Team Learning of Computable Languages

Sanjay Jain
School of Computing
National University of Singapore
Singapore 119260, Republic of Singapore
Email: sanjay@comp.nus.edu.sg

Arun Sharma
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052, Australia
Email: arun@cse.unsw.edu.au

Abstract

A team of learning machines is a multiset of learning machines. A team is said to successfully learn a concept just in case each member of some nonempty subset, of predetermined size, of the team learns the concept. Team learning of languages may be viewed as a suitable theoretical model for studying computational limits on the use of multiple heuristics in learning from examples.

Team learning of recursively enumerable languages has been studied extensively. However, it may be argued that from a practical point of view all languages of interest are computable. This paper gives theoretical results about team learnability of computable (recursive) languages. These results are mainly about two issues: redundancy and aggregation. The issue of redundancy deals with the impact of increasing the size of a team and increasing the number of machines required to be successful. The issue of aggregation deals with conditions under which a team may be replaced by a single machine without any loss in learning ability. The learning scenarios considered are:

(a) Identification in the limit of grammars for computable languages.

(b) Identification in the limit of decision procedures for computable languages.

(c) Identification in the limit of grammars for indexed families of computable languages.

(d) Identification in the limit of grammars for indexed families with a recursively enumerable class of grammars for the family as the hypothesis space.

Scenarios that can be modeled by team learning are also presented.

1 Introduction

Recently there has been considerable interest in multi-agent learning [37]—an emerging research direction at the intersection of distributed AI and machine learning. The main focus of this work has been empirical. The present paper argues that the "old" field of team learning with more than two decades [36] of development in the computational learning theory community provides an initial model for investigation of learning from examples by multiple heuristics.

To understand the idea of team learning, it is useful to consider an informal statement of a result in inductive inference due to Blum and Blum [4], called the "nonunion theorem."

According to this result, there are classes of concepts, $\mathcal{C}_1$ and $\mathcal{C}_2$, such that each of them is independently learnable, but their union $\mathcal{C}_1 \cup \mathcal{C}_2$ is not learnable. In other words, there is a learning heuristic capable of learning concepts from $\mathcal{C}_1$ and another learning heuristic capable of learning concepts from $\mathcal{C}_2$, but there is no single heuristic that can learn any concept drawn from either $\mathcal{C}_1$ or $\mathcal{C}_2$. The concept class $\mathcal{C}_1 \cup \mathcal{C}_2$ may be viewed as one of those challenging problem domains for which a single learning heuristic does not suffice. However, if one were allowed to use a "team" of heuristics with the additional weakening of the criterion of successful learning, then a learnability model can be developed under which the class $\mathcal{C}_1 \cup \mathcal{C}_2$ is learnable. We illustrate this idea next.

Let $\mathcal{H}_1$ be a learning heuristic that learns $\mathcal{C}_1$ and let $\mathcal{H}_2$ be a learning heuristic that learns $\mathcal{C}_2$. Now, if we employed a team of $\mathcal{H}_1$ and $\mathcal{H}_2$ to learn $\mathcal{C}_1 \cup \mathcal{C}_2$ and weakened the criterion of success to the requirement that success is achieved just in case any one member in the team is successful, then the class $\mathcal{C}_1 \cup \mathcal{C}_2$ becomes learnable by the team of heuristics $\mathcal{H}_1$ and $\mathcal{H}_2$ under this new criterion of success. However, a price has been paid as it is no longer possible to determine which member of the team learns which concept in $\mathcal{C}_1 \cup \mathcal{C}_2$. If it were possible to determine such information, then the team of heuristics $\mathcal{H}_1$ and $\mathcal{H}_2$ could be aggregated into a single heuristic, thereby contradicting the nonunion theorem. At first glance, this lack of information about which strategy in the team learns which concept may appear to be debilitating, we illustrate scenarios in Section 5 where such lack of information is not a hindrance.

The study of team learning has concentrated on two kinds of questions: *aggregation* and *redundancy*. The question of aggregation attempts to determine the conditions under which employing a team of learning strategies yields no advantage over employing a single learning heuristic. The question of redundancy attempts to find if introducing redundancy in the team (e.g., doubling the number of heuristics in the team and also doubling the number of heuristics required to be successful for the team to be successful) yields any extra learning ability. The present paper surveys the work on team learning of recursively enumerable languages and presents new results about team learning of computable[1] languages from the standpoint of aggregation and redundancy.

The present paper considers redundancy and aggregation results for team learning in the context of following learning problems:

(1) Identification in the limit of grammars for computable languages.

(2) Identification in the limit of decision procedures for computable languages.

(3) Identification in the limit of grammars for indexed families of computable languages where the hypothesis space is also an indexed family.

(4) Identification in the limit of grammars for indexed families with a recursively enumerable class of grammars for the family as the hypothesis space.

Results related to Item (1) above show that team identification of grammars for computable languages has similar behavior to team identification of recursively enumerable languages. However, results about Item (2) show that if attention is restricted to learning decision procedures for computable languages, then the behavior is similar to team identification of functions. Results related to Item (3) show that a similar behavior to team function identification is also displayed by team learning of indexed families of computable languages if the hypothesis space

[1] *Computable* languages, also referred to as *recursive* languages, are those languages for which there exists an algorithmic decision procedure.

is also an indexed family. However, results related to Item (4) show that if the hypothesis space is an enumerable class of grammars, then team learning of indexed families of computable languages has a different behavior. Proofs of results related to Items (1), (2), and (3) are based on earlier results, and are presented for the sake of completeness. For these results, we give a sketch of how they can be derived by adapting known techniques from the literature. The main contribution of this paper are results related to Item (4) which are proved in detail.

The paper is arranged as follows. In Section 2, we discuss the choice of languages over functions as a more appropriate model for learning from examples. In Section 3, we introduce the preliminary definitions about identification in the limit of languages by a single machine. In Section 4, we motivate and describe identification of languages by teams of machines. In the same section, we also provide a guide to the literature on team learning. In Section 5, we describe two hypothetical scenarios that may be modeled using team learning. Section 6 surveys previously known results about team learning of r.e. languages. In Section 7, we present results about team learning of computable languages (Items (1) and (2) above) and in Section 8, we present results about team learning of indexed families of computable languages (Items (3) and (4) above). Finally, in an appendix, we give proofs.

2 Concepts: Functions and Languages

Algorithmic identification in the limit of two concept classes, computable functions and recursively enumerable languages, have been investigated extensively in the computational learning theory literature. Although the subject of this paper is learnability of languages, we first shed some light on the distinction between function learning and language learning.

Let us consider the learning of a computable function f. A learning machine is fed the graph of f, $(0, f(0)), (1, f(1)), \ldots$, one ordered pair at a time, and the machine, from time to time, conjectures a sequence of computer programs. The machine is said to learn f just in case its conjectures converge to a program for f. Recently, learning of functions by teams of learning machines has become a very active area of research and has been suggested as a theoretical model for multi-agent learning from examples (for example, see [3, 2, 10, 12, 18, 24, 33, 35]).

The utility of function learning as a model for machine learning from examples, however, is somewhat limited, as it is able to model only one aspect of learning from examples. Data available to most learning systems are of two kinds: positive data and complete (both positive and negative) data. In learning from positive data a learner is only guaranteed that it will eventually see all the positive data, whereas in learning from complete data a learner will eventually be presented with all the positive and all the negative data. It turns out that function learning models only learning from complete data. The negative data is implicitly available to a learning machine because the input to a function learning machine is the graph of the function. To see this: if the ordered pair $(2, 5)$ is encountered in the graph, then a learning machine can safely assume that any pair of the form $(2, x)$, $x \neq 5$, does not belong to the function. The point is that a learner can eventually deduce all the negative data from the incoming positive data.[2]

However, this problem does not arise in the case of identification in the limit of languages (described in the next section), as both learning from positive data and complete data can be modeled. Moreover, as might be expected, results and techniques in the study of team learning of languages from complete data parallel results and techniques in the study of team learning

[2]Of course this discussion does not hold for identification of partial functions, but we are concerned here with identification of total computable functions.

of functions. Since they allow the additional possibility of modeling learning from only positive data, we have chosen languages as our vehicle for the investigation of multi-agent learning from examples.

3 Language Learning by a single machine

Let N denote the set of natural numbers, $\{0, 1, 2, \ldots\}$. As already noted our domain is the collection of recursively enumerable languages over N. A grammar for a recursively enumerable language L is a computer program that accepts L (or, equivalently, generates L [17]). For any recursively enumerable language L, the elements of L constitute its positive data and the elements of the complement, $N - L$, constitute its negative data. We next describe notions that capture the presentation of positive data and presentation of both positive and negative data.

Definition 1 A *text* for the language L is an infinite sequence (repetitions allowed) consisting of all and only the elements of L. T denotes a typical variable for texts.

So, a text for L represents an instance of positive data presentation for L. The next definition introduces a notion that represents an instance of both positive and negative data presentation for L.

Definition 2 An *informant* for L is an infinite sequence (with repetitions allowed) of ordered pairs such that for each $n \in N$ either $(n, 1)$ or $(n, 0)$ (but not both) appear in the sequence and $(n, 1)$ appears only if $n \in L$ and $(n, 0)$ appears only if $n \notin L$.

At any give time a learning machine has access to only a finite sequence of a text or an informant. For this reason it useful to introduce some notation about finite sequences. We do it for texts; a similar discussion holds for informants.

The initial sequence of text T of length n is denoted $T[n]$. The set of all finite initial sequences of texts, $\{T[n] \mid T \text{ is a text and } n \in N\}$, is denoted SEQ. We let σ and τ range over SEQ. We let Λ denote the empty sequence. The sequence resulting from the concatenation of two sequences σ followed by τ is denoted $\sigma \diamond \tau$. The *content* of a sequence σ, denoted content(σ), is the set of natural numbers in the range of σ. The *length* of σ, denoted by $|\sigma|$, is the number of elements in σ. For $n \leq |\sigma|$, the initial segment of σ of length n is denoted by $\sigma[n]$.

We now consider machines that learn from texts. Similar definitions can be made for machines that learn from informants. A learning machine (for learning from texts) may be thought of as an algorithmic device that computes a mapping from SEQ into N. The output of the learning machine may be viewed as indices for computer programs in a suitable acceptable programming system conjectured by the machine as hypotheses. We let $\mathbf{M}$, with or without decorations, denote a typical variable for learning machines. We say that a learning machine $\mathbf{M}$ converges on a text T just in case there exists an i such that for all but finitely many n, $\mathbf{M}(T[n]) = i$. We now consider what it means for a learning machine to successfully learn languages. The criterion of success considered in the present paper is Gold's [16] *identification in the limit*. We first introduce it for learning from positive data.

Definition 3 [16]

(a) **M TxtEx**-*identifies* an r.e. language L just in case $\mathbf{M}$, fed any text for L, converges to a grammar for L. In this case we say that $L \in \mathbf{TxtEx}(\mathbf{M})$.

(b) **M TxtEx**-*identifies* a class of languages, $\mathcal{L}$, just in case **M TxtEx**-identifies each language in $\mathcal{L}$.

(c) **TxtEx** denotes a collection of classes $\mathcal{L}$ of r.e. languages such that some machine **TxtEx**-identifies $\mathcal{L}$.

Thus **TxtEx** is a set theoretic summary of the capability of machines to **TxtEx**-identify classes of r.e. languages. Intuitively, if $\mathcal{L} \in$ **TxtEx**, then there exists a machine that **TxtEx**-identifies each language in $\mathcal{L}$.

It is easy to see that any class consisting of just one language is identifiable because an "oblivious" machine that ignores its input and keeps on emitting a grammar for the only language in the class is successful on that language; however, such a machine is unsuccessful on every other language. It is precisely for this reason, that we introduced Part (b) in the above definition; machines that learn only one language are not very interesting.

As an example of a class in **TxtEx**, consider **FIN**, the class of finite languages. It is easy to see that **FIN** belongs to **TxtEx** because a machine employing the heuristic of emitting a grammar for all the elements it has seen at any given time will suffice.

We now define identification from both positive and negative data.

Definition 4 [16]

(a) **M InfEx**-*identifies* an r.e. language L just in case **M**, fed any informant for L, converges to a grammar for L. In this case we say that $L \in$ **InfEx**(**M**).

(b) **M InfEx**-*identifies* a class of languages, $\mathcal{L}$, just in case **M InfEx**-identifies each language in $\mathcal{L}$.

(c) **InfEx** denotes a collection of all such classes $\mathcal{L}$ of r.e. languages such that some machine **InfEx**-identifies $\mathcal{L}$.

4 Learning by a team and related work

A team of learning machines is a multiset of learning machines.[3] Before we formally define learning by a team, it is worth considering the origins of team learning. Consider the following theorem for **TxtEx**-identification.

Theorem 1 [4] *There are classes of languages $\mathcal{L}_1$ and $\mathcal{L}_2$ such that*

(a) $\mathcal{L}_1 \in$ **TxtEx**,

(b) $\mathcal{L}_2 \in$ **TxtEx**, *but*

(c) $(\mathcal{L}_1 \cup \mathcal{L}_2) \notin$ **TxtEx**.

The above result[4], popularly referred to as the "non-union theorem," says that **TxtEx** is not closed under union. In other words, there are classes of languages that are identifiable, but the union of these classes is not identifiable. This result may be viewed as a fundamental limitation on building a general purpose device for machine learning, and, to an extent, justifies the use of heuristic methods in Artificial Intelligence. However, this result also suggests a more general criterion of identification in which a team of learning machines is employed and success of the team is the success of any member in the team. We illustrate this idea next.

Consider the classes of languages $\mathcal{L}_1$ and $\mathcal{L}_2$ in Theorem 1. Let $\mathbf{M}_1$ **TxtEx**-identify $\mathcal{L}_1$ and $\mathbf{M}_2$ **TxtEx**-identify $\mathcal{L}_2$. Now, if we employed a team consisting of $\mathbf{M}_1$ and $\mathbf{M}_2$ to identify $\mathcal{L}_1 \cup \mathcal{L}_2$ and weakened the criterion of success to the requirement that success is achieved just

[3] We use multset because there may be several copies of the same machine in the team.

[4] Taking $\mathcal{L}_1 = \{N\}$ and $\mathcal{L}_2 =$ **FIN** yields a proof because of Gold's [16] result that no class of languages that contains all the finite languages and an infinite language can be identified in the limit from only positive data.

in case any one member in the team is successful, then the class $\mathcal{L}_1 \cup \mathcal{L}_2$ becomes identifiable by the team consisting of $\mathbf{M}_1$ and $\mathbf{M}_2$ under this new criterion of success. This idea can be extended to teams of n machines out of which at least m ($m \leq n$) are required to be successful. The formal definitions for team identification of languages are presented next.

We abuse the notation slightly and use the same notation for sets and multisets; it will be clear from context which one is meant.

Definition 5 Let $m, n \in N$ and $0 < m \leq n$.

(a) A team of n machines $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\}$ is said to $\mathbf{Team}_n^m\mathbf{TxtEx}$-identify a language L just in case at least m members in the team $\mathbf{TxtEx}$-identify L. In this case we write $L \in \mathbf{Team}_n^m\mathbf{TxtEx}(\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\})$.

(b) A team of n machines $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\}$ is said to $\mathbf{Team}_n^m\mathbf{TxtEx}$-identify a class of languages, $\mathcal{L}$, just in case the team $\mathbf{Team}_n^m\mathbf{TxtEx}$-identifies each language in $\mathcal{L}$.

(c) $\mathbf{Team}_n^m\mathbf{TxtEx}$ is defined to be the collection of classes $\mathcal{L}$ of r.e. languages such that some team of n machines $\mathbf{Team}_n^m\mathbf{TxtEx}$-identifies $\mathcal{L}$.

We can similarly define $\mathbf{Team}_n^m\mathbf{InfEx}$ for team learning from both positive and negative data. We now give a brief guide to the literature on team learning.

The "nonunuon theorem" scenario first appears in the context of frequency identification, and was studied by Podnieks [34].[5] Team learning of functions was motivated by Case (cited in [35]) based on the nonunion theorem of Blum and Blum [4], and studied extensively by Smith [35]. The general case of team identification (m out of n) is due to Osherson, Stob, and Weinstein [29]. The notion of probabilistic learning turns out to be closely related to team learning and was first investigated by Freivalds [13]. Pitt [31] was the first to notice that team learning in the limit of functions and probabilistic identification in the limit of functions turn out to be equivalent (see also Pitt and Smith [33]). Jain and Sharma [19, 21, 23] investigated team learning of recursively enumerable languages.

Recently, there has been a spurt of activity in the study of teams and probabilistic machines for learning with bounded number of mind changes (see Wiehagen, Freivalds, and Kinber [38] and Daley and Kalyanasundaram [8]). Considerable work has been done for a special case of learning with bounded number of mind changes, namely finite identification (0 mind changes; also referred to as one-shot learning in the literature). We direct the reader to Freivalds [14], Jain, Sharma, and Velauthapillai [24], Daley, Pitt, Velauthapillai, and Will [12], Daley, Kalyanasundaram, and Velauthapillai [10]. The problem of teams for Popperian[6] finite identification of functions is addressed by Daley and Kalyanasundaram [9]. Allowing teams of finite learners to make up to a finite number of errors in the hypothesis conjectured has been addressed by Daley, Kalyanasundaram, and Velauthapillai [11]. Behaviorally correct function identification by teams has been studied by Daley [7].

In the context of language identification, work has hardly begun on other criteria. We direct the reader to Jain and Sharma [20, 22] for results on finite, vacillatory, and behaviorally correct identification of languages by teams. Meyer [27] has investigated probabilistic identification of indexed families of computable languages (see Meyer [28] for interaction between monotonicity constraints and probabilistic identification of indexed families of computable languages).

[5] For **Ex**-identification of functions, frequency learning was shown to be equivalent to team learning by Pitt [31]. Kinber and Zeugmann [25] extended this concept to reliable frequency identification, which in turn is equivalent to one-sided error probabilistic inference.

[6] Popperian learners are such that they only conjecture indices of total computable functions; see [6].

5 Settings for team learning

Finally, it is worth noting an aspect of team identification that cannot be overlooked, namely, it is in general not possible to determine which members in the team are successful. If it were possible to decide which members are successful then identification by teams would not yield any extra learning ability over identification by single machines. This property seems to rob team identification of any possible utility. However, we present below scenarios, first described by us in [19], in which the knowledge of which machines are successful is of no consequence, all that matters is some are.

First, consider a hypothetical situation in which an intelligent species, somewhere in outer space, is attempting to contact other intelligent species (such as humans on earth) by transmitting radio signals in some language (most likely alien to humans). Being a curious species ourselves, we would like to establish a communication link with such a species that is trying to reach out. For this purpose, we could employ a team of language learners each of which perform the following three tasks in a loop:

(a) receive and examine strings of a language (eg., from a radio telescope);

(b) guess a grammar for the language whose strings are being received;

(c) transmit messages back to outer space based on the grammar guessed in Step (b).

If one or more of the learners in the team is actually, but, possibly unknowingly, successful in learning a grammar for the alien language, a correct communication link would be established between the two species.

Consider another scenario in which two countries, A and B, are at war with each other. Country B uses a secret language to transmit movement orders to its troops. Country A, with an intention to confuse the troops of country B, wants to learn a grammar for country B's secret language so that it can transmit conflicting troop movement instructions in that secret language. To accomplish this task, country A employs a team of language learners, each of which perform the following three tasks in a loop:

(a) receive and examine strings of country B's secret language;

(b) guess a grammar for the language whose strings are being received;

(c) transmit conflicting messages based on the grammar guessed in Step (b) (so that B's troops think that these messages are from B's Generals).

If one or more of the learners in the team is actually, but possibly unknowingly, successful in correctly learning a grammar for country B's secret language, then country A achieves its purpose of confusing the troops of country B.

It should be noted that the notion of team learning models only part of the above scenario, as we ignore in our mathematical model the aspect of learners transmitting messages back. We also mathematically ignore possible detrimental effects of a learner guessing an incorrect grammar and transmitting messages that could interfere with messages from a learner that infers a correct grammar (for example, the string 'baby milk powder factory' in one language could mean the string 'ammunition storage' in another!). In no way are these issues trivial; we simply don't have a formal handle on them at this stage.

6 Previous Results: Team learning of r.e. languages

We now survey some of the results about team learning of r.e. languages. The results that we present here are about redundancy and aggregation. We direct the reader to [19, 21, 22, 23] for additional results.

First, it is easy to show the following proposition.

Proposition 1 *Let $k, m, n \in N$ such that $0 < m \leq n$ and $k \geq 1$.*
(a) $\mathbf{Team}_n^m\mathbf{TxtEx} \subseteq \mathbf{Team}_{n \cdot k}^{m \cdot k}\mathbf{TxtEx}$.
(b) $\mathbf{Team}_n^m\mathbf{InfEx} \subseteq \mathbf{Team}_{n \cdot k}^{m \cdot k}\mathbf{InfEx}$.

The above proposition says that for both texts and informants, the classes of languages that can be learned by a given team can also be learned if we multiply the size of the team and the number of machines required to be successful by the same factor. In other words, introducing redundancy does not hurt. The question is: Does it help? We consider team learning from informants first, followed by team learning from texts.

6.1 Team learning from informants

For identification from both positive and negative data, introducing redundancy in the team does not yield any extra learning ability.

Theorem 2 *(Adapted from* [33]*) Let $k, m, n \in N$ such that $0 < m \leq n$ and $k \geq 1$. Then* $\mathbf{Team}_n^m\mathbf{InfEx} = \mathbf{Team}_{n \cdot k}^{m \cdot k}\mathbf{InfEx}$.

The above result says that the classes of languages that can be identified by teams employing n machines and requiring at least m to be successful are exactly the same as those classes which can be identified by teams employing $n \cdot k$ machines and requiring at least $m \cdot k$ to be successful.

We next consider the question of aggregation, that is, under what conditions can a team be replaced by a single machine without any loss in learning ability. Part (a) of the next result says that if a majority of the members in the team are required to be successful, then employing a team does not yield any extra learning ability. Part (b) of the result says that $\frac{1}{2}$ is indeed the cutoff. In the sequel, we refer to such cutoff points as *aggregation ratios*. $\subset$ denotes proper subset.

Theorem 3 *(a)* $(\forall m, n \mid \frac{m}{n} > \frac{1}{2})[\mathbf{Team}_n^m\mathbf{InfEx} = \mathbf{InfEx}]$.
(b) $\mathbf{InfEx} \subset \mathbf{Team}_2^1\mathbf{InfEx}$.

A proof of the above result can be worked out using techniques from Pitt [32] and from Pitt and Smith [33].

6.2 Team learning from texts

Surprisingly, introducing redundancy in the team does help sometimes in the context of learning from only positive data. The following result says that there are classes of languages that can be **TxtEx**-identified by teams employing 4 machines and requiring at least 2 to be successful, but cannot be **TxtEx**-identified by any team employing 2 machines and requiring at least 1 to be successful.

Theorem 4 *(*[23]*)* $\mathbf{Team}_2^1\mathbf{TxtEx} \subset \mathbf{Team}_4^2\mathbf{TxtEx}$.

Even more surprising is the next theorem which implies that the classes of languages that can be **TxtEx**-identified by teams employing 6 machines and requiring at least 3 to be successful are exactly the same as those classes that can be **TxtEx**-identified by teams employing 2 machines and requiring at least 1 to be successful.

Theorem 5 *(*[23]*)* $(\forall j)[\mathbf{Team}_{4j+2}^{2j+1}\mathbf{TxtEx} = \mathbf{Team}_2^1\mathbf{TxtEx}]$.

The complete picture is actually quite complicated. The status of teams with success ratio $\frac{1}{2}$ is completely known, but only partial results are known for other team ratios ($\frac{1}{k}, k > 2$); we direct the reader to [23].

The next result sheds light on when a team learning languages from texts can be aggregated into a single machine without loss in learning ability. Part (a) of the result says that if *more than two-thirds* of the members in the team are required to be successful, then employing a team for learning languages from texts does not yield any extra learning ability. Part (b) of the result says that $\frac{2}{3}$ is indeed the cutoff. We refer the reader to [32, 31, 23] for proofs.

Theorem 6
(a) $(\forall m, n \mid \frac{m}{n} > \frac{2}{3})[\mathbf{Team}_n^m\mathbf{TxtEx} = \mathbf{TxtEx}]$.
(b) $\mathbf{TxtEx} \subset \mathbf{Team}_3^2\mathbf{TxtEx}$.

7 Results: Team learning of computable languages

It may justifiably be argued that recursively enumerable languages are too general to usefully model concepts of practical interest. For this reason, it is worth considering the effects of team learning on restricted classes of languages. In this section, we present results about redundancy and aggregation for computable languages. We denote the class of computable languages by **REC**.

It turns out that even for computable languages, redundancy does help sometimes. (*Notation*: The power set of a set A is denoted 2^A.)

Theorem 7 $(\mathbf{Team}_2^1\mathbf{TxtEx} \cap 2^{\mathbf{REC}}) \subset (\mathbf{Team}_4^2\mathbf{TxtEx} \cap 2^{\mathbf{REC}})$.

The proof of Theorem 4 in [23] is actually a proof of the above theorem (because the language class constructed as the witness for $\mathbf{Team}_4^2\mathbf{TxtEx}$ being a strict super set of $\mathbf{Team}_2^1\mathbf{TxtEx}$ consist only of computable languages). For similar reasons, the aggregation ratio for team identification of computable languages turns out to be $\frac{2}{3}$, as recorded in the following theorem.

Theorem 8
(a) $(\forall m, n \mid \frac{m}{n} > \frac{2}{3})$ $[(\mathbf{Team}_n^m\mathbf{TxtEx} \cap 2^{\mathbf{REC}}) = (\mathbf{TxtEx} \cap 2^{\mathbf{REC}})]$.
(b) $(\mathbf{TxtEx} \cap 2^{\mathbf{REC}}) \subset (\mathbf{Team}_3^2\mathbf{TxtEx} \cap 2^{\mathbf{REC}})$.

It may be argued that if we are restricting ourselves to learning of computable languages then we should consider identifying decision procedures instead of grammars. The following definition formalizes this notion. (*Notation*: A *characteristic function* of a language L is the function which is 1 on elements of L and 0 on nonelements of L.)

Definition 6 [16]
(a) **M TxtExCI**-*identifies* a computable language L just in case **M**, fed any text for L, converges to a program that computes the characteristic function of L. In this case we say that $L \in \mathbf{TxtExCI}(\mathbf{M})$.
(b) **M TxtExCI**-*identifies* a class of languages, $\mathcal{L}$, just in case **M TxtExCI**-identifies each language in $\mathcal{L}$.
(c) **TxtExCI** denotes the collection of classes $\mathcal{L}$ of computable languages such that some machine **TxtExCI**-identifies $\mathcal{L}$.

It should be noted that the hypothesis space of the learner in the above definition is still the set of all programs; it is only required that the final converged program compute the characteristic function of the language being learned. Osherson and Weinstein [30] observed the following fact which implies that there are classes of computable languages for which a grammar can be identified from texts, but for which a decision procedure cannot be identified from texts.

Theorem 9 [30] $\mathbf{TxtExCI} \subset (\mathbf{TxtEx} \cap 2^{\mathbf{REC}})$.

We next consider team identification of decision procedures for computable languages from texts. One can define $\mathbf{Team}_n^m\mathbf{TxtExCI}$ in a manner similar to other team learning criteria. Until now, we have seen that in the case of learning from only positive data (texts), redundancy sometimes results in increased learning ability. Surprisingly, the following theorem shows that redundancy does not pay off when the team is learning decision procedures.

Theorem 10 *Let $k, m, n \in N$ such that $0 < m \leq n$ and $k \geq 1$. Then* $\mathbf{Team}_n^m\mathbf{TxtExCI} = \mathbf{Team}_{n \cdot k}^{m \cdot k}\mathbf{TxtExCI}$.

The aggregation ratio for **TxtExCI** turns out to be $\frac{1}{2}$ as shown by the following theorem.

Theorem 11 *(a)* $(\forall m, n \mid \frac{m}{n} > \frac{1}{2})[\mathbf{Team}_n^m\mathbf{TxtExCI} = \mathbf{TxtExCI}]$.
(b) $\mathbf{TxtExCI} \subset \mathbf{Team}_2^1\mathbf{TxtExCI}$.

Note that Theorem 11(b) follows by taking $\mathcal{L} = \{N\} \cup \mathbf{FIN}$. Proof of Theorem 10 and Theorem 11(a) can be obtained by adapting techniques from Pitt [32] and Pitt and Smith [33]. We illustrate such an adaptation in the Appendix for Theorem 10.

8 Results: Indexed families of computable languages

We next consider identification of indexed families of computable languages. A sequence of nonempty languages $L_0, L_1, \ldots$ is an indexed family just in case there exists a computable function f such that for each $i \in N$ and for each $x \in N$,

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

In other words, there is a uniform decision procedure for languages in the class. Angluin [1] was the first researcher to restrict investigations to indexed families of computable languages; she was motivated by the fact that most language families of practical interest are indexed families (e.g., the class of pattern languages). [39] provides an excellant survey of research done in the context of learnability of indexed families of languages. We denote by **INDEX** the collection of all indexed families of computable languages. Again, we restrict ourselves to texts, as informants do not yield any new insight.

Since we are considering indexed families, it makes sense to consider scenarios where the hypothesis space available to the learning machine is a recursively enumerable class of grammars. We first introduce some notation.

Let Σ be a fixed terminal alphabet. $\mathbf{Lang}(G)$ is the language generated/accepted by G.

$\mathcal{G} = G_0, G_1, G_2, \ldots$ is a hypothesis space just in case $\mathcal{G}$ is a recursively enumerable family of grammars over Σ such that membership in $\mathbf{Lang}(G_i)$ is uniformly decidable for all $i \in N$

and all strings $s \in \Sigma^*$. When a learning machine emits i, we interpret it to mean that it is conjecturing the grammar G_i. We say that the class of languages $\{\mathbf{Lang}(G_i) \mid i \in N\}$ is defined by the hypothesis space $\mathcal{G}$. We also refer to the class $\{\mathbf{Lang}(G_i) \mid i \in N\}$ as range($\mathcal{G}$); it is easy to see that range($\mathcal{G}$) is an indexed family.

Below we adapt Gold's criterion of identification in the limit to the identification of indexed families with respect to a given hypothesis.

Definition 7 *Let $\mathcal{L}$ be an indexed family and let $\mathcal{G}$ be a hypothesis space.*

(a) Let $L \in \mathcal{L}$. A machine $\mathbf{M}$ **TxtEx**-identifies *L with respect to $\mathcal{G}$ just in case* $\mathbf{M}$*, fed any text for L, converges to j and $L = \mathbf{Lang}(G_j)$.*

(b) A machine $\mathbf{M}$ **TxtEx**-identifies *$\mathcal{L}$ with respect to $\mathcal{G}$ just in case for each $L \in \mathcal{L}$,* $\mathbf{M}$ **TxtEx**-*identifies L with respect to $\mathcal{G}$.*

There are three kinds of identification of indexed families that have been studied in the literature: (a) class comprising; (b) class preserving; and (c) exact [26, 39]. Since, the notion of **TxtEx**-identification of indexed families is equivalent in all of the above three forms [26], we will only consider the class comprising case in this paper.

Definition 8 $[\mathbf{TxtEx}]_{\mathbf{Index}}$ *denotes the collection of all indexed families $\mathcal{L}$ for which there is a machine* $\mathbf{M}$ *and a hypothesis space $\mathcal{G}$ such that* $\mathbf{M}$ **TxtEx**-*identifies $\mathcal{L}$ with respect to $\mathcal{G}$.*

Similar to the above, we can define a hypothesis space of decision procedures and define the collection $[\mathbf{TxtExCI}]_{\mathbf{Index}}$. It turns out that for indexed families of computable languages, learning grammars and decision procedures are equivalent.

Proposition 2 $[\mathbf{TxtEx}]_{\mathbf{Index}} = [\mathbf{TxtExCI}]_{\mathbf{Index}}$.

Hence we only consider grammar identification in our investigation of team learning for indexed families. One can then define the collection

$$[\mathbf{Team}_n^m\mathbf{TxtEx}]_{\mathbf{Index}}$$

in a manner similar to other team identification criteria.

The following two theorems summarize that in the case of learning indexed families of computable languages from texts, redundancy does not pay off and the aggregation ratio is $\frac{1}{2}$. Hence, having a more structured hypothesis space makes a difference.

Theorem 12 *Let $k, m, n \in N$ such that $0 < m \leq n$ and $k \geq 1$.*
Then $[\mathbf{Team}_n^m\mathbf{TxtEx}]_{\mathbf{Index}} = [\mathbf{Team}_{n\cdot k}^{m\cdot k}\mathbf{TxtEx}]_{\mathbf{Index}}$.

Theorem 13 *(a)* $(\forall m, n \mid \frac{m}{n} > \frac{1}{2})[[\mathbf{Team}_n^m\mathbf{TxtEx}]_{\mathbf{Index}} = [\mathbf{TxtEx}]_{\mathbf{Index}}]$.

(b) $[\mathbf{TxtEx}]_{\mathbf{Index}} \subset [\mathbf{Team}_2^1\mathbf{TxtEx}]_{\mathbf{Index}}$.

Note that Theorem 13(b) follows by taking $\mathcal{L} = \{N\} \cup \mathbf{FIN}$. A proof of Theorem 12 and Theorem 13(a) can be worked out on the lines of proofs of Theorem 10 as discussed in the Appendix.

Finally we consider team identification of indexed families where the learning machines are allowed to conjecture any r.e. index, that is, the hypothesis space is not restricted to a recursively enumerable family of grammars. This is an interesting question as it sheds light on the choice of a hypothesis space that is far richer than the concept class being learned warrants. In this case it turns out that redundancy pays off in some cases, and the aggregation ratio is $\frac{2}{3}$. The next two theorems summarize this result.

Theorem 14 $(\mathbf{Team}_2^1\mathbf{TxtEx} \cap \mathbf{INDEX}) \subset (\mathbf{Team}_4^2\mathbf{TxtEx} \cap \mathbf{INDEX})$.

Theorem 15 *(a)* $(\forall m, n \mid \frac{m}{n} > \frac{2}{3})$ $[(\mathbf{Team}_n^m\mathbf{TxtEx} \cap \mathbf{INDEX}) = (\mathbf{TxtEx} \cap \mathbf{INDEX})]$.
(b) $(\mathbf{TxtEx} \cap \mathbf{INDEX}) \subset (\mathbf{Team}_3^2\mathbf{TxtEx} \cap \mathbf{INDEX})$.

The Appendix contains a detailed proof of the above two theorems.

9 Acknowledgements

We are grateful to the referees for several valuable suggestions. A preliminary version of this paper was presented at the *Fourth Pacific Rim International Conference on Artificial Intelligence*, Cairns, Australia. Research of Arun Sharma has been partially supported by the Australian Research Council Grant A49600456.

Appendix

Proofs of Theorems 10 and 11 and of Theorems 12 and 13 can be derived by adapting proof techniques from Pitt [32] and from Pitt and Smith [33]. We illustrate such an adaptation for Theorem 10. Proofs of Theorem 14 and 15, however, require intricate diagonalization arguments which are described in detail. We first introduce some mathematical notation.

Notation

We let φ denote an acceptable programming system. Since there are countably many programs in the programming system φ, we refer to each program with its index (or, its number). We let φ_i stand for the partial computable function computed by the program with index i in the φ-system. We denote $\varphi_i(x)\downarrow$ to mean that the program with index i in the φ-system on input x is defined. We write $\varphi_i(x)\downarrow = y$, or simply $\varphi_i(x) = y$, to mean that the program with index i in the φ-system, on input x, outputs y. We write $\varphi_i(x)\uparrow$ to denote that the program with index i in the φ-system on input x does not halt.

We let Φ denote an arbitrary fixed Blum complexity measure [5, 17] for the φ-system. W_i denotes domain(φ_i). W_i is, then, the r.e. set/language $(\subseteq N)$ accepted (or equivalently, generated) by the φ-program i. We refer to i as a grammar (acceptor) for L just in case L is the domain of φ_i. We denote by $W_{i,s}$ the set $\{x \leq s \mid \Phi_i(x) < s\}$.

A computable language has a computable decision procedure. We refer to i as a decision procedure (or, the characteristic index) for a computable language L just in case $\varphi_i(x) = 1$ if $x \in L$ and $\varphi_i(x) = 0$ if $x \notin L$. Suppose i is not a decision procedure for L, then we consider two kinds of errors that i can make in deciding if an element belongs to L. Suppose $\varphi_i(x)\downarrow$ and either $\varphi_i(x) \neq 1$ when $x \in L$ or $\varphi_i(x) \neq 0$ when $x \notin L$, then we say that i makes an *error of commission* at x. On the other hand if $\varphi_i(x)\uparrow$, then we say that i makes an *error of omission* at x. Finally, for a finite set S of programs, let unify(S) be a program defined as follows:

begin $\varphi_{\text{unify}(S)}(x)$

Search for $i \in S$ such that $\varphi_i(x)\downarrow$.

If and when such an i is found, let $\varphi_{\text{unify}(S)}(x) = \varphi_i(x)$ for the first such i found.

end

Intuitively, unify(S) just computes the union of functions computed by programs in S (on inputs where more than one program in S converge but to different values, unify(S) can arbitrarily choose one of the converging programs). It is easy to observe that if S contains at least one decision procedure for L and programs that make only errors of omission in deciding membership in L, then unify(S) is a decision procedure for L. This observation will be useful in extracting (in the limit) a decision procedure for a computable language L from a set of programs F, at least one of which is a decision procedure for L, and a text for L. This is the subject of the next claim.

Claim 1 *Given a finite set of programs, F, and a text T for L, such that at least one of the programs in F is a decision procedure for L, one can find, in the limit, from F and T a decision procedure for L.*

Proof. Let F and a text T for L be given. Without loss of generality assume that range of each program in F is a subset of $\{0,1\}$. We show how to construct a decision procedure for L in the limit.

Let $S_1 = \{i \in F \mid (\exists x \in L)[\varphi_i(x)\downarrow = 0]\}$.

So, programs in S_1 are not decision procedures for L. S_2 below tries to search for programs which accept elements in the complement of L.

Let $S_2 = \{i \in F - S_1 \mid (\exists j \in F - S_1)(\exists x)[\varphi_i(x)\downarrow \neq 0 \wedge \varphi_j(x)\downarrow = 0]\}$. Note that if $i \in F - S_1$ as witnessed by x and j, then $x \notin L$ (since otherwise j would be in S_1).

It should be noted that both S_1 and S_2 can be constructed from F and T in the limit.

We now claim that unify($F - (S_1 \cup S_2)$) is a decision procedure for L. To see this first note that all programs in F which reject an element of L are in S_1. Thus all elements in $F - S_1$ either accept each element of L or diverge on elements of L. Also, since there is a decision procedure for L in F (there exists one such by the assumption), it follows that there is a decision procedure for L in $F - S_1$. Now for any element i in $F - S_1$ such that for some $x \notin L$, $\varphi_i(x)\downarrow = 1$, we have that $i \in S_2$. This follows by the definition of S_2 and the fact that there exists a decision procedure for L in $F - S_1$. Now it is straightforward to see that for each $j \in F - (S_1 \cup S_2)$, for each x, either $\varphi_j(x)\uparrow$ or $\varphi_j(x)$ correctly determines the membership of x in L; that is, j is either a decision procedure for L or only makes errors of omission. It follows that unify($F - (S_1 \cup S_2)$) is a decision procedure for L. □

We now sketch how the above claim can be used to establish Theorem 10. We will first show that for arbitrary m and n, $\mathbf{Team}_n^m\mathbf{TxtExCI} \subseteq \mathbf{Team}_{\lfloor n/m \rfloor}^1\mathbf{TxtExCI}$. From this it follows that, $\mathbf{Team}_{k \cdot n}^{k \cdot m}\mathbf{TxtExCI} \subseteq \mathbf{Team}_{\lfloor n/m \rfloor}^1\mathbf{TxtExCI}$. It is also easy to see that $\mathbf{Team}_{\lfloor n/m \rfloor}^1\mathbf{TxtExCI} \subseteq \mathbf{Team}_n^m\mathbf{TxtExCI}$. The theorem follows.

We now show that

$$\mathbf{Team}_n^m\mathbf{TxtExCI} \subseteq \mathbf{Team}_{\lfloor n/m \rfloor}^1\mathbf{TxtExCI}.$$

Suppose $\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_n$ are given. Suppose further that T is a text for L and these machines $\mathbf{Team}_n^m\mathbf{TxtExCI}$ identify L. We observe the following:

(a) The number of machines converging to a program (perhaps an incorrect one) among $\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_n$ on text T lies in one of the intervals, $[i \cdot m, (i+1) \cdot m)$ where $1 \leq i \leq \lfloor n/m \rfloor$ and

(b) if the number of converging machines lies in the interval $[i \cdot m, (i+1) \cdot m)$, then at least one of the first $i \cdot m$ converging machines on T converges to a decision procedure for L. This

is because at least m machines are guaranteed to converge to a correct decision procedure and the cardinality of the interval $(i \cdot m, (i+1) \cdot m)$ is $m - 1$.

We now construct $\lfloor n/m \rfloor$ machines as follows: machine $\mathbf{M}'_i$ where $1 \leq i \leq \lfloor n/m \rfloor$, on text T, searches for the first $i \cdot m$ machines converging on T. Let F_i be the set of programs to which these machines converge. $\mathbf{M}'_i$ then, assuming that F_i contains a decision procedure for L, using the above claim tries to find a decision procedure for L.

Note that the assumption — F_i contains a decision procedure for L — is true for at least one i, and thus at least one of the $\lfloor n/m \rfloor$ machines succeeds in **TxtExCI**-identifying L.

The above technique can easily be adapted to yield proofs for Theorems 2, 3(a), 10, and 11(a). A modification of the technique can also be used to prove Proposition 2 and Theorems 12 and 13(a).

Our proof of Theorem 14 and 15 depend upon the following weakening of the notion of a locking sequence [4].

Definition 9 (Fulk [15]) *Let machine* $\mathbf{M}$ *and language* L *be given.* σ *is said to be a* stabilizing sequence *for* $\mathbf{M}$ *on* L *just in case the following hold:*

(a) $\text{content}(\sigma) \subseteq L$, *and*

(b) $(\forall \tau \mid \text{content}(\tau) \subseteq L)\ [\mathbf{M}(\sigma) = \mathbf{M}(\sigma \diamond \tau)]$.

So a stabilizing sequence is like a locking sequence except that $\mathbf{M}$'s conjecture on it need not be a grammar for the language. The following technical lemma due to Fulk facilitates the proof of Theorems 14 and 15.

Lemma 1 (Fulk [15]) *Given any machine* $\mathbf{M}$, *one can effectively construct machine* $\mathbf{M}'$ *such that all the following conditions hold.*

(a) $\mathbf{TxtEx}(\mathbf{M}) \subseteq \mathbf{TxtEx}(\mathbf{M}')$.

(b) If there exists a locking sequence for $\mathbf{M}'$ *on* L, *then* $\mathbf{M}'$ *identifies* L.

(c) If $\mathbf{M}'$ *identifies* L, *then all texts for* L *contain a locking sequence for* $\mathbf{M}'$ *on* L.

For the following, let $\alpha_0, \alpha_1, \ldots$ denote a recursive enumeration of all the finite initial sequences.

Proof of Theorem 15

We now prove Theorem 15, which is:

Theorem 15

(a) $(\forall m, n \mid \frac{m}{n} > \frac{2}{3})\ [(\mathbf{Team}_n^m\mathbf{TxtEx} \cap \mathbf{INDEX}) = (\mathbf{TxtEx} \cap \mathbf{INDEX})]$.

(b) $(\mathbf{TxtEx} \cap \mathbf{INDEX}) \subset (\mathbf{Team}_3^2\mathbf{TxtEx} \cap \mathbf{INDEX})$.

Proof of part (a) is straightforward as it is implied by Theorem 6 (a). We show part (b), that is we show that $(\mathbf{TxtEx} \cap \mathbf{INDEX}) \subset (\mathbf{Team}_3^2\mathbf{TxtEx} \cap \mathbf{INDEX})$.

Let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ denote a recursive enumeration of learning machines such that, for all $\mathcal{L} \in \mathbf{TxtEx}$, there exists an i, such that $\mathbf{M}_i$ **TxtEx**-identifies $\mathcal{L}$. (Note that there exists such an enumeration [30]).

We assume, without loss of generality, that no machine converges on any text at the empty sequence Λ. Let $\mathbf{INIT} = \{\{x \mid x \leq n\} \mid n \in N\}$. The idea of the proof is to define an indexed family of computable languages in which the classes $\{N\}$ and **INIT** are embedded. Since $\{N\} \cup \mathbf{INIT}$ is not **TxtEx**-identifiable, it will be so arranged that the resulting class is

not in **TxtEx**. However, the embedding is done in such a way that a team of three machines can be designed, at least two of which are capable of identifying any language in the class.

For $i, j, k \in N$, define the following languages:

$A_i = \{\langle 0, i, x\rangle \mid x \in N\}$.

$B_{i,j} = \{\langle 0, i, x\rangle \mid x \leq j\}$.

$C_{i,j,k} = B_{i,j} \cup \{\langle 1, i, k\rangle\}$.

Let T^i be a text for A_i such that content$(T^i[n+1]) = B_{i,n}$. Define predicate Prop(i, j, t) to be true just in case $(\forall z \leq t \mid T^i[j] \subseteq \alpha_z \ \wedge\ \text{content}(\alpha_z) \subseteq A_i)[\mathbf{M}_i(\alpha_z) = \mathbf{M}_i(T^i[j])]$. Intuitively, Prop$(i, j, t)$ is a bounded test for whether $T^i[j]$ looks like a stabilizing sequence for $\mathbf{M}$ on A_i (t gives a bound on the extensions of $T^i[j]$ which are tested for mind change). Observe that if Prop(i, j, t) is false, then so is Prop$(i, j, t+1)$. Also, observe that if $T^i[j]$ is a stabilizing sequence for machine $\mathbf{M}_i$ on A_i then Prop(i, j, t) is true for all $t \in N$. For $i, j, k \in N$, we now define language $L_{\langle i,j,k\rangle}$ as follows.

$$L_{\langle i,j,k\rangle} = \begin{cases} A_i, & \text{if } j = 0; \\ A_i, & \text{if } j > 0 \text{ and } (\exists j' < j)[\text{Prop}(i, j', k)]; \\ B_{i,j}, & \text{if } j > 0 \text{ and } (\forall j' < j)[\neg\text{Prop}(i, j', k)] \text{ and } (\forall t)[\text{Prop}(i, j, t)]; \\ C_{i,j,l}, & \text{if } j > 0 \text{ and } (\forall j' < j)[\neg\text{Prop}(i, j', k)] \text{ and} \\ & l = \min(\{t \mid \neg\text{Prop}(i, j, t)\}) < \infty. \end{cases}$$

Let $\mathcal{L} = \{L_{\langle i,j,k\rangle} \mid i, j, k \in N\}$. Also, note that $B_{i,j} \in \mathcal{L}$ iff $j = \min(\{j' \mid T^i[j']$ is a stabilizing sequence for $\mathbf{M}_i$ on $A_i\})$. Now the result follows from the following three claims.

Claim 2 *$\mathcal{L}$ is an indexed family.*

Proof. We give a program for a computable function that takes two arguments $\langle i, j, k\rangle$ and x as input and outputs 1 if $x \in$ Ł$_{\langle i,j,k\rangle}$, 0 if $x \notin$ Ł$_{\langle i,j,k\rangle}$. In the algorithm below the phrase "Output $x \in A$" means "Output 1" if $x \in A$; otherwise "Output 0".

begin
  if $(\exists m)[x = \langle 0, i, m\rangle]$ or $(\exists n)[x = \langle 1, i, n\rangle]$ **then**
    $(*\ x$ is of the correct form $*)$
    if $(\exists n)[x = \langle 1, i, n\rangle]$ **then**
      if $(j > 0)\ \wedge\ (\forall j' < j)[\neg\text{Prop}(i, j', k)] \wedge$
      $n = \min(\{t \mid \neg\text{Prop}(i, j, t)\})$ **then**
        Output 1
      else
        Output 0
      endif
    else $(*\ x$ is of the form $\langle 0, i, m\rangle\ *)$
      if $j = 0$ **then**
        Output $x \in A_i$
      else
        if $(\exists j' < j)[\text{Prop}(i, j', k)]$ **then**
          Output $x \in A_i$
        else $(*$ i.e., $(\forall j' < j)[\neg\text{Prop}(i, j', k)]\ *)$
          Output $x \in B_{i,j}$
        endif
      endif

endif
else ($*$ x is not of the correct form $*$)
Output 0
endif
end

It is easy to verify that the above program is a uniform decision procedure for $\mathcal{L}$. □

Claim 3 $\mathcal{L} \notin \mathbf{TxtEx}$.

PROOF. Suppose for contradiction $\mathbf{M}_i$ **TxtEx**-identifies $\mathcal{L}$. Without loss of generality let $\mathbf{M}_i$ satisfy the properties described in Fulk's Lemma 1 above. Therefore, there exists a least j such that $T^i[j]$ is a locking sequence for $\mathbf{M}_i$ on A_i. Let k be such that $(\forall j' < j)[\neg\text{Prop}(i, j', k)]$. But, then $L_{i,j,k} = B_{i,j}$, and on any text for $B_{i,j}$ extending $T^i[j]$, $\mathbf{M}_i$ does not **TxtEx**-identify $B_{i,j}$. □

Claim 4 $\mathcal{L} \in \mathbf{Team}_3^2\mathbf{TxtEx}$.

PROOF. Let g be a computable function such that $W_{g(i)} = A_i$. Let h be a computable function such that

$$W_{h(i)} = \begin{cases} A_i, & \text{if } (\forall j)(\exists t)[\neg\text{Prop}(i, j, t)]; \\ B_{i,j}, & \text{if } j = \min(\{j' \mid (\forall t)[\text{Prop}(i, j, t)]\}). \end{cases}$$

Let $f(i, j, k)$ be a grammar for $C_{i,j,k}$. We now define three machines $\mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3$ which $\mathbf{Team}_3^2\mathbf{TxtEx}$-identify $\mathcal{L}$.

$$\mathbf{M}^1(\sigma) = \begin{cases} g(i), & \text{if } \emptyset \subset \text{content}(\sigma) \subseteq A_i; \\ f(i, j, k), & \text{if } \langle 1, i, k\rangle \in \text{content}(\sigma) \text{ and } j = \max(\{x \mid \langle 0, i, x\rangle \in \text{content}(\sigma)\}) \text{ and} \\ & \text{content}(\sigma) \subseteq C_{i,j,k}; \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathbf{M}^2(\sigma) = \begin{cases} h(i), & \text{if } \emptyset \subset \text{content}(\sigma) \subseteq A_i; \\ f(i, j, k), & \text{if } \langle 1, i, k\rangle \in \text{content}(\sigma) \text{ and } j = \max(\{x \mid \langle 0, i, x\rangle \in \text{content}(\sigma)\}) \text{ and} \\ & \text{content}(\sigma) \subseteq C_{i,j,k}; \\ 0, & \text{otherwise.} \end{cases}$$

Let $\text{match}(i, \sigma) = \max(\{t \mid W_{i,t} \subseteq \text{content}(\sigma) \ \wedge \ \text{content}(\sigma[t]) \subseteq W_{i,|\sigma|}\})$

$$\mathbf{M}^3(\sigma) = \begin{cases} g(i), & \text{match}(g(i), \sigma) > \text{match}(h(i), \sigma); \\ h(i), & \text{otherwise.} \end{cases}$$

Let T be a text for $L \in \mathcal{L}$. If L is of the form $C_{i,j,k}$ for some i, j, k, then clearly, $\mathbf{M}^1, \mathbf{M}^2$ **TxtEx**-identify T. So suppose $L \subseteq A_i$, for some i. Note that $g(i)$ is a grammar for A_i. Moreover, in this case if for all j, $(\exists t)[\neg\text{Prop}(i, j, t)]$, then $h(i)$ is a grammar for A_i else $h(i)$ is a grammar for $B_{i,j}$, where $j = \min(\{j' \mid (\forall t)[\text{Prop}(i, j, t)]\})$. Also, L must be either A_i, or $B_{i,j}$ (where $j = \min(\{j' \mid (\forall t)[\text{Prop}(i, j, t)]\})$). It follows that at least one of $g(i)$ and $h(i)$ is a grammar for L. Now, if $W_{g(i)} = W_{h(i)} = A_i$, then $L = A_i$ and $\mathbf{M}^1, \mathbf{M}^2$, **TxtEx**-identify L; otherwise $\mathbf{M}^3$ and one of $\mathbf{M}^1, \mathbf{M}^2$ **TxtEx**-identify L. □

The above three claims imply the result. ∎

Proof of Theorem 14

We now prove Theorem 14, which is

Theorem 14 ($\mathbf{Team}_2^1\mathbf{TxtEx} \cap \mathbf{INDEX}$) $\subset$ ($\mathbf{Team}_4^2\mathbf{TxtEx} \cap \mathbf{INDEX}$).

This proof may be viewed as a more involved version of the previous proof in the sense that the language class is constructed for diagonalization against two machines instead of one. This is achieved by constructing the language class in such a way that N and **INIT** are embedded twice. Of course the embedding is done with enough clues for a suitable team of four machines at least two of which are successful. The details are as follows.

Let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ denote a recursive enumeration of inductive inference machines such that, for all $\mathcal{L} \in \mathbf{TxtEx}$, there exists an i, such that $\mathbf{M}_i$ **TxtEx**-identifies $\mathcal{L}$. (Note that there exists such an enumeration [30]). We assume, without loss of generality, that no $\mathbf{M}_i$ converges on any text at the empty sequence Λ. Consider the following languages:

$A_{i,j} = \{\langle 0, i, j, x\rangle \mid x \in N\}$.

$B_{i,j,k} = \{\langle 0, i, j, x\rangle \mid x < k\} \cup \{\langle 0, i, j, k + 2x\rangle \mid x \in N\}$.

$C_{i,j,k,l} = \{\langle 0, i, j, x\rangle \mid x < k + 2 + l\}$.

$E_{i,j,k,l} = \{\langle 0, i, j, x\rangle \mid x < k\} \cup \{\langle 0, i, j, k + 2x\rangle \mid x < l\}$.

$F_{i,j,k,l,w} = C_{i,j,k,l} \cup \{\langle 1, i, j, w\rangle\}$.

$G_{i,j,k,l,w} = E_{i,j,k,l} \cup \{\langle 1, i, j, w\rangle\}$.

Let $T^{i,j}$ be a text for $A_{i,j}$ such that $\text{content}(T^{i,j}[n+1]) = \{\langle 0, i, j, x\rangle \mid x \leq n\}$.

Let $T^{i,j,k}$ be a text for $B_{i,j,k}$ such that

$$\text{content}(T^{i,j,k}[n]) = \begin{cases} \{\langle 0, i, j, x\rangle \mid x < n\}, & \text{if } n \leq k; \\ \{\langle 0, i, j, x\rangle \mid x < k\} \cup \{\langle 0, i, j, k + 2x\rangle \mid x < n - k\}, & \text{if } n > k. \end{cases}$$

Let $\text{Prop}(i, j, k, t)$ be true iff $(\exists m \in \{i, j\})(\forall z \leq t \mid T^{i,j}[k] \subseteq \alpha_z \wedge \text{content}(\alpha_z) \subseteq A_{i,j})[\mathbf{M}_m(\alpha_z) = \mathbf{M}_m(T^{i,j}[k])]$.

Intuitively, $\text{Prop}(i, j, k, t)$ is a bounded test for whether $T^{i,j}[k]$ looks like a stabilizing sequence for $\mathbf{M}_i$ or $\mathbf{M}_j$ on $A_{i,j}$ (t gives a bound on the extensions of $T^{i,j}[k]$ which are tested for mind change).

Note that if $\text{Prop}(i, j, k, t)$ is false, then so is $\text{Prop}(i, j, k, t+1)$. Also, if $T^{i,j}[k]$ is a stabilizing sequence for one of $\mathbf{M}_i, \mathbf{M}_j$ on $A_{i,j}$ then $\text{Prop}(i, j, k, t)$ is true for all $t \in N$.

Let $\text{Prop}'(i, j, k, l, t)$ be true iff $\text{Prop}(i, j, k, t)$ and $(\forall m \in \{i, j\})(\forall z \leq t \mid T^{i,j}[k + l] \subseteq \alpha_z \wedge \text{content}(\alpha_z) \subseteq A_{i,j})[\mathbf{M}_m(\alpha_z) = \mathbf{M}_m(T^{i,j}[k + l])]$.

Note that if $\text{Prop}'(i, j, k, l, t)$ is false, then so is $\text{Prop}'(i, j, k, l, t + 1)$. Also, if $T^{i,j}[k]$ is a stabilizing sequence for one of $\mathbf{M}_i, \mathbf{M}_j$ on $A_{i,j}$ and $T^{i,j}[k + l]$ is a stabilizing sequence for both $\mathbf{M}_i, \mathbf{M}_j$ on $A_{i,j}$ then $\text{Prop}'(i, j, k, l, t)$ is true for all $t \in N$.

Let $\text{Prop}''(i, j, k, l, t)$ be true iff $\text{Prop}(i, j, k, t)$ and $(\forall m \in \{i, j\})(\forall z \leq t \mid T^{i,j,k}[k + l] \subseteq \alpha_z \wedge \text{content}(\alpha_z) \subseteq B_{i,j,k})[\mathbf{M}_m(\alpha_z) = \mathbf{M}_m(T^{i,j,k}[k + l])]$.

Note that if $\text{Prop}''(i, j, k, l, t)$ is false, then so is $\text{Prop}''(i, j, k, l, t + 1)$. Also, if $T^{i,j}[k]$ is a stabilizing sequence for one of $\mathbf{M}_i, \mathbf{M}_j$ on $A_{i,j}$ and if $T^{i,j,k}[k + l]$ is a stabilizing sequence for both $\mathbf{M}_i, \mathbf{M}_j$ on $B_{i,j,k}$ then $\text{Prop}''(i, j, k, l, t)$ is true for all $t \in N$.

For each $i, j, k, l, w \in N$, define languages $L^1_{\langle i,j,k,l,w\rangle}$, $L^2_{\langle i,j,k,l,w\rangle}$, and $L^3_{\langle i,j,k,l,w\rangle}$ as follows.

$$L^1_{\langle i,j,k,l,w\rangle} = \begin{cases} A_{i,j}, & \text{if } k = 0; \\ A_{i,j}, & \text{if } k > 0 \text{ and } (\exists k' < k)[\text{Prop}(i, j, k, w)]; \\ B_{i,j,k}, & \text{if } k > 0 \text{ and } (\forall k' < k)[\neg\text{Prop}(i, j, k', w)] \text{ and } (\forall t)[\text{Prop}(i, j, k, t)]; \\ G_{i,j,k,l,s}, & \text{if } k > 0 \text{ and } (\forall k' < k)[\neg\text{Prop}(i, j, k', w)] \text{ and} \\ & s = \min(\{t \mid \neg\text{Prop}(i, j, k, t)\}) < \infty. \end{cases}$$

$$L^2_{\langle i,j,k,l,w\rangle} = \begin{cases} A_{i,j}, & \text{if } [(\exists k' < k)[\text{Prop}(i,j,k',w)] \text{ OR } (\exists l' < l)[\text{Prop}'(i,j,k,l',w)]]; \\ C_{i,j,k,l}, & \text{if } (\forall k' < k)[\neg\text{Prop}(i,j,k',w)] \text{ and } (\forall l' < l)[\neg\text{Prop}'(i,j,k,l',w)] \text{ and} \\ & (\forall t)[\text{Prop}(i,j,k,t)] \text{ and } (\forall t)[\text{Prop}'(i,j,k,l,t)]; \\ F_{i,j,k,l,s}, & \text{if } (\forall k' < k)[\neg\text{Prop}(i,j,k',w)] \text{ and } (\forall l' < l)[\neg\text{Prop}'(i,j,k,l',w)] \text{ and} \\ & s = \min(\{t \mid \neg\text{Prop}(i,j,k,t) \ \vee \ \neg\text{Prop}'(i,j,k,l,t)\}). \end{cases}$$

$$L^3_{\langle i,j,k,l,w\rangle} = \begin{cases} A_{i,j}, & \text{if } [(\exists k' < k)[\text{Prop}(i,j,k',w)] \text{ OR } (\exists l' < l)[\text{Prop}''(i,j,k,l',w)]]; \\ E_{i,j,k,l}, & \text{if } (\forall k' < k)[\neg\text{Prop}(i,j,k',w)] \text{ and } (\forall l' < l)[\neg\text{Prop}''(i,j,k,l',w)] \text{ and} \\ & (\forall t)[\text{Prop}(i,j,k,t)] \text{ and } (\forall t)[\text{Prop}''(i,j,k,l,t)]; \\ G_{i,j,k,l,s}, & \text{if } (\forall k' < k)[\neg\text{Prop}(i,j,k',w)] \text{ and } (\forall l' < l)[\neg\text{Prop}''(i,j,k,l',w)] \text{ and} \\ & s = \min(\{t \mid \neg\text{Prop}(i,j,k,t) \ \vee \ \neg\text{Prop}''(i,j,k,l,t)\}). \end{cases}$$

It is easy to verify that $\mathcal{L} = \{L^m_{\langle i,j,k,l,w\rangle} \mid i,j,k,l,w \in N, \ \wedge \ m \in \{1,2,3\}\}$ is an indexed family. We claim that $\mathcal{L}$ witnesses the theorem.

Suppose for contradiction that machines $\mathbf{M}_i, \mathbf{M}_j$ $\mathbf{Team}^1_2\mathbf{TxtEx}$-identify $\mathcal{L}$. Without loss of generality, assume that $\mathbf{M}_i, \mathbf{M}_j$ satisfy Lemma 1. Then, there exists a least k such that $T^{i,j}[k]$ is a stabilizing sequence for one of $\mathbf{M}_i$, $\mathbf{M}_j$ on $A_{i,j}$. Without loss of generality, let us assume that it is $\mathbf{M}_i$. Thus, $B_{i,j,k} \in \mathcal{L}$ ($L^1_{i,j,k,l,w}$ for large enough w will be $B_{i,j,k}$). Now suppose $W_{\mathbf{M}_i(T^{i,j}[k])}$ enumerates $\langle 0,i,j,k+1\rangle$ (a similar argument can be given for the case where $W_{\mathbf{M}_i(T^{i,j}[k])}$ does not enumerate $\langle 0,i,j,k+1\rangle$). Thus, $\mathbf{M}_i$ cannot identify $B_{i,j,k}$ or any of $E_{i,j,k,l}$. Thus, there must exists a least l such that $T^{i,j,k}[k+l]$ is a stabilizing sequence for $\mathbf{M}_j$ on $B_{i,j,k}$. Thus, $E_{i,j,k,l} \in \mathcal{L}$ ($L^3_{i,j,k,l,w}$ for large enough w is $E_{i,j,k,l}$). However, $\mathbf{M}_j$ can $\mathbf{TxtEx}$-identify at most one of $B_{i,j,k}$, and $E_{i,j,k,l}$. It follows that $\mathbf{M}_i, \mathbf{M}_j$ do not $\mathbf{Team}^1_2\mathbf{TxtEx}$-identify $\mathcal{L}$.

We now show that $\mathcal{L} \in \mathbf{Team}^2_4\mathbf{TxtEx}$. Let g be recursive function such that for all i,j, $W_{g(i,j)} = A_{i,j}$. Let recursive functions h, f_1, f_2 be as defined below:

$$W_{h(i,j)} = \begin{cases} A_{i,j}, & \text{if } (\forall k)(\exists w)[\neg\text{Prop}(i,j,k,w)]; \\ B_{i,j,k}, & \text{if } k = \min(\{k' \mid (\forall w)[\text{Prop}(i,j,k',w)]\}) < \infty. \end{cases}$$

$$W_{f_1(i,j,k)} = \begin{cases} A_{i,j}, & \text{if } (\forall l)(\exists w)[\neg\text{Prop}'(i,j,k,l,w)]; \\ C_{i,j,k,l}, & \text{if } l = \min(\{l' \mid (\forall w)[\text{Prop}'(i,j,k,l',w)]\}). \end{cases}$$

$$W_{f_2(i,j,k)} = \begin{cases} B_{i,j,k}, & \text{if } (\forall l)(\exists w)[\neg\text{Prop}''(i,j,k,l,w)]; \\ E_{i,j,k,l}, & \text{if } l = \min(\{l' \mid (\forall w)[\text{Prop}''(i,j,k,l',w)]\}). \end{cases}$$

Define $\mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3, \mathbf{M}^4$ as follows.

Let F be a recursive function such that $W_{F(D)} = D$, for all finite sets D.

$$\mathbf{M}^1(\sigma) = \begin{cases} g(i,j), & \text{if content}(\sigma) \subseteq A_{i,j}; \\ F(\text{content}(\sigma)), & \text{if } \neg(\exists i,j)[\text{content}(\sigma) \subseteq A_{i,j}]. \end{cases}$$

$$\mathbf{M}^2(\sigma) = \begin{cases} h(i,j), & \text{if content}(\sigma) \subseteq A_{i,j}; \\ F(\text{content}(\sigma)), & \text{if } \neg(\exists i,j)[\text{content}(\sigma) \subseteq A_{i,j}]. \end{cases}$$

Note that if the input text is for a language $L \in \mathcal{L}$, which is not a subset of any $A_{i,j}$, then $\mathbf{M}^1, \mathbf{M}^2$ identify T. Further note that if, for all k, $(\exists w)[\neg\text{Prop}(i,j,k,w)]$, then $W_{g(i,j)} = W_{h(i,j)} = A_{i,j}$, and $A_{i,j}$ is the only subset of $A_{i,j}$, which is in the class $\mathcal{L}$. Thus, for the definition of $\mathbf{M}^3, \mathbf{M}^4$ below we assume that the input language is contained in some $A_{i,j}$, and for this

value of i, j, there exists a minimum k such that $(\forall w)[\text{Prop}(i, j, k, w)]$. We let $k_{i,j}$ denote such a k. We assume that the function match is as defined in the proof of Theorem 15.

$$\mathbf{M}^3(\sigma) = \begin{cases} f_1(i,j,k_{i,j}), & \text{if content}(\sigma) \subseteq A_{i,j}, \text{ and} \\ & \text{match}(g(i,j),\sigma) > \text{match}(h(i,j),\sigma); \\ f_2(i,j,k_{i,j}), & \text{otherwise.} \end{cases}$$

$$\mathbf{M}^4(\sigma) = \begin{cases} g(i,j), & \text{if content}(\sigma) \subseteq A_{i,j} \text{ and match}(g(i,j),\sigma) \geq \\ & \max(\{\text{match}(g(i,j),\sigma), \text{match}(h(i,j),\sigma), \\ & \text{match}(f_1(i,j,k_{i,j}),\sigma), \text{match}(f_2(i,j,k_{i,j}),\sigma)\}); \\ h(i,j), & \text{if content}(\sigma) \subseteq A_{i,j} \text{ and match}(h(i,j),\sigma) \geq \\ & \max(\{\text{match}(g(i,j),\sigma), \text{match}(h(i,j),\sigma), \\ & \text{match}(f_1(i,j,k_{i,j}),\sigma), \text{match}(f_2(i,j,k_{i,j}),\sigma)\}); \\ f_1(i,j,k_{i,j}), & \text{if content}(\sigma) \subseteq A_{i,j} \text{ and match}(f_1(i,j,k_{i,j}),\sigma) \geq \\ & \max(\{\text{match}(g(i,j),\sigma), \text{match}(h(i,j),\sigma), \\ & \text{match}(f_1(i,j,k_{i,j}),\sigma), \text{match}(f_2(i,j,k_{i,j}),\sigma)\}); \\ f_2(i,j,k_{i,j}), & \text{if content}(\sigma) \subseteq A_{i,j} \text{ and match}(f_2(i,j,k_{i,j}),\sigma) \geq \\ & \max(\{\text{match}(g(i,j),\sigma), \text{match}(h(i,j),\sigma), \\ & \text{match}(f_1(i,j,k_{i,j}),\sigma), \text{match}(f_2(i,j,k_{i,j}),\sigma)\}). \end{cases}$$

Now for any L in $\mathcal{L}$ such that $L \subseteq A_{i,j}$, and $W_{g(i,j)} \neq W_{h(i,j)}$, it is easy to verify that, at least one and at most two of $g(i,j)$, $h(i,j)$, $f_1(i,j,k_{i,j})$, $f_2(i,j,k_{i,j})$ is a grammar for L. Furthermore, if two of the above are grammars for L, then they must either be $g(i,j)$, and $f_1(i,j,k_{i,j})$ or $h(i,j)$ and $f_2(i,j,k_{i,j})$. Further, if $g(i,j), f_1(i,j,k_{i,j})$ are grammars for L, then $\text{match}(T[n], g(i,j)) > \text{match}(T[n], h(i,j))$ for any text T for L and large enough n. Similarly, if $h(i,j), f_2(i,j,k_{i,j})$ are grammars for L, then $\text{match}(T[n], g(i,j)) < \text{match}(T[n], h(i,j))$ for any text T for L and large enough n. It then follows from the above that at least two of $\mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3, \mathbf{M}^4$ **TxtEx**-identify L. The theorem follows. ∎

References

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

[2] K Apsitis, R. Freivalds, and C. Smith. Choosing a learning team: a topological approach. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computation*, pages 283–289, 1994.

[3] K Apsitis, R. Freivalds, and C. Smith. Asymmetric team learning. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pages 90–95. ACM Press, 1997.

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

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

[6] J. Case, S. Jain, and S. Ngo Manguelle. Refinements of inductive inference by Popperian and reliable machines. *Kybernetika*, 30:23–52, 1994.

[7] R. Daley. On the error correcting power of pluralism in BC-type inductive inference. *Theoretical Computer Science*, 24:95–104, 1983.

[8] R. Daley and B. Kalyanasundaram. Capabilities of probabilistic learners with bounded mind changes. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 182–191. ACM Press, 1993.

[9] R. Daley and B. Kalyanasundaram. Use of reduction arguments in determining Popperian FIN-type learning capabilities. In K. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Algorithmic Learning Theory: Fourth International Workshop (ALT '93)*, volume 744 of *Lecture Notes in Artificial Intelligence*, pages 173–186. Springer-Verlag, 1993.

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

[11] R. Daley, B. Kalyanasundaram, and M. Velauthapillai. Capabilities of fallible finite learning. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 199–208. ACM Press, 1993.

[12] R. Daley, L. Pitt, M. Velauthapillai, and T. Will. Relations between probabilistic and team one-shot learners. In L. Valiant and M. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 228–239. Morgan Kaufmann, 1991.

[13] R. Freivalds. Functions computable in the limit by probabilistic machines. In *Mathematical Foundations of Computer Science*, volume 28 of *Lecture Notes in Computer Science*, pages 77–87. Springer-Verlag, 1974.

[14] R. Freivalds. Finite identification of general recursive functions by probabilistic strategies. In *Proceedings of the Conference on Fundamentals of Computation Theory*, pages 138–145. Akademie-Verlag, Berlin, 1979.

[15] M. Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.

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

[17] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[18] S. Jain and A. Sharma. Finite learning by a team. In M. Fulk and J. Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 163–177. Morgan Kaufmann, 1990.

[19] S. Jain and A. Sharma. Language learning by a team. In M. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 153–166. Springer-Verlag, 1990.

[20] S. Jain and A. Sharma. On aggregating teams of learning machines. In K. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Algorithmic Learning Theory: Fourth International Workshop (ALT '93)*, volume 744 of *Lecture Notes in Artificial Intelligence*, pages 150–163. Springer-Verlag, 1993.

[21] S. Jain and A. Sharma. Probability is more powerful than team for language identification. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 192–198. ACM Press, 1993.

[22] S. Jain and A. Sharma. On aggregating teams of learning machines. *Theoretical Computer Science*, 137:85–108, 1995.

[23] S. Jain and A. Sharma. Computational limits on team identification of languages. *Information and Computation*, 130:19–60, 1996.

[24] S. Jain, A. Sharma, and M. Velauthapillai. Finite identification of function by teams with success ratio 1/2 and above. *Information and Computation*, 121:201–213, 1995.

[25] E. Kinber and T. Zeugmann. One-sided error probabilistic inductive inference and reliable frequency identification. *Information and Computation*, 92:253–284, 1991.

[26] S. Lange and T. Zeugmann. Learning recursive languages with a bounded number of mind changes. *International Journal of Foundations of Computer Science*, 4(2):157–178, 1993.

[27] L. Meyer. Probabilistic language learning under monotonicity constraints. In K. Jantke, T. Shinohara, and T. Zeugmann, editors, *Algorithmic Learning Theory: Sixth International Workshop (ALT '95)*, volume 997 of *Lecture Notes in Artificial Intelligence*, pages 169–184. Springer-Verlag, 1995.

[28] L. Meyer. Monotonic and dual-monotonic probabilistic language learning. In S. Ben-David, editor, *Third European Conference on Computational Learning Theory*, volume 1208 of *Lecture Notes in Artificial Intelligence*, pages 66–78. Springer-Verlag, 1997.

[29] D. Osherson, M. Stob, and S. Weinstein. Aggregating inductive expertise. *Information and Control*, 70:69–95, 1986.

[30] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, 1986.

[31] L. Pitt. *A characterization of probabilistic inference*. PhD thesis, Yale University, 1984.

[32] L. Pitt. Probabilistic inductive inference. *Journal of the ACM*, 36:383–433, 1989.

[33] L. Pitt and C. Smith. Probability and plurality for aggregations of learning machines. *Information and Computation*, 77:77–92, 1988.

[34] K. Podnieks. Comparing various concepts of function prediction, Part I. In *Theory of Algorithms and Programs, vol. 1*, pages 68–81. Latvian State University, Riga, Latvia, 1974.

[35] C. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982.

[36] C. Smith. Three decades of team learning. In S. Arikawa and K. Jantke, editors, *Algorithmic learning theory: Fourth International Workshop on Analogical and Inductive Inference (AII '94) and Fifth International Workshop on Algorithmic Learning Theory (ALT '94)*, volume 872 of *Lecture Notes in Artificial Intelligence*, pages 211–228. Springer-Verlag, 1994.

[37] G. Weiss and S. Sen, editors. *Adaptation and Learning in Multi-Agent Systems*, volume 1042 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1996.

[38] R. Wiehagen, R. Freivalds, and E. Kinber. On the power of probabilistic strategies in inductive inference. *Theoretical Computer Science*, 28:111–133, 1984.

[39] T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In K. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer-Verlag, 1995.