



Trees and learning

Wolfgang Merkle^{a,*},¹ and Frank Stephan^{b,2}

^a *Ruprecht-Karls-Universität Heidelberg, Institut für Informatik, Im Neuenheimer Feld 294, 69120 Heidelberg, Germany*

^b *NICTA, Sydney Node, The University of New South Wales, Sydney NSW 2052, Australia*

Received 11 November 2002; revised 6 August 2003

Abstract

We characterize FIN-, EX- and BC-learning, as well as the corresponding notions of team learning, in terms of isolated branches on effectively given sequences of trees. The more restrictive models of FIN-learning and strong-monotonic BC-learning are characterized in terms of isolated branches on a single tree. Furthermore, we discuss learning with additional information where the learner receives an index for a strongly recursive tree such that the function to be learned is isolated on this tree. We show that EX-learning with this type of additional information is strictly more powerful than EX-learning.

© 2003 Elsevier Inc. All rights reserved.

1. Introduction

Inductive inference [1,2,4,6,9,16,17] deals with the learning of classes of recursive functions in the limit. A learner is a Turing machine that receives for a function $f: \mathbb{N} \rightarrow \mathbb{N}$ successively the prefixes

$$f(0), \quad f(0)f(1), \quad f(0)f(1)f(2), \dots;$$

for each prefix, the learner guesses an index for f (or abstains from guessing by outputting a special symbol “?”) and the learner succeeds in learning f if almost all guesses correctly describe f . Depending on the type of convergence required, we obtain different models of learning. In the setting of behaviorally correct learning (BC), we simply require semantical convergence, i.e., it is required that almost all guesses are correct in the sense that they compute f . Explanatory learning

*Corresponding author.

E-mail addresses: merkle@math.uni-heidelberg.de (W. Merkle), fstephan@cse.unsw.edu.au (F. Stephan).

¹Partially supported by the EU network *Complexity, Logic and Recursion Theory* (COLORET) EU Contract ERBCHRXCT930415.

²Supported by the Deutsche Forschungsgemeinschaft (DFG) under Grants Am 60/9-1 and Ste 967/1-1.

(EX) is more restrictive; here, in order to learn a function, the learner has to converge syntactically, i.e., the output must be almost always the same correct program for f . While the EX-learner might change its guess finitely often before converging to the correct program, in the most restrictive setting of finite learning (FIN) the learner may first abstain from guessing (again indicated by outputting the special symbol “?” for “no guess”) but eventually has to output a correct program for f , then sticking to the first guessed program forever.

Now, if for any of these settings there is a single learner that learns all functions in some class \mathcal{S} , then \mathcal{S} is said to be in BC, EX, or FIN, respectively. Furthermore, for each of these settings we can define a corresponding notion of team learning where the inference is done by a fixed team of finitely many machines working in parallel [20]. The team learns a single function if and only if it is learned by some machine in the team and the team learns a class of functions if and only if there is some fixed team that learns all functions in the class.

We show that all concepts of learning in the limit mentioned so far have natural characterizations in terms of effectively given sequences of trees. For example in the case of $\{0, 1\}$ -valued functions, a class S is in BC if and only if there is a uniformly recursive sequence T_0, T_1, \dots of binary trees such that each $f \in S$ is an isolated branch on some tree T_n and is not on any other tree T_m . These characterizations correspond to the well-known fact from recursion theory that each isolated branch of a recursively bounded recursive tree is itself recursive, see Lemma 2.2 for details.

Wiehagen et al. [23,24] give alternate characterizations of FIN, EX, and BC via enumerations of indices of partial recursive functions that satisfy additional requirements. We show in the case of $\{0, 1\}$ -valued functions that for all these models of learning there is an easy and natural way to pass from our characterizations in terms of sequences of trees to the corresponding characterizations in terms of enumerations.

Furthermore, we characterize the learnable classes in more restrictive models by means of single effectively given trees. For finite learning with and without the oracle K we obtain a characterization in terms of fully isolated branches on recursively enumerable trees and we show that strong-monotonic BC, as introduced by Wiehagen [22], can be characterized in terms of the isolated branches on a single recursive tree.

Finally we discuss learning with additional information where the learner receives an index for a strongly recursive tree such that the function to be learned is isolated on this tree. Case et al. [3] show that the class of all recursive functions cannot be learned in the corresponding model of EX-learning but can be learned in the corresponding model of BC-learning, hence in particular the latter model is strictly more powerful than standard BC-learning. These results suggest to ask whether additional information of this type helps an EX-learner at all. We give an affirmative answer to this question; EX-learning with additional information as described above is strictly more powerful than EX-learning without.

2. Notation and some basic facts

All functions are total unless explicitly attributed as being partial. Furthermore, the term function refers to a function from \mathbb{N} to \mathbb{N} unless explicitly stated otherwise. A string is a finite sequence of natural numbers and a string is binary if and only if it contains only the numbers 0

and 1. For a string σ , we write $|\sigma|$ for its LENGTH and we identified σ in the natural way with an \mathbb{N} -valued function with domain $\text{dom}(\sigma) = \{0, \dots, |\sigma| - 1\}$. Moreover, we let the NORM of σ be defined by

$$\|\sigma\| = |\sigma| + \sum_{x \in \text{dom}(\sigma)} \sigma(x).$$

The EMPTY STRING is denoted by λ . We write $\tau \preceq \sigma$ if τ is a PREFIX of σ , i.e., if $\text{dom}(\tau)$ is contained in $\text{dom}(\sigma)$ and τ and σ agree on $\text{dom}(\tau)$. Similarly, for a function f from \mathbb{N} to \mathbb{N} , we write $\tau \preceq f$ if τ agrees on its domain with f . We let $\varphi_0, \varphi_1, \dots$ be the standard enumeration of all partial recursive functions from \mathbb{N} to \mathbb{N} , and we write $\tau \preceq \varphi_i$ if and only if at all places in the domain of τ , φ_i is defined and agrees with τ .

In subsequent characterizations of FIN-, EX- and BC-learning we will use the following notation and facts related to trees. As usual, a TREE is a set of strings that is closed downwards under the prefix-relation, and a tree is binary if and only if it contains only binary strings.

Definition 2.1. Let T be a tree, let f be a function, and let σ be a string.

- The function f is an INFINITE BRANCH of T (or: is ON T) if and only if all prefixes $\sigma \preceq f$ are in T .
- T is FINITELY BRANCHING ABOVE σ if for every $\tau \in T$ with $\sigma \preceq \tau$, the set $\{y \in \mathbb{N}: \tau y \in T\}$ is finite.
- The function f is ISOLATED ON T ABOVE σ if and only if, first, f is the only function on T that satisfies $\sigma \preceq f$ and, second, T is finitely branching above σ . The function f is FULLY ISOLATED ON T ABOVE σ if and only if $\sigma \preceq f$ and among all $\tau \succ \sigma$ exactly those with $\tau \preceq f$ are in T . The function f is ISOLATED (FULLY ISOLATED) ON T if and only if f is isolated (fully isolated) on T above some σ .
- The CLOSURE of T is the set $\text{Cl}[T] = \{\sigma z: z \in \mathbb{N} \wedge \exists y \in \mathbb{N}[y \geq z \wedge \sigma y \in T]\}$. The tree T is STRONGLY RECURSIVE if and only if T and $\text{Cl}[T]$ are both recursive. A program e that decides (the marked union $\{0\sigma: \sigma \in T\} \cup \{1\tau: \tau \in \text{Cl}[T]\}$ of) both sets is called a STRONGLY RECURSIVE INDEX for T . A sequence T_0, T_1, \dots of trees is UNIFORMLY STRONGLY RECURSIVE if and only if there is a recursive function g such that $g(k)$ is a strongly recursive index for T_k .

If restricted to finitely branching trees, the concept of isolated branch can be equivalently defined by requiring that some prefix of the branch is not extended by any other infinite branch on the given tree. In particular, the latter remark applies to recursively bounded trees, i.e., to trees where there is a recursive function that on input σ gives an upper bound on the numbers y such that the tree contains σy . It is well known that if an infinite branch is isolated on a recursively bounded recursive tree, then the branch is recursive (see [15], the proof of Proposition V.5.27 and the remarks on recursively bounded trees at the end of the same section). The latter result extends by virtually the same proof to isolated branches on a strongly recursive tree.

Lemma 2.2. *There is a recursive function ib such that whenever e is a strongly recursive index for a tree T and f is isolated on T above σ , then $\varphi_{\text{ib}(e,\sigma)} = f$.*

Proof. We specify an effective procedure to compute the function $\varphi_{\text{ib}(e,\sigma)}$. For a start, we let $\varphi_{\text{ib}(e,\sigma)}$ agree with σ on all places in $\text{dom}(\sigma)$ and then the function is extended inductively.

Assuming that $\varphi_{\text{ib}(e,\sigma)}$ has already been defined on the set $\{0, \dots, x - 1\}$, where we denote the restriction of $\varphi_{\text{ib}(e,\sigma)}$ to this set by τ , in the induction step we try to define $\varphi_{\text{ib}(e,\sigma)}(x)$ as follows.

- Search for a natural number z such that τz is not in $\text{Cl}[T]$, i.e., such that T does not contain any string of the form τy with $y \in \mathbb{N}$ and $z \leq y$.
- Enumerate the set E_τ of all $y < z$ for which the set $\{\eta \in T : \tau y \preceq \eta\}$ is finite, until all but a single number $y_0 < z$ are enumerated into E_τ .
- Let $\varphi_{\text{ib}(e,\sigma)}(x) = y_0$.

It is to be understood that every next instruction of the induction step is reached if and only if all preceding instructions have been completed successfully and that in case the last instruction is not reached, $\varphi_{\text{ib}(e,\sigma)}$ will remain undefined at x . In the latter case, $\varphi_{\text{ib}(e,\sigma)}$ will also be undefined at all places larger than x .

Assume now that f is isolated on T_e above σ . We show inductively that $\varphi_{\text{ib}(e,\sigma)}(x) \downarrow = f(x)$ for all x . This holds by definition for all $x \in \text{dom}(\sigma)$. So assume that in the induction step we are given τ and x as in the inductive definition of $\varphi_{\text{ib}(e,\sigma)}$, where x is not in $\text{dom}(\sigma)$. By the induction hypothesis, we can assume $\sigma \preceq \tau \preceq f$, hence T is not only finitely branching above σ , but also above τ . Thus the search in the first instruction terminates and yields an upper bound z for $f(x)$. By König’s Lemma, moreover, for every natural number y , the set $\{\eta \in T : \tau y \preceq \eta\}$ is infinite if and only if there is an infinite branch on T that extends τy . But by f being isolated above σ such an infinite branch exists exactly for $y = f(x)$, hence among all natural numbers $y < z$, exactly the $y \neq f(x)$ are enumerated into E_τ and $\varphi_{\text{ib}(e,\sigma)}(x)$ is set equal to $f(x)$. \square

Remark 2.3. The function ib obeys a “monotonicity property” that we will use in the proof of Theorem 7.4. Let e be a strongly recursive index e for a tree T , let f be a function on T , and let σ_0 and σ_1 be strings with $\sigma_0 \preceq \sigma_1 \preceq f$. Then $\varphi_{\text{ib}(e,\sigma_1)}$ extends $\varphi_{\text{ib}(e,\sigma_0)}$ in the sense that whenever $\varphi_{\text{ib}(e,\sigma_0)}$ is defined, $\varphi_{\text{ib}(e,\sigma_1)}$ is defined, too, and the two functions have the same value.

For a proof, observe that by construction of ib for $i = 0, 1$, if $\varphi_{\text{ib}(e,\sigma_i)}$ is defined at all, then it must agree with f . First assume that $\varphi_{\text{ib}(e,\sigma_0)}$ is undefined at some place in $\text{dom}(\sigma_1)$, hence is undefined at all places outside of $\text{dom}(\sigma_1)$. Then we are done, because $\varphi_{\text{ib}(e,\sigma_1)}$ is defined at all places in $\text{dom}(\sigma_1)$. On the other hand, if $\varphi_{\text{ib}(e,\sigma_0)}$ is defined at all places in $\text{dom}(\sigma_1)$, then $\varphi_{\text{ib}(e,\sigma_0)}$ and $\varphi_{\text{ib}(e,\sigma_1)}$ are identical because both agree with f on $\text{dom}(\sigma_1)$, and at the remaining places they are defined by the same inductive process that starts with σ_1 .

The following Lemma 2.5 will be useful in connection with the subsequent characterizations of learnable classes in terms of isolated branches of uniformly strongly recursive sequences of trees.

Definition 2.4. A tree T is STRONGLY CO-R.E. if and only if the complements of T and of $\text{Cl}[T]$ are both r.e. A program e that enumerates (the marked union of) the two latter sets is called a STRONGLY CO-R.E. INDEX for T . A sequence T_0, T_1, \dots of trees is UNIFORMLY STRONGLY CO-R.E. if and only if there is a recursive function g such that $g(k)$ is a strongly co-r.e. index for T_k .

Lemma 2.5. *If U_0, U_1, \dots is a uniformly strongly co-r.e. sequence of trees then there is a uniformly strongly recursive sequence of trees T_0, T_1, \dots such that for all $n \in \mathbb{N}$,*

- (a) $U_n \subseteq T_n$,
- (b) U_n and T_n have the same infinite branches,
- (c) for all functions f and all σ , f is isolated above σ on U_n if and only if f is isolated above σ on T_n (hence in particular U_n and T_n have the same isolated branches).

Proof. Fix a recursive function g such that $g(n)$ is a strongly co-r.e. index for U_n . For all n and t , let U_n^t and $\text{Cl}[U_n]^t$ be the sets of all strings that have not been enumerated into the complement of U_n and $\text{Cl}[U_n]$, respectively, after t steps of $g(n)$, i.e., the sequence U_n^0, U_n^1, \dots converges decreasingly to U_n , and likewise for the $\text{Cl}[U_n]^t$ and $\text{Cl}[U_n]$. Then we obtain a sequence T_0, T_1, \dots as required if we let for all strings σ and all natural numbers z

$$\sigma z \in T_n \Leftrightarrow \left[(\forall \tau \preceq \sigma z) [\tau \in U_n^{||\sigma z||}] \right] \text{ and } (\forall y \preceq z) [\sigma y \in \text{Cl}[U_n]^{||\sigma z||}]. \quad (1)$$

Fix n . First consider an arbitrary string $\rho = \sigma z$ in U_n . Then $g(n)$ will neither enumerate a prefix of σ into the complement of U_n nor any string σy with $y \preceq z$ into the complement of $\text{Cl}[U_n]$, hence ρ is in T_n . As a consequence, U_n is a subset of T_n and in particular every infinite branch on U_n is also on T_n . Now assume for a contradiction that there is a function f that is an infinite branch on T_n but not on U_n . Then there is a $\tau \prec f$ that is not in U_n and hence is in U_n^t for some t . But then by definition, T_n will not contain any extension of τ of norm at least t , which contradicts our assumption that all prefixes of f are contained in T_n . Finally, the sets of isolated branches on U_n and T_n coincide by (b) and because by the second conjunct in (1) for all σ , the set $\{z \in \mathbb{N} : \sigma z \in U_n\}$ is finite if and only if the set $\{z \in \mathbb{N} : \sigma z \in T_n\}$ is finite. \square

3. Characterizations of the basic models

In this section, we characterize the FIN-, EX-, and BC-learnable classes in terms of isolated paths and uniformly strongly recursive sequences of trees.

Theorem 3.1. *A class of functions \mathcal{S} is in EX if and only if there is a uniformly strongly recursive sequence T_0, T_1, \dots of trees such that for every $f \in \mathcal{S}$, there is some n where f is isolated on T_n above λ and f is not on T_m for all $m \neq n$.*

Proof. (\Leftarrow): Assume that T_0, T_1, \dots is a uniformly strongly recursive sequence of trees as in the theorem. Fix a recursive function g such that $g(n)$ is a strong recursive index for T_n and let ib be the recursive function from Lemma 2.2, i.e., in particular $\text{ib}(g(n), \lambda)$ is an index for f if f is isolated on T_n above λ .

Now the following inference algorithm learns \mathcal{S} as required. On input σ the algorithm outputs $\text{ib}(g(n), \lambda)$ for the first $n \leq |\sigma|$ such that σ is on T_n and, if there is no such n , the algorithm outputs the symbol “?” to indicate that the algorithm makes no guess. In order to verify that this

algorithm identifies in the limit every function $f \in \mathcal{S}$, it suffices to observe that by assumption for every such f there is some n such that the function f is isolated on T_n above λ but is not on any other tree T_m with $m < n$. As a consequence, every sufficiently long $\sigma \preceq f$ is contained in T_n but not in T_0, T_1, \dots, T_{n-1} and the algorithm eventually converges to the index $\text{ib}(g(n), \lambda)$ for f .

(\Rightarrow): Assume that a learner M EX-learns a class of functions S . By applying the padding-lemma we can assume $M(\sigma) > |\sigma|$ whenever M has a mind change at σ , that is, whenever $M(\tau a) \neq M(\tau)$ where $\sigma = \tau a$. By this assumption, if M converges to some index n on some function f to be learned, then M converges already on prefixes of f of length smaller than n . As a consequence, for all functions f and all n , M converges on f to n if and only if f is on the tree

$$R_n = \{\sigma : (\forall \tau \prec \sigma)[|\tau| \geq n \Rightarrow M(\tau) = n]\}.$$

We have $R_n = \text{Cl}[R_n]$ because membership of a string in R_n does not depend on its last symbol, i.e., for all ρ , the tree R_n either contains all strings of the form ρy , $y \in \mathbb{N}$, or contains none of these strings. As a consequence, the sequence R_0, R_1, \dots , which is obviously uniformly recursive, is indeed uniformly strongly recursive. Furthermore, for every n , we let P_n be the set of prefixes of the total extensions of the n th partial recursive function φ_n , i.e.,

$$P_n = \{\sigma : \forall x \in \text{dom}(\sigma)[\varphi_n(x) \uparrow \text{ or } \varphi_n(x) \downarrow = \sigma(x)]\} \tag{2}$$

and we let $U_n = P_n \cap R_n$. Then the sequence U_0, U_1, \dots is uniformly strongly co-r.e., because both the sequences R_0, R_1, \dots and P_0, P_1, \dots are, hence by Lemma 2.5 it is sufficient to show that the U_n satisfy the specification of the T_n . First observe that a function cannot be on R_m unless M converges on this function to m , hence every function is on at most one of the trees U_n . Next fix f in S . Then M converges on f to some index n with $\varphi_n = f$. So by construction f is on P_n and on R_n , and hence on U_n . Furthermore, f is isolated on U_n above λ because P_n consists just of the prefixes of f . \square

The proof of Theorem 3.1 can be modified to obtain a characterization of finite learning.

Theorem 3.2. *A class of functions \mathcal{S} is in FIN if and only if there is a uniformly strongly recursive sequence T_0, T_1, \dots of trees such that for every function $f \in \mathcal{S}$, there is some n where f is isolated on T_n above λ while for all $m \neq n$ the string $f(0)f(1)\dots f(m)$ is not in T_m .*

Proof. (\Leftarrow): Assume that T_0, T_1, \dots is a uniformly strongly recursive sequence of trees as in the theorem. Fix a recursive function g such that $g(n)$ is a strong recursive index for T_n . Then we obtain a finite learner for \mathcal{S} as follows. On data $f(0)f(1)\dots$, the learner waits for the first n with $f(0)\dots f(n) \in T_n$ and then outputs $\text{ib}(g(n), \lambda)$, where ib is the function from Lemma 2.2. The verification of the inference algorithm is similar to the corresponding argument in the proof of Theorem 3.1 and is left to the reader.

(\Rightarrow): Assume that M is a finite learner for the class \mathcal{S} . Then the set

$$S = \{\sigma : M(\sigma) \neq ? \wedge (\forall \tau \prec \sigma)[M(\tau) = ?]\}$$

of all strings on which M makes a guess for the first time, is recursive. The set S is prefix-free, i.e., for every pair σ, ρ of distinct strings in S , we have $\sigma \not\prec \rho$ and $\rho \not\prec \sigma$. Fix an appropriate effective enumeration $\sigma_0, \sigma_1, \dots$ of all strings where each string appears exactly once and for all n , we have

$|\sigma_n| \leq n$. For every n , let $U_n = \emptyset$ if and only if $\sigma_n \notin S$. Otherwise, i.e., if $\sigma_n \in S$, let U_n be the tree $P_{M(\sigma_n)}$ as defined in (2). Then the sequence U_0, U_1, \dots is uniformly strongly co-r.e. Apply Lemma 2.5 in order to obtain a uniformly strongly recursive sequence V_0, V_1, \dots and let $T_n = \{\tau \in V_n: \tau \preceq \sigma_n \vee \sigma_n \preceq \tau\}$ in case $\sigma_n \in S$ and, otherwise, let $T_n = \emptyset$.

In order to show that the T_n have the required property, fix f in \mathcal{S} and let σ_n be the unique prefix of f in S . Then $M(\sigma_n)$ is an index for f and the tree $P_{M(\sigma_n)}$ coincides with the set of all prefixes of f , thus f is isolated above λ on U_n , hence also on V_n and T_n . Furthermore, for all $m \neq n$, in case $\sigma_m \notin S$ the tree T_m is empty, and in case $\sigma_m \in S$ the strings σ_m and σ_n are incompatible, hence $\sigma_m \not\preceq f$ and thus, by definition of T_m , neither the string $f(0)f(1)\dots f(|\sigma_m|)$ nor its extension $f(0)f(1)\dots f(|m|)$ is in T_m . \square

Next we characterize BC-learnable classes. The proof of the corresponding Theorem 3.3 uses Podnieks' characterization $BC = NV''$ [4,19], where by definition a class \mathcal{S} is in NV'' if and only if there is a partial recursive function γ such that

$$(\forall f \in \mathcal{S}) (\forall^\infty x) [\gamma(f(0)f(1)\dots f(x-1)) \downarrow = f(x)], \quad (3)$$

i.e., $\mathcal{S} \in NV''$ if and only if some partial recursive function γ predicts for all $f \in \mathcal{S}$ almost all values $f(x)$ from the data $f(0), f(1), \dots, f(x-1)$. Note that γ might fail to predict f either by computing an incorrect value or by being undefined.

Theorem 3.3. *A class of functions \mathcal{S} is in BC if and only if there is a uniformly strongly recursive sequence T_0, T_1, \dots of trees such that for every $f \in \mathcal{S}$ there is some n where f is isolated on T_n and is not on T_m for all $m \neq n$.*

Proof. (\Leftarrow): This part of the proof is pretty similar to the corresponding argument in the proof of Theorem 3.1 for the case of EX. Assume that T_0, T_1, \dots is a uniformly strongly recursive sequence of trees as in Theorem 3.3. Fix a recursive function g such that $g(n)$ is a strong recursive index for T_n and let ib be the recursive function from Lemma 2.2. Recall that if f is isolated above σ on some T_n , then $\text{ib}(g(n), \sigma)$ is an index for f .

Now the following inference algorithm learns \mathcal{S} as required. On input σ , the algorithm outputs the index $\text{ib}(g(n), \sigma)$ for the least $n \leq |\sigma|$ such that σ is on T_n ; in case there is no such n , the algorithm outputs the symbol “?” in order to indicate that the algorithm makes no guess. By assumption on the T_n , for all sufficiently large σ , we indeed select some fixed n such that, first, f is isolated on T_n and, second, f is isolated on the latter tree above σ , i.e., $\text{ib}(g(n), \sigma)$ is an index for f .

(\Rightarrow): According to $BC = NV''$, fix some partial recursive γ that satisfies (3). Let the set F contain the empty string λ and all strings $\sigma = \rho a$ where $\gamma(\rho) \downarrow \neq a$, that is, all strings σ where γ makes an incorrect prediction for the last place of σ . The set F is obviously r.e., so fix some appropriate effective enumeration $\sigma_0, \sigma_1, \dots$ where each string in F appears exactly once. Define the trees U_n by

$$U_n = \{\tau: \tau \preceq \sigma_n\} \cup \{\sigma: \sigma_n \prec \sigma \text{ and } (\forall \tau)[\sigma_n \prec \tau \preceq \sigma \Rightarrow \tau \notin F]\},$$

i.e., U_n contains an extension σ of σ_n if and only if for all strings $\tau = \rho a$ with $a \in \mathbb{N}$ and $\sigma \prec \tau \preceq \sigma_n$, the value $\gamma(\rho)$ is either undefined or agrees with a .

The trees U_n are uniformly strongly co-r.e. First observe that, while using the given enumeration of F , we can uniformly in n compute σ_n and enumerate the complement of U_n . Second, in order to enumerate for given n the complement of $\text{Cl}[U_n]$, we compute σ_n and enumerate all strings of length at most $|\sigma_n|$ that are not prefixes of σ . In addition, we proceed as follows in parallel for all ρ of length strictly larger than $|\sigma_n|$. In case $\gamma(\rho)$ is defined, enumerate ρy for all natural numbers $y > \gamma(\rho)$; in addition enumerate ρy for all $y \leq \gamma(\rho)$ in case ρ is eventually enumerated into the complement of U_n . The correctness of this construction follows because for every n and for every string ρ , the set $\{y \in \mathbb{N} : \rho y \in U_n\}$ either is equal to \mathbb{N} (in case $\rho \in U_n$ and $\gamma(\rho)$ is undefined) or is a singleton containing just $\gamma(\rho)$ (in case $\rho \in U_n$ and $\gamma(\rho)$ is defined), or is empty (in case $\rho \notin U_n$).

By Lemma 2.5, it is sufficient to show that the U_n meet the specification of the T_n . By construction, a function f is on U_n if and only if the length of σ_n and thus n is maximal with $\sigma_n \preceq f$, hence each function f is on at most one tree U_n and each $f \in \mathcal{S}$ is on some tree U_n . It remains to show that in the latter case f is isolated on U_n . If we choose $\rho_0 \prec f$ such that for all τ and a where $\rho_0 \preceq \tau a \prec f$, the value $\gamma(\tau)$ is defined and agrees with a , then in fact f is fully isolated on U_n above ρ_0 . If not, there is a least $\sigma \in U_n$ where $\rho_0 \preceq \sigma \not\prec f$. But then this σ is in F and is a strict extension of ρ_0 and thus of σ_n , hence $\sigma \notin U_n$ follows by definition of U_n . \square

Remark 3.4. Podnieks' result $\text{BC} = \text{NV}''$ [4,19] can be adjusted in order to obtain a characterization of EX. A class \mathcal{S} is in EX if and only if there is a function γ that satisfies (3) and whose "last error" on each $f \in \mathcal{S}$ is always due to predicting an incorrect value, i.e., is never due to being undefined. Using this characterization of EX, we obtain an alternate proof of one direction of Theorem 3.1 that is similar to the proof of the corresponding direction of Theorem 3.3.

In Corollary 3.5 we state characterizations of the concepts of EX, FIN, and BC-learning of classes of $\{0,1\}$ -valued functions. These characterizations are special cases of Theorems 3.1, 3.2, and 3.3, and are obtained by observing that, first, a binary tree is strongly recursive if and only if it is recursive and, second, an infinite branch on a binary tree is isolated above λ if and only if it is the only infinite branch on this tree. The second statement follows by the remark made after Definition 2.1.

Corollary 3.5. *Let \mathcal{S} be a class of $\{0,1\}$ -valued functions.*

- (a) \mathcal{S} is in FIN if and only if there is a uniformly recursive sequence of binary trees T_0, T_1, \dots such that for all $f \in \mathcal{S}$ there is some n where f is the only infinite branch on T_n and $f(0)f(1)\dots f(m) \notin T_m$ for all $m \neq n$.
- (b) \mathcal{S} is in EX if and only if there is a uniformly recursive sequence of binary trees T_0, T_1, \dots such that for all $f \in \mathcal{S}$ there is some n where f is the only infinite branch on T_n and f is not on T_m for all $m \neq n$.
- (c) \mathcal{S} is in BC if and only if there is a uniformly recursive sequence of binary trees T_0, T_1, \dots such that for all $f \in \mathcal{S}$ there is some n where f is isolated on T_n and f is not on T_m for all $m \neq n$.

The next theorem states that there is a finitely branching recursive tree T where all functions on T are recursive and isolated on T , but the class of all functions on T cannot be BC-learned. This

implies, first, that in the characterization of BC according to Theorem 3.3 the condition on strong recursiveness is indeed necessary. Second, the counterexample T shows that it is reasonable to include the condition on finite branching in the definition of the concept isolated. The tree T can be transformed into the strongly recursive tree $U = \{\sigma a : \sigma \in T \wedge a \in \mathbb{N}\}$, where each function f on U is recursive and is isolated in the weaker sense that there is some prefix of f that is not extended by any other function on T . However, each function is on T if and only if it is on U . Consequently, by assumption on T , the class of branches on U that are isolated in the mentioned weaker sense is not in BC.

Theorem 3.6. *There is a finitely branching recursive tree T such that each function on T is recursive and isolated on T , but the class of all functions on T is not in BC.*

Proof. We define a uniformly recursive sequence of trees T_0, T_1, \dots where T_n consists of all prefixes and some extensions of the string $1^n 0$. Then we let T be equal to the union of the T_n , hence in particular T is recursive and 1^∞ is on T . During the construction we ensure for all n that T_n is finitely branching and has exactly one infinite branch f_n , where the branch f_n is recursive and is not almost always predicted correctly by the n th Turing machine M_n , i.e., there are infinitely many m where $M_n(f_n(0) \dots f_n(m))$ either is undefined or differs from $f(m+1)$. By the latter, Podnieks' result $\text{BC} = \text{NV}''$ —which we have already used in the proof of Theorem 3.3—implies that the class \mathcal{S} of all infinite branches on T is not in BC.

The construction of T_n is done in stages. During stage $m > 0$, we receive a string σ_m as input, we add strings to T_n , and in case the stage eventually completes, we define the string σ_{m+1} . For a start, we let $\sigma_0 = 1^n 0$ and we let T_n contain all prefixes of σ_0 . During stage $m+1$, we proceed as follows.

In case M_n does not predict the number 0 for any input $\sigma_m 0^t$, let σ_{m+1} be undefined and put $\sigma_m 0^k$ into T_n for all $k > 0$.

In case there is a least pair (s, t) such that M_n on input $\sigma_m 0^t$ predicts 0 within $s > 0$ steps, let σ_{m+1} be equal to $\sigma_m 0^s$ and put $\sigma_m 0^k$ into T_n for $k = 0, 1, \dots, t + s$.

If all σ_m are defined then let $f_n = \lim_m \sigma_m$ and, otherwise, let $f_n = \sigma_m 0^\infty$ for the maximal m such that σ_m is defined. We leave to the reader the routine tasks to show that each T_n is finitely branching and uniformly recursive in n and that f_n is the only infinite branch on T_n . By the latter all infinite branches of T , i.e., 1^∞ and all the f_n , are recursive. Furthermore, with the T_n also T is finitely branching.

So it remains to show $\mathcal{S} \notin \text{NV}''$. Here it suffices to observe that for every n , the machine M_n fails to predict f_n because either $f_n = \sigma_m 0^\infty$ and M_n predicts none of the zeroes beyond σ_m or $f_n = \lim_m \sigma_m$ and M_n fails at least m times on any prefix σ_m of f . \square

4. Characterizations of refutable and reliable learning

In this section we give characterizations of various types of learning of functions with refutation, i.e., learning models where the learner is not only required to learn all concepts in a

given class but also has to explicitly refute concepts outside the class by outputting a distinguished refutation symbol. Learning with refutation is usually only considered in connection with learning under the criterion EX; for simplicity, in the remainder of this section learning always means EX-learning, i.e., a learner **LEARNS** a function if the learner converges on this function to some fixed index for the function.

The most restrictive concept of learning with refutation is refuting learning, where the learner has to refute after a finite number of steps by outputting the refutation symbol. In the more liberal model of limit-refuting learning, the learner is only required to “refute in the limit” by converging to the refutation symbol. While a refuting learner continues forever to output refutation symbols after having output a refutation symbol once, a limit-refuting learner might alternate between indices and refutation symbols in an arbitrary way before converging. Another even more liberal model where refuting means to output a refutation symbol infinitely often can be shown [7,8,11] to be equivalent to Minicozzi’s [12] concept of reliable learning.

Definition 4.1. A learner **REFUTES** a function if on this function the learner first outputs at most finitely many numbers (without outputting any refutation symbol) and then outputs nothing but refutation symbols. A learner is **REFUTING** if for any function, either the learner refutes the function or the learner converges on this function to an index for the function without ever outputting a refutation symbol. A class is **REFUTING LEARNABLE** if and only if the class is learned by a refuting learner.

A learner **LIMIT-REFUTES** a function if the learner converges on the function to the refutation symbol. A learner is **LIMIT-REFUTING** if the learner either learns or limit-refutes any function. A class is **LIMIT-REFUTING LEARNABLE** if and only if the class is learned by a limit-refuting learner.

A learner is **RELIABLE** if and only if for any function, the learner either learns the functions or has infinitely many mind changes on this function (i.e., if the function is not learned, then the learner does not converge to a fixed index). A class is **RELIABLY LEARNABLE** if the class is learned by a reliable learner.

Refuting learning has been introduced by Mukouchi and Arikawa [13,14]. Limit-refuting learning is considered by Jain et al. [7,8] in the setting of function learning from informant and by Merkle and Stephan [11] in the setting of set learning from text. Note that Jain et al. focus on a variant of limit-refuting learning where the learner only has to refute all *recursive* functions that are not learned.

Theorem 4.2. *Let \mathcal{S} be a class of functions.*

- (a) *The class \mathcal{S} is reliably learnable if and only if there is a uniformly strongly recursive sequence T_0, T_1, \dots of trees such that*
 - (i) *every $f \in \mathcal{S}$ is on some T_n ,*
 - (ii) *every function on every T_n is isolated above λ .*
- (b) *The class \mathcal{S} is limit-refuting learnable if and only if there is a uniformly strongly recursive sequence T_0, T_1, \dots of trees and a uniformly recursive sequence U_0, U_1, \dots of trees such that (i) and (ii) above hold and*
 - (iii) *any function is either on some T_n or on some U_m but not both.*

- (c) The class \mathcal{S} is refuting learnable if and only if there is a uniformly strongly recursive sequence T_0, T_1, \dots of trees and a recursive tree U such that (i) and (ii) above hold and (iii') for any function, the function is on U if and only if it is on some T_n .

Proof. We first show the backwards directions of the three equivalences (a), (b), and (c). In all three cases, we are given a uniformly strongly recursive sequence of trees T_0, T_1, \dots that satisfies (i) and (ii). In this situation, fix a recursive function g such that $g(n)$ is a strong recursive index for T_n . If there is any function on T_n at all, then by (ii) an index for this function is given by $\text{ib}(g(n), \lambda)$, where ib is the recursive function from Lemma 2.2. For any σ let

$$n(\sigma) = \min\{j \leq |\sigma| : \sigma \in T_j\} \quad \text{and} \quad e(\sigma) = \text{ib}(g(n(\sigma)), \lambda),$$

where both values are undefined in case the minimization is over the empty set. Given a function f , consider the values of $n(\sigma)$ for strings $\sigma \preceq f$ of increasing length. If f is on some of the trees T_0, T_1, \dots , then the $n(\sigma)$ will converge to the least index n such that n is on T_n , hence the $e(\sigma)$ converge to an index for f . On the other hand, if f is on none of the trees T_0, T_1, \dots and it is not the case that almost all $n(\sigma)$ are undefined, then the defined values of $n(\sigma)$ go nonascendingly to infinity.

Next we describe learners as required. In all three cases, the learner learns exactly the functions that are on some tree T_n and hence by (i) learns all functions in \mathcal{S} .

In case (a), we obtain a reliable learner for \mathcal{S} as follows. On the empty string, the learner outputs an arbitrary index and on any input $\sigma = \eta y$, $y \in \mathbb{N}$, the learner outputs $e(\sigma)$ in case the values $n(\eta)$ and $n(\sigma)$ are both defined and are the same and, otherwise, in order to achieve divergence, the learner outputs $|\sigma|$.

In case (b), we are given in addition a uniformly strongly recursive sequence of trees U_0, U_1, \dots such that (iii) is satisfied. Similarly to $n(\sigma)$, let $m(\sigma)$ be the minimum index $j \leq |\sigma|$ such that σ is on U_m . A limit-refuting learner for \mathcal{S} works as follows. The learner outputs $e(\sigma)$ in case $n(\sigma)$ is defined while $m(\sigma)$ is either undefined or is larger than $n(\sigma)$. Otherwise, i.e., in case $n(\sigma)$ is undefined or is defined and is strictly smaller than $m(\sigma)$, the learner outputs a refutation symbol.

In case (c), we are given in addition a strongly recursive tree U such that (iii') is satisfied. A refuting learner as required simply outputs $e(\sigma)$ in case $\sigma \in U$ and, otherwise, outputs a refutation symbol.

Next we show the forwards directions of the three equivalences (a), (b), and (c). In all three cases we are given a learner M that converges on any function in \mathcal{S} to an index for that function, i.e., in particular this learner learns \mathcal{S} under the criterion EX. By applying the construction in the proof of Theorem 3.1 to M , we obtain a strongly recursive sequence T_0, T_1, \dots that satisfies (i) and (ii). Assertion (i) is immediate because the theorem states that for every $f \in \mathcal{S}$, there is some n where f is isolated on T_n above λ . Assertion (ii) follows by the following observations on the construction of the T_n . Any function f might be on T_n only if M converges on f to the index n . In case of a reliable, limit-refuting, or refuting learner the latter implies that M learns f , hence n is an index for f and in particular φ_n is total. But if φ_n is total, then T_n contains just the prefixes of this function.

The discussion in the preceding paragraph finishes the proof of (a). In order to obtain sets U_0, U_1, \dots and U as required by (b) and (c), we let

$$U_m = \{\sigma: \text{there are at most } m \text{ prefixes of } \sigma \text{ on which } M \text{ does not output the refutation symbol}\},$$

$$U = \{\sigma: \text{there is no prefix of } \sigma \text{ on which } M \text{ outputs the refutation symbol}\};$$

the standard verification of the fact that the latter sets have the desired properties is omitted. \square

5. Application to team learning

We review the concept of team learning; for details and references see Smith [20].

Definition 5.1. Let m and n be two natural numbers with $m \leq n$ and let \mathcal{S} be a class of functions.

The class \mathcal{S} is in $[m, n]$ EX if there is a team (i.e., a finite sequence) of n learners such that every $f \in \mathcal{S}$ is EX-learned by at least m learners in the team. The class \mathcal{S} is in $[m, n]$ BC if there is a team of n learners such that every $f \in \mathcal{S}$ is BC-learned by at least m learners in the team.

Remark 5.2. In the proof of Theorem 3.1, we have constructed from a class \mathcal{S} in EX a uniformly recursive sequence T_0, T_1, \dots of trees. While this would not have been strictly necessary for proving the theorem, in the verification of the construction we have actually shown that each function f —even $f \notin \mathcal{S}$ or nonrecursive f —is on at most one tree in this sequence. The latter remark extends to the sequence of trees that has been constructed from a given class in BC in order to prove Theorem 3.3.

Theorem 5.3. Let m and n be two natural numbers with $m \leq n$ and let \mathcal{S} be a class of functions.

- (a) The class \mathcal{S} is in $[m, n]$ EX if and only if there is a uniformly strongly recursive sequence of trees such that each $f \in \mathcal{S}$ is on at most n of these trees and is isolated above λ on at least m of these trees.
- (b) The class \mathcal{S} is in $[m, n]$ BC if and only if there is a uniformly strongly recursive sequence of trees such that each $f \in \mathcal{S}$ is on at most n of these trees and is isolated on at least m of these trees.

Proof. We show the first assertion concerning EX-learning and leave the almost identical considerations for BC-learning to the interested reader. First assume that we are given a sequence T_0, T_1, \dots as in the first assertion of the theorem. In order to construct a team of learners M_1, \dots, M_n as required, pick a recursive function g such that $g(n)$ is a strongly recursive index for T_n and recall the properties of the function ib from Lemma 2.2. Moreover, for all strings σ , let $I_\sigma = \{n \leq |\sigma|: \sigma \in T_n\}$. Then we let $M_k(\sigma) = ?$ in case I_σ contains less than k indices and, otherwise, we let $M_k(\sigma) = \text{ib}(g(i_k), \lambda)$ where i_k is the k th element in I_σ . Now fix $f \in \mathcal{S}$. Then there is some l where $m \leq l \leq n$ such that f is exactly on the trees T_{i_1}, \dots, T_{i_l} . Thus for almost all prefixes σ of f , the

least l elements in I_σ are just i_1, \dots, i_l , hence for $k \leq l$, the learner M_k converges to $\text{ib}(g(i_k), \lambda)$, which is indeed an index for f for the at least m indices i_k such that f is isolated above λ on T_{i_k} .

Now, conversely, assume that we are given a class \mathcal{S} in $[m, n]\text{EX}$. Fix a corresponding team of n learners and for $k = 1, \dots, n$, let \mathcal{S}_k be the class of all functions that are EX-learned by the k th learner in this team. Moreover, apply Theorem 3.1 to \mathcal{S}_k in order to obtain a strongly uniformly recursive sequence $T_0^k, T_1^k, T_2^k, \dots$ of trees. Thus every function in \mathcal{S}_k is isolated above λ on some tree in the latter sequence and—according to Remark 5.2—we can assume that every function is on at most one tree in this sequence. Using an appropriate bijection $\langle \cdot, \cdot \rangle$ from $\mathbb{N} \times \{1, \dots, n\}$ to \mathbb{N} , we let $T_{\langle i, k \rangle} = T_i^k$. Then the sequence T_0, T_1, \dots is uniformly strongly recursive. Moreover, by construction each function is on at most n trees in the latter sequence and each function $f \in \mathcal{S}$, i.e., each function f that is in at least m of the classes \mathcal{S}_k , is isolated above λ on at least m trees in this sequence. \square

Another application of the characterization of EX-learning in terms of sequences of trees is an alternative proof for the result of Pitt and Smith [18] that in the case of EX-learning, single learners are as powerful as teams in which more than half of the learners have to be successful.

Theorem 5.4 (Pitt and Smith [18, Corollary 13]). *For all m, n with $\frac{m}{n} > \frac{1}{2}$, we have $[m, n]\text{EX} = \text{EX}$.*

Proof. The inclusion of EX in $[m, n]\text{EX}$ is trivially true. In order to show the reverse inclusion, fix m and n as in the theorem. Furthermore, fix an arbitrary team of n EX-learners and let \mathcal{S} be the class of all functions that are EX-learned by at least m learners in this team. We have to show that \mathcal{S} can be EX-learned. Here we use that the proof of Theorem 3.1 as given above in fact yields the following slightly modified version of the theorem.

$\mathcal{S} \in \text{EX}$ if and only if there is a uniformly recursive sequence T_0, T_1, \dots of trees such that every $f \in \mathcal{S}$ is on some tree T_n and in case an $f \in \mathcal{S}$ is on a tree T_m then f is isolated above λ on T_m .

For $k = 1, \dots, n$, let \mathcal{S}_k be the class of all functions that are EX-learned by the k th learner in the given team. Moreover, apply Theorem 3.1 to \mathcal{S}_k in order to obtain a strongly uniformly recursive sequence $T_0^k, T_1^k, T_2^k, \dots$ of trees. Let T_0, T_1, \dots be a uniformly strongly recursive sequence of trees that contains exactly the trees that can be written as intersection of m trees that are each taken from a different sequences as just defined, i.e., the T_i comprise exactly the trees in the class

$$\mathcal{T} = \{T_{i_1}^{k_1} \cap \dots \cap T_{i_m}^{k_m} : i_1, k_1, \dots, i_m, k_m \in \mathbb{N}, 1 \leq k_1 < \dots < k_m \leq n\}.$$

Now fix $f \in \mathcal{S}$. Then f is learned by at least m learners in the given team, i.e., there are pairwise different indices k_1, \dots, k_m such that f is in the classes $\mathcal{S}_{k_1}, \dots, \mathcal{S}_{k_m}$. As a consequence, first, f is on trees of the form $T_{i_1}^{k_1}, \dots, T_{i_m}^{k_m}$, hence f is on some tree in \mathcal{T} . Second, assuming that f is on a tree $T \in \mathcal{T}$, then T is the intersection of trees $T_{i_1}^{l_1}, \dots, T_{i_m}^{l_m}$ where the l_j are pairwise different. But then by $m + m > n$, the intersection of $\{k_1, \dots, k_m\}$ and $\{l_1, \dots, l_m\}$ is nonempty, thus for some index l_j , the function f is in \mathcal{S}_{l_j} and is on $T_{i_j}^{l_j}$. But then f is isolated above λ on the latter tree by choice of the trees $T_0^{l_j}, T_1^{l_j}, \dots$, hence f is isolated above λ on T . \square

6. Relations to characterizations by effective enumerations

Wiehagen and Jung [23] give equivalent characterization of FIN, EX, and BC in terms of effective enumerations of partial recursive functions that satisfy additional requirements. They use algorithms of the type “learning by enumeration” in order to show that the classes described by such enumerations are in FIN, in EX, or in BC, respectively. Concerning the more involved reverse direction of their equivalence results, in the case of learning of sets we obtain proofs by passing from our representations in terms of sequences of trees to enumerations with the required properties. In Theorem 6.1 and its proof we give a full account of this transition for the case of EX and afterwards we sketch how to proceed in the cases of FIN and BC.

Theorem 6.1. Let \mathcal{S} be a class of $\{0, 1\}$ -valued functions. (Wiehagen and Jung [23, Satz 1 (Theorem 1)]). If $\mathcal{S} \in \text{EX}$ then there is an effective enumeration ψ of partial recursive functions and a recursive function h such that all $f \in \mathcal{S}$ satisfy

- (a) $f = \psi_n$ for at least one and at most finitely many n ,
- (b) for almost all n , if $(\forall x \leq h(n)) [\psi_n(x) \downarrow = f(x)]$ then $f = \psi_n$.

Proof. Fix any \mathcal{S} in EX. According to Theorem 3.1, we can pick a uniformly strongly recursive sequence T_0, T_1, \dots of trees such that every $f \in \mathcal{S}$ is isolated above λ on some T_n and is not on T_m for all $m \neq n$. Let g be a recursive function such that $g(n)$ is a strong recursive index for T_n and recall the definition of the function ib from Lemma 2.2. Let $\psi_n = \varphi_{\text{ib}(g(n), \lambda)}$. By construction, for any function f we have $f = \psi_n$ if and only if f is the only infinite branch of T_n , hence (a) follows by choice of the T_n .

By Remark 5.2 we can assume that for $n \neq m$, the trees T_m and T_n do not have a common infinite branch. Therefore, by König’s Lemma [15, Theorem V.5.23], the intersection $T_m \cap T_n$ is finite and the function

$$h(n) = \max\{|\sigma| : (\exists m < n)[\sigma \in T_n \cap T_m]\} \quad (4)$$

is recursive. By assumption, every $f \in \mathcal{S}$ is on some tree T_m . In case there was some $n > m$ with $(\forall x \leq h(n))[\psi_n(x) \downarrow = f(x)]$ then the string $f(0)f(1)\dots f(h(n))$ of length $h(n) + 1$ would be in $T_m \cap T_n$, thus contradicting the choice of h . Consequently, (b) holds for almost all n . \square

The characterizations of FIN and BC in terms of enumerations given by Wiehagen and Jung are similar to the one for EX stated in Theorem 6.1. In both cases, condition (a) is relaxed by just requiring that every f in \mathcal{S} is equal to one of the ψ_n , and in the case of FIN, in addition condition (b) is strengthened to hold for all instead of almost all n . Such enumerations can be obtained easily from the characterizations of FIN and EX in terms of sequences of trees according to Theorems 3.2 and 3.3. In the case of FIN, in order to pass from a representation of a class in FIN as sequence of trees to the desired enumeration, we define the functions ψ_n as in the proof of Theorem 6.1 and let $h(n) = n$. In the case of BC, we let $\psi_{\langle n, j \rangle} = \varphi_{\text{ib}(g(n), \sigma(n, j))}$ where $\sigma(n, j)$ denotes string number j in T_n with respect to an appropriate ordering, ib and g are defined as in the proof of Theorem 6.1, and $\langle \cdot, \cdot \rangle$ is the usual pairing function [21]. The function h is defined similar to

(4). More precisely, $h(\langle n, i \rangle) = |\sigma|$ for the longest string σ that T_n shares with some tree T_m such that $m \neq n$ and there is some j where $\langle m, j \rangle < \langle n, i \rangle$. \square

7. Finite learning and single trees

While the characterizations of learning models in Section 3 are based on sequences of trees, we will now consider characterizations in terms of single trees. We show that characterizations of the latter type exist for restricted models of learning such as finite or strong monotonic learning, whereas Remark 7.1 indicates that such characterizations are unlikely to exist for BC- and EX-learning.

Remark 7.1. Suppose there were a characterization of EX (or alternatively of BC) in terms of single trees as follows. There is a certain property such that a class \mathcal{S} of functions is in EX if and only if there is some tree T where for all f in \mathcal{S} , first, f is on T and, second, the pair (f, T) satisfies the property under consideration. Now assume for a contradiction that there were such a property. For any function g , define the class

$$\mathcal{S}_g = \{f: f(x) = g(x) \text{ for almost all } x\}$$

and observe that for recursive g , the class \mathcal{S}_g is in EX. If we let F be the tree that contains all strings, then obviously for every single class \mathcal{S}_g the tree F is the unique tree where all f in \mathcal{S}_g are on F . As a consequence, for each recursive g , F is the unique tree that corresponds to \mathcal{S}_g with respect to our assumed equivalence, hence in particular the element g of \mathcal{S}_g is on F and the pair (g, F) satisfies the property under consideration. But then, using the reverse direction of our assumed equivalence, we obtain as a contradiction that the class of all recursive functions is in EX.

We first give a characterization of $\text{FIN}[K]$, i.e., of the classes that can be finitely learned by oracle Turing machines that receive the halting problem K as an oracle.

Theorem 7.2. *A class \mathcal{S} is in $\text{FIN}[K]$ if and only if there is an r.e. tree T such that every $f \in \mathcal{S}$ is fully isolated on T .*

Proof. (\Leftarrow): Let T be an r.e. tree as in the theorem. Fix a recursive function g such that $g(\tau)$ is an index for the function f whenever f is fully isolated above τ on T . Consider the following K -recursive algorithm.

On input σ , output $g(\tau)$ for the least $\tau \preceq \sigma$ such that the K -recursive query

$$\exists \rho_0, \rho_1 \succ \tau [\rho_0, \rho_1 \in T \wedge \rho_0 \not\leq \rho_1 \wedge \rho_1 \not\leq \rho_0]$$

is answered negatively, i.e., τ is the least prefix of σ such that there are no two incompatible nodes above τ in T . If there is no such τ , abstain from guessing by outputting “?”.

This algorithm infers finitely every function that is fully isolated on T and thus in particular every function that is in \mathcal{S} .

(\Rightarrow): Assume that the class \mathcal{S} is finitely learned with oracle K by some oracle Turing machine M . We define a recursive function r that maps each string σ to a natural number.

If there is some $\tau \preceq \sigma$ such that $M^{K_{|\sigma|}}(\tau) \neq ?$ can be computed in less than $|\sigma|$ steps, then let $r(\sigma) = M^{K_{|\sigma|}}(\tau)$ for the least such τ and, otherwise, let $r(\sigma) = 0$.

Here K_s is the set of all strings that have been enumerated into K after s steps of some fixed Turing machine that enumerates K . Let $V = \{\sigma : \sigma \preceq \varphi_{r(\sigma)}\}$, i.e., σ is in V if and only if $\varphi_{r(\sigma)}$ is defined at all places in the domain of σ and agrees there with σ . Furthermore, let the tree T be the closure of V under taking prefixes. Then T is r.e. since r is recursive. Now fix $f \in \mathcal{S}$ and let the string τ be the shortest prefix of f such that $M^K(\tau)$ converges to an index e of f . Pick m such that for all $\rho \preceq \tau$, the computation of $M^{K_m}(\rho)$ converges in less than m steps and queries its oracle only at places where K_m agrees with K . Then by definition of r , for every extension σ of τ of length at least m , $r(\sigma) = M^K(\tau)$ is an index for f , hence by definition of T , $\sigma \in T$ if and only if $\sigma \preceq f$. So f is fully isolated on T and \mathcal{S} is contained in the class of fully isolated branches of T . \square

The following characterization of FIN is less appealing than the one obtained for $\text{FIN}[K]$ because it requires another r.e. set F in order to signal when the learner has to make its guess. In contrast to $\text{FIN}[K]$, however, we are able to characterize FIN in terms of single recursive trees.

Theorem 7.3. *For every class \mathcal{S} of functions, the following conditions are equivalent.*

- (a) $\mathcal{S} \in \text{FIN}$.
- (b) *There is an r.e. tree T and an r.e. set F such that for all $f \in \mathcal{S}$,*
 - f is fully isolated on T and
 - for every $\sigma \preceq f$, σ is in F if and only if f is fully isolated on T above σ .
- (c) *There is a strongly recursive tree T and an r.e. set F such that for all $f \in \mathcal{S}$,*
 - f is isolated on T and
 - for every $\sigma \preceq f$, σ is in F if and only if f is isolated on T above σ .

Proof. (a) \Rightarrow (b): Pick a recursive learner M for \mathcal{S} that always first outputs the symbol “?” and whenever it switches to a guess e , then continues to guess e on any further input. Moreover, let for all strings σ ,

$$\sigma \in F \Leftrightarrow M(\sigma) \neq ?$$

$$\sigma \in T \Leftrightarrow (\forall \tau \prec \sigma M(\tau) = ?) \vee (M(\sigma) = e \wedge \sigma \preceq \varphi_e).$$

Consider an arbitrary function f that is inferred by M . Then f is on T because for any $\sigma \preceq f$ either we have $M(\tau) = ?$ for all $\tau \preceq \sigma$ or we have $M(\sigma) = e$ where $\sigma \preceq f = \varphi_e$. Next fix any $\sigma \preceq f$. In case $\sigma \notin F$, we have $M(\sigma) = ?$, hence $\sigma b \in T$ for all b and thus f is not fully isolated on T above σ . Conversely, in case $\sigma \in F$, we have $M(\sigma) = e$ for some index e , hence $f = \varphi_e$ and by construction the tree T contains just the extensions η of σ where $\eta \preceq f$, thus f is fully isolated on T above σ . In summary, we have $\sigma \in F$ if and only if f is fully isolated on T above σ . The latter equivalence then implies that f is indeed fully isolated on T because almost all $\sigma \preceq f$ are in F .

(b) \Rightarrow (c): Let (b) hold via an r.e. set F and an r.e. tree T' . We construct a strongly recursive tree T such that F and T satisfy (c). We say that two strings ρ_0 and ρ_1 witness that a string σ is irrelevant if and only if both strings are in $F \cap T'$, $\rho_0 \prec \rho_1$ and $\rho_0 \prec \sigma$, while ρ_1 and σ are incompatible. This notation is justified because in the latter situation σ is not a prefix of any $f \in \mathcal{S}$. For a proof by contradiction, assume $\sigma \preceq f$ for some $f \in \mathcal{S}$. Then $\rho_0 \preceq f$ and f is fully isolated on T' above ρ_0 , hence only one of the two incompatible extensions ρ_1 and σ of ρ_0 can be in T' , i.e., σ is not in T' . But this contradicts our assumption that f is on T' . Now the tree T is defined by

$$\sigma \in T \text{ if and only if } \sigma \text{ being irrelevant is not witnessed by strings } \rho_0 \text{ and } \rho_1 \text{ in } F_{\|\sigma\|} \cap T'_{\|\sigma\|}.$$

Here F_s and T'_s are the sets of all strings that have been enumerated within at most s steps by appropriate enumerations of F and T' , respectively. The tree T is strongly recursive. First, T is recursive by definition. Second, the predicate “ $(\exists y \geq x) [\sigma y \in T]?$ ” can be answered effectively as follows. In case $\sigma x \in T$, the answer to the query is “Yes”. Otherwise, we find strings ρ_0 and ρ_1 in $F_{\|\sigma x\|} \cap T'_{\|\sigma x\|}$ that witness that σx is irrelevant. But then for every $y \geq x$, the two latter strings are in $F_{\|\sigma y\|} \cap T'_{\|\sigma y\|}$, hence σy is in T if and only if $\sigma y \preceq \rho_1$. As a consequence, the answer is “Yes” if and only if there is a $y \geq x$ where $\sigma y \preceq \rho_1$.

So it remains to show that for every $f \in \mathcal{S}$ and every $\sigma \preceq f$, the string σ is in F if and only if f is isolated on T above σ . (That f is isolated on T then follows because F contains some prefix of f .) First assume $\sigma \notin F$. Then for all $a \in \mathbb{N}$, no proper prefix of σa is in F and so there are no witnesses for σa being irrelevant, hence T contains all such strings σa and f is not isolated on T above σ . Next assume that σ is in F . Fix a string $\tau \succ \sigma$ and let

$$E_\tau = \{\eta \in T: \tau y \preceq \eta \text{ for some } y \neq f(|\tau|)\}.$$

Then τ and $\tau f(|\tau|)$ witness irrelevance of every string in E_τ . The two former strings are in $F \cap T'$ and thus are in $F_t \cap T'_t$ for some t , hence the set E_τ contains only strings of norm at most t and thus is finite. As a consequence, the subtree of T above σ is finitely branching and contains no infinite branch besides f , i.e., f is isolated on T above σ .

(c) \Rightarrow (a): Let e be a strongly recursive index for T and let ib be the recursive function from Lemma 2.2, i.e., in particular $\text{ib}(e, \sigma)$ is an index for f whenever f is isolated above σ on T . On input σ , the learning algorithm searches for a $\tau \preceq \sigma$ of minimal length such that

$$\text{some prefix of } \tau \text{ is enumerated into } F \text{ within } |\tau| \text{ steps.} \quad (5)$$

If there is no such τ then the algorithm abstains from guessing and, otherwise, it guesses $\text{ib}(e, \tau)$. For every $f \in \mathcal{S}$, there is some prefix of f in F , hence there is a prefix τ of f that satisfies (5). Moreover, on the least such τ the learning algorithm outputs $\text{ib}(e, \tau)$, which is indeed an index for f because every τ that satisfies (5) has a prefix in F and hence f is isolated on T above τ . \square

Wiehagen [22] introduced the concept of strong-monotonic learning of functions. A machine M infers a function f STRONG-MONOTONICALLY if and only if for all σ and τ with $\sigma \preceq \tau \preceq f$ we have

$$\varphi_{M(\sigma)}(x) \downarrow = y \Rightarrow \varphi_{M(\tau)}(x) \downarrow = y$$

(hence in particular $\varphi_{M(\sigma)}(x) = f(x)$ whenever $\varphi_{M(\sigma)}(x)$ is defined for some $\sigma \preceq f$).

The concept of strong-monotonic learning can be combined with the various types of convergence for inductive inference machines in the obvious way. In particular the concept of

strong monotonic BC learning is obtained by combining strong-monotonic learning and behaviorally correct convergence. Note that a class \mathcal{S} is in strong-monotonic BC if and only if \mathcal{S} is in NV'' via a partial recursive function γ that never incorrectly predicts any $f \in \mathcal{S}$ at any x , that is, either γ is undefined or γ yields already the correct value $f(x)$.

The following theorem shows that strong-monotonic BC can be characterized in terms of the isolated branches on a single strongly recursive tree.

Theorem 7.4. *\mathcal{S} is in strong-monotonic BC if and only if there is a strongly recursive tree T such that every $f \in \mathcal{S}$ is isolated on T .*

Proof. (\Rightarrow): If \mathcal{S} is in strong-monotonic BC then there is a partial recursive predictor γ such that for all $f \in \mathcal{S}$ and for almost all strings $\tau \preceq f$, the value $\gamma(\tau)$ is defined and is equal to $f(|\tau|)$, while $\gamma(\tau)$ is undefined on the remaining $\tau \preceq f$. Let T contain all strings σ such that for all $\tau < \sigma$, either the computation of $\gamma(\tau)$ does not terminate within $\|\sigma\|$ steps or $\gamma(\tau)$ is equal to $\sigma(|\tau|)$. Since T is closed under prefixes, T is a tree.

T is strongly recursive: The predicate “ $\sigma \in T$?” is decidable by the definition of T while the predicate “ $(\exists y \geq x) [\sigma y \in T]$?” can be answered by first checking whether $\sigma x \in T$. There are three cases:

- (a) $\sigma x \in T$. The answer to the query is “Yes”.
- (b) $\sigma x \notin T$ and $\gamma(\sigma)$ does not converge within $\|\sigma x\|$ steps. Then the error occurs at some proper prefix of σ and no string of the form σy with $y \geq x$ is in T . The answer is “No”.
- (c) $\sigma x \notin T$ and $\gamma(\sigma) = y$ within $\|\sigma x\|$ steps. Then the answer is “Yes”, if $y \geq x$ and $\sigma y \in T$, and the answer is “No”, otherwise.

Now let $f \in \mathcal{S}$. Since γ on input $\sigma \preceq f$ either outputs $f(|\sigma|)$ or diverges, any $\sigma \preceq f$ is in T and f is an infinite branch on T . In order to show that f is isolated on T , let $\tau_0 \preceq f$ be minimal such that $\gamma(\tau) = f(|\tau|)$ for all τ with $\tau_0 \preceq \tau \preceq f$. Fix a string τ of the latter type and consider the set

$$E_\tau = \{\eta \in T: \tau y \preceq \eta \text{ for some } y \neq f(|\tau|)\}.$$

By definition of T , the set E_τ contains only strings of norm less than the number of steps needed to compute $\gamma(\tau)$, i.e., E_τ is finite. As a consequence, the subtree of T above τ_0 is finitely branching and contains no infinite branch besides f , hence f is isolated on T .

(\Leftarrow): On any input σ the learner just outputs $\text{ib}(e, \sigma)$ where e is a strong recursive index for the tree T and ib is the function from Lemma 2.2. Since f is isolated on T above almost all $\sigma \preceq f$, the index $\text{ib}(e, \sigma)$ computes for almost all $\sigma \preceq f$ the function f . From Remark 2.3 it follows that whenever $\sigma \preceq \tau \preceq f$ for some infinite branch T of f then $\varphi_{\text{ib}(e, \tau)}$ extends $\varphi_{\text{ib}(e, \sigma)}$. Therefore the learning algorithm is strong-monotonic. \square

By an argument similar to the one given in Remark 7.1 we obtain as a corollary to Theorem 7.4 that strong-monotonic BC is strictly contained in BC. For example, this separation is witnessed by the class of all functions that are almost everywhere 0.

Corollary 7.5. *Strong-monotonic BC is strictly contained in BC.*

Proof. Let \mathcal{S} be the class of all functions f where $f(x) = 0$ for almost all $x \in \mathbb{N}$. It is easy to see that \mathcal{S} is in BC. Now assume for a contradiction that \mathcal{S} were in strong-monotonic BC. Then by Theorem 7.4 there is a tree on which every function in \mathcal{S} is isolated. But such a tree must contain all prefixes of all functions in \mathcal{S} , i.e., the tree contains all strings and hence has no isolated paths at all. \square

8. Learning with additional information

In the setting of learning with additional information, an inductive inference machine receives as input a sequence $e, f(0), f(1), \dots$ and is required to learn f in the limit under the assumption that e relates to f in some specific way. For example, Freivalds and Wiehagen [5] considered the case where e is assumed to be an upper bound on the size of some program for f . They showed that in this setting every recursive function can be learned in EX-style, that is, under the convergence criterion that corresponds to EX.

Case et al. [3] showed that if e is assumed to be an index of a strongly recursive tree such that f is an isolated infinite branch of this tree, then the class of all recursive functions can be learned in BC-style. The latter can in fact be strengthened to strong-monotonic BC-style, because in Theorem 7.4 the transition from a strongly recursive tree T to a machine that witnesses that the class of isolated branches of T is in strong-monotonic BC can be chosen to be effective in an index for T . It follows from the results of Case et al. [3] that in the case of EX-learning, even with the additional information of an index for some tree as describe above, not all recursive functions can be learned. In view of this result, one might ask whether additional information of this type helps an EX-learner at all. The next theorem answers this question in the affirmative.

Theorem 8.1. *There is a class \mathcal{S} of recursive functions and an inference machine M such that $\mathcal{S} \notin \text{EX}$, but for all $f \in \mathcal{S}$ and for every strongly recursive index e of a tree on which f is isolated, the inference machine M converges on input $e, f(0), f(1), \dots$ to some fixed program for f .*

Proof. The proof is a modification of a proof used by Kummer and Stephan [10, Theorem 7.1] in order to show that there is a class in EX^1 that can be $\text{EX}[A]$ -learned only if A is high.

The principal function p of the complement \bar{V} of a set V is defined by

$$p(0) = \min(\bar{V}) \quad \text{and} \quad p(n+1) = \min(\bar{V} - \{p(0), p(1), \dots, p(n)\}).$$

There is an r.e. set V such that the principal function of its complement dominates all recursive functions; for such V , any infinite set $E \subseteq^* \bar{V}$ has high Turing degree [21, XI.1], where $X \subseteq^* Y$ means that all but at most finitely many elements of X are also elements of Y .

The class \mathcal{S} is defined as

$$\mathcal{S} = \{f \in \text{REC}_{0,1} : (\exists x)[f \text{ extends } \varphi_{g(x)}]\},$$

where $\text{REC}_{0,1}$ is the class of recursive $\{0, 1\}$ -valued functions and g is a recursive function that we will define shortly. In the construction of g , we ensure that the sequence $\varphi_{g(0)}, \varphi_{g(1)}, \dots$ has the following properties.

- (a) $0^x 1 \preceq \varphi_{g(x)}$ and $\varphi_{g(x)}$ is $\{0, 1\}$ -valued.
- (b) $V \subseteq \text{dom}(\varphi_{g(x)})$.
- (c) If W_x is finite and e is a program that computes a total extension of $\varphi_{g(x)}$, then $e > |W_x|$.
- (d) If W_x is finite and e is a recursive index of a tree T such that some total extension of $\varphi_{g(x)}$ is an isolated branch on T , then $e > |W_x|$.
- (e) If W_x is finite, then $\varphi_{g(x)}(y) \downarrow = 1$ only for finitely many y .
- (f) If W_x is infinite, then $\varphi_{g(x)}$ is total.

For given x , the function $\varphi_{g(x)}$ is the limit of partial functions $\varphi_{g(x),s}$, which are constructed in stages. Initially, the function $\varphi_{g(x),0}$ is set to $0^x 1$ and during each stage $s \geq 0$, the procedure tries to extend the function $\varphi_{g(x),s}$ to $\varphi_{g(x),s+1}$ such that in the limit the requirements (b), (c), and (d) are met. The construction is effective in x , hence g can be chosen to be recursive. As usual, we say two partial functions α and β are consistent if and only if $\alpha(y) = \beta(y)$ for all arguments y on which both of α and β are defined.

Stage $s \equiv 0 \pmod{3}$ (stage of type (b)):

Let $\varphi_{g(x),s+1}(y) = 0$ for all $y \in V_s - \text{dom}(\varphi_{g(x),s})$.

Stage $s \equiv 1 \pmod{3}$ (stage of type (c)):

Pick the least $e \leq |W_{x,s}|$ such that $\varphi_{g(x),s}$ and $\varphi_{e,s}$ are consistent and there is a number y that is in the domain of $\varphi_{e,s}$ but not in the domain of $\varphi_{g(x),s}$. Ensure that $\varphi_{g(x)}$ differs from φ_e at the least such y by letting $\varphi_{g(x),s+1}(y)$ be equal to a value in $0, 1$ different from $\varphi_{e,s}(y)$.

If there is no such e , then let $\varphi_{g(x),s+1} = \varphi_{g(x),s}$.

Stage $s \equiv 2 \pmod{3}$ (stage of type (d)):

Pick the least $e \leq |W_{x,s}|$ such that we had not yet diagonalized against e during a previous stage of type (d) and such that there is $\sigma \in \{0, 1\}^*$ that is consistent with $\varphi_{g(x),s}$ and where $T_{e,s}(\sigma) \downarrow = 0$, i.e., where σ is not contained in the tree given by e . Diagonalize against e by picking the least such σ , then letting $\varphi_{g(x),s+1}(y) = \sigma(y)$ for all y that are in the domain of σ but not in the domain of $\varphi_{g(x),s}$.

If there is no such e , let $\varphi_{g(x),s+1} = \varphi_{g(x),s}$.

First it is verified that the construction meets the six conditions. Condition (a) is satisfied by the initialization of $\varphi_{g(x)}$ and because the values of g are always chosen from $\{0, 1\}$. During the stages of type (b), $\varphi_{g(x)}$ gets defined on all elements of V_s . Therefore V is contained in the domain of $\varphi_{g(x)}$, i.e., (b) holds. Conditions (c), (d), and (e) have to be satisfied only if W_x is finite and (f) only if W_x is infinite.

Assume that W_x is finite. Then there are only finitely many stages of type (c) or (d) during which the domain of $\varphi_{g(x)}$ is properly extended, while during stages of type (b), $\varphi_{g(x)}$ becomes defined

just on arguments in V where the assigned value is 0. Consequently, first, $\varphi_{g(x)}$ maps only finitely many arguments to 1 and (e) is satisfied. Second, the domain of $\varphi_{g(x)}$ contains at most finitely many places that are not in V , hence there are infinitely many undefined places. Now assume for a proof by contradiction that (c) was false, i.e., there is an index $e \leq |W_x|$ such that φ_e is a total extension of $\varphi_{g(x)}$. Then $\varphi_{g(x),s}$ and $\varphi_{e,s}$ are consistent for all s and the set $\text{dom}(\varphi_{e,s}) - \text{dom}(\varphi_{g(x)})$ is nonempty for almost all s , i.e., for all sufficiently large stages s of type (c) the index e satisfies the condition in the definition of the stage. As a consequence, and contrary to our assumption, after at most e stages of type (c) at which indices smaller than e are handled, eventually there is a stage during which $\varphi_{g(x)}$ is ensured to be inconsistent with φ_e .

Condition (d) follows by a similar argument. Assume that $e \leq |W_x|$ is a recursive index for the tree T . In case some total extension of $\varphi_{g(x)}$ is not on T , then eventually during substage (d) we let $\varphi_{g(x)}$ agree with some σ not in T on the domain of σ , thereby ensuring that no total extension of $\varphi_{g(x)}$ is on T . Consequently, either all or none of the total extensions of $\varphi_{g(x)}$ are on T . In the former case none of the total extensions of $\varphi_{g(x)}$ is isolated on T (since $\varphi_{g(x)}$ is undefined at infinitely many places) and in the latter case no extension is on T . So, also (d) holds.

Assume that W_x is infinite. For each y there is a function $\varphi_{u(y)}$ with domain $\{y\}$. As long as $\varphi_{g(x),s}$ is not defined on y , $\varphi_{g(x),s}$ and $\varphi_{u(y),s}$ are consistent. For all sufficiently large stages s , we have $u(y) \leq |W_{x,s}|$ and $\varphi_{u(y),s}(y) \downarrow$. So, in some substage of type (c), the value $\varphi_{g(x),s+1}(y)$ will be defined in order to diagonalize against $u(y)$, if it has not been defined before. Therefore $\varphi_{g(x)}$ is total and (f) is satisfied.

Second it is shown that \mathcal{S} is learnable in the limit with additional information. The inference algorithm for \mathcal{S} works as follows.

On input $e\sigma \preceq ef(0)f(1)f(2)\dots$, compute the x with $0^x 1 \preceq f$ and check whether $e < |W_{x,|\sigma|}|$.

If so, output the index $g(x)$.

Otherwise, output a canonical index for the set $\sigma 0^\infty$.

In order to verify the algorithm, fix any input of the form $ef(0)f(1)f(2)\dots$ where $f \in \mathcal{S}$, e is an index for a strongly recursive tree and f is isolated on T_e ; note that for other inputs there is nothing to prove. Then, by definition of f , there is an x such that f extends $\varphi_{g(x)}$ and $0^x 1 \preceq f$.

If W_x is infinite, then $g(x)$ is an index of a total function according to (f), thus $\varphi_{g(x)}$ is its own total extension. In this case, for all sufficiently large σ , the algorithm goes into the yes-case and outputs the correct index $g(x)$.

Next assume that W_x is finite. Let $H_f = \{y: f(y) = 1\}$. Then f extends the partial function $\varphi_{g(x)}$, which is defined on the whole set V by (b) and assumes the value 1 at most finitely often according to (e), hence $H_f \subseteq {}^*V$. Furthermore, the set H_f is recursive and hence is finite by assumption on V , i.e., f maps only finitely many arguments to 1. By construction, f is isolated on T_e only if $e > |W_x|$. Therefore $e > |W_{x,|\sigma|}|$ for all $\sigma \preceq f$ and the algorithm always outputs a canonical index for $\sigma 0^\infty$. If $\sigma \preceq f$ is sufficiently long, the equation $f = \sigma 0^\infty$ holds, hence the algorithms almost always outputs the same correct index for f .

Third it has to be shown that $\mathcal{S} \notin \text{EX}$. Assume by way of contradiction that there is a learner M that EX-learns all functions in \mathcal{S} , i.e., M learns any recursive, $\{0,1\}$ -valued extension of any

partial function of the form $\varphi_{g(x)}$. Consider the set

$$T = \{\sigma: \sigma \text{ is a binary string and for all } i = \langle x, y \rangle \text{ in the domain of } \sigma \text{ holds that} \\ \text{if the computation of } \varphi_{g(x)}(y) \text{ terminates in at most } |\sigma| \text{ steps, then } \sigma(i) = \varphi_{g(x)}(y)\}.$$

By definition, the set T is recursive and closed under taking prefixes, thus T is a recursive tree. Furthermore, the set of infinite branches of T is nonempty. Hence by the Low Basis Theorem [21, VI.5.13], the tree T has an infinite branch A that is of low degree (the infinite branch is considered as a set in the obvious way).

By construction, $A(\langle x, y \rangle) = \varphi_{g(x)}(y)$ whenever $\varphi_{g(x)}(y) \downarrow$. Furthermore, for all x , the set $A_x = \{y: \langle x, y \rangle \in A\}$ is recursive. Either W_x is infinite and then $\varphi_{g(x)}$ is a characteristic function for A_x or W_x is finite and then $\varphi_{g(x)}(y) \downarrow = 1$ for only finitely many y , hence $A_x \subseteq {}^*\bar{V}$ and since A_x is low, A_x is not high and thus finite. As a consequence, M has to learn all the sets A_x . Since A is low there is a K -recursive algorithm that computes the limit $e(x)$ of the values $M(A_x(0)A_x(1)\dots A_x(y))$. Since $e(x)$ is an index of the characteristic function of A_x , condition (c) postulates that $e(x) \geq |W_x|$ whenever W_x is finite. Now the K -recursive test whether $|W_x| > e(x)$ decides whether W_x is infinite. If $|W_x| > e(x)$ then W_x is infinite and if $|W_x| \leq e(x)$ then W_x is finite. So the set $\{x: W_x \text{ is finite}\}$ would be recursive in K —this contradicts the fact that this set is Σ_2 -complete [21, Theorem IV.3.2]. Hence such an M does not exist. \square

Acknowledgments

We would like to thank William Gasarch for helpful discussion and the anonymous referees of the *Conference on Learning Theory 1996* and of the *Journal of Computer and System Sciences* for their comments.

References

- [1] D. Angluin, C.H. Smith, Inductive inference: theory and methods, *Computing Surveys* 15 (1983) 237–269.
- [2] L. Blum, M. Blum, Towards a mathematical notion of inductive inference, *Inform. and Control* 28 (1975) 125–155.
- [3] J. Case, S. Kaufmann, E. Kinber, M. Kummer, Learning recursive functions from approximations, *J. Comput. System Sci.* 55 (1997) 183–196.
- [4] J. Case, C.H. Smith, Comparison of identification criteria for machine inductive inference, *Theoret. Comput. Sci.* 25 (1983) 193–220.
- [5] R.V. Freivalds, R. Wiehagen, Inductive inference with additional information, *Elektron. Informationsverarbeitung Kybernet.* 15 (1979) 179–185.
- [6] E.M. Gold, Language identification in the limit, *Inform. and Control* 10 (1967) 447–474.
- [7] S. Jain, E.B. Kinber, R. Wiehagen, T. Zeugmann, On learning of functions refutably, *Theoret. Comput. Sci.* 298 (2003) 111–143.
- [8] S. Jain, E. Kinber, R. Wiehagen, T. Zeugmann, On learning of functions refutably, Manuscript, 2001.
- [9] S. Jain, D.N. Osherson, J.S. Royer, A. Sharma, *Systems that Learn: An Introduction to Learning Theory*, MIT Press, Cambridge, MA, 1999 (2nd Edition of [18]).
- [10] M. Kummer, F. Stephan, On the structure of degrees of inferability, *J. Comput. System Sci. (Special Issue COLT 1993)* 52 (1996) 214–238.

- [11] W. Merkle, F. Stephan, Refuting learning revisited, *Theoret. Comput. Sci.* 298 (2003) 145–177.
- [12] E. Minicozzi, Some natural properties of strong-identification in inductive inference, *Theoret. Comput. Sci.* 2 (1976) 345–360.
- [13] Y. Mukouchi, S. Arikawa, Inductive inference machines that can refute hypothesis spaces, in: K.P. Jantke, et al., (Eds.), *Algorithmic Learning Theory 1993*, Lecture Notes in Computer Science, Vol. 744, Springer, New York, 1993, pp. 123–136.
- [14] Y. Mukouchi, S. Arikawa, Towards a mathematical theory of machine discovery from facts, *Theoret. Comput. Sci.* 137 (1995) 53–84.
- [15] P. Odifreddi, *Classical Recursion Theory*, Vol. I, North-Holland, Amsterdam, 1989.
- [16] P. Odifreddi, *Classical Recursion Theory*, Vol. II, Elsevier, Amsterdam, 1999.
- [17] D. Osherson, M. Stob, S. Weinstein, *Systems that Learn—An Introduction to Learning Theory for Cognitive and Computer Scientists*, MIT Press, Cambridge, MA, 1986.
- [18] L. Pitt, C.H. Smith, Probability and plurality for aggregations of learning machines, *Inform. and Comput.* 77 (1988) 77–92.
- [19] K.M. Podnieks, Comparing various concepts of function prediction, Part I, *Učēnye Zapiski Latvījskovo Universitet* 210 (1974) 68–81 (in Russian).
- [20] C.H. Smith, Three decades of team-learning, in: S. Arikawa, et al., (Eds.), *Algorithmic Learning Theory 1994*, Lecture Notes in Computer Science, Vol. 872, Springer, New York, 1994, pp. 211–228.
- [21] R. Soare, *Recursively Enumerable Sets and Degrees*, Springer, Heidelberg, 1987.
- [22] R. Wiehagen, A thesis in inductive inference, in: J. Dix, et al., (Eds.), *Nonmonotonic and Inductive Logic*, Lecture Notes in Computer Science, Vol. 543, Springer, New York, 1990, pp. 184–207.
- [23] R. Wiehagen, H. Jung, Recursion-theoretic characterizations of identifiable classes of recursive functions, *Elektron. Informationsverarbeitung Kybernet.* 18 (1977) 385–397 (in German).
- [24] R. Wiehagen, W. Liepke, Characteristic properties of identifiable classes of recursive functions, *Elektronische Informationsverarbeitung und Kybernetik* 12 (1976) 421–438 (in German).