely learn a language perfectly, where the a represents an upper bound on the numer of *anomalies* permitted in final grammars. The $a > 0$ case is from Case and Lynes (1982), but Osherson and Weinstein (1982a), independently, introduced the $a = *$ case. For these and the other success criteria of this paper, we have that tolerating more anomalies leads to being able to learn larger classes of languages (Case and Lynes (1982), Case (1998), Baliga and Case (1993)). Gold's model of language learning from text (positive information) and by machine (Gold (1967)) has been very influential in contemporary theories of natural language and in mathematical work motivated by its possible connection to human language learning (cf. e.g. Pinker (1979), Wexler and Culicover (1980), Wexler (1982), Osherson, Stob and Weinstein (1982), Osherson, Stob and Weinstein (1984), Berwick (1985); Gleitman (1986), Case (1986), Osherson, Stob and Weinstein (1986b), Osherson, Stob and Weinstein (1986a), Fulk (1985), Fulk (1990a), Kirsh (1992), Baliga, Case and Jain (1995)).

We sometimes write **TxtEx** for $\mathbf{TxtEx}^0_*$ and $\mathbf{TxtEx}^a$ for $\mathbf{TxtEx}^a_*$.

2.3.2 Vacillatory and Behaviorally Correct Language Identification

Definition 2 We say $\mathbf{M}(T)$ *finitely-converges* (written: $\mathbf{M}(T)\Downarrow$) $\overset{\text{def}}{\Leftrightarrow}$ $\{\mathbf{M}(\tau) \mid \tau \subset T\}$ is a non-empty finite subset of N. If $\mathbf{M}(T)\Downarrow$, then $\mathbf{M}(T)$ is defined $= \{p \mid (\exists^\infty \tau \subset T)[\mathbf{M}(\tau) = p]\}$; otherwise, $\mathbf{M}(T)$ is undefined.

Definition 3 $\mathbf{M}$, $\mathbf{TxtFex}^a_b$-*identifies* an r.e. language L (written:$L \in \mathbf{TxtFex}^a_b(\mathbf{M})$) $\overset{\text{def}}{\Leftrightarrow}$ ($\forall$ texts T for $L)[\mathbf{M}(T)\Downarrow =$ a non-empty set of cardinality $\leq b$ and $(\forall p \in \mathbf{M}(T))[W_p =^a L]]$. $\mathbf{TxtFex}^a_b$ denotes the set of all *classes* $\mathcal{L}$ of languages such that some learning machine $\mathbf{TxtFex}^a_b$-identifies each language in $\mathcal{L}$.

In $\mathbf{TxtFex}^a_b$-identification the b is a "bound" on the number of final grammars and the a a "bound" on the number of anomalies allowed in these final grammars. In general a "bound" of $*$ just means *un*bounded, but *finite*. Intuitively, $\mathcal{L} \in \mathbf{TxtFex}^a_b \overset{\text{def}}{\Leftrightarrow}$ there is an algorithmic procedure $\mathbf{p}$ such that, if $\mathbf{p}$ is given any listing of any language $L \in \mathcal{L}$, it outputs a sequence of grammars *converging* in a non-empty *set* of no more than b grammars, and each of these grammars makes no more than a mistakes in generating L.

N.B. The b in '$\mathbf{TxtEx}^a_b$' has a totally different meaning from the b in '$\mathbf{TxtFex}^a_b$'; the former is a bound on mind changes for convergence to a single final program, the latter is a bound on the number of different programs an associated machine eventually vacillates between in the limit.

We sometimes write $\mathbf{TxtFex}_b$ for $\mathbf{TxtFex}^0_b$.

$\mathbf{TxtFex}^a_1$-identification is clearly equivalent to $\mathbf{TxtEx}^a_*$. Osherson and Weinstein (1982a) were the first to define the notions of $\mathbf{TxtFex}^0_*$ and $\mathbf{TxtFex}^*_*$, and the other cases of $\mathbf{TxtFex}^a_b$-identification are from Case (1986) and Case (1998).

Definition 4 (1) $\mathbf{M}$ $\mathbf{TxtBc}^a$-*identifies* L (written: $L \in \mathbf{TxtBc}^a(\mathbf{M})$) $\overset{\text{def}}{\Leftrightarrow}$ ($\forall$ texts T for $L)(\forall^\infty n)[W_{\mathbf{M}(T[n])} =^a L]$.

(2) $\mathbf{TxtBc}^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtBc}^a(\mathbf{M})]\}$.

In a completely computable universe all texts must be recursive (synonym: computable). This motivates the following

Definition 5 (1) $\mathbf{M}$ $\mathbf{RecTxtBc}^a$-*identifies* L (written: $L \in \mathbf{RecTxtBc}^a(\mathbf{M})$) $\overset{\text{def}}{\Leftrightarrow}$ ($\forall$ *recursive* texts T for $L)(\forall^\infty n)[W_{\mathbf{M}(T[n])} =^a L]$.

(2) $\mathbf{RecTxtBc}^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{RecTxtBc}^a(\mathbf{M})]\}$.

Definition 4 is from Case and Lynes (1982). The $a \in \{0, *\}$ cases were independently introduced in Osherson and Weinstein (1982a) and Osherson and Weinstein (1982b). $\mathbf{RecTxtBc}^a \neq \mathbf{TxtBc}^a$ (Case and Lynes (1982), Freivalds (1985)); however, the restriction to recursive texts doesn't affect learning power for $\mathbf{TxtFex}^a_b$-identification (Case (1998)).

We sometimes write **TxtBc** for $\mathbf{TxtBc}^0$, etc.

Definition 6 σ is called a $\mathbf{TxtEx}^a$-*locking sequence* for $\mathbf{M}$ on L, $\overset{\text{def}}{\Leftrightarrow}$ content$(\sigma) \subseteq L$, $W_{\mathbf{M}(\sigma)} =^a L$, and $(\forall \tau \mid \sigma \subseteq \tau \ \wedge \ \text{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \mathbf{M}(\tau)]$.

σ is called a $\mathbf{TxtBc}^a$-*locking sequence* for $\mathbf{M}$ on L, iff content$(\sigma) \subseteq L$, and $(\forall \tau \mid \sigma \subseteq \tau \ \wedge \ \text{content}(\tau) \subseteq L)[W_{\mathbf{M}(\tau)} =^a L]$.

It can be shown (cf. e.g. Blum and Blum (1975), Osherson and Weinstein (1982a), Osherson, Stob and Weinstein (1986a), Case (1998)) that if $\mathbf{M}$ $\mathbf{TxtEx}^a$-identifies L, then there exists a $\mathbf{TxtEx}^a$-locking sequence for $\mathbf{M}$ on L. Similarly it can be shown that, if $\mathbf{M}$ $\mathbf{TxtBc}^a$-identifies L, then there exists a $\mathbf{TxtBc}^a$-locking sequence for $\mathbf{M}$ on L.

Lemma 4.2.2B in Osherson, Stob and Weinstein (1986a) easily generalizes to cover all learning criteria considered in this paper thereby providing a recursive enumeration $\mathbf{M}_0, \mathbf{M}_1, \ldots$ of (total) language learning machines such that, for any inference criteria $\mathbf{I}$, every $\mathcal{L} \in \mathbf{I}$ is $\mathbf{I}$-identified by some machine in this enumeration. Moreover, this enumeration satisfies the s-m-n property, i.e., given a description, algorithmic in x, of a behavior of a machine $\mathbf{M}$, one can algorithmically find a machine $\mathbf{M}_{f(x)}$ whose learning behavior is at least as good as that of $\mathbf{M}$. In the following we fix such an arbitrary enumeration $\mathbf{M}_0, \mathbf{M}_1, \ldots$.

3 Results

In Section 3.1 we present our positive and negative results on synthesizing learning machines from grammars. In Section 3.2 we present such results on synthesizing learning machines from decision procedures.

3.1 Synthesizing From Uniform Grammars

We can formally define the classes $\mathcal{C}_x$ from Section 1 as follows.

Definition 7 $\mathcal{C}_x \overset{\text{def}}{=} \{W_p \mid p \in W_x\}$.

One can think of the x in '$\mathcal{C}_x$' as representing a *uniform* grammar for generating (accepting) the languages in $\mathcal{C}_x$.

The following theorem removes some of the sting from the negative result (Osherson, Stob and Weinstein (1988)) motivating the present paper. It does this by relaxation of the criterion for successful learning.

Theorem 8 $(\exists$ *recursive* $f)(\forall x \mid W_x$ *is finite*$)$ $[\mathcal{C}_x \subseteq \mathbf{TxtFex}_{\text{card}(W_x)}(\mathbf{M}_{f(x)})]$.

Proof. Let $\text{match}(i, T[n]) = \min(\{n\} \cup (W_i^n \boldsymbol{\Delta} \text{content}(T[n])))$. Intuitively match finds the minimum point of disagreement between W_i and $T[n]$. Note that [i is a grammar for content$(T) \Leftrightarrow \lim_{n\to\infty} \text{match}(i, T[n]) = \infty$].

If $W_x^n = \emptyset$, then let $\mathbf{M}_{f(x)}(T[n]) =?$. Otherwise, let

$$\mathbf{M}_{f(x)}(T[n]) = \text{ the least } i \in W_x^n \text{ which maximizes match}(i, T[n]).$$

Fix x such that W_x is finite. Let $L \in \mathcal{C}_x$ and T be a text for L. Since, W_x is finite, it is easy to verify that, for all but finitely many n, $\mathbf{M}_{f(x)}(T[n])$ is in W_x and is a grammar for $L = \text{content}(T)$. The theorem follows. ∎

The above proof also demonstrates that

Corollary 9 $(\exists$ *recursive* $f)(\forall n, x \mid W_x$ *is finite and* $\max(\{\text{card}(\{i \in W_x \mid W_i = L\}) \mid L \in \mathcal{C}_x\}) = n)[\mathcal{C}_x \subseteq \mathbf{TxtFex}_n(\mathbf{M}_{f(x)})]$.

This nicely generalizes the special case from Osherson, Stob and Weinstein (1988) presented as Corollary 10 below.

Corollary 10 (Osherson, Stob and Weinstein (1988)) $(\exists$ recursive $f)(\forall x \mid W_x$ is finite $\wedge$ $(\forall$ distinct $i, j \in W_x)[W_i \neq W_j])[\mathcal{C}_x \subseteq \mathbf{TxtEx}(\mathbf{M}_{f(x)})]$.

That the bound in Theorem 8 above is tight is witnessed by the following strong lower bound

Theorem 11 *For all* $n \geq 1$, $\neg(\exists f \in \mathcal{LR})(\forall i_0, i_1, \ldots, i_n)$ $[\{W_{i_0}, W_{i_1}, \ldots W_{i_n}\} \subseteq \mathbf{TxtFex}_n^*(\mathbf{M}_{f(i_0, i_1, \ldots, i_n)})]$.

The $n = 1$ case of Theorem 11 just above, with 'limiting recursive' replaced by 'recursive' and with the $*$ removed, is just the negative result from Osherson, Stob and Weinstein (1988) that inspired the present paper, but, of course, Theorem 11 is stronger and more general. Theorem 11 follows by a direct $(n+1)$-ary recursion theorem argument and also quickly but indirectly from the fact from Case (1998) that $(\mathbf{TxtFex}_{n+1} - \mathbf{TxtFex}_n^*) \neq \emptyset$.

The following theorem implies that it is possible to synthesize algorithmically, from uniform grammars, behaviorally correct learners for classes which can be learned in one-shot (i.e., without any mind changes). As Theorem 14 further below shows, the cost of passing from *no* mind changes in the input classes to *infinitely* many in the *synthesized* learning machines is in some cases necessary (but see the second paragraph in Section 4 below).

Recall that $2* \overset{\text{def}}{=} *$.

Theorem 12 $(\exists$ *recursive* $f)(\forall x)[\mathcal{C}_x \in \mathbf{TxtEx}_0^a \Rightarrow \mathcal{C}_x \subseteq \mathbf{TxtBc}^{2a}(\mathbf{M}_{f(x)})]$.

Proof. Fix x such that $\mathcal{C}_x \in \mathbf{TxtEx}_0^a$.

Claim 13 $(\forall L \in \mathcal{C}_x)(\exists S \subseteq L)(\forall L' \mid S \subseteq L')[L' \in \mathcal{C}_x \Rightarrow L' =^{2a} L]$.

Proof. Suppose $\mathbf{M}$ $\mathbf{TxtEx}_0^a$–identifies $\mathcal{C}_x$. Suppose by way of contradiction that $L \in \mathcal{C}_x$ is such that $(\forall S \subseteq L)(\exists L' \mid S \subseteq L')[(L' \in \mathcal{C}_x) \ \wedge \ (L' \neq^{2a} L)]$. Let σ be a $\mathbf{TxtEx}^a$ locking sequence for $\mathbf{M}$ on L. Let $L' \in \mathcal{C}_x$ be such that $L' \supseteq \text{content}(\sigma)$ and $L' \neq^{2a} L$. Since $W_{\mathbf{M}(\sigma)} =^a L$, it follows that $W_{\mathbf{M}(\sigma)} \neq^a L'$. Therefore, $L' \notin \mathbf{TxtEx}_0^a(\mathbf{M})$, a contradiction. □(Claim 13)

Let g be a recursive function such that for all x and σ, $W_{g(x,\sigma)} = W_y$, where $y \in W_x$ is an integer satisfying $\text{content}(\sigma) \subseteq W_y$, if such an integer exists; otherwise, $W_{g(x,\sigma)} = \emptyset$. Let f be a recursive function such that for all x, $\mathbf{M}_{f(x)}(\sigma) = g(x,\sigma)$.

Fix $L \in \mathcal{C}_x$ and a text T for L. Let $S \subseteq L$ witness that L satisfies the above claim. Let n_0 be such that $\text{content}(T[n_0]) \supseteq S$. It follows using Claim 13 that for all $n \geq n_0$, $W_{\mathbf{M}_{f(x)}(T[n])} =^{2a} L$. ■ (Theorem 12)

It is open whether, for $a > 0$, the $2a$ in Theorem 12 just above is also a lower bound; however, we do know the following

Theorem 14 $\neg(\exists f \in \mathcal{LR})(\forall x)[\mathcal{C}_x \in \mathbf{TxtEx}_0 \Rightarrow \mathcal{C}_x \subseteq \mathbf{TxtFex}_*^*(\mathbf{M}_{f(x)})]$.

Proof. Suppose by way of contradiction that $f \in \mathcal{LR}$ is such that $(\forall x \mid \mathcal{C}_x \in \mathbf{TxtEx}_0)[\mathcal{C}_x \subseteq \mathbf{TxtFex}_*^*(\mathbf{M}_{f(x)})]$. Let g be a recursive function such that, for all x, $f(x) = \lim_{t\to\infty} g(x,t)$. By the Operator Recursion Theorem (Case (1974)), there exists a recursive function p such that the languages $W_{p(i)}$, $i \geq 0$, are defined in stages as follows. Initially, the $W_{p(i)}$'s are empty and σ_1 is the empty sequence. Go to stage 1.

```
Stage $s$
1. For each $i$ such that $1 \leq i \leq s$, enumerate $p(i)$ into $W_{p(0)}$ and let $W_{p(i)} = \text{content}(\sigma_s)$.
2. Let $x_s = 1 + \max(\{x \mid \text{content}(\sigma_s) \cap \{\langle x, i\rangle \mid i \geq 0\} \neq \emptyset\})$.
3. Dovetail steps 4, 5 and 6. If step 4 succeeds (before step 5, if ever) then go to step 7. If step 5 succeeds
      (before step 4, if ever) then go to step 8.
4. Search for $\sigma' \supset \sigma_s$ such that $\text{card}(\{\mathbf{M}_{g(p(0),s)}(\sigma'') \mid \sigma'' \subseteq \sigma'\}) \geq s$.
5. Search for a $s' > s$, such that $g(p(0), s) \neq g(p(0), s')$.
6. For each $i$ such that $1 \leq i \leq s$, enumerate more and more elements of $\{\langle x_s + i - 1, j\rangle \mid j \geq 0\}$ into $W_{p(i)}$.
7. Let $\sigma_{s+1} \supset \sigma'$ be the least sequence such that $\text{content}(\sigma_{s+1}) \supseteq \{j \mid j \leq 1 +$
      $\max(\bigcup_{1 \leq i \leq s} W_{p(i)}$ enumerated until now$)\}$.
   Go to stage $s + 1$.
8. Let $\sigma_{s+1} \supset \sigma_s$ be the least sequence such that $\text{content}(\sigma_{s+1}) \supseteq \{j \mid j \leq 1 +$
      $\max(\bigcup_{1 \leq i \leq s} W_{p(i)}$ enumerated until now$)\}$.
   Go to stage $s + 1$.
End stage $s$
```

We consider 2 cases.

Case 1: All stages terminate.

In this case, it can be verified that $W_{p(0)} = \{p(i) \mid i \geq 1\}$ and for all $i \geq 1$, $W_{p(i)} = N$. Thus, $\mathcal{C}_{p(0)} \in \mathbf{TxtEx}_0$. Since step 5 can succeed in only finitely many stages, step 4 succeeds in almost every stage. Thus, $\mathbf{M}_{f(p(0))}$ outputs infinitely many distinct grammars on $\bigcup_{s=1}^{\infty} \sigma_s$, a text for N. Hence $\mathcal{C}_{p(0)} \not\subseteq \mathbf{TxtFex}_*^*(\mathbf{M}_{f(p(0))})$.

Case 2: Stage s starts but does not terminate.

In this case, $W_{p(0)} = \{p(i) \mid 1 \leq i \leq s\}$. From step 6 it is clear that, for all i such that $1 \leq i \leq s$, $W_{p(i)} =^* \{\langle x_s + i - 1, j\rangle \mid j \geq 0\}$. Hence $(\forall i \leq s, j \leq s \mid i \neq j)[W_{p(i)} \neq^* W_{p(j)}]$. Also, since step 5 did not succeed in stage s, $f(p(0)) = g(p(0), s)$. Thus, since step 4 did not succeed in stage s, $\mathcal{C}_{p(0)} \not\subseteq \mathbf{TxtFex}_*^*(\mathbf{M}_{f(p(0))})$. Furthermore, it is clear that $\mathcal{C}_{p(0)}$ is finite and the languages in it are pairwise incomparable (by $\subset$). Hence, $\mathcal{C}_{p(0)} \in \mathbf{TxtEx}_0$. ▮

In most of the diagonalization results below, we will prove the theorem only for recursive f. Generalization to limiting recursive f can be obtained by using a trick similar to use of step 5 in the construction above.

It is interesting to generalize Theorem 12 above about synthesis from one-shot learnable $\mathcal{C}_x$'s to the case of *two*-shot learnable $\mathcal{C}_x$'s. The next two Theorems (Theorems 15 and 17) provide our best to date upper and lower bounds, respectively, for the two-shot cases. Other possibilities are open.

Theorem 15 $(\exists f \in \mathcal{R})(\forall x)[[\mathcal{C}_x \in \mathbf{TxtEx}_1 \ \wedge\ (\forall$ *distinct* $i, j \in W_x)[W_i \neq W_j]] \Rightarrow \mathcal{C}_x \subseteq \mathbf{TxtBc}^*(\mathbf{M}_{f(x)})]$.

PROOF. Suppose $\mathcal{C}_x \in \mathbf{TxtEx}_1$ and $(\forall i, j \in W_x)[W_i = W_j \Rightarrow i = j]$.

We say that $L \in \mathcal{C}_x$ satisfies property $\text{Prop}_0 \overset{\text{def}}{\Leftrightarrow}$ there exists a finite subset S_L of L such that $(\forall L' \in \mathcal{C}_x)[S_L \subseteq L' \Rightarrow L = L']$. Intuitively all L satisfying Prop_0 have a finite subset which uniquely determines L (from $\mathcal{C}_x$).

We say that $L \in \mathcal{C}_x$ satisfies property $\text{Prop}_1 \overset{\text{def}}{\Leftrightarrow}$ there exists a finite subset S_L of L such that $(\forall L' \in \mathcal{C}_x \mid L' \neq L)[S_L \subseteq L' \Rightarrow L'$ satisfies $\text{Prop}_0]$.

Note that if L satisfies Prop_0, then, trivially, L satisfies Prop_1.

Claim 16 *All languages $L \in \mathcal{C}_x$ satisfy Prop_1.*

PROOF. Suppose by way of contradiction $L \in \mathcal{C}_x$ does not satisfy Prop_1. Suppose $\mathbf{M}$ $\mathbf{TxtEx}_1$-identifies $\mathcal{C}_x$. Let σ be a $\mathbf{TxtEx}$-locking sequence for $\mathbf{M}$ on L. Let $L' \in \mathcal{C}_x$ be such that $L' \neq L$, L' does not satisfy Prop_0, and $\text{content}(\sigma) \subseteq L'$ (such an L' exists, since L does not satisfy Prop_1). Let σ' be an extension of σ such that σ' is a $\mathbf{TxtEx}$-locking sequence for $\mathbf{M}$ on L'. Let $L'' \in \mathcal{C}_x$ be such that $L'' \neq L'$ and $\text{content}(\sigma') \subseteq L''$. Such an L'' exists since L' does not satisfy Prop_0.

Now it is easy to see that $\mathbf{M}$ does not $\mathbf{TxtEx}_1$–identify at least one language in $\{L, L', L''\}$ since, if $\mathbf{M}$ $\mathbf{TxtEx}_1$ identifies L and L', then on a text for L'' which extends σ' it needs at least two mind changes. □ (Claim 16)

Thus, all languages in $\mathcal{C}_x$ satisfy Prop_1. We will utilize this property in algorithmically synthesizing a machine for $\mathbf{TxtBc}^*$–identifying $\mathcal{C}_x$. Let f be a recursive function such that for all x and σ, $W_{\mathbf{M}_{f(x)}(\sigma)}$ is defined in stages as follows:

```
Stage $s$
1. Let $n = |\sigma|$, $X = W_x^n$ and $Y = \{i \in X \mid \text{content}(\sigma) \subseteq W_i^s\}$.
2. For $i \in Y$, let $Z_i = W_i^s[n]$.
3. Let $p_1 = \min(Y)$ and let $p_2 = \min(\{i \in Y \mid Z_i \subset Z_{p_1}\})$. Intuitively $W_{p_2}$ is the first language in $\mathcal{C}_x$ which
     "looks like" a proper subset of $W_{p_1}$.
4. If $p_2\uparrow$ (recall that $\min(\emptyset)\uparrow$) then enumerate all the elements of $W_{p_1}^s$ into $W_{\mathbf{M}_{f(x)}(\sigma)}$ and go to stage $s+1$.
5. Otherwise,
   5.1 Let $S_{p_1} = \min(\{S' \mid (S' \subseteq W_{p_1}^s)$ and $(\forall i \in W_x^s \mid i \neq p_1)[S' \not\subseteq W_i^s]\})$.
```

```
        (* Note that according to our convention, the minimum over a collection of finite sets, is the set
           in the collection with the least canonical index. *)
  5.2   Let $S_{p_2} = \min(\{S' \mid (S' \subseteq W^s_{p_2})$ and $(\forall i \in W^s_x \mid i \neq p_2)[S' \not\subseteq W^s_i]\})$.
  5.3   If $S_{p_1} < S_{p_2}$, then enumerate $W^s_{p_2}$ into $W_{\mathbf{M}_{f(x)}(\sigma)}$. Otherwise enumerate $W^s_{p_1}$ into $W_{\mathbf{M}_{f(x)}(\sigma)}$.
        (* Here, if $S_{p_2}$ (respectively, $S_{p_1}$) is undefined but $S_{p_1}$ (respectively, $S_{p_2}$) is defined, then we assume
           $S_{p_1} < S_{p_2}$ (respectively, $S_{p_2} < S_{p_1}$). *)
  5.4   Go to stage $s + 1$
End stage $s$
```

Fix $L \in \mathcal{C}_x$ and T, a text for L. Let $i_1 \in W_x$ be the unique grammar such that $W_{i_1} = L$. We consider two cases.

Case 1: L satisfies Prop_0.

Let S_L witness that L satisfies Prop_0. Let $n_0 \geq \max(S_L)$ be the least value such that $S_L \subseteq \text{content}(T[n_0])$ and $i_1 \in W^{n_0}_x$. Fix n such that $n \geq n_0$. We claim that $W_{\mathbf{M}_{f(x)}(T[n])} =^* L$. Let s_0 be the least number such that $\text{content}(T[n]) \subseteq W^{s_0}_{i_1}$. It is clear that for all stages $s \geq s_0$, the set Y computed in step 1 (of stage s) is the singleton set $\{i_1\}$ (since L satisfies Prop_0). Thus, for all stages $s \geq s_0$, step 4 will be executed and $W_{\mathbf{M}_{f(x)}(T[n])} =^* L$.

Case 2: L satisfies Prop_1 but not Prop_0.

Let S_L witness that L satisfies Prop_1. Let n_0 be so large so that the following are satisfied:

(i) $S_L \subseteq \text{content}(T[n_0])$; (ii) for $i \leq i_1$, if $\text{content}(T[n_0]) \subseteq W_i$, then $L \subseteq W_i$; (iii) for $i < j < i_1$, if $W_i \not\supseteq W_j$, then $W_i[n_0] \not\supseteq W_j[n_0]$; (iv) $W^{n_0}_x \supseteq W_x[i_1]$.

Fix $n \geq n_0$. We claim that $W_{\mathbf{M}_{f(x)}(T[n])} =^* L$. Let s_0 be so large that $(\forall i \in W^n_x)[W^{s_0}_i \supseteq W_i[\max(\{n\} \cup \text{content}(T[n]))]]$. Thus, values of Y, Z_i, p_1, p_2 as defined in steps 1,2,3 do not change beyond stage s_0. Below, let Y, Z_i, p_1, p_2 be as computed in stage s_0 and beyond. There are two possibilities.

Suppose p_2 is not defined. In this case, for all stages $s' \geq s_0$, step 4 will be executed. Note that $i_1 \in Y$ and for any $i < i_1$, if $i \in Y$, then (by (ii) above) $L = W_{i_1} \subseteq W_i$. Thus $p_1 = i_1$ and $W_{\mathbf{M}_{f(x)}(T[n])} =^* W_{i_1} = L$.

Now suppose p_2 is defined. Then in stage $s \geq s_0$, step 4 will not be executed; instead, step 5 will be executed. Clearly, $i_1 \in \{p_1, p_2\}$ (otherwise $W_{i_1} \subset W_{p_2} \subset W_{p_1}$ — by condition (ii) and (iii) for choice of n_0 — which implies $\mathcal{C}_x \notin \mathbf{TxtEx}_1$). Suppose $i_1 = p_1$ (the argument is similar if $i_1 = p_2$). Since $S_L \subseteq W_{p_2}$, it follows that $L' = W_{p_2}$ satisfies Prop_0. Now since W_{p_2} satisfies Prop_0 but W_{p_1} does not, it follows that for all but finitely many stages s, $S_{p_1} > S_{p_2}$. It thus follows, due to step 5.3, that $W_{\mathbf{M}_{f(x)}(T[n])} =^* W_{p_1} = L$. ▮ (Theorem 15)

Theorem 17 $(\forall f \in \mathcal{LR})(\forall n)(\exists x \mid \mathcal{C}_x \in \mathbf{TxtEx}_1)[\mathcal{C}_x \not\subseteq \mathbf{RecTxtBc}^n(\mathbf{M}_{f(x)})]$.

Proof. We prove here the following restricted version of the theorem only.

$$(\forall f \in \mathcal{R})(\forall n)(\exists x \mid \mathcal{C}_x \in \mathbf{TxtEx}_1)[\mathcal{C}_x \not\subseteq \mathbf{RecTxtBc}^n(\mathbf{M}_{f(x)})].$$

The lift from $\mathcal{R}$ to $\mathcal{LR}$ is straightforward.

Fix $f \in \mathcal{R}$ and n. By the Operator Recursion Theorem there exists a 1-1 increasing recursive function p such that the languages $W_{p(i)}$, $i \geq 0$, are defined as follows. Enumerate $p(1)$ in $W_{p(0)}$. $W_{p(1)}$ will be a subset of **ODD**. The construction will use a set O. Initially, let $O = \emptyset$. Informally, O is the set of odd numbers we have decided to keep out of $W_{p(1)}$. Let σ_2 be the empty sequence. Go to stage 2.

```
Stage $s$
1.  Enumerate $p(s)$ into $W_{p(0)}$. Dovetail the execution of steps 2 and 3. If and when step 3 succeeds, go to
       step 4.
2.  Enumerate one-by-one, in increasing order, the elements of $\mathbf{ODD} - O$ into $W_{p(s)}$.
3.  Search for $\sigma_{s+1} \supset \sigma_s$ and set $P_s$ containing exactly $n+1$ distinct odd numbers such that $\text{content}(\sigma_{s+1}) \subseteq$
       $\mathbf{ODD} - O$ and $P_s \subseteq (W_{\mathbf{M}_{f(p(0))}(\sigma_{s+1})} - \text{content}(\sigma_{s+1}))$.
```

4.

```
    4.1  Enumerate content($\sigma_{s+1}$) into $W_{p(1)}$.
    4.2  Enumerate the (even) number $2s$ into $W_{p(s)}$.
    4.3  Let $O = O \cup P_s$, where $P_s$ is the set found above in step 3.
    4.5  Go to stage $s + 1$.
```

End stage s.

We consider two cases.

Case 1: Stage s starts but does not terminate.

In this case $W_{p(0)} = \{p(i) \mid 1 \leq i \leq s\}$. Observe that for each i such that $2 \leq i \leq s-1$, $2i$ is the only even number in $W_{p(i)}$. $W_{p(1)}$ is a finite subset of **ODD** and $W_{p(s)} =$ **ODD** $- O$, is an infinite subset of **ODD**. It is then easy to verify that $\mathcal{C}_{p(0)} \in \mathbf{TxtEx}_1$.

Let $T \supset \sigma_s$ be a recursive text for $W_{p(s)}$. It is clear that for all $(\sigma \mid \sigma_s \subset \sigma \subset T)[W_{\mathbf{M}_{f(p(0))}(\sigma)} \cap \mathbf{ODD}$ is finite]. Thus, **M** does not $\mathbf{RecTxtBc}^n$-identify $W_{p(s)}$.

Case 2: All stages terminate.

In this case, clearly, for all $i > 1$, $W_{p(i)}$ is finite and contains exactly one even number, namely $2i$. Also, $W_{p(1)}$ contains only odd numbers. Thus $\mathcal{C}_{p(0)}$ belongs to $\mathbf{TxtEx}_1$.

We claim that **M** does not $\mathbf{RecTxtBc}^n$-identify $W_{p(1)}$. Let $T = \bigcup_{s \geq 2} \sigma_s$. Clearly, T is a recursive text with content exactly $W_{p(1)}$. Consider any stage $s \geq 2$. It is clear by steps 3 and 4 that, for all s, card$(W_{\mathbf{M}_{f(p(0))}(\sigma_s)} - W_{p(1)}) \geq n + 1$. Thus, T is a recursive text witnessing that **M** does not $\mathbf{RecTxtBc}^n$–identify $W_{p(1)}$. ∎

3.2 Synthesizing From Uniform Decision Procedures

Definition 18

(1) We say that x is a *uniform decision procedure* for a class $\mathcal{L}$ of recursive languages $\overset{\text{def}}{\Leftrightarrow}$ $[\varphi_x \in \mathcal{R}_{0,1} \wedge \mathcal{L} = \{L \mid (\exists i)[\chi_L(\cdot) = \varphi_x(i, \cdot)]\}]$.

(2) Suppose x *is* a uniform decision procedure. Then $\mathcal{U}_x$ is (by definition) the class of languages for which x is a uniform decision procedure.

(3) $\mathcal{L}$ is a *uniformly decidable class of languages* $\overset{\text{def}}{\Leftrightarrow}$ $(\exists x$, a uniform decision procedure$)[\mathcal{L} = \mathcal{U}_x]$.

(4) When a fixed uniform decision procedure x is understood, we sometimes then write U_i for the language whose characteristic function is $\varphi_x(i, \cdot)$.

(5) By $U_i[s]$ we mean $\{x \in U_i \mid x \leq s\}$.

It is straightforward to show that uniform decision procedures and indexes of indexed families are intercompilable and, hence, that the uniformly decidable classes of languages formally defined just above (Definition 18) are exactly the indexed families of languages. In the formal statements and proofs of our results we will employ the terminology from Definition 18.

As noted in detail in Section 1, there has been considerable interest in the computational learning theory community in learnability, from positive data, of uniformly decidable classes of recursive languages (indexed families) (Angluin (1980b), Zeugmann and Lange (1995)).

Angluin (1980b) deals with so-called *exact* (Lange and Zeugmann (1993)) learning in which, for each learnable class, the programs learned derive naturally from the defining uniform decision procedure for that class.[8] Herein, we will synthesize learning machines whose hypothesis spaces *in many cases of necessity* go beyond hypothesis spaces naturally associated with the defining uniform decision procedures for the classes (Lange and Zeugmann (1993)).[9]

[8]Typically, in the literature one uses i as a "program" for the U_i from Definition 18 above. The s-m-n in the φ-system (Rogers (1967)) clearly yields corresponding φ-decision procedures or grammars as one wishes. See the next footnote.

[9]For example, if x is a uniform decision procedure (as in Definition 18 above), by s-m-n, there are computable d and g such that, for all i, $\varphi_{d(x,i)}(\cdot) = \varphi_x(i, \cdot)$, and $W_{g(x,i)} = \varphi^{-1}_{d(x,i)}(1)$. The hypothesis spaces $\{d(x,i) \mid i \in N\}$ and $\{g(x,i) \mid i \in N\}$ are each naturally associated with x, but, we will, in most cases, need to employ much more general spaces of programs.

The next two theorems (Theorems 19 and 20) deal with the special case where the uniformly decidable classes are *finite*. The first is an even more positive result than its analog for uniformly r.e. classes (Theorem 8 in Section 3.1 above). It's proof is straightforward, hence, omitted. The second shows the first is best possible.

Theorem 19 $(\forall n \geq 1)(\exists$ *recursive* $f)(\forall x \mid x$ *is a uniform decision procedure and* $\mathrm{card}(\mathcal{U}_x) = n)[\mathcal{U}_x \subseteq \mathbf{TxtEx}_{n-1}(\mathbf{M}_{f(x)})]$.

Theorem 20 *For all* $n \geq 1$, *there exists a uniformly decidable class* $\mathcal{L}$ *such that* $\mathrm{card}(L) = n+1$ *and* $\mathcal{L} \notin \mathbf{TxtEx}^*_{n-1}$.

PROOF. Let $L_j = \{\langle x, y\rangle \mid y \in N \ \wedge \ x \leq j\}$. It is easy to verify that $\{L_j \mid j \leq n\}$ is uniformly decidable but not in $\mathbf{TxtEx}^*_{n-1}$. ∎

The next two theorems (Theorems 21 and 24) concern synthesis from one-shot learnable uniformly decidable classes, and the first provides a much more positive result than its analog for uniformly r.e. classes (Theorem 12 in Section 3.1 above). The second shows the first is best possible and that the cost of passing from *no* mind changes in the input classes to *finitely* many in the *synthesized* learning machines is necessary.

Theorem 21 $(\forall a \in N \cup \{*\})(\exists f \in \mathcal{R})(\forall x \mid x$ *is a uniform decision procedure*$)[\mathcal{U}_x \in \mathbf{TxtEx}^a_0 \Rightarrow \mathcal{U}_x \subseteq \mathbf{TxtEx}^a(\mathbf{M}_{f(x)})]$.

PROOF. Fix a. Let $\mathcal{U}_x \in \mathbf{TxtEx}^a_0$ be given. We first show the following Claim.

Claim 22 *For all* $L \in \mathcal{U}_x$, *there exist finite sets* S_L, S^1_L *and* S^2_L *such that*
(a) $S_L \subseteq L$, *and*
(b) for the least i *such that* $S_L \subseteq U_i$:

$$(\forall L' \in \mathcal{U}_x \mid S_L \subseteq L')[(U_i - S^1_L) \cup S^2_L =^a L']$$

PROOF. Suppose $\mathcal{U}_x \in \mathbf{TxtEx}^a_0(\mathbf{M})$. Suppose $L \in \mathcal{U}_x$. Then there exists a sequence σ, such that $\mathrm{content}(\sigma) \subseteq L$ and $W_{\mathbf{M}(\sigma)} \neq ?$. This implies that for all $L' \in \mathcal{U}_x$ such that $\mathrm{content}(\sigma) \subseteq L'$, $W_{\mathbf{M}(\sigma)} =^a L'$. Let $S_L = \mathrm{content}(\sigma)$. Let i be the least number such that $S_L \subseteq U_i$. Let $S^1_L = U_i - W_{\mathbf{M}(\sigma)}$ and $S^2_L = W_{\mathbf{M}(\sigma)} - U_i$. It is easy to verify that (a) and (b) are satisfied. □

Claim 23 *Suppose* T *is a text for* $L \in \mathcal{U}_x$. *Let* S^1_T, S^2_T *and* S_T *be such that*
(a) $S_T \subseteq \mathrm{content}(T)$, *and*
(b) for the least i *such that* $S_T \subseteq U_i$:

$$(\forall L' \in \mathcal{U}_x \mid S_T \subseteq L')[(U_i - S^1_T) \cup S^2_T =^a L'].$$

Then, for the least i *such that* $S_T \subseteq U_i$:

$$[(U_i - S^1_T) \cup S^2_T] =^a L.$$

PROOF. Since, $S_T \subseteq L$, it follows from clause (b) that $[(U_i - S^1_T) \cup S^2_T] =^a L$. □

Since one can verify in the limit, for given S^1_T, S^2_T and S_T, whether clauses (a) and (b) in Claim 23 above are satisfied, one can algorithmically search for (some lexicographically least) such S^1_T, S^2_T, S_T. This is what gives us $\mathbf{M}_{f(x)}$ given below.

Let $\mathrm{Gram}(i, S_1, S_2)$ be a grammar obtained algorithmically from i and finite sets S_1 and S_2, such that $W_{\mathrm{Gram}(i,S_1,S_2)} = (U_i - S_1) \cup S_2$.

$\mathbf{M}_{f(x)}(T[n])$:

1. Let $S_n, S_n^1, S_n^2 \subseteq \{x \leq n\}$ be (lexicographically least (if any)) finite sets such that
   (a) $S_n \subseteq \text{content}(T[n])$
   (b) for the least i such that $S_n \subseteq U_i$:

$$(\forall j \leq n \mid S_n \subseteq U_j)[(U_i[n] - S_n^1) \cup S_n^2 =^a U_j[n]]$$

2. If no such S_n, S_n^1, S_n^2 is found in the search above, then output 0. Else let S_n, S_n^1, S_n^2 be the lexicographically least such set. For the least i such that $S_n \subseteq U_i$, output $\text{Gram}(i, S_n^1, S_n^2)$.

End $\mathbf{M}_{f(x)}$

Using Claim 22, it is easy to verify that, for any T for $L \in \mathcal{U}_x$, S_n, S_n^1, S_n^2 as found in step 1 above converges to S_T, S_T^1, S_T^2, which satisfy (a) and (b) in Claim 23. It thus follows that $\mathbf{M}_{f(x)}$ $\mathbf{TxtEx}^a$-identifies $\mathcal{U}_x$. ∎

Theorem 24 $(\forall n)\neg(\exists f \in \mathcal{LR})(\forall x \mid x \textit{ is a uniform decision procedure})[\mathcal{U}_x \in \mathbf{TxtEx}_0 \Rightarrow \mathcal{U}_x \subseteq \mathbf{TxtEx}_n^*(\mathbf{M}_{f(x)})]$.

Proof. Fix n. For simplicity of presentation, we give the proof only for recursive f. The proof can be straightforwardly generalized to limiting recursive f. By implicit use of the recursion theorem there exists an x such that $\mathcal{U}_x$ may be described as follows.

Let σ_0 be a sequence, if any, such that $\mathbf{M}_{f(x)}(\sigma_0) \neq ?$. For $i \leq n$, if σ_i is defined, then try to define σ_{i+1} as follows: let σ_{i+1} be a sequence, if any, such that $\sigma_i \subseteq \sigma_{i+1}$ and $\mathbf{M}_{f(x)}(\sigma_i) \neq \mathbf{M}_{f(x)}(\sigma_{i+1})$.

Let $L_0 = \{\langle 0, y\rangle \mid y \in N\}$. For i such that σ_i is defined in the process above, define L_{2i+1}, L_{2i+2} as follows: $L_{2i+1} = \text{content}(\sigma_i) \cup \{\langle 2i+1, y\rangle \mid y \in N\}$. $L_{2i+2} = \text{content}(\sigma_i) \cup \{\langle 2i+2, y\rangle \mid y \in N\}$.

Let $\mathcal{U}_x = \{L_0\} \cup \{L_{2i+1}, L_{2i+2} \mid \sigma_i \text{ is defined in the process above}\}$. Since $\mathcal{U}_x$ is finite and all languages in $\mathcal{U}_x$ are pairwise incomparable (by $\subset$), it follows that $\mathcal{U}_x \in \mathbf{TxtEx}_0$. We claim that $\mathcal{U}_x \not\subseteq \mathbf{TxtEx}_n^*(\mathbf{M}_{f(x)})$.

If σ_{n+1} is defined then, $\mathbf{M}_{f(x)}$ makes at least $n+1$ mind changes on σ_{n+1} and thus does not $\mathbf{TxtEx}_n^*$-identify $L_{2n+3}, L_{2n+4} \in \mathcal{U}_x$.

If σ_0 is not defined, then $L_0 \in \mathcal{U}_x$, but $\mathbf{M}_{f(x)}$ does not $\mathbf{TxtEx}^*$-identify L_0.

If, for some $i \leq n$, σ_i is defined but σ_{i+1} is not defined, then: $L_{2i+1}, L_{2i+2} \in \mathcal{U}_x$, but $\mathbf{M}_{f(x)}$ does not $\mathbf{TxtEx}^*$-identify at least one of L_{2i+1}, L_{2i+2} (since $\text{content}(\sigma_i) \subseteq L_{2i+1} \cap L_{2i+2}$, $L_{2i+1} \neq^* L_{2i+2}$ and $\mathbf{M}_{f(x)}$ on any extension of σ_i outputs $\mathbf{M}_{f(x)}(\sigma_i)$).

The theorem follows. ∎

Next we present our two Main Corollaries (Corollaries 25 and 26) which *completely characterize* the uniformly decidable classes in $\mathbf{TxtBc}^a$ and are easy consequences of Lemmas 27 and 29 following them.[10] The first corollary is a very important case of the second which we've separated out for special attention.

As we noted in Section 1 above, Angluin (1980b) completely characterized the uniformly decidable classes in **TxtEx**.[11] Essentially she showed that, for any fixed uniform decision procedure x, $\mathcal{U}_x \in \mathbf{TxtEx} \Leftrightarrow$ Condition 1 holds, where Condition 1 states:[12]

There is an *r.e.* sequence of (r.e. indices of) finite sets $S_0, S_1, \ldots$ (called *tell tales*) such that,

$$(\forall i)[S_i \subseteq U_i \ \wedge \ (\forall j \mid S_i \subseteq U_j)[U_j \not\subset U_i]]. \tag{1}$$

[10] It seems pedagogically useful to present the results in this order.

[11] Mukouchi (1992), Lange and Zeugmann (1992) characterized the uniformly decidable classes in $\mathbf{TxtEx}_0$. de Jongh and Kanazawa (1996) surprisingly characterizes the *r.e.* classes in **TxtEx** and presents other interesting results.

[12] Recall that the U_i's are defined in Definition 18 above.

As noted in Section 1 above, she also considered a Condition 2 just like Condition 1 *except* that the sequence of finite sets (tell tales) is *not* required to be r.e. and showed that Condition 2 is *not* sufficient. Our characterization of uniformly decidable classes in **TxtBc** is, surprisingly, just Angluin's Condition 2! As mentioned above, it is referred to as the *subset principle*, a necessary condition preventing overgeneralization in learning from positive data (cf. e.g. Angluin (1980b), Berwick (1985), Zeugmann, Lange and Kapur (1995), Kapur and Bilardi (1992), Case (1998)).[13]

Corollary 25 $\mathcal{U}_x \in \mathbf{TxtBc} \Leftrightarrow (\forall U \in \mathcal{U}_x)(\exists S \subseteq U \mid S \textit{ is finite})(\forall U' \in \mathcal{U}_x \mid S \subseteq U')[U' \not\subset U]$.

It is surprising and important that the subset principle alone (Angluin's Condition 2) *without the added constructivity conditions of Angluin's Condition 1* characterizes the uniformly decidable classes $\mathcal{U}_x \in \mathbf{TxtBc}$.

Osherson, Stob and Weinstein (1986a) notes that a class of r.e. languages $\mathcal{U}$ can be learned in the limit from positive data *by a not necessarily algorithmic procedure* iff Angluin's Condition 2 holds for $\mathcal{U}$. Hence, Corollary 25 together with this observation entails that *for uniformly decidable classes* $\mathcal{U}$, $\mathcal{U}$ can be learned in the limit from positive data *by a not necessarily algorithmic procedure* iff $\mathcal{U} \in \mathbf{TxtBc}$. Therefore, for uniformly decidable classes, algorithmicity of the learning procedure doesn't matter for behaviorally correct identification! It is open whether there are other types of classes $\mathcal{U}$ (besides uniformly decidable) for which algorithmicity of the learning procedure doesn't matter for behaviorally correct identification.

Suppose x is a uniform decision procedure. Corollary 25, immediately above also provides the following characterization.

$$\mathcal{U}_x \in (\mathbf{TxtBc} - \mathbf{TxtEx}) \Leftrightarrow \mathcal{U}_x \text{ satisfies Condition 2 but not Condition 1.}$$

Hence, since Angluin (1980b) provided an example $\mathcal{U}_x$ satisfying Condition 2 and not Condition 1, her example is a uniformly decidable class witnessing that $(\mathbf{TxtBc} - \mathbf{TxtEx}) \neq \emptyset$.

Our characterization for $\mathbf{TxtBc}^a$ is next.[14]

Corollary 26 $\mathcal{U}_x \in \mathbf{TxtBc}^a \Leftrightarrow (\forall U \in \mathcal{U}_x)(\exists S \subseteq U \mid S \textit{ is finite})(\forall U' \mid S \subseteq U' \in \mathcal{U}_x \ \wedge \ U' \subseteq U)[\mathrm{card}(U - U') \leq 2a]$.

As we will see, our Main Theorem (Theorem 30) below is an immediate consequence of Lemmas 27 and 29 to follow.

Lemma 27 *Suppose* $\mathcal{U}_x \in \mathbf{TxtBc}^a$*. Then* $(\forall L \in \mathcal{U}_x)(\exists X_L \subseteq L \mid X_L \textit{ is finite})(\forall L' \in \mathcal{U}_x \mid X_L \subseteq L')[L' \not\subseteq L \ \vee \ \mathrm{card}(L - L') \leq 2a]$.

Proof. Suppose $\mathcal{U}_x \in \mathbf{TxtBc}^a(\mathbf{M})$. Suppose $L \in \mathcal{U}_x$. Then there exists a $\mathbf{TxtBc}^a$-locking sequence σ for $\mathbf{M}$ on L. Let $X_L = \mathrm{content}(\sigma)$. Note that $W_{\mathbf{M}(\sigma)} =^a L$. Now, for any $L' \in \mathcal{U}_x$ such that $X_L \subseteq L' \subseteq L$, we must have $W_{\mathbf{M}(\sigma)} =^a L'$ (otherwise, $\mathbf{M}$ does not $\mathbf{TxtBc}^a$-identify L'). It follows that $L =^{2a} L'$. ∎

Before presenting our Main Lemma (Lemma 29), we present a slightly weaker version of it: in Lemma 29, the $\mathbf{TxtBc}^{2a}$ in Lemma 28, is replaced by just $\mathbf{TxtBc}^a$.

Lemma 28 *There exists an* $f \in \mathcal{R}$ *such that the following is satisfied.*

Suppose x *is a uniform decision procedure. Further suppose* $(\forall L \in \mathcal{U}_x)(\exists X_L \subseteq L \mid X_L \textit{ is finite})(\forall L' \in \mathcal{U}_x \mid X_L \subseteq L')[L' \not\subseteq L \ \vee \ \mathrm{card}(L - L') \leq 2a]$*. Then,* $[\mathcal{U}_x \subseteq \mathbf{TxtBc}^{2a}(\mathbf{M}_{f(x)})]$.

Proof. Suppose x is a uniform decision procedure satisfying the hypothesis of the theorem. We describe the construction of $\mathbf{M}_{f(x)}$. It is easy to see that the construction is algorithmic in x.

$\mathbf{M}_{f(x)}(T[n]) = \mathrm{Proc}(T[n])$, where $W_{\mathrm{Proc}(T[n])}$ is defined as follows.

[13] See Kapur, Lust, Harbert and Martohardjono (1993), Wexler (1993) for discussion regarding the possible connection between this subset principle and a more traditionally linguistically oriented one in Manzini and Wexler (1987).

[14] The characterizing condition is a variant of Angluin's Condition 2.

We also have a variant of Angluin's characterization above, but for $\mathbf{TxtEx}^*$ in place of $\mathbf{TxtEx}$, which characterization is just like hers *except* that (1) above is replaced by

$$(\forall i)[S_i \subseteq U_i \ \wedge \ (\forall j \mid S_i \subseteq U_j \subseteq U_i)[U_j =^* U_i]]. \quad (2)$$

```
$W_{\mathrm{Proc}(T[n])}$
1. Let $P_n = \{i \leq n \mid \mathrm{content}(T[n]) \subseteq U_i\}$.
2. Go to stage 0.
Begin Stage $s$
3.  Let $DelS_n^s = \{i \in P_n \mid (\exists i' \in P_n \mid i' < i)[U_i[s] \not\subseteq U_{i'}[s]]\}$.
    (* Note that $DelS_n^s \subseteq DelS_n^{s+1}$. Intuitively $DelS_n^s$ consists of grammars we want to delete from $P_n$,
      since they seem to be bad (see analysis below) *).
4.  Let $S_n^s = P_n - DelS_n^s$
    (* Note that $(\forall i, i' \in S_n^s \mid i' < i)[U_i[s] \subseteq U_{i'}[s]]$ *)
5.  Let $i_n^s = \max(S_n^s)$. Enumerate $U_{i_n^s}[s]$.
6.  Go to stage $s+1$.
End stage $s$
End $W_{\mathrm{Proc}(T[n])}$
```

Suppose $L \in \mathcal{U}_x$ and T is a text for L. We claim that for all but finitely many n, $W_{\mathrm{Proc}(T[n])} =^{2a} L$. This will prove the theorem. Let X_L be as given in the hypothesis. It follows that for all $L' \in \mathcal{U}_x$, $[X_L \subseteq L'] \Rightarrow [L' \not\subseteq L \ \vee \ L' =^{2a} L]$. Let j be the minimal number such that $U_j = L$.

Let n be large enough so that,

(i) For all $i < j$ such that $U_i \not\supseteq U_j$, $\mathrm{content}(T[n]) \not\subseteq U_i$.

(ii) $j < n$.

(iii) $X_L \subseteq \mathrm{content}(T[n])$.

We claim that $W_{\mathrm{Proc}(T[n])} =^{2a} U_j = L$. Let $P_n, S_n^s, DelS_n^s, i_n^s$ be as defined in $Proc(T[n])$ above. They satisfy the following properties:

(a) $j \in P_n$.

(b) $(\forall i < j)[i \in P_n \Rightarrow U_i \supset L]$.

(c) $(\forall i > j)[i \in P_n \Rightarrow [U_i \not\subseteq L \ \vee \ U_i =^{2a} L]]$.

(d) for all s: $j \notin DelS_n^s$; thus $j \in S_n^s$.

It immediately follows from (d) above and the comment after step (4) that $W_{\mathrm{Proc}(T[n])} \subseteq U_j$.

Now note that, since $DelS_n^s \subseteq DelS_n^{s+1}$, we have $S_n^s \supseteq S_n^{s+1}$. Thus $\lim_{s\to\infty} i_n^s$ is defined. Let this limiting value be i_n. It follows that $U_{i_n} \subseteq W_{Proc(T[n])} \subseteq U_j = L$. Now since, $U_{i_n} \subseteq U_j$, property (c) above implies that $U_{i_n} =^{2a} U_j$. Thus, $W_{Proc(T[n])} =^{2a} L$. It follows that $\mathbf{M}_{f(x)}$ $\mathbf{TxtBc}^{2a}$-identifies L. ∎

Lemma 29 *There exists a recursive f satisfying the following. Suppose x is a uniform decision procedure. Further suppose $(\forall L \in \mathcal{U}_x)(\exists X_L \subseteq L \mid X_L$ is finite$)(\forall L' \in \mathcal{U}_x \mid X_L \subseteq L')[L' \not\subseteq L \ \vee \ \mathrm{card}(L - L') \leq 2a]$. Then, $[\mathcal{U}_x \subseteq \mathbf{TxtBc}^a(\mathbf{M}_{f(x)})]$.*

Proof. The proof of the lemma is a careful modification of the proof of Lemma 28 to reduce the errors. The weaker version, Lemma 28, above suffices to obtain the $a \in \{0, *\}$ cases. So suppose $a \in N$.

Consider the following machine $\mathbf{M}_{f(x)}$ for $\mathcal{U}_x$. $\mathbf{M}_{f(x)}(T[n]) = Proc(T[n])$, where $W_{Proc(T[n])}$ is as defined below.

```
$W_{Proc(T[n])}$.
Let $P_n = \{i \leq n \mid \mathrm{content}(T[n]) \subseteq U_i\}$.
Go to stage $n$ (* we start from stage $n$ just for ease of writing the proof. *)
Begin stage $s$
1.  Let $DelS_n^s = \{i \in P_n \mid (\exists i' \in P_n \mid i' < i)[U_i[s] \not\subseteq U_{i'}[s]]\}$.
    (* Note that $DelS_n^s \subseteq DelS_n^{s+1}$ *).
```

(* Intuitively $DelS_n^s$ consists of grammars we want to delete from P_n, since they seem to be bad (see analysis below) *).

2. Let $S_n^s = P_n - DelS_n^s$.

   (* Note that after the above deletion, we have the following property: $(\forall i > i' \in S_n^s)[U_i[s] \subseteq U_{i'}[s]]$. Thus the members of S_n^s form a reverse subset chain (for elements $\leq s$) *)

3. Let $k_n^s = \max(S_n^s)$. (* Note: k_n^s is a non-increasing function of s *).

4. Let $\text{Cancel}_n^s = \{i \in S_n^s \mid \text{card}(U_i[n] - U_{k_n^s}[n]) > 2a\}$.

   (* Note: We form Cancel_n^s looking at enumeration of elements $\leq n$. This Cancel set may become smaller as stages go on! Intuitively Cancel just tries to delete sets which are too big compared to the input. *)

5. Let $Q_n^s = S_n^s - \text{Cancel}_n^s$.

   Let $i_n^s = \min(Q_n^s)$.

   (* Note: since k_n^s was a non-increasing function of s; it is easy to see that $\max(Cancel_n^s)$ will be a non-increasing function of s. It does not necessarily follow that i_n^s is non-increasing function of s. However it is nearly so — what "nearly" means will be clearer in the proof after the construction*).

   Let $D_n^s = U_{i_n^s}[n] - U_{k_n^s}[n]$. Let $A_n^s \subseteq D_n^s$ be a set of $\min(\text{card}(D_n^s), a)$ elements such that the following property is satisfied:

   if $z_1 \in A_n^s$ and $z_2 \in D_n^s - A_n^s$, then

   (A) $\max(\{i \in Q_n^s \mid z_2 \in U_i\}) < \max(\{i \in Q_n^s \mid z_1 \in U_i\})$ OR

   (B) $\max(\{i \in Q_n^s \mid z_2 \in U_i\}) = \max(\{i \in Q_n^s \mid z_1 \in U_i\})$ and $[z_2 \in A_n^{s-1} \Rightarrow z_1 \in A_n^{s-1}]$.

   (* Intuitively we select an A_n^s in the following form: select the a elements of A_n^s so that elements which are enumerated by more decision procedures in Q_n^s get priority. Breaking of ties is done in a manner consistent with earlier priority settings. *)

   Enumerate $(U_{i_n^s}[s] - D_n^s) \cup A_n^s$.

6. Go to stage $s + 1$.

End stage s

End

Now fix $L \in \mathcal{U}_x$. Let j be the minimal number such that $U_j = L$.

Let X_L be as given in the hypothesis of the Lemma. Fix a text T for L. Assume that $n > j$ is so large that the following properties (a) to (d) are satisfied. For these big enough n, we will claim below that $W_{Proc(T[n])} =^a L$.

(a) $X_L \subseteq \text{content}(T[n])$.

(b) $(\forall i < j)[U_i \not\supseteq L \Rightarrow \text{content}(T[n]) \not\subseteq U_i]$.

(c) $(\forall i, i' \in W_x \mid i < i' < j)[U_{i'} \not\subseteq U_i \Rightarrow U_{i'}[n] \not\subseteq U_i[n]]$.

Intuitively, (c) ensures that if $i < j$, is in $\bigcup_s DelS_n^s$, then it is in $DelS_n^n$ (note that we started in stage n).

(d) $(\forall i < j \mid U_i \supseteq L)[\text{card}(\{y < n \mid y \in U_i - L\}) \geq \min(\{2a + 1, \text{card}(U_i - L)\})]$.

Intuitively (d) says that either all elements of $U_i - L$ are below n, or there are at least $2a + 1$ elements of $U_i - L$ below n; This second part ensures that if U_i is too big then i will be in Cancel_n^s for every stage s. The earlier part makes sure that all the elements which are in $U_i - L$ have been already noticed and thus would not be enumerated in step 5 except as part of A_n^s.

For all of following we assume that $n > j$ is big enough so that (a) to (d) are satisfied. We will consider what $Proc(T[n])$ enumerates. So let all the variables below be as in $Proc(T[n])$.

Let $Big = \{i < j \mid U_i \supseteq L \ \wedge \ \text{card}(U_i - L) \leq 2a \ \wedge \ (\forall i' \mid i' < i < j \ \wedge \ U_{i'} \supseteq L)[U_i \subseteq U_{i'}]\}$.

Note that for all $i < j$, if $i \notin Big$, then $i \notin Q_n^s$ for any s (each $i < j$ in $W_x - Big$ would be either in $DelS_n^n$ (note that we started at stage n) or in Cancel_n^s for each s). So $i < j$ which are not in Big are never in Q_n^s.

Note the following properties of $DelS_n^s$ and Cancel_n^s.

(e) $j \notin DelS_n^s$ for all s (thus $j \in S_n^s$ for every s). Note that for finitely many stages s, j maybe in Cancel_n^s but that will not hurt us.

(f) $DelS_n^s \subseteq DelS_n^{s+1}$ (and thus $S_n^{s+1} \subseteq S_n^s$).

(g) Since elements of S_n^s form a reverse subset chain (for elements $\leq s$), members of Cancel_n^s have the initial segment property within S_n^s, i.e., if $i, i' \in S_n^s$, $i < i'$ and $i' \in \text{Cancel}_n^s$ then $i \in \text{Cancel}_n^s$.

(h) $\text{Cancel}_n^{s+1} \subseteq \text{Cancel}_n^s$ (this follows since k_n^s is a non-increasing function of s). Thus $\max(\text{Cancel}_n^s)$ is non-increasing function of s.

Let $D = \{y \mid (\exists i \in Big)[y \in U_i - L]\}$.

What we will first show is that $Proc(T[n])$ enumerates a subset of $L \cup D$. We will then show that what is enumerated within D has cardinality appropriately bounded; this will complete the proof.

Now consider any stage s:

If in stage s, $i_n^s \geq j$, then clearly, whatever is enumerated in stage s is contained in L (since $U_{i_n^s}[s] \subseteq U_j[s]$; otherwise i_n^s would be in $DelS_n^s$). If $i_n^s < j$, then clearly, $U_{i_n^s} \subseteq L \cup D$ since i_n^s must be in Big.

We now consider what exactly is the difference between L and the set enumerated by $Proc(T[n])$. Clearly, $DelS_n^s$, Cancel_n^s and D_n^s, i_n^s, k_n^s, A_n^s etc must achieve a limiting value (as s goes to ∞). Let the limiting values of these variable be $DelS_n$, Cancel_n, D_n, i_n, k_n, A_n etc.

Consider the first stage when i_n^s achieves a value $\leq j$. (There must exists such a stage: note that $X_L \subseteq U_{k_n} \subseteq L$; thus, by hypothesis, $\text{card}(L - U_{k_n}) \leq 2a$). Since $\max(\text{Cancel}_n^s)$ is a non-increasing function of s, and $S_n^s \cap \{i \leq j\} = S_n \cap \{i \leq j\}$ for all s, we have that, if i_n^s achieves a value $\leq j$, it is non-increasing from that point onwards (this is what we meant by nearly non-increasing in step 5 of the construction above). Furthermore, since $k_n^s \geq j$, for all s, it follows that $D \cap D_n^s \subseteq D \cap D_n^{s+1}$. (To see this note that all elements of D are $\leq n$. Moreover, the elements of Q_n^s form a reverse subset chain, with respect to elements $\leq n$. Thus since $\min(\{j, i_n^{s+1}\}) \leq i_n^s$, any elements of D which were in D_n^s would also be in D_n^{s+1}). Thus $A_n^s \cap D \subseteq A_n^{s+1}$ (from the way A_n^s was chosen, since if $i_n^s \geq j$, then $D \cap A_n^s$ is empty. On the other hand if $i_n^s < j$, then $i_n^s \geq i_n^{s+1}$ (due to non-increasing property of i_n^s once it becomes $\leq j$) and thus based on the subset chain property, the above holds (the priority ordering among the elements of D_n does not change!)).

We now have following property:

$W_{Proc(T[n])} - L \subseteq A_n$ (due to the fact that $D \cap A_n^{s+1} \subseteq A_n$). Also $L - W_{Proc(T[n])} \subseteq D_n - A_n$, since all other elements of L are enumerated in the stages beyond the point where A_n, D_n and i_n get their limiting values. Now suppose $A_n \subseteq L$. In this case, the difference between L and $W_{Proc(T[n])}$ is subset of $D_n - A_n$ which is of size $\leq a$.

If $A_n \not\subseteq L$. Then by the priority ordering selected for chosing elements of A_n, we know that elements of $D_n - A_n$ do not belong to L. Thus the difference in L and $W_{Proc(T[n])}$ is a subset of A_n which is of size $\leq a$. ∎

Next we have our Main Theorem (Theorem 30) which follows immediately from Lemmas 27 and 29 above. It says that we *can* synthesize, from uniform decision procedures for classes in $\mathbf{TxtBc}^a$, $\mathbf{TxtBc}^a$-learning machines! Hence, in passing from learnable uniformly decidable classes to algorithmically synthesized learning machines for them, we get a fixed point, for each a, at $\mathbf{TxtBc}^a$-identification!

Theorem 30 $(\forall a \in N \cup \{*\})(\exists f \in \mathcal{R})(\forall x \mid$ *x is a uniform decision procedure*$)[\mathcal{U}_x \in \mathbf{TxtBc}^a \Rightarrow \mathcal{U}_x \subseteq \mathbf{TxtBc}^a(\mathbf{M}_{f(x)})]$.

The next and last theorem of this section contrasts nicely with Theorems 21 and 30 above. It also shows that the cost of passing from *one* mind change in the input classes to *infinitely* many in the *synthesized* learning machines is necessary. This is so, as in our other lower bound results above, even if we employed the stronger limiting recursive procedures for synthesis of learning machines from algorithmic descriptions of the class to be learned!

Theorem 31 $\neg(\exists f \in \mathcal{LR})(\forall x \mid$ *x is a uniform decision procedure*$)[\mathcal{U}_x \in \mathbf{TxtEx}_1 \Rightarrow \mathcal{U}_x \subseteq \mathbf{TxtFex}_*^*(\mathbf{M}_{f(x)})]$.

PROOF. We prove a simpler version of the theorem:

$$\neg(\exists f \in \mathcal{R})(\forall x \mid x \text{ is a uniform decision procedure})$$
$$[\mathcal{U}_x \in \mathbf{TxtEx}_1 \Rightarrow \mathcal{U}_x \subseteq \mathbf{TxtEx}^*(\mathbf{M}_{f(x)})].$$

The proof can be straightforwardly generalized to take care of $\mathbf{TxtFex}^*_*$ and limiting recursive f.

Suppose by way of contradiction otherwise. Let f be given. By the Operator Recursion Theorem there exists a 1-1 increasing, p such that the following holds.

We let $\mathcal{L} = \{L \mid (\exists i \geq 1)[\chi_L = \varphi_{p(i)}]\}$.

It will be easily seen that, for $i > 0$, $\varphi_{p(i)}$ is either a characteristic function or an empty function. Hence, it follows that a uniform decision procedure for $\mathcal{L}$ (defined just above) exists and can be found algorithmically. Let $p(0)$ be uniform decision procedure for $\mathcal{L}$.

We note that all $\varphi_{p(i)}$, $i \geq 1$, *which are considered in the construction below* will be characteristic functions.

If $\varphi_{p(i)}$ is a characteristic function, we will abuse notation slightly and refer to the language for which it's a characteristic function as $U_{p(i)}$.

We will have that $\mathcal{U}_{p(0)} \in \mathbf{TxtEx}_1$. Thus $\mathbf{M}_{f(p(0))}$ $\mathbf{TxtEx}^*$ identifies $\mathcal{U}_{p(0)}$. We let $U_{p(1)} = \mathbf{ODD}$.

Furthermore, for all $i > 1$, $U_{p(i)}$ will be finite. In addition the construction will ensure that at least one of the following properties (A) and (B) is satisfied.

(A) For $i > 1$, $U_{p(i)}$ contains exactly one even number. Moreover, for all $i > i' > 1$, $U_{p(i)} \cap U_{p(i')} \cap \mathbf{EVEN} = \emptyset$.

(B) There exists a $j > 1$ such that, B.1, B.2 and B.3 are satisfied.

(B.1) for all $i > j$, $\varphi_{p(i)}(0)\uparrow$.

(B.2) for all i such that $1 < i < j$, $U_{p(i)}$ contains exactly one even number. Moreover, for all i, i' such that $1 < i < i' < j$: $W_{p(i)} \cap W_{p(i')} \cap \mathbf{EVEN} = \emptyset$.

(B.3) $W_{p(j)}$ is finite, does not contain any even number, and for all i such that $1 < i < j$: $W_{p(j)} \not\subseteq W_{p(i)}$.

Note that the above properties imply that $\mathcal{C}_{p(0)} \in \mathbf{TxtEx}_1$. In case (A) clearly, a machine can first output a grammar for $\mathbf{ODD}$. Then if it sees an even number it can output a grammar for the corresponding finite set using appropriate $p(i)$. In case (B) $\mathcal{C}_{p(0)}$ is finite and machine can $\mathbf{TxtEx}_1$-identify by first waiting until it gets an even number or sees $W_{p(j)}$ in the input and then output the corresponding grammar. The machine now needs to change its conjecture only if the input is for $\mathbf{ODD}$, requiring at most 1 mind change.

We now proceed to define $W_{p(i)}$ for $i > 1$. Let σ_0 be a sequence such that content$(\sigma) = \{1\}$. Go to stage 0.

```
Stage $s$
0.  Dovetail steps 1 and 2 until, if ever, step 2 succeeds. If and when step 2 succeeds, go to step 3.
1.  Start defining $p(s+2)$ so that it will be a characteristic function for content$(\sigma_s)$, unless step 2 below
      succeeds.
2.  Search for an extension $\sigma$ of $\sigma_s$ such that content$(\sigma) \subseteq \mathbf{ODD}$ and $\mathbf{M}_{f(p(0))}(\sigma) \neq \mathbf{M}_{f(p(0))}(\sigma_s)$.
3.  If and when such a $\sigma$ (in step 2) is found,
        Let $e$ be an even number large enough so that $\varphi_{p(s+2)}(e)$ has not yet been defined and $e$ is bigger
          than all the even numbers considered in previous stages.
        Let $\varphi_{p(s+2)}$ be characteristic function for content$(\sigma_s) \cup \{e\}$.
        Let $\sigma_{s+1}$ be an extension of $\sigma$ such that content$(\sigma_{s+1}) \supset$ content$(\sigma_s) \cup \{2x+1 \mid x \leq$
          max(content$(\sigma_s)$)$\}$.
4.  Go to stage $s+1$.
End stage $s$
```

Now consider the following cases.

Case 1: All stages terminate.

In this case clearly, for all $i > 1$, $U_{p(i)}$ is finite, and $p(i)$ satisfy the property (A) above. Moreover, $\mathbf{M}_{f(p(0))}$ makes infinitely many mind changes on $\bigcup_s \sigma_s$ which is a text for $\mathbf{ODD}$.

Case 2: stage s starts but does not terminate.

In this case clearly, property (B) above is satisfied with $j = s+2$. Moreover, $\mathbf{M}_{f(p(0))}$ can $\mathbf{TxtEx}^*$-identify at most one of $U_{p(s+2)}$ and $U_{p(1)} = \mathbf{ODD}$, since step 2 does not succeed.

This proves the theorem. ∎

N.B. Angluin's proof of her characterization (by Condition 1) of uniformly decidable classes in **TxtEx** essentially provides an algorithm for transforming *two* things into a learning machine which **TxtEx**-identifies $\mathcal{L}$:[15]

(1) a uniform decision procedure for a class of recursive languages $\mathcal{L}$ that is **TxtEx**-identifiable *and*

(2) *a program for generating an associated r.e. sequence of tell tale sets* $S_0, S_1, \ldots$ *as featured in Equation (1) above.*

The *additional input information* of a program for generating the tell tales therefore makes a huge difference in the mind change complexity of the synthesized learning machine!

4 Future Directions

Bārzdiņš and Freivalds (1972) first considered improvements of archetypal enumeration techniques, involving a majority vote strategy which has better mind-change complexity. It would be interesting to look into variants of our algorithms above for synthesizing learning machines with improved mind-change complexity at least for interesting special cases.

Corollary 9 and 10 above suggest to us ones exploring the relevance to our paper's topics of r.e. classes of languages which can be enumerated with $\leq n+1$ duplications but not with $\leq n$ (Pour-El and Howard (1964), Pour-El and Putnam. (1965)). In this interest we have a preliminary result complementing Theorem 12 above (in Section 3.1) as follows.

Theorem 32 $(\exists$ *recursive* $f)(\forall x \mid \mathcal{C}_x \in \mathbf{TxtEx}_0 \ \wedge\ (\forall$ *distinct* $i, j \in W_x)[W_i \neq W_j])[\mathcal{C}_x \subseteq \mathbf{TxtEx}(\mathbf{M}_{f(x)})]$.

We also know that, in this result, $\mathbf{TxtEx}(\mathbf{M}_{f(x)})$ cannot be improved to $\mathbf{TxtEx}_n(\mathbf{M}_{f(x)})$.

It would be interesting to explore extensions of the present paper for the cases of adding small amounts of negative information to the input data (Baliga, Case and Jain (1995)). In the light of Theorem 22 in Baliga, Case and Jain (1995) and the discussion following the proof of Theorem 31 above, it is reasonable to hope for some resultant improvements in mind change complexity for synthesized machines.

Case, Jain and Sharma (1997) present positive and negative results regarding synthesizers of language learners which tolerate noisy data, where noise is modeled as in Stephan (1995), Case, Jain and Stephan (1996). Furthermore, the proofs of the positive results provide characterizations of corresponding noise-tolerantly learnable language classes. It would be interesting to extend the results of Case, Jain and Sharma (1997) and of this paper to probablistically correct inference (cf. e.g. Freivalds (1979b), Freivalds (1979a), Pitt (1984), Valiant (1984), Wiehagen, Freivalds and Kinber (1984), Daley (1985), Pitt (1989), Pitt and Smith (1988), Daley (1988), Kinber and Zeugmann (1991), Viksna (1991), Daley, Pitt, Velauthapillai and Will (1991), Schapire (1992), Daley, Kalyanasundaram and Velauthapillai (1992), Kearns and Vazirani (1994), Ambainis (1996), Case, Kaufmann, Kinber and Kummer (1997), Mitchell (1997)).

References

Ambainis, A. (1996). Probabilistic PFIN-type learning. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, (pp. 157–168). ACM Press.

[15]Actually Angluin's synthesized machine learns decision procedures rather than grammars, and, for **TxtEx**-identification of uniformly decidable classes of languages, one can learn grammars ⇔ one can learn decision procedures. We can prove this latter is *not* the case for **TxtBc**-identification (of uniformly decidable classes of languages).

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

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

Angluin, D. (1982). Inference of reversible languages. *Journal of the ACM*, *29*, 741–765.

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

Arimura, H. & Shinohara, T. (1994). Inductive inference of Prolog programs with linear data dependency from positive data. In Jaakkola, H., Kangassalo, H., Kitahashi, T., & Markus, A. (Eds.), *Proc. Information Modelling and Knowledge Bases V*, (pp. 365–375). IOS Press.

Baliga, G. & Case, J. (1993). Learnability: Admissible, co–finite and hypersimple languages. In A. Lingas, R. Karlsson, S. C. (Ed.), *Proceedings of the 20th International Colloquium on Automata, Languages and Programming, Lund, Sweden*, volume 700 of *Lecture Notes in Computer Science*, (pp. 289–300). Springer-Verlag.

Baliga, G., Case, J., & Jain, S. (1995). Language learning with some negative information. *Journal of Computer and System Sciences*, *51*(5), 273–285.

Baliga, G., Case, J., & Jain, S. (1996). Synthesizing enumeration techniques for language learning. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, (pp. 169–180). ACM Press.

Bārzdiņš, J. (1974). Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1* (pp. 82–88). Latvian State University. In Russian.

Bārzdiņš, J. & Freivalds, R. (1972). On the prediction of general recursive functions. *Soviet Mathematics Doklady*, *13*, 1224–1228.

Berwick, R. (1985). *The Acquisition of Syntactic Knowledge.* MIT Press.

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

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

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

Case, J. (1986). Learning machines. In W. Demopoulos & A. Marras (Eds.), *Language Learning and Concept Acquisition.* Ablex Publishing Company.

Case, J. (1998). The power of vacillation in language learning. *SIAM Journal on Computing.* To Appear (Preliminary Version Appeared in COLT 88).

Case, J., Jain, S., & Sharma, A. (1997). Synthesizing noise-tolerant language learners. In Li, M. & Maruoka, A. (Eds.), *Algorithmic Learning Theory: Eighth International Workshop (ALT '97)*, volume 1316 of *Lecture Notes in Artificial Intelligence*, (pp. 228–243).

Case, J., Jain, S., & Stephan, F. (1996). Vacillatory and BC learning on noisy data. In Arikawa, S. & Sharma, A. (Eds.), *Algorithmic Learning Theory: Seventh International Workshop (ALT '96)*, volume 1160 of *Lecture Notes in Artificial Intelligence*, (pp. 285– 298). Springer-Verlag.

Case, J., Kaufmann, S., Kinber, E., & Kummer, M. (1997). Learning recursive functions from approximations. *Journal of Computer and System Sciences*, *55*, 183–196. Special Issue for EuroCOLT'95.

Case, J. & Lynes, C. (1982). Machine inductive inference and language identification. In Nielsen, M. & Schmidt, E. M. (Eds.), *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, (pp. 107–115). Springer-Verlag.

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

Daley, R. (1985). Inductive inference hierarchies: Probabilistic vs. pluralistic. In Bibel, W. & Jantke, K. (Eds.), *Mathematical Methods of Specification and Synthesis of Software Systems, Wendisch-Rietz, GDR*, volume 215 of *Lecture Notes in Computer Science*, (pp. 73–82). Springer-Verlag.

Daley, R. (1988). Transformation of probabilistic learning strategies into deterministic learning strategies. In Haussler, D. & Pitt, L. (Eds.), *Proceedings of the Workshop on Computational Learning Theory*, (pp. 157–163). Morgan Kaufmann.

Daley, R., Kalyanasundaram, B., & Velauthapillai, M. (1992). Breaking the probability 1/2 barrier in FIN-type learning. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, (pp. 203–217). ACM Press.

Daley, R., Pitt, L., Velauthapillai, M., & Will, T. (1991). Relations between probabilistic and team one-shot learners. In Valiant, L. & Warmuth, M. (Eds.), *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, (pp. 228–239). Morgan Kaufmann.

de Jongh, D. & Kanazawa, M. (1996). Angluin's thoerem for indexed families of r.e. sets and applications. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, (pp. 193–204). ACM Press.

Freivalds, R. (1979a). Finite identification of general recursive functions by probabilistic strategies. In *Proceedings of the Conference on Fundamentals of Computation Theory* (pp. 138–145). Akademie-Verlag, Berlin.

Freivalds, R. (1979b). On the principle capabilities of probabilistic algorithms in inductive inference. *Semiotika Inform*, *12*, 137–140.

Freivalds, R. (1985). Recursiveness of the enumerating functions increases the inferrability of recursively enumerable sets. *Bulletin of the European Association for Theoretical Computer Science*, *27*, 35–40.

Fulk, M. (1985). *A Study of Inductive Inference Machines*. PhD thesis, SUNY/Buffalo.

Fulk, M. (1990a). Prudence and other conditions on formal language learning. *Information and Computation*, *85*, 1–11.

Fulk, M. (1990b). Robust separations in inductive inference. In *31st Annual IEEE Symposium on Foundations of Computer Science*, (pp. 405–410). IEEE Computer Society Press.

Gleitman, L. (1986). Biological dispositions to learn language. In W. Demopoulos & A. Marras (Eds.), *Language Learning and Concept Acquisition*. Ablex Publ. Co.

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

Hopcroft, J. & Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Jantke, K. (1979). Automatic synthesis of programs and inductive inference of functions. In *Int. Conf. Fundamentals of Computations Theory*, (pp. 219–225). Akademie-Verlag, Berlin.

Kapur, S. (1991). *Computational Learning of Languages*. PhD thesis, Cornell University.

Kapur, S. & Bilardi, G. (1992). Language learning without overgeneralization. In Finkel, A. & Jantzen, M. (Eds.), *Proceedings of the Ninth Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, (pp. 245–256). Springer-Verlag.

Kapur, S., Lust, B., Harbert, W., & Martohardjono, G. (1993). Universal grammar and learnability theory: The case of binding domains and the 'subset principle'. In E. Reuland & W. Abraham (Eds.), *Knowledge and Language, Volume I* (pp. 185–216). Kluwer.

Kearns, M. & Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA.

Kinber, E. & Zeugmann, T. (1991). One-sided error probabilistic inductive inference and reliable frequency identification. *Information and Computation*, *92*, 253–284.

Kirsh, D. (1992). PDP learnability and innate knowledge of language. In S. Davis (Ed.), *Connectionism: Theory and Practice* (pp. 297–322). Oxford University Press, NY.

Lange, S. & Zeugmann, T. (1992). Types of monotonic language learning and their characterization. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, (pp. 377–390). ACM Press.

Lange, S. & Zeugmann, T. (1993). Language learning in dependence on the space of hypotheses. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, (pp. 127–136). ACM Press.

Lavarač, N. & Džeroski, S. (1994). *Inductive Logic Programming*. Ellis Horwood, New York.

Machtey, M. & Young, P. (1978). *An Introduction to the General Theory of Algorithms*. North Holland, New York.

Manzini, R. & Wexler, K. (1987). Parameters, binding theory and learnability. *Linguistic Inquiry*, *18*, 413–444.

Mitchell, T. (1997). *Machine Learning.* McGraw Hill.

Muggleton, S. & Raedt, L. D. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, *19/20*, 669–679.

Mukouchi, Y. (1992). Characterization of finite identification. In Jantke, K. (Ed.), *Analogical and Inductive Inference, Proceedings of the Third International Workshop*, (pp. 260–267).

Nix, R. (1983). Editing by examples. Technical Report 280, Department of Computer Science, Yale University, New Haven, CT, USA.

Osherson, D., Stob, M., & Weinstein, S. (1982). Ideal learning machines. *Cognitive Science*, *6*, 277–290.

Osherson, D., Stob, M., & Weinstein, S. (1984). Learning theory and natural language. *Cognition*, *17*, 1–28.

Osherson, D., Stob, M., & Weinstein, S. (1986b). An analysis of a learning paradigm. In W. Demopoulos & A. Marras (Eds.), *Language Learning and Concept Acquisition.* Ablex Publ. Co.

Osherson, D., Stob, M., & Weinstein, S. (1986a). *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press.

Osherson, D., Stob, M., & Weinstein, S. (1988). Synthesising inductive expertise. *Information and Computation*, *77*, 138–161.

Osherson, D. & Weinstein, S. (1982a). Criteria of language learning. *Information and Control*, *52*, 123–138.

Osherson, D. & Weinstein, S. (1982b). A note on formal learning theory. *Cognition*, *11*, 77–88.

Pinker, S. (1979). Formal models of language learning. *Cognition*, *7*, 217–283.

Pitt, L. (1984). *A characterization of probabilistic inference.* PhD thesis, Yale University.

Pitt, L. (1989). Probabilistic inductive inference. *Journal of the ACM*, *36*, 383–433.

Pitt, L. & Smith, C. (1988). Probability and plurality for aggregations of learning machines. *Information and Computation*, *77*, 77–92.

Pour-El, M. & Howard, W. (1964). A structural criterion for recursive enumeration without repetition. *Zeitschr. j. math. Logik und Grundlagen d. Math. Bd.*, *10*, 105–114.

Pour-El, M. & Putnam., H. (1965). Recursively enunumerable classes and their application to recursive sequences of formal theories. *Arch. f. Math. Log. Grund.*, *8*, 104–121.

Rogers, H. (1958). Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, *23*, 331–341.

Rogers, H. (1967). *Theory of Recursive Functions and Effective Computability.* McGraw-Hill. Reprinted, MIT Press 1987.

Royer, J. (1987). *A Connotational Theory of Program Structure*, volume 273 of *Lecture Notes in Computer Science.* Springer-Verlag.

Schapire, R. (1992). *The Design and Analysis of Efficient Learning Algorithms.* MIT Press: Cambridge, MA.

Shapiro, N. (1971). Review of "Limiting recursion" by E.M. Gold and "Trial and error predicates and the solution to a problem of Mostowski" by H. Putnam. *Journal of Symbolic Logic*, *36*, 342.

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

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

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

Shinohara, T. & Arikawa, A. (1995). Pattern inference. In Jantke, K. P. & Lange, S. (Eds.), *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, (pp. 259–291). Springer-Verlag.

Stephan, F. (1995). Noisy inference and oracles. In Jantke, K., Shinohara, T., & Zeugmann, T. (Eds.), *Algorithmic Learning Theory: Sixth International Workshop (ALT '95)*, volume 997 of *Lecture Notes in Artificial Intelligence*, (pp. 185–200). Springer-Verlag.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, *27*, 1134–1142.

Viksna, J. (1991). Probabilistic inference of approximations. In Jantke, K. (Ed.), *Proceedings of the Conference on Nonmonotonic Logic and Inductive Inference*, volume 659 of *Lecture Notes in Artificial Intelligence*, (pp. 323–332). Springer-Verlag, Berlin.

Wexler, K. (1982). On extensional learnability. *Cognition*, *11*, 89–95.

Wexler, K. (1993). The subset principle is an intensional principle. In E. Reuland & W. Abraham (Eds.), *Knowledge and Language, Volume I* (pp. 217–239). Kluwer.

Wexler, K. & Culicover, P. (1980). *Formal Principles of Language Acquisition*. MIT Press.

Wiehagen, R., Freivalds, R., & Kinber, E. (1984). On the power of probabilistic strategies in inductive inference. *Theoretical Computer Science*, *28*, 111–133.

Wright, K. (1989). Identification of unions of languages drawn from an identifiabl e class. In Rivest, R., Haussler, D., & Warmuth, M. (Eds.), *Proceedings of the Second Annual Workshop on Computational Learning Theory*, (pp. 328–333). Morgan Kaufmann Publishers, Inc.

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

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