

Robust Learning is Rich

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

Carl Smith
Department of Computer Science
University of Maryland
College Park, MD 20742
USA
smith@cs.umd.edu

Rolf Wiehagen
Department of Computer Science
University of Kaiserslautern
D-67653 Kaiserslautern
Germany
wiehagen@informatik.uni-kl.de

Abstract

Intuitively, a class of objects is robustly learnable if not only this class itself is learnable but all of its computable transformations remain learnable as well. In that sense, being learnable robustly seems to be a desirable property in all fields of learning.

We will study this phenomenon within the paradigm of inductive inference. Here a class of recursive functions is called robustly learnable under a success criterion **I** iff all of its images under general recursive operators are learnable under the criterion **I**. Fulk [Ful90] showed the existence of a non-trivial class which is robustly learnable under the criterion **Ex**. However, several of the hierarchies (such as the anomaly hierarchies for **Ex** and **Bc**) do not stand robustly. Hence, up to now it was not clear if robust learning is really rich. The main intention of this paper is to give strong evidence that robust learning *is* rich.

Our main result proved by a priority construction is that the mind change hierarchy for **Ex** stands robustly. Moreover, the hierarchies of team learning for both **Ex** and **Bc** stand robustly as well. In several contexts, we observe the surprising fact that a more complex topological structure of the classes to be learned leads to positive robustness results, whereas an easy topological structure yields negative results. We also show the counter-intuitive fact that even some self-referential classes can be learned robustly. Some of our results point out the difficulty of robust learning when only a bounded number of mind changes is allowed. Further results concerning uniformly robust learning are derived.

1 Introduction

Consider the following basic learning scenario. A learning machine has to learn some unknown object, that is, based on some information the machine creates one or more hypotheses which finally yield a correct global description of that object. Then consider the extent to which such a machine can be a general purpose learner in that it can learn each object from a, finite or even infinite, class of objects. In various learning models this can be shown to be possible. A next question to ask then could be the following: How “stable” is the property that a class is learnable?

That is, under which, small or not so small, “transformations” the transformed classes remain learnable? Of course, this may depend on the class under consideration; some classes may be more stable in that sense than others. In the best case, there could be learnable classes such that *all* of their “derivatives” remain learnable. Such classes, if any, we call *robustly* learnable. Do they exist at all? And, if they do, how “rich” are they? Are they worth studying? It may be interesting to answer these questions in any concrete learning model. We will do so within the paradigm of inductive inference. Our main intention is to give strong evidence that in this model, and hence, hopefully, in others too, robust learning is really rich.

The basic learning situation in inductive inference may be described as follows. A learner receives as input a graph of a function f , one element at a time. As the learner is receiving its input, it conjectures a sequence of programs as hypotheses. The learner is said to **Ex**-identify f iff the sequence of programs output by it converges to a program for f . This is essentially the model of identification introduced by Gold [Gol67] (see the formal definitions in Section 2).

In this paper we will restrict our attention to computable learners of (classes of) total and computable, i.e. recursive, functions. Let us consider the extent to which a machine \mathbf{M} can be a general purpose learner, i.e., to what extent it can, say, **Ex**-identify each function f in a class of functions. For example, it is not too difficult to show that a suitable machine \mathbf{M} can **Ex**-identify the class of all the polynomials. Gold [Gol67] even showed that one can **Ex**-identify every recursively enumerable class of recursive functions. This can be done as follows: Suppose \mathcal{C} is a recursively enumerable class of recursive functions. Let p_0, p_1, \dots be an effective sequence of programs which compute exactly the functions in \mathcal{C} . Consider a machine \mathbf{M} which behaves as follows: on any input data, \mathbf{M} searches for the least i such that the function computed by program p_i is consistent with the input data; \mathbf{M} then outputs p_i . It can be shown that \mathbf{M} acting as above will **Ex**-identify each function in \mathcal{C} . The above technique is often called *identification by enumeration*. The naturalness of this strategy led Gold to conjecture that *any* class of functions which can be **Ex**-identified, can also be **Ex**-identified using identification by enumeration. In other words, Gold’s conjecture was: every **Ex**-identifiable class is contained in a recursively enumerable class of functions. However, as Bärzdiņš proved in [Bär71], this conjecture is false. He exhibited the following “self-describing” class SD of recursive functions, $\text{SD} = \{f \mid f(0) \text{ is a program for } f\}$. A machine can **Ex**-identify each function f in SD by just outputting the program $f(0)$. On the other hand, no recursively enumerable class of recursive functions contains SD .

In the 1970’s Bärzdiņš came up with a more sophisticated version of Gold’s conjecture designed to transcend such self-referential counterexamples as above. He reasoned that if a class of functions is identifiable only by way of a self-referential property, then there would be an “effective transformation” that would transform the class into an *unidentifiable* one. The idea is that if a learning device is able to find the embedded self-referential information in the elements of a class, so can an effective transformation, which can then weed out this information. Naturally, the notion of an effective transformation can be made precise in several ways. In the present paper we therefore use the concept of general recursive operators, i.e. effective and total mappings from total functions to total functions (see Definition 6); below we will discuss this choice and possible alternatives in more detail. In order to illustrate Bärzdiņš’ intuition in the context of the class SD above, consider the operator Θ weeding out the self-referential information $f(0)$ as follows: $\Theta(f) = g$, where $g(x) = f(x + 1)$. One can show that $\Theta(\text{SD}) = \{\Theta(f) \mid f \in \text{SD}\} = \mathcal{R}$, the class of all the recursive functions. Thus, $\Theta(\text{SD})$ is *not* **Ex**-identifiable [Gol67]. Informally, Bärzdiņš’ conjecture then can be stated as follows: If all the projections of a class of functions under all general recursive operators are identifiable (or, in other words, if the class is identifiable *robustly*), then the class is contained

in a recursively enumerable class of recursive functions and, consequently, it is identifiable by enumeration. This was how the notion of robust learning appeared in inductive inference historically. Zeugmann [Zeu86] and Kurtz and Smith [KS89] then dealt with Bärzdiņš’ conjecture. The paper [Zeu86] is remarkable in several respects. It gives a formal statement of Bärzdiņš’ conjecture for the first time. To this end, effective operators (see [Rog67]) rather than general recursive operators are used to make the notion of effective transformations mathematically precise. Then this conjecture has been verified for several learning criteria, namely \mathbf{Ex}_0 -identification (see Definition 1), $\mathcal{R}\mathbf{Ex}$ -identification and $\mathcal{T}\mathbf{Ex}$ -identification (reliable learning; here the learning machine is not allowed to converge on a recursive or total function, respectively, which cannot be learned by the machine). For \mathbf{Ex} -identification, Bärzdiņš’ conjecture remained open. The paper [KS89] sparked Fulk’s interest in Bärzdiņš’ conjecture. He then showed in [Ful90] that Bärzdiņš’ conjecture as stated above is false by exhibiting a class of functions which is robustly \mathbf{Ex} -identifiable, but not contained in any recursively enumerable class of recursive functions. This result can be taken as the first non-trivial step to show that, in the model of inductive inference, robust learning may be really interesting and rich.

Since Gold [Gol67] many criteria of inference have been proposed by researchers all over the world, see for example [AS83, BB75, CS83, Fre91, KW80, JORS99]. These have usually been accompanied by proofs showing the differences between the new and old criteria of inference. The proof techniques used to show separations between the criteria often involve classes with self-referential properties. Thus, it would be interesting to study whether these separations hold robustly. For example, Fulk [Ful90] showed that the anomaly hierarchies for \mathbf{Ex} and \mathbf{Bc} -identification (see formal definitions in Section 2) do not hold robustly. Hence, one may expect some celebrated results of inductive inference (especially the hierarchies) not to stand robustly.

In this paper we further study robust identification. Our main result, Theorem 24 and Corollary 31, shows that the mind change hierarchy with respect to \mathbf{Ex} -identification stands robustly! Contrast this with the fact that the anomaly hierarchies for \mathbf{Ex} and \mathbf{Bc} -identification do not stand robustly. The proof of this result uses a complicated priority construction. We can also show that the hierarchies of team learning for both \mathbf{Ex} and \mathbf{Bc} [Smi82] stand robustly as well (Theorem 41 and Corollary 42). Moreover, we exhibit the counter-intuitive fact that even some of the self-describing classes can be learned robustly (Theorem 36). In a sense, this yields also a “second order” disproof of Bärzdiņš’ conjecture, since we disprove it even with self-describing classes, i.e. with classes for which it was commonly believed that this conjecture does hold. Consequently, there are two kinds of self-describing classes, namely robustly learnable ones, and not robustly learnable ones (such as SD mentioned above). In order to find out where this difference may come from, we made a surprising observation, namely that the robust self-describing class from Theorem 36 has a much more complex topological structure (more exactly, it possesses some kind of “accumulation point”) than the non-robustly learnable class SD (which is “extremely discrete” in that $f(0) \neq g(0)$, for all distinct f and g in SD). Moreover, it is precisely this topological structure which apparently yields the corresponding robustness property (see the more detailed discussion after Theorem 36, where we can rely on the corresponding proof). Note that both SD and the class from Theorem 36 are not contained in any recursively enumerable class of recursive functions. Interestingly, a similar type of phenomenon is observed also for recursively enumerable classes. There, again, the more complex topological structure leads to a positive robustness result (concerning the existence of an infinite robustly learnable subclass), whereas the “trivial” structure yields a corresponding negative result (see Theorem 22, Corollary 33, and the discussion following Corollary 33).

Several of our results show the difficulty of robustly identifying even simple classes when only

a bounded number of mind changes is allowed. For example, Theorem 16 shows that no infinite class of functions can be robustly finitely identified (i.e. **Ex**-identified without any mind changes). Theorem 18 states that no infinite recursively enumerable class of functions can be robustly identified with a bounded number of mind changes. Theorem 22 points out that some simple classes such as the class of all constant functions, do not even contain any infinite subclass which can be robustly **Ex**-identified with a bounded number of mind changes.

We have also considered *uniformly* robust learning. Informally, a class \mathcal{C} is uniformly robustly learnable if \mathcal{C} is robustly learnable, and, moreover, given any general recursive operator Θ , one can effectively generate a machine which learns the class $\Theta(\mathcal{C})$. In other words, the images of \mathcal{C} are all not only learnable in that learning machines for them *exist*, but one even has learning machines for them effectively *at hand*. For this strengthened version of robust learning, we have both positive and negative results. Actually, for **Ex**-learning with a bounded number of mind changes, uniformly robust learning is possible only for very restricted classes, whereas for standard **Ex**-learning as well as for some kind of generalized **Ex**-learning, uniformly robust learning seems to be achievable for quite rich classes.

We have mainly concentrated on robustness under transformations by *general recursive* operators. This is mainly due to the fact that we deal with learning of *recursive* functions, and general recursive operators transform any class of recursive functions “automatically” into a class of recursive functions again. Also in the papers [CJO⁺98] and [OS99] dealing with robust learning, the set of general recursive operators or subsets of this set such as the primitive recursive operators are used, respectively. On the other hand, it may be useful to consider robustness under transformations by *recursive* operators which are only required to take the functions in the class being learned to total functions, but need not map functions outside this class to total functions. In Section 6, we note that some of our positive results, such as the mind change hierarchy, hold even for this strengthened form of robustness. Moreover, again some self-referential classes turn out to be robustly learnable in this strengthened sense. Hence, as a non-expected consequence, even recursive operators are not capable of weeding out all self-referential codings! However, other results specifically use the properties of general recursive operators, and thus do not go through for this strengthened form of robustness. As already mentioned above, there are further approaches to make the notion of an “effective transformation” mathematically precise. Recall that in [Zeu86] effective operators are used for this purpose. At this moment, we do not see any “master approach” to this end. Actually, each approach seems to be justified if it yields interesting results.

In [CJO⁺98] robust learning has been studied for another specific learning scenario, namely learning aided by context. The intuition behind this model is to present the functions to be learned not in a pure fashion to the learner, but together with some “context” which is intended to help in learning. It is shown then that within this scenario several results hold robustly as well. In [OS99] the notion of hyperrobust learning is introduced. A class of recursive functions is called hyperrobustly learnable if there is one and the same learner which learns not only this class itself but also all of its images under all primitive recursive operators. Hence this learner must be capable to learn the *union* of all these images. This definition is then justified by the following results. First, it is shown that the power of hyperrobust learning does not change if the class of primitive recursive operators is replaced by any larger, still recursively enumerable class of general recursive operators. Second, based on this stronger definition Bärzdiņš’ conjecture is proved by showing that a class of recursive functions is hyperrobustly **Ex**-learnable iff this class is contained in a recursively enumerable class of recursive functions.

On a philosophical side, Herman Weyl [Wey52] started to describe the famous Erlangen program

on founding geometry algebraically due to Felix Klein as follows: “If you are to find deep properties of some object, consider all the natural transformations that preserve your object (i.e. under which the object remains invariant).” Since general recursive operators (or other types of operators) can be looked upon as natural transformations, it is interesting to consider robust identification from a purely philosophical point of view too. Note that in [AF96] a problem dual to ours is investigated. While we will search for *learnable classes* which remain learnable under all general recursive operators, Ambainis and Freivalds - even more in the spirit of the Erlangen program - search for such *general recursive operators* which map all learnable classes to learnable classes. However, being robustly identifiable is a desirable property worth studying *on its own*. Actually, we feel it fully justified to find out both, which classes of objects are not only learnable themselves but also all of their “derivatives” and where this property comes from.

The paper is organized as follows. In Section 2 the needed definitions and some basic results are presented. In Section 3 robust learning with a bounded number of mind changes is investigated. In Section 4 the hierarchies on robust team identification are derived. Section 5 deals with uniformly robust learning. In Section 6 we discuss the results obtained and point out some directions for future work. Finally, note that the present paper is an extended version of both [JW97] and [JSW98].

2 Notation and Preliminaries

Recursion-theoretic concepts not explained below are treated in [Rog67]. N denotes the set of natural numbers. $*$ denotes a non-member of N and is assumed to satisfy $(\forall n)[n < * < \infty]$. Let $\in, \subseteq, \subset, \supseteq, \supset$, respectively denote the membership, subset, proper subset, superset and proper superset relations for sets. The empty set is denoted by \emptyset . We let $\text{card}(S)$ denote the cardinality of the set S . So “ $\text{card}(S) \leq *$ ” means that $\text{card}(S)$ is finite. The minimum and maximum of a set S are denoted by $\min(S)$ and $\max(S)$, respectively. We take $\max(\emptyset)$ to be 0 and $\min(\emptyset)$ to be ∞ . For a set A , 2^A denotes the power set of A .

$\langle \cdot, \cdot \rangle$ denotes a 1-1 computable mapping from pairs of natural numbers onto natural numbers. π_1, π_2 are the corresponding projection functions. $\langle \cdot, \cdot \rangle$ is extended to n -tuples of natural numbers in a natural way. Λ denotes the empty function. η , with or without decorations¹, ranges over partial functions. If η_1 and η_2 are both undefined on input x , then, we take $\eta_1(x) = \eta_2(x)$. We say that $\eta_1 \subseteq \eta_2$ iff for all x in domain of η_1 , $\eta_1(x) = \eta_2(x)$. For $a \in N \cup \{*\}$, $\eta_1 =^a \eta_2$ means that $\text{card}(\{x \mid \eta_1(x) \neq \eta_2(x)\}) \leq a$. $\eta_1 \neq^a \eta_2$ means that $\neg[\eta_1 =^a \eta_2]$. If $\eta =^a f$, then we often call a program for η as an a -error program for f . We let $\text{domain}(\eta)$ and $\text{range}(\eta)$ respectively denote the domain and range of the partial function η . $\eta(x) \downarrow$ denotes that $\eta(x)$ is defined. $\eta(x) \uparrow$ denotes that $\eta(x)$ is undefined. We say that a partial function η is *consistent* with η' iff for all $x \in \text{domain}(\eta) \cap \text{domain}(\eta')$, $\eta(x) = \eta'(x)$. η is *inconsistent* with η' iff there exists an x such that $\eta(x) \downarrow \neq \eta'(x) \downarrow$.

f, g, h, F and H , with or without decorations, range over total functions. \mathcal{R} denotes the class of all *recursive* functions, i.e., total computable functions with arguments and values from N . \mathcal{C} and \mathcal{S} , with or without decorations, range over subsets of \mathcal{R} . \mathcal{P} denotes the class of all *partial recursive* functions over N . φ denotes a *fixed* acceptable programming system. φ_i denotes the partial recursive function computed by program i in the φ -system. Note that in this paper all programs are interpreted with respect to the φ -system. We let Φ be an arbitrary Blum complexity measure [Blu67] associated with the acceptable programming system φ ; many such measures exist

¹Decorations are subscripts, superscripts, primes, and the like.

for any acceptable programming system [Blu67].

A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be recursively enumerable (r.e.) iff there exists an r.e. set X such that $\mathcal{C} = \{\varphi_i \mid i \in X\}$. For any non-empty recursively enumerable class \mathcal{C} , there exists a recursive function f such that $\mathcal{C} = \{\varphi_{f(i)} \mid i \in N\}$. A function g is said to be an *accumulation point* of a class $\mathcal{C} \subseteq \mathcal{R}$ iff $g \in \mathcal{R}$ and $(\forall n \in N)(\exists f \in \mathcal{C})(\forall x \leq n)[f(x) = g(x)] \wedge f \neq g$. Note that the accumulation point may or may not belong to the class. The following functions and classes are commonly considered below. Zero is the everywhere 0 function, i.e., $\text{Zero}(x) = 0$, for all $x \in N$. $\text{CONST} = \{f \mid (\forall x)[f(x) = f(0)]\}$ denotes the class of the constant functions. $\text{FINSUP} = \{f \mid (\forall^\infty x)[f(x) = 0]\}$ denotes the class of all recursive functions of finite support.

2.1 Function Identification

We first describe inductive inference machines. We assume, without loss of generality, that the graph of a function is fed to a machine in canonical order. For $f \in \mathcal{R}$ and $n \in N$, we let $f[n]$ denote the finite initial segment $\{(x, f(x)) \mid x < n\}$. Clearly, $f[0]$ denotes the empty segment. SEG denotes the set of all finite initial segments, $\{f[n] \mid f \in \mathcal{R} \wedge n \in N\}$. We let σ and τ , with or without decorations, range over SEG . Let $|\sigma|$ denote the length of σ . We often identify (partial) functions with their graphs. Thus for example, for $\sigma = f[n]$ and for $x < n$, $\sigma(x)$ denotes $f(x)$. An *inductive inference machine* (IIM) [Gol67] is an algorithmic device that computes a mapping from SEG into $N \cup \{?\}$. Intuitively, “?” above denotes the case when the machine may not wish to make a conjecture. Although it is not necessary to consider learners that issue “?” for identification in the limit, it becomes useful when the number of mind changes a learner can make is bounded (see Definition 1 below). In this paper, we assume, without loss of generality, that once an IIM has issued a conjecture on some initial segment of a function, it outputs a conjecture on all extensions of that initial segment. This is without loss of generality because a machine wishing to emit “?” after making a conjecture can instead be thought of as repeating its previous conjecture. We let \mathbf{M} , with or without decorations, range over learning machines. Since the set of all finite initial segments, SEG , can be coded onto N , we can view these machines as taking natural numbers as input and emitting natural numbers or ?’s as output. We say that $\mathbf{M}(f)$ converges to i (written: $\mathbf{M}(f)\downarrow = i$) iff $(\forall^\infty n)[\mathbf{M}(f[n]) = i]$; $\mathbf{M}(f)$ is undefined if no such i exists. The next definitions describe several criteria of function identification.

Definition 1 [Gol67, BB75, CS83] Let $a, b \in N \cup \{*\}$. Let $f \in \mathcal{R}$.

- (a) $\mathbf{M} \text{ Ex}_b^a$ -identifies f (written: $f \in \text{Ex}_b^a(\mathbf{M})$) just in case there exists an a -error program i for f such that $\mathbf{M}(f)\downarrow = i$ and $\text{card}(\{n \mid ? \neq \mathbf{M}(f[n]) \neq \mathbf{M}(f[n+1])\}) \leq b$ (i.e., \mathbf{M} makes no more than b mind changes on f).
- (b) $\mathbf{M} \text{ Ex}_b^a$ -identifies \mathcal{S} iff $\mathbf{M} \text{ Ex}_b^a$ -identifies each $f \in \mathcal{S}$.
- (c) $\text{Ex}_b^a = \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \text{Ex}_b^a(\mathbf{M})]\}$.

Note that in part (a) above, change of conjecture from ? to some $i \in N$ is not considered a mind change.

We often write Ex_b for Ex_b^0 , Ex^a for Ex_*^a , and Ex for Ex_*^0 . Ex_0 is also referred to as *finite identification*. By the definition of convergence, only finitely many data points from a function f have been observed by an IIM \mathbf{M} at the (unknown) point of convergence. Hence, some form of learning must take place in order for \mathbf{M} to learn f . For this reason, hereafter the terms *identify*, *learn* and *infer* are used interchangeably.

Definition 2 [Bär74, CS83] Let $a \in N \cup \{*\}$. Let $f \in \mathcal{R}$.

- (a) $\mathbf{M} \mathbf{Bc}^a$ -identifies f (written: $f \in \mathbf{Bc}^a(\mathbf{M})$) iff, for all but finitely many $n \in N$, $\mathbf{M}(f[n])$ is an a -error program for f .
- (b) $\mathbf{M} \mathbf{Bc}^a$ -identifies \mathcal{S} iff $\mathbf{M} \mathbf{Bc}^a$ -identifies each $f \in \mathcal{S}$.
- (c) $\mathbf{Bc}^a = \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Bc}^a(\mathbf{M})]\}$.

We often write \mathbf{Bc} for \mathbf{Bc}^0 .

Definition 3 $\mathbf{NUM} = \{\mathcal{C} \mid (\exists \mathcal{C}' \mid \mathcal{C} \subseteq \mathcal{C}' \subseteq \mathcal{R})[\mathcal{C}' \text{ is recursively enumerable}]\}$.

Some relationships between the above criteria are summarized in the following theorem.

Theorem 4 [CS83, BB75, Bär71, Gol67]

- (a) Let $b \in N \cup \{*\}$. Then, $\mathbf{Ex}_b = \mathbf{Ex}_b^0 \subset \mathbf{Ex}_b^1 \subset \mathbf{Ex}_b^2 \subset \dots \subset \mathbf{Ex}_b^*$.
- (b) Let $a \in N \cup \{*\}$. Then, $\mathbf{Ex}_0^a \subset \mathbf{Ex}_1^a \subset \mathbf{Ex}_2^a \subset \dots \subset \mathbf{Ex}_*^a$.
- (c) Let $a, b, c, d \in N \cup \{*\}$. Then, $\mathbf{Ex}_b^a \subseteq \mathbf{Ex}_d^c$ iff $a \leq c$ and $b \leq d$.
- (d) $\mathbf{NUM} \subseteq \mathbf{Ex}$.
- (e) $\mathbf{Ex}_0 - \mathbf{NUM} \neq \emptyset$.
- (f) $\mathbf{NUM} - \bigcup_{m \in N} \mathbf{Ex}_m \neq \emptyset$.
- (g) $\mathbf{FINSUP} \not\subseteq \bigcup_{m \in N} \mathbf{Ex}_m$.
- (h) $\mathbf{Ex}_*^* \subset \mathbf{Bc} = \mathbf{Bc}^0 \subset \mathbf{Bc}^1 \subset \mathbf{Bc}^2 \subset \dots \subset \mathbf{Bc}^* = 2^{\mathcal{R}}$.

We let \mathbf{I} and \mathbf{J} range over identification criteria defined above. Below we will mainly deal with \mathbf{NUM} , \mathbf{Ex}_0 , \mathbf{Ex}_m , $\bigcup_{m \in N} \mathbf{Ex}_m$ and \mathbf{Ex} . Using essentially the idea in [JORS99, Proposition 4.22], (for \mathbf{Ex} -identification), for all criteria \mathbf{I} of inference considered in this paper, one can show that:

There exists an r.e. sequence $\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2, \dots$, of inductive inference machines such that, for any \mathbf{I} , for all $\mathcal{C} \in \mathbf{I}$, there exists an $i \in N$ such that $\mathcal{C} \subseteq \mathbf{I}(\mathbf{M}_i)$.

We assume $\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2, \dots$ to be one such sequence of machines.

2.2 Operators

Definition 5 [Rog67] A *recursive operator* is an effective total mapping, Θ , from (possibly partial) functions to (possibly partial) functions, which satisfies the following properties:

- (a) Monotonicity: For all functions η, η' , if $\eta \subseteq \eta'$ then $\Theta(\eta) \subseteq \Theta(\eta')$.
- (b) Compactness: For all η , if $(x, y) \in \Theta(\eta)$, then there exists a finite function $\alpha \subseteq \eta$ such that $(x, y) \in \Theta(\alpha)$.
- (c) Recursiveness: For all finite functions α , one can effectively enumerate (in α) all $(x, y) \in \Theta(\alpha)$.

Definition 6 [Rog67] A recursive operator Θ is called *general recursive* iff Θ maps all total functions to total functions.

For each recursive operator Θ , we can effectively (from Θ) find a recursive operator Θ' such that,

- (d) for each finite function α , $\Theta'(\alpha)$ is finite, and its canonical index can be effectively determined from α , and
- (e) for all total functions f , $\Theta'(f) = \Theta(f)$.

This allows us to get a nice effective sequence of recursive operators.

Proposition 7 *There exists an effective enumeration, $\Theta_0, \Theta_1, \dots$ of recursive operators satisfying condition (d) above such that, for all recursive operators Θ , there exists an $i \in \mathbb{N}$ satisfying:*

$$\text{for all total functions } f, \Theta(f) = \Theta_i(f).$$

PROOF. Let $\Theta^0, \Theta^1, \Theta^2, \dots$ denote a recursive enumeration of all the operators satisfying properties (b) Compactness and (c) Recursiveness above. Note that there exists such a recursive enumeration of operators. (Θ^i however may not be monotone). Define Θ_i as follows. We will define Θ_i on elements of SEG. This Θ_i can then be extended to all partial functions by taking $\Theta_i(\eta) = \bigcup\{\Theta_i(\sigma) \mid \sigma \subseteq \eta \wedge \sigma \in \text{SEG}\}$. Let $\Theta_i(\Lambda) = \Lambda$. Let $S_i(f[n+1]) = \bigcup_{m \leq n+1} [\Theta^i(f[m])]$ enumerated in $n+1$ steps]. For $n \geq 1$, let

$$\Theta_i(f[n+1]) = \begin{cases} S_i(f[n+1]), & \text{if } S_i(f[n+1]) \text{ denotes a partial function;} \\ \Theta_i(f[n]), & \text{otherwise.} \end{cases}$$

Note that $S_i(f[n+1])$ may not be a partial function (i.e. it may be multiply defined on some arguments). This is so, since Θ^i may not satisfy monotonicity, and hence S_i may not be monotone.

It is easy to verify that each Θ_i is a recursive operator (i.e. satisfies conditions (a) Monotonicity, (b) Compactness and (c) Recursiveness). Moreover, if Θ^i is a recursive operator, then $\Theta_i(f) = \Theta^i(f)$, for any total function f . The proposition follows. ■

Since we will be mainly concerned with the properties of operators on total functions, for diagonalization purposes, one can restrict attention to operators in the above enumeration $\Theta_0, \Theta_1, \dots$

Definition 8 Let \mathbf{I}, \mathbf{J} be identification criteria.

$$(\mathbf{I}, \mathbf{J})\text{-robust} = \{\mathcal{C} \mid \mathcal{C} \in \mathbf{I} \wedge (\forall \text{ general recursive operators } \Theta)[\Theta(\mathcal{C}) \in \mathbf{J}]\}.$$

Note that traditionally only (\mathbf{I}, \mathbf{I}) -robust identification is considered and referred to as robust \mathbf{I} -identification (as we did in Section 1). The above definition is a generalization of this notion. The reason we consider such a generalization is that there are classes which are not in (\mathbf{I}, \mathbf{I}) -robust, but they are in (\mathbf{I}, \mathbf{J}) -robust for \mathbf{J} a weaker identification criterion than \mathbf{I} , i.e. $\mathbf{I} \subset \mathbf{J}$. Alternatively, one may interpret a positive result on (\mathbf{I}, \mathbf{J}) -robustness as “how simple” (namely, even from \mathbf{I}) a robustly \mathbf{J} -identifiable class can be. Also, as seen by the following proposition, we always have (\mathbf{I}, \mathbf{J}) -robust as a subset of (\mathbf{J}, \mathbf{J}) -robust.

Proposition 9 (a) *Suppose $\mathbf{I} \subseteq \mathbf{I}'$, $\mathbf{J} \subseteq \mathbf{J}'$. Then (\mathbf{I}, \mathbf{J}) -robust \subseteq $(\mathbf{I}', \mathbf{J}')$ -robust.*

(b) (\mathbf{I}, \mathbf{J}) -robust = $(\mathbf{I} \cap \mathbf{J}, \mathbf{J})$ -robust.

PROOF. Follows easily from definitions. ■

Proposition 10 (a) $\text{NUM} = (\text{NUM}, \text{NUM})$ -robust.

(b) $\text{NUM} \subseteq (\mathbf{Ex}, \mathbf{Ex})$ -robust.

PROOF. (a) Suppose \mathcal{C} is recursively enumerable. Then for any general recursive operator Θ , $\Theta(\mathcal{C})$ is also recursively enumerable. Part (a) follows.

(b) Follows using part (a), Theorem 4(d) and Proposition 9(a). ■

2.3 Some Useful Propositions

In this subsection we prove some useful propositions. Some of these are folklore and likely to have been proven by others, either explicitly or implicitly. We include them here in order to make the paper self-contained. The following proposition is useful in proving the main result of the paper.

Proposition 11 *Suppose $n \in \mathbb{N}$, $\mathcal{S} \in \mathbf{Ex}_n$ and \mathcal{C} is finite. Then $\mathcal{S} \cup \mathcal{C} \in \mathbf{Ex}_{n+1}$.*

PROOF. Suppose $\mathcal{C} = \{f_0, f_1, \dots, f_m\}$, where f_i are distinct. Let s be such that, for each distinct $i, j \leq m$, $f_i[s] \neq f_j[s]$. For $i \leq m$, let p_i denote a program for f_i . Suppose $\mathbf{M} \in \mathbf{Ex}_n$ -identifies \mathcal{S} . Define \mathbf{M}' as follows:

$$\mathbf{M}'(f[z]) = \begin{cases} ?, & \text{if } z \leq s; \\ p_i, & \text{if } z > s, i \leq m, \text{ and } f_i[z] = f[z]; \\ \mathbf{M}(f[z]), & \text{if } z > s, \text{ and } (\forall i \leq m)[f_i[z] \neq f[z]]. \end{cases}$$

Note that \mathbf{M}' \mathbf{Ex} -identifies each function in $\mathcal{C} \cup \mathcal{S}$. Moreover, (i) $\mathbf{M}' \in \mathbf{Ex}_0$ -identifies \mathcal{C} , and (ii) \mathbf{M}' makes at most one extra mind change than \mathbf{M} on any function f . It follows that $\mathbf{M}' \in \mathbf{Ex}_{n+1}$ -identifies $\mathcal{S} \cup \mathcal{C}$. ■

Proposition 12 *Suppose $m \in \mathbb{N}$. For any infinite class $\mathcal{C} \in \mathbf{Ex}_m$, there exists an infinite subclass $\mathcal{C}' \subseteq \mathcal{C}$ such that $\mathcal{C}' \in \mathbf{Ex}_0$. Moreover, if \mathcal{C} is recursively enumerable, then \mathcal{C}' can be chosen to be recursively enumerable too.*

PROOF. Suppose $\mathcal{C} \in \mathbf{Ex}_m$ as witnessed by \mathbf{M} . For $i \leq m + 1$, let $\mathcal{C}_i = \{f \in \mathcal{C} \mid \mathbf{M} \text{ on } f \text{ makes at least } i \text{ mind changes}\}$. Note that $\mathcal{C}_{m+1} = \emptyset$, and $\mathcal{C}_0 = \mathcal{C}$. Moreover, if \mathcal{C} is recursively enumerable, then each \mathcal{C}_i is recursively enumerable. Let j be the least number such that \mathcal{C}_j is finite. Note that $1 \leq j \leq m + 1$. Let $\mathcal{C}' = \mathcal{C}_{j-1} - \mathcal{C}_j$. We now show the following properties of \mathcal{C}' :

- (a) $\mathcal{C}' \in \mathbf{Ex}_0$: To see this, let \mathbf{M}'_j be an IIM which, on input function f , outputs (only) the j -th conjecture, if any, output by \mathbf{M} on f . It is easy to verify that $\mathbf{M}'_j \in \mathbf{Ex}_0$ -identifies $\mathcal{C}_{j-1} - \mathcal{C}_j$.
- (b) \mathcal{C}' is infinite: \mathcal{C}_{j-1} is infinite, and \mathcal{C}_j is finite, thus \mathcal{C}' is infinite.
- (c) If \mathcal{C} is recursively enumerable, then so is \mathcal{C}' : If \mathcal{C} is recursively enumerable, then each \mathcal{C}_i , $i \leq m + 1$, is recursively enumerable. Now \mathcal{C}' being recursively enumerable follows from \mathcal{C}_j being finite.

The above properties prove the proposition. ■

The following proposition shows that, if a class contains an accumulation point for itself, then it cannot be finitely identified.

Proposition 13 [*Lin72*] *Suppose $\mathcal{C} \in \mathbf{Ex}_0$. Then \mathcal{C} does not contain any accumulation point of \mathcal{C} .*

In particular we will be using the following example. Let $h_0 = \text{Zero}$. For $k \in \mathbb{N}$, let

$$h_{k+1}(x) = \begin{cases} 1, & \text{if } x = k + 1; \\ 0, & \text{otherwise.} \end{cases}$$

Consequently, h_0 is an accumulation point of every infinite subclass of $\{h_k \mid k \geq 1\}$. Suppose $h_0 \in \mathcal{C}$ and $\mathcal{C} \in \mathbf{Ex}_0$. By Proposition 13, \mathcal{C} can contain at most finitely many h_k 's.

Our next proposition allows one to effectively construct a class diagonalizing against any given machine (for \mathbf{Ex}_m -identification). We will use this proposition with the same notation in the proof of Theorem 18 below.

Proposition 14 *Suppose $k, l \in N$ and $\sigma \in SEG$ are given. Then, for $m = 2^{l+2}$, one can effectively (in k, l, σ) enumerate a sequence $F_{k,l}^0, F_{k,l}^1, F_{k,l}^2, \dots, F_{k,l}^{m-1}$ of (not necessarily distinct) functions such that*

- (a) for $i < m$, $\sigma \subseteq F_{k,l}^i$.
- (b) $\{F_{k,l}^0, F_{k,l}^1, \dots, F_{k,l}^{m-1}\} \not\subseteq \mathbf{Ex}_l(\mathbf{M}_k)$.

PROOF. Below we will give the construction of $F_{k,l}^i$. It will be easy to see that the construction is effective in k, l, σ . Initially, let $S_0 = \{i \mid i < 2^{l+2}\}$. For all $i \in S_0$ and $x \in \text{domain}(\sigma)$, let $F_{k,l}^i(x) = \sigma(x)$. Let $x_0 = \max(\text{domain}(\sigma)) + 1$. Go to stage 0.

Stage s

1. If $\text{card}(S_s) = 1$,

Then,

For $i \in S_s$, for $x \geq x_s$, let $F_{k,l}^i(x) = 0$.

Halt.

2. (* For the following $\text{card}(S_s) > 1$. *)

Let S_s^0, S_s^1 be a partition of S_s into two equal size subsets.

Let $x = x_s$.

3. Repeat

- 3.1 Let i_0, i_1 be some members of S_s^0 and S_s^1 , respectively.

- 3.2 If $\mathbf{M}_k(F_{k,l}^{i_0}[x_s]) \neq \mathbf{M}_k(F_{k,l}^{i_1}[x])$,

Then,

Let $S_{s+1} = S_s^0$.

For $i \in S_s^1$, for $y \geq x$, let $F_{k,l}^i(y) = 1$.

Go to stage $s + 1$.

- 3.3 ElseIf $\mathbf{M}_k(F_{k,l}^{i_1}[x_s]) \neq \mathbf{M}_k(F_{k,l}^{i_0}[x])$,

Then,

Let $S_{s+1} = S_s^1$.

For $i \in S_s^0$, for $y \geq x$, let $F_{k,l}^i(y) = 0$.

Go to stage $s + 1$.

- 3.4 Else

For $i \in S_s^0$, let $F_{k,l}^i(x) = 0$.

For $i \in S_s^1$, let $F_{k,l}^i(x) = 1$.

Let $x = x + 1$

Forever

End

Now fix k, l . Let $F_{k,l}^i$ be as defined above. Clearly, clause (a) in the proposition is satisfied. It is easy to verify that each $F_{k,l}^i$ is total. Also, for $s > 0$, for each $f \in S_s$, \mathbf{M}_k on $f[x_s]$ makes at least $s - 1$ mind changes. We now consider two cases.

Case 1: Stage $l + 2$ is entered. In this case, S_{l+2} must have cardinality 1. Let $i \in S_{l+2}$. Now \mathbf{M}_k on $F_{k,l}^i$ makes at least $l + 2 - 1$ mind changes.

Case 2: Stage $s < l + 2$ is entered but never exited. In this case, \mathbf{M}_k outputs the same program (in the limit) on each $F_{k,l}^i$, for $i \in S_s$. However, for $i_0 \in S_s^0$ and $i_1 \in S_s^1$, $F_{k,l}^{i_0} \neq^* F_{k,l}^{i_1}$. Thus \mathbf{M}_k does not \mathbf{Ex}_l -identify at least one of $F_{k,l}^{i_0}$ and $F_{k,l}^{i_1}$.

From the above cases, (b) follows. ■

The next technical result turns out to be useful in proving classes robustly unlearnable with a bounded number of mind changes. Moreover, this proposition is interesting on its own.

Proposition 15 *There exists an infinite r.e. class \mathcal{S} such that, for all m , no infinite subset of \mathcal{S} belongs to \mathbf{Ex}_m .*

PROOF. We will construct a recursive $S : N \times N \rightarrow \text{SEG}$ satisfying the following four properties ((A) to (D)).

(A) For all $i, t \in N$, $S(i, t)$ is an initial segment of $S(i + 1, t)$.

(B) For all $i \in N$, $\lim_{t \rightarrow \infty} S(i, t)$ converges. Let $\sigma_i = \lim_{t \rightarrow \infty} S(i, t)$.

(C) For all $i \in N$, $\sigma_i \subseteq \sigma_{i+1}$.

(D) For all $i \in N$, for all $j < i$, either (D1) or (D2) is satisfied.

(D1) \mathbf{M}_j on σ_{i+1} makes at least i mind changes.

(D2) $(\forall \sigma \supseteq \sigma_{i+1})[\mathbf{M}_j(\sigma_{i+1}) = \mathbf{M}_j(\sigma)]$; in other words, \mathbf{M}_j does not make a mind change on any extension of σ_{i+1} .

Assuming such an S , for each $k \in N$, define H_k as follows:

$$H_k(x) = \begin{cases} S(k, k)(x), & \text{if } x \in \text{domain}(S(k, k)); \\ k, & \text{otherwise.} \end{cases}$$

Note that H_k 's are pairwise different and one can compute $H_k(x)$ effectively from k and x . Let $\mathcal{S} = \{H_k \mid k \in N\}$. We claim that \mathcal{S} satisfies the properties claimed in the proposition. First note that \mathcal{S} is infinite and r.e. Now fix \mathbf{M}_j and m . We claim that \mathbf{M}_j cannot \mathbf{Ex}_m -identify any infinite subset of \mathcal{S} . To see this, let $i = 1 + \max(\{j, m\})$. Now, by (A) and (B) it follows that, for all but finitely many k , $\sigma_{i+1} \subseteq S(k, k)$. Thus, for all but finitely many k , $\sigma_{i+1} \subseteq H_k$. However, by (D), \mathbf{M}_j can \mathbf{Ex}_m -identify at most one extension of σ_{i+1} (since either \mathbf{M}_j makes at least $m + 1$ mind changes on σ_{i+1} or it never changes its mind on any extension of σ_{i+1}). It follows that \mathbf{M}_j can \mathbf{Ex}_m -identify at most finitely many of H_k 's.

It remains to construct S as claimed. We implicitly assume a canonical indexing of all elements of SEG, and often identify elements of SEG with their canonical indices. Thus, when we say $\tau < i$, we mean that the canonical index of τ is less than i . Similarly, when we say, $\min(X)$, where $X \subseteq \text{SEG}$, then we mean the minimum based on the canonical indexing.

Let $\text{Mindchange}(\mathbf{M}, \tau)$ denote the number of mind changes made by \mathbf{M} on τ . For all t , let $S(0, t)$ be the empty sequence. For $i \in N$, define $S(i + 1, t)$ as follows. Suppose $\tau = S(i, t)$. Let $X = \{\tau' \mid \tau \subseteq \tau' \wedge (\forall j < i)[\text{Mindchange}(\mathbf{M}_j, \tau') \geq i \vee (\forall \tau'' \mid \tau' \subseteq \tau'' \wedge \tau'' < t)[\mathbf{M}_j(\tau'') = \mathbf{M}_j(\tau')]]\}$. Note that X is non-empty. Let $S(i + 1, t) = \min(X)$.

It is now easy to verify that the properties (A) to (D) are satisfied. ■

3 Robust Learning with a Bounded Number of Mind Changes

We start with some results pointing out the difficulty of robust learning when only a bounded number of mind changes is allowed. The following theorem shows that no infinite class can be robustly finitely identified. Thus, robust finite identification is very weak. Note that the analogous result has been proved in [Zeu86] for effective operators instead of general recursive operators, as it will be done in Theorem 16 below. We use essentially the same proof idea as in [Zeu86] and include the proof for completeness.

Theorem 16 *For any $\mathcal{C} \subseteq \mathcal{R}$, $\mathcal{C} \in (\mathbf{Ex}_0, \mathbf{Ex}_0)$ -robust iff \mathcal{C} is finite.*

PROOF. Suppose by way of contradiction that \mathcal{C} is infinite and belongs to $(\mathbf{Ex}_0, \mathbf{Ex}_0)$ -robust. Suppose \mathbf{M} \mathbf{Ex}_0 -identifies \mathcal{C} . Let $X = \{f[n+1] \mid \mathbf{M}(f[n]) = ? \wedge \mathbf{M}(f[n+1]) \neq ?\}$ (i.e. X denotes the initial segments on which \mathbf{M} outputs its conjecture for the first time). Let g be a fixed function in \mathcal{C} . Let $\sigma \in X$ be such that $\sigma \subseteq g$. Let $\sigma_0, \sigma_1, \dots$, be a 1-1 recursive enumeration of X such that $\sigma_0 = \sigma$. Note that no total function can have two different σ_i 's as its initial segment.

Define h_i as follows. $h_0 = \text{Zero}$.

$$h_{k+1}(x) = \begin{cases} 1, & \text{if } x = k + 1; \\ 0, & \text{otherwise.} \end{cases}$$

Now define Θ as follows:

$\Theta(\eta)$

1. For k such that $\sigma_k \subseteq \eta$, let $h_k \subseteq \Theta(\eta)$.
2. For k such that,
 - for all $k' < k$, $\text{domain}(\sigma_{k'}) \subseteq \text{domain}(\eta)$ and $\sigma_{k'} \not\subseteq \eta$,
 - let $h_0[k] \subseteq \Theta(\eta)$.

End

We first show that Θ is general recursive. Fix any $\eta \in \mathcal{P}$. By definition of σ_i , $i \in N$, there can be at most one i such that $\sigma_i \subseteq \eta$. If for all i , $\sigma_i \not\subseteq \eta$, then $\Theta(\eta) \subseteq h_0$, and is thus a partial function. On the other hand, if i is such that $\sigma_i \subseteq \eta$, then $\Theta(\eta) = h_i$ (note that $h_0[i] \subseteq h_i$, and if step 2 makes $h_0[k] \subseteq \Theta(\eta)$, then $k \leq i$). It follows that Θ maps partial functions to partial functions. Also Θ satisfies (a) monotonicity and (b) compactness, since Θ is based only on whether $\sigma_i \subseteq \eta$, for various values of i . Θ satisfies (c) recursiveness since σ_i , $i \in N$, is a recursive enumeration. Now suppose η is a total function. If there exists a k such that $\sigma_k \subseteq \eta$, then clearly $\Theta(\eta)$ is total due to step 1. On the other hand, if for all k , $\sigma_k \not\subseteq \eta$, then by step 2, $h_0[k] \subseteq \Theta(\eta)$, for all k . Thus, $\Theta(\eta) = h_0$. It follows that Θ is general recursive.

We now show that $\Theta(\mathcal{C}) \notin \mathbf{Ex}_0$. Note that $h_0 \in \Theta(\mathcal{C})$, since $\sigma_0 \subseteq g$, and thus $\Theta(g) = h_0$. Moreover, each $f \in \mathcal{C}$ is mapped to a different h_k , since each $f \in \mathcal{C}$ extends one and only one σ_k . Thus, $\Theta(\mathcal{C})$ contains infinitely many h_k . It follows from Proposition 13 that $\Theta(\mathcal{C}) \notin \mathbf{Ex}_0$. ■

Corollary 17 *For any \mathbf{I} , no infinite class belongs to $(\mathbf{I}, \mathbf{Ex}_0)$ -robust.*

PROOF. By Proposition 9, $(\mathbf{I}, \mathbf{Ex}_0)$ -robust $\subseteq (\mathbf{Ex}_0, \mathbf{Ex}_0)$ -robust. The corollary follows from Theorem 16. \blacksquare

The next theorem shows that no infinite recursively enumerable class can be robustly identified with a bounded number of mind changes.

Theorem 18 *Suppose \mathcal{C} is an infinite r.e. class. Then, for any \mathbf{I} , $\mathcal{C} \notin (\mathbf{I}, \bigcup_{m \in N} \mathbf{Ex}_m)$ -robust.*

PROOF. If \mathcal{C} is an infinite r.e. class and $\mathcal{C} \in (\mathbf{I}, \bigcup_{m \in N} \mathbf{Ex}_m)$ -robust, then $\mathcal{C} \in \bigcup_{m \in N} \mathbf{Ex}_m$. Thus, $\mathcal{C} \in \mathbf{Ex}_n$ for some $n \in N$. Now, by Proposition 12, \mathcal{C} contains an infinite r.e. subclass in \mathbf{Ex}_0 . Hence, without loss of generality, it suffices to show that no infinite r.e. class belongs to $(\mathbf{Ex}_0, \bigcup_{m \in N} \mathbf{Ex}_m)$ -robust.

Suppose \mathcal{C} is an infinite r.e. class in $(\mathbf{Ex}_0, \bigcup_{m \in N} \mathbf{Ex}_m)$ -robust. Let h_0, h_1, \dots denote a 1-1, recursive enumeration of \mathcal{C} . Let \mathbf{M} be such that $\mathcal{C} \subseteq \mathbf{Ex}_0(\mathbf{M})$. Let $\sigma_i \subseteq h_i$ be such that $\mathbf{M}(\sigma_i) \neq ?$ (thus $\mathbf{M}(\sigma_i)$ must be a program for h_i).

For fixed k, l , let $F_{k,l}^i$, $0 \leq i < 2^{l+2}$, denote a sequence of 2^{l+2} total functions, such that (a) for $x \leq \langle k, l \rangle$, $F_{k,l}^i(x) = 0$; and (b) \mathbf{M}_k fails to \mathbf{Ex}_l -identify $\{F_{k,l}^i \mid i < 2^{l+2}\}$. Note that such a sequence of functions can be effectively constructed by Proposition 14. Let $n_0 < n_1 < \dots$ be a sequence of increasing numbers such that, $n_{\langle k,l \rangle+1} - n_{\langle k,l \rangle} = 2^{l+2}$. Now define Θ as follows.

$\Theta(\eta)$

1. If there exist $k, l \in N$ and $i < 2^{l+2}$ such that $\sigma_{n_{\langle k,l \rangle+1}+i} \subseteq \eta$, then $\Theta(\eta) = F_{k,l}^i$.
2. If $\sigma_0 \subseteq \eta$, then $\Theta(\eta) = \text{Zero}$.
3. If for all $k' \leq n_k$, $\text{domain}(\sigma_{k'}) \subseteq \text{domain}(\eta)$ and $\sigma_{k'} \not\subseteq \eta$, then $\text{Zero}[k] \subseteq \Theta(\eta)$.

End

Note that step 3 above is consistent with steps 1 and 2, since $\text{Zero}[\langle k, l \rangle] \subseteq F_{k,l}^i$. Clearly, Θ is general recursive. Hence, $\Theta(\mathcal{C}) \supseteq \{F_{k,l}^i \mid k, l \in N \wedge i < 2^{l+2}\}$. Thus, for any k and l , \mathbf{M}_k does not \mathbf{Ex}_l -identify $\Theta(\mathcal{C})$. It follows that $\Theta(\mathcal{C}) \notin \bigcup_{m \in N} \mathbf{Ex}_m$. \blacksquare

The following corollaries can be derived from Theorem 18.

Corollary 19 *No infinite r.e. class is in $\bigcup_{m \in N} (\mathbf{Ex}_m, \mathbf{Ex}_m)$ -robust.*

Corollary 20 $\text{NUM} - \bigcup_{m \in N} (\mathbf{Ex}_m, \mathbf{Ex}_m)$ -robust $\neq \emptyset$.

Contrasting Corollary 20 with the fact that **NUM** is contained in $(\mathbf{Ex}, \mathbf{Ex})$ -robust, Proposition 10(b), we have

Corollary 21 $(\mathbf{Ex}, \mathbf{Ex})$ -robust $- \bigcup_{m \in N} (\mathbf{Ex}_m, \mathbf{Ex}_m)$ -robust $\neq \emptyset$.

Note that $\text{CONST} \notin \bigcup_{m \in N} (\mathbf{Ex}_m, \mathbf{Ex}_m)$ -robust follows directly from Corollary 19. The following gives both an easy direct proof and an idea for why **CONST** behaves so “negatively” with respect to robust learning with a bounded number of mind changes. Recall from Section 2 that **FINSUP** denotes the set of functions of finite support. Suppose F_i denotes (in some recursive enumeration) the i -th function in **FINSUP**. Define Θ as follows: $\Theta(f) = F_{f(0)}$. Clearly, $\Theta(\text{CONST}) = \text{FINSUP}$.

Thus $\Theta(\text{CONST}) \notin \bigcup_{m \in N} \mathbf{Ex}_m$, by Theorem 4(g). The following theorem is a generalization of the above idea to show that not only CONST , but even none of its infinite subclasses belongs to $\bigcup_{m \in N} (\mathbf{Ex}_m, \mathbf{Ex}_m)$ -robust.

Theorem 22 *Suppose \mathcal{C} is an infinite subset of CONST . Then, for any \mathbf{I} , $\mathcal{C} \notin (\mathbf{I}, \bigcup_{m \in N} \mathbf{Ex}_m)$ -robust.*

PROOF. Let \mathcal{S} be as in Proposition 15. Let H_0, H_1, \dots be a 1–1 enumeration of \mathcal{S} .

Define Θ as follows:

$$\Theta(\eta) = \begin{cases} H_i, & \text{if } \eta(0) = i; \\ \Lambda, & \text{if } \eta(0) \text{ is undefined.} \end{cases}$$

Clearly, Θ is general recursive. Since H_i 's are pairwise distinct, it follows that $\Theta(\mathcal{C})$ is infinite. It follows by Proposition 15 that, for all m , $\Theta(\mathcal{C}) \notin \mathbf{Ex}_m$. The theorem follows. \blacksquare

In the remainder of this section we derive several positive results on robust learning with a bounded number of mind changes. We start with our main result. This result shows that the mind change hierarchy, $\mathbf{Ex}_0 \subset \mathbf{Ex}_1 \subset \mathbf{Ex}_2 \dots$, stands robustly, that is, for all $n \in N$, $(\mathbf{Ex}_n, \mathbf{Ex}_n)$ -robust is properly contained in $(\mathbf{Ex}_{n+1}, \mathbf{Ex}_{n+1})$ -robust. Note that Fulk [Ful90] showed that, for all $a \in N \cup \{*\}$, $(\mathbf{Ex}^a, \mathbf{Ex}^a)$ -robust = $(\mathbf{Ex}, \mathbf{Ex})$ -robust, and, for all $a \in N$, $(\mathbf{Bc}^a, \mathbf{Bc}^a)$ -robust = $(\mathbf{Bc}, \mathbf{Bc})$ -robust. Thus, several hierarchies do not stand for robust identification. The $(\mathbf{Ex}_n, \mathbf{Ex}_n)$ -robust hierarchy result interestingly contrasts with the above collapses. Furthermore, Theorem 16 showed that $(\mathbf{Ex}_0, \mathbf{Ex}_0)$ -robust contains only finite classes. Thus, the fact that even $(\mathbf{Ex}_0, \mathbf{Ex}_1)$ -robust contains infinite classes is interesting on its own, see Corollary 33 and Theorem 35 below.

Definition 23 Suppose $m \in N$, and $0 = x_0 < x_1 < x_2 < \dots < x_m < \infty$. Then $\text{Step}(x_1, x_2, \dots, x_m)$ denotes the function f defined as: $f(x) = i$, if $x_i \leq x < x_{i+1}$, $i < m$; $f(x) = m$, if $x_m \leq x$;

$\text{Step}()$ denotes Zero.

For $n \in N$, let $\mathcal{STEP}_n = \{\text{Step}(x_1, x_2, \dots, x_m) \mid m \leq n, \text{ and } 0 < x_1 < x_2 < \dots < x_m\}$. We allow $m = 0$ in this definition, that is \mathcal{STEP}_n includes $\text{Step}() = \text{Zero}$.

Intuitively, $\text{Step}(x_1, x_2, \dots, x_m)$ is an m -step function with steps at x_1, x_2, \dots, x_m . We only consider steps of size 1, without explicitly mentioning it. Let Base be a function such that $\text{Base}(\text{Step}(x_1, \dots, x_m)) = x_m$. By convention, $\text{Base}(\text{Step}()) = 0$.

Theorem 24 *For every $n \geq 1$, there exists a class $\mathcal{C}_n \in (\mathbf{Ex}_n, \mathbf{Ex}_n)$ -robust such that $\mathcal{C}_n \notin \mathbf{Ex}_{n-1}$.*

PROOF. We use a priority construction to determine the functions in \mathcal{C}_n . \mathcal{C}_n will be a subset of \mathcal{STEP}_n that contains Zero.

Definition 25 A *label* is a finite sequence of the form (a_1, \dots, a_m) , where $m \leq n$, and $a_i \in N$. As usual $()$ is also a label.

Think of the labels as nodes of a tree, where $()$ is the root, and $(a_1, a_2, \dots, a_m, a_{m+1})$ is the a_{m+1} -th child of (a_1, a_2, \dots, a_m) (we start with 0-th child). The tree above is a depth n tree, where each node (except the leaves which are at depth n) has infinitely many children. Often below we will refer to parent, ancestors, etc. of a label. We mean the parent, ancestors, etc. in this tree.

Below we will define, for each label (a_1, \dots, a_m) , a function $F_{(a_1, \dots, a_m)}$, where $m \leq n$, and $a_i \in N$. $F_{(\dots)}$ will satisfy the following properties (in addition to some other properties considered later on).

- (A) $F_{()} = \text{Step}() = \text{Zero}$. Each $F_{(a_1, a_2, \dots, a_m)}$ will be of the form $\text{Step}(x_1, x_2, \dots, x_m)$ for some x_1, x_2, \dots, x_m .
- (B) Suppose $F_{(a_1, a_2, \dots, a_m)}$ is $\text{Step}(x_1, x_2, \dots, x_m)$, and $m < n$. Then $F_{(a_1, a_2, \dots, a_m, a_{m+1})}$ is $\text{Step}(x_1, x_2, \dots, x_m, x_{m+1})$, for some $x_{m+1} > x_m$. Moreover, if $F_{(a_1, a_2, \dots, a_m, a_{m+1})}$ is $\text{Step}(x_1, x_2, \dots, x_m, x_{m+1})$ and $F_{(a_1, a_2, \dots, a_m, a'_{m+1})}$ is $\text{Step}(x_1, x_2, \dots, x_m, x'_{m+1})$, then, $[a_{m+1} < a'_{m+1} \text{ iff } x_{m+1} < x'_{m+1}]$.

(A) and (B) are important properties to be maintained throughout the construction. We take \mathcal{C}_n to be the collection of all $F_{(a_1, \dots, a_m)}$, where (a_1, \dots, a_m) , $m \leq n$, is a label.

Claim 26 \mathcal{C}_n is in \mathbf{Ex}_n , but not in \mathbf{Ex}_{n-1} .

PROOF. Since $\mathcal{STEP}_n \in \mathbf{Ex}_n$, it follows that $\mathcal{C}_n \in \mathbf{Ex}_n$. Suppose by way of contradiction that \mathcal{C}_n is in \mathbf{Ex}_{n-1} as witnessed by \mathbf{M} . We will define a sequence of labels, $L_0 = ()$, $L_1 = (a_1)$, $L_2 = (a_1, a_2)$, \dots , $L_n = (a_1, a_2, \dots, a_n)$ as follows. Label L_0 is $()$. Let y_0 be such that $\mathbf{M}(F_{L_0}[y_0])$ is a program for F_{L_0} (there must exist such a y_0 , since otherwise \mathbf{M} does not \mathbf{Ex}_{n-1} -identify F_{L_0}). Suppose we have already defined L_0, L_1, \dots, L_i and y_0, y_1, \dots, y_i , where $i < n$. Suppose $L_i = (a_1, a_2, \dots, a_i)$. We now define L_{i+1} as follows. Pick a_{i+1} such that $\text{Base}(F_{(a_1, \dots, a_i, a_{i+1})}) > y_i$ (there exists such an a_{i+1} due to properties (A) and (B) above). Let $L_{i+1} = (a_1, \dots, a_i, a_{i+1})$. Choose $y_{i+1} > y_i$, such that $\mathbf{M}(F_{L_{i+1}}[y_{i+1}])$ is a program for $F_{L_{i+1}}$ (note that there exists such a y_{i+1} since otherwise \mathbf{M} does not \mathbf{Ex}_{n-1} -identify $F_{L_{i+1}}$). Continuing in this way we can define the labels L_0, \dots, L_n .

Label $L_n = (a_1, \dots, a_n)$ is such that $F_{L_n} \in \mathcal{C}_n$, but \mathbf{M} on F_{L_n} outputs at least $n + 1$ different programs (one for each of F_{L_0}, \dots, F_{L_n}). Thus \mathbf{M} does not \mathbf{Ex}_{n-1} -identify \mathcal{C}_n . \square

The construction below defines $F_{(\dots)}$'s using a priority construction. For this purpose, to each label, we will assign a function in \mathcal{STEP}_n . The functions associated with a label may change over time (higher priority labels may spoil lower priority labels). However, the functions associated with any particular label will stabilize in the limit. We take $F_{(a_1, \dots, a_m)}$ as the function associated (in the limit) with label (a_1, \dots, a_m) . We will use $f_{(a_1, \dots, a_m)}$ to denote the function currently associated with label (a_1, \dots, a_m) .

We assign priority as follows. Let pr denote a computable bijective function from the labels to N . (Thus pr^{-1} is a computable bijective function from N to labels). We take $pr(a_1, \dots, a_m)$ as the priority of label (a_1, \dots, a_m) . Lower pr value means higher priority. We assume pr is nice, in the sense that, for all a_1, a_2, \dots, a_{m+1} : $pr(a_1, \dots, a_m) < pr(a_1, \dots, a_m, a_{m+1})$ and $pr(a_1, \dots, a_m) < pr(a_1, \dots, a_m + 1)$.

For ease of notation, we usually identify a label with its image under pr . Thus, if $pr(a_1, a_2, \dots, a_m) = i$, then we will often say label i to mean label (a_1, \dots, a_m) . Also, we will often use f_i (F_i) to mean $f_{(a_1, \dots, a_m)}$ ($F_{(a_1, \dots, a_m)}$), where $pr(a_1, a_2, \dots, a_m) = i$. We may also talk of i as being parent of j , where in the tree mentioned above $pr^{-1}(i)$ is the parent of $pr^{-1}(j)$. Intuitively, the aim of the construction is to define the $F_{(\dots)}$ in such a way so that Claims 29 and 30 below are satisfied. Properties ensured by these claims then allows us to \mathbf{Ex}_n -identify $\Theta_r(\mathcal{C}_n)$, for each general recursive Θ_r . This is described in more detail later.

In the following we will sometimes place stars on the labels. There are infinitely many different kinds of stars, one for each Θ_i . We denote the star used for Θ_i by $*_i$. Intuitively, when we place $*_i$ on label k at stage s , we mean that $\Theta_i(f_k[s])$ is inconsistent with $\Theta_i(f_j[s])$, for all $j < k$. Initially, we assign $\text{Step}()$ to label $()$. All other labels are assigned functions in some manner consistent with (A) and (B) above. Initially, for all i , we place $*_i$ on label $()$.

Stage s .

1. If there exists an $i < s$ such that
 - (a) For some $k \leq i$,
 - there is no $*_i$ placed on label k and
 - for each $j < k$, $\Theta_i(f_k[s])$ is inconsistent with $\Theta_i(f_j[s])$
 - OR
 - (b) For some $k < i$,
 - there is no $*_k$ placed on label i , and
 - for all $j < i$, $\Theta_k(f_i[s])$ is inconsistent with $\Theta_k(f_j[s])$

Then go to step 2. Otherwise go to stage $s + 1$.

2. Pick the least i , satisfying 1(a) or 1(b) above.
 - 2a. If i satisfies 1(a): place $*_i$ on label k and Go to step 3.
 - 2b. If i satisfies 1(b): place $*_k$ on label i and Go to step 3.

(In case of many k 's satisfying 1(a)/1(b), choose an arbitrary one. Choosing the least i is important, but the corresponding k can be any of the successful ones).

Note that in case of 1(b) being successful, $s > \text{Base}(f_i)$ must hold (otherwise, there cannot be inconsistency with $f_{\text{parent}(i)}$).
3. (Labels greater than i get spoiled). A new f_j is assigned to labels $j > i$ in such a way that
 - (1) For $j > i$, $\text{Base}(f_j) > s$.
 - (2) New assignments satisfy properties (A) and (B) above.
 - (3) All $*$'s on labels $j > i$ are removed.

NOTE: For i as above, let $S_s = \{f_r[s] \mid r \leq i\}$. Now (1) and (2) above imply that each f_j (the new function assigned to label j) is an extension of some element of S_s . Thus all functions currently assigned to labels are extensions of some element of S_s . This property will be used later in the proof.

4. Go to stage $s + 1$.

End Stage s

Note that in the construction above, in step 1, (b) is the important condition; (a) is used merely to get around the “initial functions” problem.

Claim 27 *For each r , the function assigned to label r eventually stabilizes.*

PROOF. By induction on r . Suppose all labels less than r eventually get stable functions but label r does not. Suppose all labels less than r get stable functions before stage s . Now we consider how many times label r may be spoiled beyond stage s . Since, all labels less than r get stable functions by stage s , beyond stage s , label r can be spoiled only if $i = r - 1$ in step 1.

Due to each of (a) or (b) in step 1, label r may be spoiled at most r times, since there are only r possible values for $k \leq r - 1$. Thus, r can get spoiled only finitely often beyond stage s . Thus, the function assigned to label r stabilizes. \square

We let F_0, F_1, \dots denote the functions eventually assigned to the labels.

Claim 28 *Suppose at some stage s , step 1 succeeds in finding an i satisfying (a) or (b). Let i be the least one as chosen in step 2. Let $S_s = \{f_r[s] \mid r \leq i\}$, where f_r is as at step 2 of stage s (note that f_r , $r \leq i$, is not changed in stage s). Then,*

- (1) All f_r as at the end of step 3 of stage s , are extensions of some element of S_s .
- (2) All functions ever assigned to some label beyond stage s , are extensions of some element of S_s .
- (3) All functions in \mathcal{C}_n are extensions of some element of S_s .

PROOF. (1) is straight from construction (see comment at the end of step 3 of the construction). (2) follows by induction on the stages greater than or equal to s , and (3) follows from (2). \square

Claim 29 *Suppose at some stage s , step 2b is executed with $i = \ell$ and $k = r$. Let f_ℓ be as at the end of stage s . Then,*

- (1) *For all w such that $\Theta_r(f_\ell[s]) \subseteq \Theta_r(F_w)$, we must have $f_\ell[s] \subseteq F_w$.*
- (2) *Either ℓ never gets spoiled beyond stage s , OR, for all w , $\Theta_r(f_\ell[s]) \not\subseteq \Theta_r(F_w)$.*

PROOF. (1) Note that by Claim 28, F_w must be an extension of some element of S_s . However, due to success of step 1(b) in stage s , with $i = \ell$ and $k = r$, if F_w is not an extension of $f_\ell[s]$, then $\Theta_r(F_w)$ will be inconsistent with $\Theta_r(f_\ell[s])$. A contradiction. Thus F_w must be an extension of $f_\ell[s]$.

(2) If ℓ gets spoiled in some stage greater than s , then let s' be first such stage. Let ℓ' be the value of i as chosen in stage s' . Let f_u be the value of functions assigned to label u as at the end of stage s . Note that functions assigned to labels less than or equal to ℓ' do not change between stage s and s' . Thus, by Claim 28, all functions in \mathcal{C}_n must be an extension of some element in $S_{s'} = \{f_u[s'] \mid u \leq \ell'\}$. Thus, every function in \mathcal{C}_n is inconsistent with $f_\ell[s]$ (due to success of step 1(b) in stage s , with $i = \ell$ and $k = r$). Thus, no element of \mathcal{C}_n extends $f_\ell[s]$. (2) now follows from (1). \square

Claim 30 *Suppose, $w > r$, and $\Theta_r(F_w)$ is inconsistent with $\Theta_r(F_u)$, for each $u < w$. Then there exists a stage s such that, (1) w never gets spoiled at or beyond stage s (thus, for $u \leq w$, f_u (at stage s) is the same as F_u) and (2) at stage s , step 2b is executed with $i = w$ and $k = r$.*

PROOF. Suppose stage s' is the least stage at the end of which all functions assigned to labels less than or equal to w are stabilized (i.e. stage s' is the last stage in which step 1 succeeds with an $i < w$). Now, by hypothesis, for all but finitely many $s'' > s'$, we have: for all $u < w$, $\Theta_r(F_w[s''])$ is inconsistent with $\Theta_r(F_u[s''])$. Thus there must be a stage $s > s'$, in which step 1 (b) succeeds with $i = w$ and $k = r$. (Note: Step 2b may choose $i = w$ and some other $k \neq r$; however this can happen only finitely often, and eventually $k = r$ must be taken by step 2b). \square

Now we are in a position to show the robustness of \mathcal{C}_n . For this, fix r such that Θ_r is general recursive. Claims 29 and 30 allow us an easy way to identify $\Theta_r(\mathcal{C}_n)$. For the time being only consider $\mathcal{C}'_n = \mathcal{C}_n - \{F_j \mid (\exists i \leq r)[\Theta_r(F_i) = \Theta_r(F_j)]\}$. Note that $\Theta_r(\mathcal{C}'_n) = \Theta_r(\mathcal{C}_n) - \Theta_r(\{F_i \mid i \leq r\})$.

Suppose $g \in \mathcal{C}'_n$ and $\Theta_r(g)$ is the input function. Initially \mathbf{M} outputs ?, and “thinks” the input function to be $\Theta_r(F_0)$. (Note that \mathbf{M} does not actually output a program for $\Theta_r(F_0)$.) Let $L_0 = ()$, currfunc = F_0 and $s_0 = 0$. Then \mathbf{M} executes the following loop (starting with iteration 1).

Loop iteration p

1. Search for a stage $s_p > s_{p-1}$, a descendant L_p of L_{p-1} such that
 - 1.1 $\Theta_r(\text{currfunc}[s_p])$ is inconsistent with $\Theta_r(g)$ (note that $\Theta_r(g)$ is the input function, and we are not calculating it).

- 1.2 In the construction of \mathcal{C}_n above at stage s_p : (a) Step 2b is executed with $i = pr(L_p)$ and $k = r$, where $i > k$ and (b) $\Theta_r(f_{L_p}[s_p]) \subseteq \Theta_r(g)$, where f_{L_p} is the function assigned to label L_p at stage s_p .

Then,

fix one such s_p and corresponding L_p . Output a program for $\Theta_r(f_{L_p})$, and set $\text{currfunc} = f_{L_p}$.

End

Suppose $g \in \mathcal{C}'_n$. Suppose the sequence of stages and labels considered by \mathbf{M} in the above procedure are s_1, s_2, \dots , and L_1, L_2, \dots , respectively. For $u = 1, 2, \dots$, let f_{L_u} be the corresponding function assigned to label L_u at stage s_u . Then, by Claim 29 part (1), we must have $f_{L_u}[s_u] \subseteq g$, for $u = 1, 2, \dots$, and by Claim 29 part (2), we must additionally have $f_{L_u} = F_{L_u}$. Since the depth of the label tree is finite, \mathbf{M} 's final output stabilizes. This must be a program for $\Theta_r(g)$, since otherwise, by Claim 30, the search by machine \mathbf{M} would have succeeded. (Note that the limiting value of currfunc may not be equal to g . However, we will have: $\Theta_r(\text{limiting value of currfunc}) = \Theta_r(g)$.) The number of outputs of \mathbf{M} is bounded by n (i.e. $n - 1$ mind changes), since the depth of the tree is at most n (Note: \mathbf{M} did NOT output a program for $\Theta_r(F_0)$ in the beginning). Thus, \mathbf{M} \mathbf{Ex}_{n-1} -identifies $\Theta_r(\mathcal{C}'_n)$.

Finally, since $\Theta_r(\mathcal{C}_n) - \Theta_r(\mathcal{C}'_n)$ is finite, it follows that $\Theta_r(\mathcal{C}_n) \in \mathbf{Ex}_n$ (using Proposition 11). This proves the theorem. \blacksquare

Corollary 31 *For all $n \in \mathbb{N}$, $(\mathbf{Ex}_n, \mathbf{Ex}_n)$ -robust $\subset (\mathbf{Ex}_{n+1}, \mathbf{Ex}_{n+1})$ -robust.*

PROOF. Immediately from Theorem 24. \blacksquare

Corollary 32 *For every $n \geq 1$, $(\mathbf{Ex}_{n-1}, \mathbf{Ex}_n)$ -robust $- (\mathbf{Ex}_{n-1}, \mathbf{Ex}_{n-1})$ -robust $\neq \emptyset$.*

PROOF. Let $\mathcal{C}_n, F_{(\dots)}$ be as in the proof of Theorem 24. Let $\mathcal{C}'_n = \mathcal{C}_n - \{\text{Zero}\}$. Note that $\mathcal{C}'_n \in (\mathbf{Ex}_{n-1}, \mathbf{Ex}_n)$ -robust. We claim that $\mathcal{C}'_n \notin (\mathbf{Ex}_{n-1}, \mathbf{Ex}_{n-1})$ -robust. Let $g = F_{(0,0,\dots,0)}$ (there are n 0's in the subscript). Let $b = \text{Base}(g)$. Let Θ be defined as follows:

$$\Theta(\eta) = \begin{cases} \text{Zero}, & \text{if } g[b+1] \subseteq \eta; \\ \eta, & \text{if } g[b+1] \not\subseteq \eta \text{ and} \\ & \{x \mid x \leq b\} \subseteq \text{domain}(\eta); \\ \Lambda, & \text{if } \{x \mid x \leq b\} \not\subseteq \text{domain}(\eta). \end{cases}$$

Note that such a Θ can be easily constructed. Moreover, Θ is general recursive. Further note that $\Theta(\mathcal{C}'_n) = \mathcal{C}_n - \{g\}$ (since $\Theta(g) = \text{Zero}$, and, for all $f \in \mathcal{C}'_n - \{g\}$, $\Theta(f) = f$). However, it is easy to show (using a proof similar to that of Claim 26, by taking $y_0 > b$) that $\mathcal{C}_n - \{g\} \notin \mathbf{Ex}_{n-1}$. The corollary follows. \blacksquare

Hence, using the $n = 1$ case of Corollary 32, we have

Corollary 33 *Let $\mathcal{S} = \mathcal{STEP}_1 - \{\text{Zero}\}$. Then there exists an infinite subclass of \mathcal{S} which belongs to $(\mathbf{Ex}_0, \mathbf{Ex}_1)$ -robust.*

Consequently, though $(\mathbf{Ex}_0, \mathbf{Ex}_0)$ -robust classes are trivial (Theorem 16), already $(\mathbf{Ex}_0, \mathbf{Ex}_1)$ -robust learning is much richer. Moreover, contrasting Corollary 33 with Theorem 22 we see that subsets of CONST are “much harder” to robustly learn than subsets of STEP_1 which is, on the surface, counter-intuitive. We now consider some of the reasons for this difference. Let $\text{Equalupto}(f, g)$ denote the least x such that $f(x) \neq g(x)$. The proof of Theorem 24 can easily be modified to work when we replace the functions $\text{Step}(x_1, x_2, \dots, x_m)$ by recursive functions $G(x_1, x_2, \dots, x_m)$, where

- (i) a program for $G(x_1, x_2, \dots, x_m)$ can be effectively obtained from x_1, x_2, \dots, x_m , and
- (ii) for all $x_1, x_2, \dots, x_m, y_1, y_2$, such that $0 < x_1 < x_2 < \dots < x_m < y_1 < y_2$,

$$\begin{aligned} & \text{Equalupto}(G(x_1, x_2, \dots, x_{m-1}), G(x_1, x_2, \dots, x_m)) \\ & < \text{Equalupto}(G(x_1, x_2, \dots, x_{m-1}, x_m), G(x_1, x_2, \dots, x_m, y_1)) \\ & < \text{Equalupto}(G(x_1, x_2, \dots, x_{m-1}, x_m), G(x_1, x_2, \dots, x_m, y_2)). \end{aligned}$$

Thus, we have the following.

Corollary 34 *Let $\mathcal{C} \subseteq \mathcal{R}$ be any recursively enumerable class for which there exists an accumulation point. Then, there exists an infinite subclass of \mathcal{C} belonging to $(\mathbf{Ex}_0, \mathbf{Ex}_1)$ -robust.*

On the other hand, CONST , the class from Theorem 22, clearly does not possess an accumulation point. Moreover, any two constant functions differ “from the very beginning,” i.e. at argument 0, from each other. Thus, we have the surprising fact that the more complex topological structure leads to a positive robustness result, whereas the “trivial” structure yields a negative result. Note that an analogous phenomenon can also be observed for classes not contained in any recursively enumerable class (see Theorem 36 below and the discussion thereafter).

Notice that Theorem 22 remains valid if we replace CONST by any recursively enumerable class \mathcal{S} for which there exists a computable functional F , mapping every function $f \in \mathcal{R}$ to some $F(f) \in N$, such that $\{F(f) \mid f \in \mathcal{S}\}$ is infinite, and for every $f \in \mathcal{S}$, $\{g \in \mathcal{S} \mid F(f) = F(g)\}$ is finite. This holds, since one can construct $\Theta(f) = c_{F(f)}$, where c_i is the constant i function. Now, for any infinite subset \mathcal{S}' of \mathcal{S} , $\Theta(\mathcal{S}')$ is an infinite subset of CONST . Our claim then follows from Theorem 22 (since composition of general recursive operators gives a general recursive operator).

Though the above cases do not exhaust all the recursively enumerable classes, they give us an idea about the kind of properties one may look for, to determine whether a recursively enumerable class has an infinite $(\mathbf{Ex}_m, \mathbf{Ex}_n)$ -robust subclass. It is an open problem to characterize which recursively enumerable classes have infinite $(\mathbf{Ex}_m, \mathbf{Ex}_n)$ -robust subclasses.

The following theorem generalizes Corollary 33.

Theorem 35 *Suppose $m, n \in N$. For every infinite \mathcal{C} in $(\mathbf{Ex}_m, \mathbf{Ex}_n)$ -robust, there exists an infinite subset of \mathcal{C} in $(\mathbf{Ex}_0, \mathbf{Ex}_1)$ -robust.*

PROOF. Since $(\mathbf{Ex}_m, \mathbf{Ex}_n)$ -robust $\subseteq (\mathbf{Ex}_{\max\{m, n\}}, \mathbf{Ex}_{\max\{m, n\}})$ -robust, without loss of generality we may assume $m = n$ (otherwise one may just take each of m and n to be maximum of old values of m and n). Suppose \mathcal{C} is as given.

Below we will formally define $\mathcal{C}_i, \mathcal{S}_i$ satisfying the following five properties for each $i \in N$ (we give these properties *before* the formal definition of $\mathcal{C}_i, \mathcal{S}_i$ in order to provide intuition on what we are going to achieve with these definitions):

- (1) \mathcal{S}_i contains exactly i elements;
- (2) $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$ and $\mathcal{C}_{i+1} \subseteq \mathcal{C}_i$;
- (3) $\mathcal{C}_{i+1} \cup \mathcal{S}_{i+1} \subseteq \mathcal{C}_i \cup \mathcal{S}_i \subseteq \mathcal{C}$;
- (4) \mathcal{C}_i is infinite;
- (5) If Θ_i is general recursive, then $\Theta_i(\mathcal{C}_{i+1}) \in \mathbf{Ex}_0$.

It follows that:

- (A) $(\bigcup_{j \in \mathbb{N}} \mathcal{S}_j)$ is an infinite subset of \mathcal{C} (by (1) and (3));
- (B) $(\bigcup_{j \in \mathbb{N}} \mathcal{S}_j) \subseteq \mathcal{C}_i \cup \mathcal{S}_i$, for each i (by (2) and (3));
- (C) if Θ_i is general recursive then, $\Theta_i(\mathcal{C}_{i+1} \cup \mathcal{S}_{i+1}) \in \mathbf{Ex}_1$ (by (5), $\Theta_i(\mathcal{S}_{i+1})$ being finite, and Proposition 11);
- (D) $(\bigcup_{j \in \mathbb{N}} \mathcal{S}_j)$ is in $(\mathbf{Ex}_m, \mathbf{Ex}_1)$ -robust, (by (A), (B) and (C));
- (E) there exists an infinite subset of $\bigcup_{j \in \mathbb{N}} \mathcal{S}_j$ in $(\mathbf{Ex}_0, \mathbf{Ex}_1)$ -robust (by (D), and using Proposition 12).

We now inductively define \mathcal{C}_i and \mathcal{S}_i satisfying (1) – (5) as follows. Notice that this definition is not effective. Let $\mathcal{C}_0 = \mathcal{C}$ and $\mathcal{S}_0 = \emptyset$. Suppose \mathcal{C}_i and \mathcal{S}_i have been defined. Let \mathcal{C}_{i+1} and \mathcal{S}_{i+1} be defined as follows:

Case 1: Θ_i is not general recursive.

Let h be an element of \mathcal{C}_i . Then let $\mathcal{C}_{i+1} = \mathcal{C}_i - \{h\}$ and $\mathcal{S}_{i+1} = \mathcal{S}_i \cup \{h\}$.

Case 2: Θ_i is general recursive.

Case 2a: $\Theta_i(\mathcal{C}_i)$ is a finite class.

Define \mathcal{C}_{i+1} and \mathcal{S}_{i+1} as in Case 1.

Case 2b: $\Theta_i(\mathcal{C}_i)$ is an infinite class.

Note that $\Theta_i(\mathcal{C}_i)$ is in \mathbf{Ex}_n , since $\mathcal{C} \in (\mathbf{Ex}_n, \mathbf{Ex}_n)$ -robust, and $\mathcal{C}_i \subseteq \mathcal{C}$. Let $\mathcal{H} \subseteq \Theta_i(\mathcal{C}_i)$ be an infinite class in \mathbf{Ex}_0 (such \mathcal{H} exists by Proposition 12). Let $\mathcal{C}_{i+1} = \Theta^{-1}(\mathcal{H}) \cap \mathcal{C}_i$. Without loss of generality, assume \mathcal{C}_{i+1} is a proper subset of \mathcal{C}_i (otherwise just remove one element from \mathcal{C}_{i+1}). Suppose $h \in \mathcal{C}_i - \mathcal{C}_{i+1}$. Let $\mathcal{S}_{i+1} = \mathcal{S}_i \cup \{h\}$.

Properties (1)–(4) are satisfied by construction. (5) is satisfied, since $\Theta_i(\mathcal{C}_{i+1}) \subseteq \mathcal{H} \in \mathbf{Ex}_0$. ■

Our motivation for the following theorem started with the search for *simple* classes which disprove Bärzdiņš' conjecture. We were quite surprised to find such a class in \mathbf{Ex}_0 . Moreover, as the proof shows, even some self-referential classes can be robustly identified! Thus, one cannot claim that general recursive operators are capable of removing all coding. The self-referential class we use below is the rendition of the class SD used in [BB75].

Theorem 36 *There exists a \mathcal{C} in $(\mathbf{Ex}_0 - \mathbf{NUM}, \mathbf{Ex})$ -robust.*

PROOF. Let $\mathcal{C} = \{f \mid f \neq \mathbf{Zero} \wedge \varphi_{\min(\{x \mid f(x) \neq 0\})} = f\}$, i.e., for $f \in \mathcal{C}$, the minimum x such that $f(x)$ is non-zero, is a program for f .

Claim 37 $\mathcal{C} \in \mathbf{Ex}_0 - \mathbf{NUM}$.

PROOF. $\mathcal{C} \in \mathbf{Ex}_0$ is obvious. Suppose by way of contradiction, \mathcal{C}' is a recursively enumerable superset of \mathcal{C} . Let f be a recursive function such that $\mathcal{C}' = \{\varphi_{f(i)} \mid i \in N\}$. Now by implicit use of recursion theorem [Rog67], there exists an e such that,

$$\varphi_e(x) = \begin{cases} 0, & \text{if } x < e; \\ 1, & \text{if } x = e; \\ \varphi_{f(x-e-1)}(x) + 1, & \text{if } x > e. \end{cases}$$

Now $\varphi_e \in \mathcal{C}$. However, for each i , $\varphi_e(i + e + 1) = 1 + \varphi_{f(i)}(i + e + 1)$, thus $\varphi_e \neq \varphi_{f(i)}$. It follows that $\varphi_e \notin \mathcal{C}'$, contradicting $\mathcal{C} \subseteq \mathcal{C}'$. \square

We now show that,

Claim 38 *For any general recursive operator Θ , $\Theta(\mathcal{C}) \in \mathbf{Ex}$.*

PROOF. Suppose Θ is a general recursive operator. Let z be a program for $\Theta(\text{Zero})$. Let ProgUnion be a recursive function, mapping finite sets P of programs to programs, such that $\varphi_{\text{ProgUnion}(P)}$ may be defined as follows. Running on input x , program $\text{ProgUnion}(P)$ searches for a y , using a fixed dovetailing procedure, such that $(x, y) \in \bigcup_{i \in P} \Theta(\varphi_i)$; if and when such a y is found, $\varphi_{\text{ProgUnion}(P)}(x)$ is defined to be y . Define \mathbf{M} as follows.

$\mathbf{M}(\sigma)$

1. If $\Theta(\text{Zero})$ is consistent with σ , then output z , the program for $\Theta(\text{Zero})$.
2. Otherwise let n be the least number such that $\Theta(\text{Zero}[n])$ is inconsistent with σ .
(* Note that this implies that the input function, if from $\Theta(\mathcal{C})$, is one of $\Theta(\varphi_i)$, $i \leq n$. *)
3. Let $P = \{i \mid i \leq n \wedge \Theta(\varphi_i) \text{ (as enumerated in } |\sigma| \text{ steps) is consistent with } \sigma\}$.
4. Output $\text{ProgUnion}(P)$.

End

Now consider any input function $f \in \Theta(\mathcal{C})$. If f is consistent with $\Theta(\text{Zero})$, then \mathbf{M} \mathbf{Ex} -identifies f due to step 1. If f is not consistent with $\Theta(\text{Zero})$, then let n be least number such that $\Theta(\text{Zero}[n])$ is inconsistent with f . Now f must be one of $\Theta(\varphi_i)$, $i \leq n$, due to the definition of \mathcal{C} . Thus steps 3, 4 ensure that \mathbf{M} \mathbf{Ex} -identifies f . \square

The theorem follows from Claims 37 and 38. ■

The above proof uses a self-referential class. One may wish to find out the differences between the self-referential classes which allow robust \mathbf{Ex} -identification, and which do not, such as the class SD from Section 1. We make the following observation. On one hand, the class SD is topologically “extremely discrete” in that $f(0) \neq g(0)$, for all distinct f and g in SD. On the other hand, the class \mathcal{C} from Theorem 36 has a much more complex topological structure. Actually, \mathcal{C} possesses an accumulation point g , namely $g = \text{Zero}$. Moreover, what we have used in proving \mathcal{C} robustly \mathbf{Ex} -learnable is the following: For each n , one can effectively enumerate a finite set of programs, P , such that $\{f \in \mathcal{C} \mid \text{Equalupto}(f, g) < n\} \subseteq \{\varphi_i \mid i \in P\}$. The above properties are enough to show that \mathcal{C} is robustly \mathbf{Ex} -identifiable for an *arbitrary* such class \mathcal{C} .

Note that there are learning criteria \mathbf{I} such that $\mathbf{Ex}_0 \not\subseteq \mathbf{I}$; for example, $\mathbf{I} = \mathcal{T}\mathbf{Ex}$ [BB75] where the learning machine is not allowed to converge on a total function which it *cannot* learn, or $\mathbf{I} = \mathcal{T}\mathbf{Cons}$ [WZ95] where the learning machine is required to be consistent with the input data on *all* inputs. Consequently, for these criteria \mathbf{I} , $(\mathbf{I} - \mathbf{NUM}, \mathbf{Ex})$ -robust $\neq \emptyset$ does not follow directly

from Theorem 36. However, also for these \mathbf{I} , we can show the nonemptiness of $(\mathbf{I} - \mathbf{NUM}, \mathbf{Ex})$ -robust. This can be done by using the class \mathcal{S} (instead of the class \mathcal{C} from the proof of Theorem 36) defined as follows; recall that Φ denotes any Blum complexity measure [Blu67]. Let

$$f_e(x) = \begin{cases} 0, & \text{if } x < e; \\ 1, & \text{if } x = e; \\ \Phi_e(x - e - 1), & \text{otherwise} \end{cases}$$

and then let $\mathcal{S} = \{f_e \mid \Phi_e \text{ is total}\}$. Clearly, $\mathcal{S} \in \mathcal{TEx}$ and $\mathcal{S} \in \mathcal{TCons}$. Furthermore, $\mathcal{S} \notin \mathbf{NUM}$, since otherwise $\mathcal{R} \in \mathbf{NUM}$ would follow, a contradiction. Finally, for any general recursive operator Θ , $\Theta(\mathcal{S}) \in \mathbf{Ex}$ can be proved using the technique of the proof of Theorem 36 above.

4 Robust Team Identification

Smith [Smi82] considered identification by a team of machines.

Definition 39 Suppose \mathbf{I} is an identification criterion. A team (multi-set) \mathcal{M} of n machines is said to **Team_nI-identify** \mathcal{C} iff, for each $f \in \mathcal{C}$, there exists an $\mathbf{M} \in \mathcal{M}$ which \mathbf{I} -identifies f .

Team_nI = $\{\mathcal{C} \mid (\exists \mathcal{M} \text{ consisting of } n \text{ machines})[\mathcal{M} \text{ Team}_n \mathbf{I}\text{-identifies } \mathcal{C}]\}$.

Smith [Smi82] showed that the team hierarchies for \mathbf{Ex} and \mathbf{Bc} -identification are infinite. Here we show that these hierarchies are robustly infinite. Team learning has been generalized to consider the case when $m \leq n$ out of n machines are correct instead of 1 out of n as considered in Definition 39 [OSW86]. We do not consider the generalized definition here since m out of n teams are equivalent to 1 out of $\lfloor n/m \rfloor$ teams for the identification types \mathbf{Ex} and \mathbf{Bc} [PS88].

First we consider the following Lemma, which is a modified (somewhat effective) version of the team hierarchy theorem. Let **IIMTeam_n** denote the set of all teams of IIMs of size n . We identify members of \mathbf{SEG} with the finite functions they represent.

Lemma 40 Suppose $n \in \mathbb{N}$, $\sigma \in \mathbf{SEG}$, and $\mathcal{M} \in \mathbf{IIMTeam}_n$ are given. Then one can define $F_i^{\sigma, \mathcal{M}}$, for $1 \leq i \leq n + 1$ such that the following properties are satisfied.

- (A) There exists a (unique) i , $1 \leq i \leq n + 1$, such that $F_i^{\sigma, \mathcal{M}}$ is a total function.
- (B) Suppose i is the unique number such that $1 \leq i \leq n + 1$ and $F_i^{\sigma, \mathcal{M}}$ is total. Then, $\sigma \subseteq F_i^{\sigma, \mathcal{M}}$ and \mathcal{M} does not **Team_nBc-identify** $F_i^{\sigma, \mathcal{M}}$.
- (C) There exists a recursive function $g : \mathbb{N} \times \mathbf{SEG} \times \mathbf{IIMTeam}_n \times \mathbb{N} \rightarrow \mathbb{N}$ such that if $F_j^{\sigma, \mathcal{M}}$ is total then $\lim_{t \rightarrow \infty} g(j, \sigma, \mathcal{M}, t)$ converges to a program for $F_j^{\sigma, \mathcal{M}}$.

PROOF. The idea of the proof is to modify the **Team_{n+1}Ex – Team_nBc** diagonalization proof (as given in [Smi82]). Intuitively, consider the team hierarchy diagonalization for **Team_{n+1}Ex** vs **Team_nBc**. This team hierarchy diagonalization can be viewed (see Figure 1) as being an $n + 1$ level construction where the first level either gives a total function f_1 which is not \mathbf{Bc} -identified by any $\mathbf{M} \in \mathcal{M}$ or gives a τ_1 and a $\mathbf{M}^1 \in \mathcal{M}$ such that \mathbf{M}^1 does not \mathbf{Bc} -identify any extension of τ_1 ; the second level then tries to extend this τ_1 to construct either a f_2 which is not \mathbf{Bc} -identified by any machine in $\mathcal{M} - \{\mathbf{M}^1\}$ or gives a τ_2 and $\mathbf{M}^2 \in \mathcal{M} - \{\mathbf{M}^1\}$ such that \mathbf{M}^2 does not \mathbf{Bc} -identify any extension of τ_2 ; and so on.

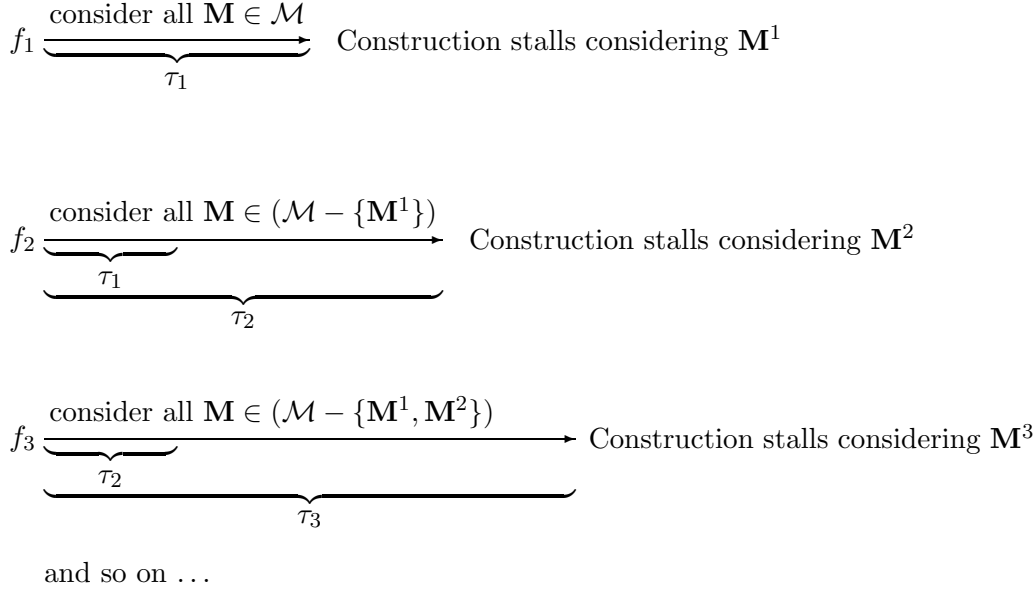


Figure 1: Construction of the functions f_1, f_2, \dots in team hierarchy.

Now, $F_j^{\sigma, \mathcal{M}}$ is taken to be f_j or τ_j as given by the construction above (for the purposes of the lemma, we make σ to be a subset of f_1 or τ_1 , respectively). g is obtained by essentially effectivizing the above construction.

We now proceed with the formal construction.

Suppose $\mathcal{M} = \{\mathbf{M}^1, \mathbf{M}^2, \dots, \mathbf{M}^n\}$. We will define below $F_i^{\sigma, \mathcal{M}}$, for $i = 0, 1, \dots$ in order of increasing i . (Though $F_0^{\sigma, \mathcal{M}}$ is not needed for the statement of lemma, it is easier to give the construction by defining $F_0^{\sigma, \mathcal{M}}$). Along with it we will also define S_i , for $1 \leq i \leq n + 1$. S_i will be a subset of $\{x \mid 1 \leq x \leq n\}$ of size $n - i + 1$. It will be the case that $S_i \supseteq S_{i+1}$. The following invariant will be satisfied.

Invariant (I): If $F_i^{\sigma, \mathcal{M}}$ is finite for all $i \leq k$, then for all total extensions f of $F_k^{\sigma, \mathcal{M}}$, none of the machines in $\{\mathbf{M}^r \mid r \notin S_{k+1}\}$, **Bc**-identifies f .

Also, if some $F_j^{\sigma, \mathcal{M}}$ is defined to be a total function, then $F_i^{\sigma, \mathcal{M}}$ is the empty function for $j < i \leq n + 1$ (in which case we do not need S_i , for $j < i \leq n + 1$). Thus, we only need to inductively define $F_j^{\sigma, \mathcal{M}}$ (and S_{j+1} if needed) for j such that, for all $i < j$, $F_i^{\sigma, \mathcal{M}}$ is finite.

Let $F_0^{\sigma, \mathcal{M}} = \sigma$. Let $S_1 = \{x \mid 1 \leq x \leq n\}$. Suppose we have already defined $F_i^{\sigma, \mathcal{M}}$ for $i < j$ (where none of $F_i^{\sigma, \mathcal{M}}$, $i < j$, is total), and S_i for $i \leq j$. We then define $F_j^{\sigma, \mathcal{M}}$ (and possibly S_{j+1}) as follows.

Intuitively, invariant (I), for $k = j - 1$, means that $F_{j-1}^{\sigma, \mathcal{M}}$ has diagonalized against all machines in $\{\mathbf{M}^r \mid r \notin S_j\}$. The job of $F_j^{\sigma, \mathcal{M}}$ is to diagonalize against at least one more machine in $\{\mathbf{M}^r \mid r \in S_j\}$.

Definition of $F_j^{\sigma, \mathcal{M}}$

1. Let $\tau = F_{j-1}^{\sigma, \mathcal{M}}$.

For $x \in \text{domain}(\tau)$, let $\mathbf{F}_j^{\sigma, \mathcal{M}}(x) = \tau(x)$.

2. If $j = n + 1$, then let $F_j^{\sigma, \mathcal{M}}(x) = 0$, for $x \notin \text{domain}(\tau)$. Else proceed with the rest of the construction.

3. Let w_0, w_1, \dots be an effective infinite sequence such that each element of S_j appears infinitely often in the sequence.

Let $\tau_0 = \tau$.

Go to stage 0.

4. Stage s

4.1 Search for an extension $\tau'_s \in \text{SEG}$ of τ_s such that $\varphi_{\mathbf{M}^{w_s}(\tau'_s)}(\max(\text{domain}(\tau'_s)) + 1) \downarrow$.

4.2 If and when such a τ'_s is found, let

4.2.1 $F_j^{\sigma, \mathcal{M}}(x) = \tau'_s(x)$, for $x \in \text{domain}(\tau'_s)$.

4.2.2 $F_j^{\sigma, \mathcal{M}}(\max(\text{domain}(\tau'_s)) + 1) = \varphi_{\mathbf{M}^{w_s}(\tau'_s)}(\max(\text{domain}(\tau'_s)) + 1) + 1$.

4.2.3 Let τ_{s+1} be $F_j^{\sigma, \mathcal{M}}[\max(\text{domain}(\tau'_s)) + 2]$. (That is, τ_{s+1} is $F_j^{\sigma, \mathcal{M}}$ defined upto now).
(* Note that definition in 4.2.1 above is consistent with the already defined portion of $F_j^{\sigma, \mathcal{M}}$, since $\tau_s \subseteq \tau'_s \subseteq \tau_{s+1}$. *)

4.2.4 Go to stage $s + 1$.

End stage s

End of Definition of $F_j^{\sigma, \mathcal{M}}$

Definition of S_{j+1}

If there are only finitely many stages in the above construction of $F_j^{\sigma, \mathcal{M}}$, then let $S_{j+1} = S_j - \{w_s\}$, where s is the last stage which is entered but does not finish.

End of Definition of S_{j+1}

Note that $F_j^{\sigma, \mathcal{M}}$ as defined above is an extension of $F_{j-1}^{\sigma, \mathcal{M}}$ due to step 1 of the construction.

We first show property (A) of Lemma. By our definition, if $F_j^{\sigma, \mathcal{M}}$ is total, then $F_i^{\sigma, \mathcal{M}}$ is empty, for $j < i \leq n + 1$. It immediately follows that at most one of $F_k^{\sigma, \mathcal{M}}$, $1 \leq k \leq n + 1$ is total. Also, if all $F_i^{\sigma, \mathcal{M}}$, $1 \leq i \leq n$, are finite, then $F_{n+1}^{\sigma, \mathcal{M}}$ is total (by step 2 of the construction above). Property (A) follows.

Suppose k is such that $F_k^{\sigma, \mathcal{M}}$ is total. $\sigma \subseteq F_k^{\sigma, \mathcal{M}}$ now follows since $\sigma = F_0^{\sigma, \mathcal{M}} \subseteq F_1^{\sigma, \mathcal{M}} \subseteq \dots \subseteq F_k^{\sigma, \mathcal{M}}$. (This shows the first part of property (B)).

We now show the remaining part of property (B). For this we also need to show that invariant (I) is satisfied.

Assume inductively that Invariant (I) is satisfied, for $k = j - 1$ in the statement of invariant (I), and for all $i < j$, $F_i^{\sigma, \mathcal{M}}$ is finite.

We then show that invariant (I) is maintained by the definition of $F_j^{\sigma, \mathcal{M}}$ (and S_{j+1}). Along with it we will show that if $F_j^{\sigma, \mathcal{M}}$ is total, then property (B) of lemma is satisfied.

For this we consider the following cases in the definition of $F_j^{\sigma, \mathcal{M}}$. Note that if $j = n + 1$, then clearly, $F_{n+1}^{\sigma, \mathcal{M}}$ would be total, and by invariant (I) (with $k = n$), no machine in \mathcal{M} **Bc**-identifies $F_{n+1}^{\sigma, \mathcal{M}}$. So assume $1 \leq j \leq n$, and consider the construction of $F_j^{\sigma, \mathcal{M}}$.

Case 1: There exist infinitely many stages.

In this case $F_j^{\sigma, \mathcal{M}}$ is total. Moreover, in each stage s we witness that $\mathbf{M}^{w_s}(\tau'_s)$ is not a program for $F_{\sigma, \mathcal{M}}^j$ (due to diagonalization in step 4.2.2). Since each $i \in S_j$ appears infinitely often in the sequence w_0, w_1, \dots , we have that for each $i \in S_j$, \mathbf{M}^i does not **Bc**-identify $F_j^{\sigma, \mathcal{M}}$. This along with invariant (I) for $k = j - 1$, gives us that $F_j^{\sigma, \mathcal{M}}$ is not **Bc**-identified by any machine in \mathcal{M} .

Case 2: Some stage s starts but does not halt.

Note that in this case $F_j^{\sigma, \mathcal{M}}$ is finite, and S_{j+1} is defined. Moreover, \mathbf{M}^{w_s} does not **Bc**-identify any total extension of $F_j^{\sigma, \mathcal{M}}$, since otherwise the search in step 4.1 would succeed. It follows that invariant (I) is satisfied for $k = j$.

From the above cases we have that property (B) of lemma holds. To see property (C) of lemma, suppose σ, \mathcal{M} and j are given such that $F_j^{\sigma, \mathcal{M}}$ is total. Then, one can determine in the limit, $F_i^{\sigma, \mathcal{M}}$, for $i < j$ (since the construction for each of these has only finitely many stages each) and determine S_i , for $i \leq j$. Given $F_{j-1}^{\sigma, \mathcal{M}}$ and S_j , a program for $F_j^{\sigma, \mathcal{M}}$ can then be easily determined. This proves the lemma. \blacksquare

Theorem 41 *For each $n \in \mathbb{N}$, there exists a class \mathcal{C} such that $\mathcal{C} \in (\mathbf{Team}_{n+1}\mathbf{Ex}, \mathbf{Team}_{n+1}\mathbf{Ex})$ -robust but $\mathcal{C} \notin \mathbf{Team}_n\mathbf{Bc}$.*

PROOF. Let n be given. The idea is to superimpose the construction given by Lemma 40 over the robustness construction of [Ful90].

For $i \in \mathbb{N}$, define σ_i, τ_i such that

(D) For all $j > i$, $\tau_i \subseteq \sigma_j$ and $\tau_i \subseteq \tau_j$.

(E) For all j , for all $k \leq j$, either:

(E1) $\Theta_k(\tau_j)$ is inconsistent with $\Theta_k(\sigma_j)$ or

(E2) For all τ_{ext} extending τ_j , for all σ_{ext} extending σ_j , $\Theta_k(\tau_{ext})$ is consistent with $\Theta_k(\sigma_{ext})$.

(F) One can obtain σ_i and τ_i limit recursively from i .

Now define \mathcal{C} as follows.

Let $\mathcal{M}_0, \mathcal{M}_1, \dots$ denote a recursive enumeration of all members of $\mathbf{IIMTeam}_n$. Let $F_j^{\sigma, \mathcal{M}}$ be as in Lemma 40. Let $\mathcal{C} = \{F_j^{\sigma_i, \mathcal{M}_i} \mid i \in \mathbb{N} \wedge 1 \leq j \leq n+1 \wedge F_j^{\sigma_i, \mathcal{M}_i} \in \mathcal{R}\}$. Clearly, $\mathcal{C} \notin \mathbf{Team}_n\mathbf{Bc}$ (by properties (A) and (B) in Lemma 40). We claim that $\mathcal{C} \in (\mathbf{Team}_{n+1}\mathbf{Ex}, \mathbf{Team}_{n+1}\mathbf{Ex})$ -robust. For this, it suffices to show that, for $1 \leq j \leq n+1$, $\mathcal{C}_j = \{F_j^{\sigma_i, \mathcal{M}_i} \mid i \in \mathbb{N} \wedge F_j^{\sigma_i, \mathcal{M}_i} \in \mathcal{R}\}$ is in $(\mathbf{Ex}, \mathbf{Ex})$ -robust. Thus, it suffices to show that, for each k , $\mathcal{C}'_j = \{\Theta_k(F_j^{\sigma_i, \mathcal{M}_i}) \mid F_j^{\sigma_i, \mathcal{M}_i} \in \mathcal{R} \wedge i \geq k\}$ is in \mathbf{Ex} (since **Ex**-identifiability is invariant under union of any finite set of functions). Below is the informal description of the machine which **Ex**-identifies \mathcal{C}'_j .

\mathbf{M} on any input g searches for the least $i \geq k$ such that $\Theta_k(\sigma_i) \subseteq g$. Note that such an i , if any, can be found in the limit (there exists such an i , if $g \in \mathcal{C}'_j$). If $\Theta_k(\sigma_i)$ and $\Theta_k(\tau_i)$ are consistent, then \mathbf{M} outputs (in the limit) a program for $\bigcup\{\Theta_k(\gamma) \mid \sigma_i \subseteq \gamma \text{ or } \tau_i \subseteq \gamma\}$; otherwise, \mathbf{M} outputs

(in the limit) a program for $\Theta_k(F_j^{\sigma_i, \mathcal{M}_i})$. Note that using property (C) of Lemma 40, one can find (in the limit) a program for $F_j^{\sigma_i, \mathcal{M}_i}$, and thus a program for $\Theta_k(F_j^{\sigma_i, \mathcal{M}_i})$.

We now show that the above **M Ex**-identifies \mathcal{C}'_j . Let g be the input function and i be as computed by **M** above. If $\Theta_k(\sigma_i)$ is consistent with $\Theta_k(\tau_i)$, then for any total function f extending σ_i or τ_i , $\Theta_k(f)$ is consistent with $\bigcup\{\Theta_k(\gamma) \mid \sigma_i \subseteq \gamma \text{ or } \tau_i \subseteq \gamma\}$. Thus, **M Ex**-identifies g . On the other hand, if $\Theta_k(\tau_i)$ is inconsistent with $\Theta_k(\sigma_i)$ (and hence with g), then g must be $\Theta_k(F_j^{\sigma_i, \mathcal{M}_i})$ (otherwise $g \notin \mathcal{C}'_j$). Thus, **M Ex**-identifies g . This proves the theorem. \blacksquare

Corollary 42 For all $n \in \mathbb{N}$,

- (a) $(\mathbf{Team}_n \mathbf{Ex}, \mathbf{Team}_n \mathbf{Ex})$ -robust \subset $(\mathbf{Team}_{n+1} \mathbf{Ex}, \mathbf{Team}_{n+1} \mathbf{Ex})$ -robust,
- (b) $(\mathbf{Team}_n \mathbf{Bc}, \mathbf{Team}_n \mathbf{Bc})$ -robust \subset $(\mathbf{Team}_{n+1} \mathbf{Bc}, \mathbf{Team}_{n+1} \mathbf{Bc})$ -robust.

PROOF. Immediately from Theorem 41. \blacksquare

5 Uniformly Robust Learning

Intuitively, for $\mathcal{C} \in \mathbf{I}$ to be *uniformly* in (\mathbf{I}, \mathbf{J}) -robust, one should be able to *effectively* (in i) find a machine to **J**-identify $\Theta_i(\mathcal{C})$. In other words, the images of \mathcal{C} are all not only learnable in that learning machines for them *exist*, but in a sense one even has learning machines for them *effectively in hand*. For this strengthened version of robust learning, we have both positive and negative results. Actually, for **Ex**-learning with a bounded number of mind changes, uniformly robust learning is possible only for very restricted (finite!) classes. On the other hand, for standard **Ex**-learning as well as for some kind of generalized **Ex**-learning, uniformly robust learning seems to be achievable.

Definition 43 A class \mathcal{C} is said to be in (\mathbf{I}, \mathbf{J}) -uniformly robust iff $\mathcal{C} \in \mathbf{I}$, and there exists a recursive function G such that $(\forall i \mid \Theta_i \text{ is general recursive})[\Theta_i(\mathcal{C}) \subseteq \mathbf{J}(\mathbf{M}_{G(i)})]$.

We now show that for a bounded number of mind changes, uniformly robust identification is quite weak.

Theorem 44 Suppose $n \in \mathbb{N}$. Consider any \mathcal{C} which contains at least 2^{n+2} distinct functions. Then $\mathcal{C} \notin (\mathbf{I}, \mathbf{Ex}_n)$ -uniformly robust. So, in particular, no infinite class is in $(\mathbf{I}, \mathbf{Ex}_n)$ -uniformly robust.

PROOF. Suppose by way of contradiction that G is a recursive function such that for all e , if Θ_e is general recursive, then $\mathbf{M}_{G(e)}$ **Ex** $_n$ -identifies $\Theta_e(\mathcal{C})$.

Let $m = 2^{n+2}$. Let f^0, f^1, \dots, f^{m-1} be m distinct functions in \mathcal{C} . Let t be such that for all $i < j < m$, $f^i[t] \neq f^j[t]$. Let $F_{k,n}^i$ be as in Proposition 14 (note that $F_{k,n}^i$ can be generated effectively from k, n , and i).

Now by the Kleene recursion theorem [Rog67], there exists an e such that Θ_e may be defined as follows.

$$\Theta_e(\eta) = \begin{cases} \lambda x. \uparrow, & \text{if } \{0, \dots, t\} \not\subseteq \text{domain}(\eta); \\ F_{G(e),n}^i, & \text{if } \{0, \dots, t\} \subseteq \text{domain}(\eta) \text{ and } \eta[t] \subseteq f^i[t]; \\ \lambda x. 0, & \text{if } \{0, \dots, t\} \subseteq \text{domain}(\eta) \text{ and for all } i < m, \eta[t] \not\subseteq f^i[t]. \end{cases}$$

It is easy to see that Θ_e is general recursive and $\Theta_e(\{f^0, \dots, f^{m-1}\}) = \{F_{G(e),n}^0, \dots, F_{G(e),n}^{m-1}\}$.

It thus follows from Proposition 14 that $\mathbf{M}_{G(e)}$ does not \mathbf{Ex}_n -identify $\Theta_e(\mathcal{C})$. ■

Note that the above theorem still holds if we require G to be partial recursive with domain a superset of $\{k \mid \Theta_k \text{ is general recursive}\}$. To see this, given a program i for such partial recursive G , one can construct a total function G' (effectively from i) such that

$$\mathbf{M}_{G'(e)}(\sigma) = \begin{cases} ?, & \text{if } G(e) \text{ is not defined within } |\sigma| \text{ steps} \\ & \text{(here step counting is with respect to the program } i) \\ \mathbf{M}_{G(e)}(\sigma), & \text{otherwise.} \end{cases}$$

Now Theorem 44 can be applied to G' .

Our next results imply that uniformly robust identification in the limit is really rich. First, it is easy to verify that

Proposition 45 $\mathbf{NUM} \subseteq (\mathbf{Ex}, \mathbf{Ex})$ -uniformrobust.

Theorem 46 now shows that it is also possible to learn classes outside of \mathbf{NUM} uniformly robustly.

Theorem 46 *There exists a class \mathcal{C} in $(\mathbf{Ex}_0 - \mathbf{NUM}, \mathbf{Ex})$ -uniformrobust.*

PROOF. Let \mathcal{C} be the class from Theorem 36. Then it is easy to see that the proof of Theorem 36 can be “uniformized”. ■

It is open at present whether $(\mathbf{Ex}, \mathbf{Ex})$ -robust = $(\mathbf{Ex}, \mathbf{Ex})$ -uniformrobust. The next result shows that for some non-trivial classes, uniformly robust learning can be achieved not only for *general* recursive operators, but even for *all partial* recursive operators. Therefore, we need the following generalization \mathbf{BlumEx} of \mathbf{Ex} which was introduced in [BB75]. Intuitively, a *partial* function η is \mathbf{BlumEx} -identifiable if in the limit an index i of an *extension* of η , i.e. $\eta \subseteq \varphi_i$, can be discovered.

For identification of a partial function, η , \mathbf{M} receives as input a graph of η , in arbitrary order. For this purpose we define the notion of texts for partial function as follows. A *text* is a mapping from N to $(N \times N) \cup \{\#\}$. The content of T , denoted $\text{content}(T)$, is $\text{range}(T) - \{\#\}$. T is a text for η iff $\text{content}(T) = \{(x, \eta(x)) \mid \eta(x) \downarrow\}$. $T[n]$ denotes the initial segment of T of length n . We let α range over initial segments of texts. $\text{content}(\alpha)$ denotes $\text{range}(\alpha) - \{\#\}$. $|\alpha|$ denotes the length of α . \mathbf{M} converges on T to i (written: $\mathbf{M}(T) \downarrow = i$) iff $(\forall^\infty n)[\mathbf{M}(T[n]) = i]$.

Definition 47 [BB75]

- (a) \mathbf{M} \mathbf{BlumEx} -identifies η (written: $\eta \in \mathbf{BlumEx}(\mathbf{M})$) iff for each text T for η , there is an i such that $\mathbf{M}(T) \downarrow = i$ and $\eta \subseteq \varphi_i$.
- (b) \mathbf{M} \mathbf{BlumEx} -identifies a class \mathcal{S} of partial functions, iff \mathbf{M} \mathbf{BlumEx} -identifies each $\eta \in \mathcal{S}$.

Theorem 48 *There exist a class $\mathcal{C} \subseteq \mathcal{R}$, $\mathcal{C} \notin \mathbf{NUM}$ and a recursive function G such that, for all k , $\Theta_k(\mathcal{C}) \subseteq \mathbf{BlumEx}(\mathbf{M}_{G(k)})$.*

PROOF. The proof is based on a modification of the construction given in [Ful90]. First we show the following.

Claim 49 *There exist $\sigma_i \in \text{SEG}$ and $\eta_i \in \mathcal{P}$, where σ_i and a program for η_i can be obtained limit recursively in i , such that, for $i \in N$, the following three conditions are satisfied.*

(A) For all $j > i$, $\sigma_i \subseteq \sigma_j$;

(B) η_i extends σ_i in such a way that: If

(i) φ_i is total, and

(ii) $\mathcal{S}_i = \{\varphi_w \mid w \in \text{range}(\varphi_i)\} \subseteq \mathcal{R}$,

then η_i is total and does not belong to \mathcal{S}_i ;

(C) For all j , for all $i, k < j$, either:

(C1) $\Theta_k(\sigma_j)$ is inconsistent with $\Theta_k(\eta_i)$ or

(C2) For all σ_{ext} extending σ_j , $\Theta_k(\eta_i)$ is consistent with $\Theta_k(\sigma_{ext})$.

PROOF. Let $\sigma_0 = \Lambda$.

Suppose we have defined σ_j , for $j \leq i$. Then define η_i, σ_{i+1} as follows.

$$\eta_i(x) = \begin{cases} \sigma_i(x), & \text{if } x \in \text{domain}(\sigma_i); \\ \varphi_{\varphi_i(x - \max(\text{domain}(\sigma_i)) - 1)}(x) + 1, & \text{otherwise.} \end{cases}$$

Let σ_{i+1} be an extension of σ_i such that (C) is satisfied. It is easy to verify that one can determine (one suitable) σ_i limit effectively from $(\sigma_j)_{j < i}$. A program for η_i can be determined effectively from σ_i . The claim follows. \square

For $i, k < j$, let $\text{met}_{i,j}^k$ be a predicate which is true iff $\Theta_k(\eta_i)$ is inconsistent with $\Theta_k(\sigma_j)$. Note that $\text{met}_{i,j}^k$ implies that $\Theta_k(\eta_i)$ is inconsistent with $\Theta_k(\eta_{j'})$ for all $j' \geq j$. Moreover, $\text{met}_{i,j}^k$ can be determined limit effectively from i, j, k . Let $\mathcal{C} = \{\eta_i \mid \eta_i \text{ is total}\}$. We claim that \mathcal{C} satisfies the theorem. $\mathcal{C} \notin \mathbf{NUM}$ follows directly from (B).

A class of partial functions S is said to be recursively enumerable iff S is empty or there exists a recursive function f such that $S = \{\varphi_{f(i)} \mid i \in N\}$. Suppose S is an r.e. class of partial functions, where the ‘‘enumeration’’ f of S is fixed implicitly. Then, let $\text{Union}(S)$ denote a (partial) function η such that for any $x \in N$, $\eta(x)$ is the first y found (in some systematic search), if any, such that, for some i , $\varphi_{f(i)}(x) = y$. (Here if $S = \emptyset$, then we take $\text{Union}(S)$ to be the everywhere undefined function.)

Now for any $i, k \in N$, define β_i^k as follows. Let $\beta_i^k = \text{Union}(\{\Theta_k(\sigma_{ext}) \mid \sigma_{ext} \supseteq \sigma_{i+1}\})$. For $i \geq k$, define δ_i^k as follows.

$$\delta_i^k(x) = \begin{cases} \Theta_k(\eta_i), & \text{if } \text{met}_{i,i+1}^k; \\ \Theta_k(\eta_i) \cup \beta_i^k, & \text{if NOT } \text{met}_{i,i+1}^k. \end{cases}$$

Note that programs for β_i^k and δ_i^k can be found limit effectively in k and i . To see this, note that one can determine σ_{i+1} in the limit from i . Once σ_{i+1} is determined, a program for β_i^k can be effectively found from σ_{i+1} . Furthermore, since one can limit effectively (in i and k) determine whether $\text{met}_{i,i+1}^k$ holds, one can limit effectively (in i and k) determine a program for δ_i^k using the programs for η_i and β_i^k .

Let σ_i^s , $s \in N$, denote a recursive (in i and s) sequence which converges to σ_i . Let $p\eta_i^s$ denote a recursive (in i and s) sequence of programs which converges to a program for η_i . Similarly, let $p\beta_i^{k,s}$ and $p\delta_i^{k,s}$, respectively, denote a recursive (in i, k and s) sequence of programs which converge to a program for β_i^k and δ_i^k , respectively. Let $\eta_i^s, \beta_i^{k,s}, \delta_i^{k,s}$ denote the functions computed by $p\eta_i^s, p\beta_i^{k,s}$, and $p\delta_i^{k,s}$, respectively.

Now the construction of a machine $\mathbf{M}_{G(k)}$ witnessing the theorem is as follows. Below when we say η_i^s restricted to s steps of computation, we mean the partial function computed by $p\eta_i^s$ within s steps. Similarly, for $\delta_m^{k,s}$ restricted to s steps of computation.

Suppose k is given. Define $\mathbf{M}_{G(k)}$ (effectively in k) to **Ex**-identify $\Theta_k(\mathcal{C})$ as follows:

$\mathbf{M}_{G(k)}(\alpha)$:

1. Let $s = |\alpha|$.
2. Let $S_s = \{i \leq k \mid \Theta_k(\eta_i^s \text{ restricted to } s \text{ steps of computation}) \text{ is consistent with } \text{content}(\alpha)\}$.
3. Let $m_s \leq s$ be the minimum value greater than k such that $\delta_{m_s}^{k,s}$ (restricted to s steps of computation) is consistent with $\text{content}(\alpha)$ (where if no such $m_s \leq s$ exists, then we take m_s to be s).
4. If $\text{content}(\alpha)$ is inconsistent with $\Theta_k(\sigma_{k+1}^s)$, Then output a program for $\text{Union}(\{\Theta_k(\eta_i^s) \mid i \in S_s\})$.

Else output a program for $\text{Union}(\{\delta_{m_s}^{k,s}\} \cup \{\Theta_k(\eta_i^s) \mid i \in S_s\})$.

End

We now argue that $\mathbf{M}_{G(k)}$ above suffices.

Suppose the input function is $\Theta_k(\eta_i)$. Clearly, on input text for $\Theta_k(\eta_i)$, S_s as calculated by $\mathbf{M}_{G(k)}$ converges (as s goes to infinity).

Case 1: $i \leq k$.

Case 1a: $\Theta_k(\sigma_{k+1})$ is inconsistent with $\Theta_k(\eta_i)$. In this case, in step 4, If statement will succeed (for large enough s). Moreover, the limiting value of S_s contains i . It follows that $\mathbf{M}_{G(k)}$ **BlumEx**-identifies $\Theta_k(\eta_i)$.

Case 1b: $\Theta_k(\sigma_{k+1})$ is consistent with $\Theta_k(\eta_i)$. In this case $\text{met}_{i,k+1}^k$ is false. Thus, for all σ_{ext} extending σ_{k+1} , $\Theta_k(\sigma_{ext})$ is consistent with $\Theta_k(\eta_i)$. In particular δ_{k+1}^k is consistent with $\Theta_k(\eta_i)$. Thus m_s , as calculated by the procedure for $\mathbf{M}_{G(k)}$, converges to $k+1$ (as s goes to infinity). Now again as in Case 1a, we immediately see that $\mathbf{M}_{G(k)}$ will use Else clause in step 4 and (for large enough s) a correct program will be output.

Case 2: $i > k$.

In this case, clearly, m_s as computed by $\mathbf{M}_{G(k)}$ above, converges (as s goes to infinity) to something less than or equal to i . Below let m denote this limiting value. We first claim that $\Theta_k(\eta_i) \subseteq \delta_m^k$. To see this, first note that if $m = i$, then this is certainly true. If $m < i$, then note that $\text{met}_{m,m+1}^k$ must be false (if $\text{met}_{m,m+1}^k$ is true, then δ_m^k cannot be consistent with $\Theta_k(\eta_i)$, due to the fact that $\sigma_{m+1} \subseteq \eta_i$). It thus follows that $\text{domain}(\Theta_k(\eta_i)) \subseteq \text{domain}(\beta_m^k) \subseteq \text{domain}(\delta_m^k)$. Now consistency of δ_m^k with $\Theta_k(\eta_i)$ implies that $\Theta_k(\eta_i) \subseteq \delta_m^k$. Thus, again by the Else part of step 4, $\mathbf{M}_{G(k)}$ **BlumEx**-identifies $\Theta_k(\eta_i)$. ■

6 Conclusions

In this paper we considered robust identification, and provided several positive and negative results. Our main result, Theorem 24 and Corollary 31, shows that the mind change hierarchy with respect

to **Ex**-identification stands robustly! This contrasts with the fact that some other hierarchies, such as the anomaly hierarchies for **Ex** and **Bc**-identification, do not stand robustly. Moreover, we proved that the team hierarchies for both **Ex** and **Bc** stand robustly as well (Theorem 41 and Corollary 42). We also showed the counter-intuitive fact that even some of the self-describing classes can be learned robustly (Theorem 36). We also discussed on the possible reasons behind why some self-referential classes can be robustly identified while others cannot. Thereby, in several contexts, we observed the surprising fact that a more complex topological structure of the classes to be learned (expressed by the existence of an accumulation point, Corollary 34, possibly in an “effectively approximable” manner, Theorem 36) leads to positive robustness results, whereas an easy or even “trivial” topological structure (class SD from Section 1 and class CONST from Theorem 22, in each of these classes all the functions are pairwise different as witnessed by argument 0) leads to negative results. Interestingly, this phenomenon holds both *inside* **NUM**, the world of recursively enumerable classes (see Corollary 34 versus Theorem 22), and *outside* **NUM** (see Theorem 36 versus class SD from Section 1).

Several of our results show the difficulty of robustly identifying even simple classes when only a bounded number of mind changes is allowed. For example, Theorem 16 shows that no infinite class of functions can be robustly **Ex**₀-identified. Theorem 18 states that no recursively enumerable class of functions can be robustly identified with a bounded number of mind changes, and Theorem 22 points out that CONST does not even contain any infinite subclass which can be robustly $\bigcup_{m \in \mathbb{N}} \mathbf{Ex}_m$ -identified.

In the present paper we have confined ourselves to *general recursive* operators for realizing the transformations of the classes to be learned. Clearly, this is not the most general approach. One could allow *recursive* operators instead, which map the functions in the class to be learned to total functions, but need not map functions outside the class to total functions. Let **(I, J)**-*recrobust* = $\{\mathcal{C} \mid \mathcal{C} \in \mathbf{I} \wedge (\forall \text{ recursive operators } \Theta \mid \Theta(\mathcal{C}) \subseteq \mathcal{R})[\Theta(\mathcal{C}) \in \mathbf{J}]\}$. Clearly, the negative results hold also for this extended notion of robustness. However, some of our positive results do not carry over to recursive operators. For example, Theorem 35 and Corollary 33 do not hold for this extended notion of robustness. This follows as a corollary to the following theorem.

Theorem 50 *For all $m \in \mathbb{N}$, no infinite class belongs to $(\mathbf{Ex}_0, \mathbf{Ex}_m)$ -recrobust.*

PROOF. Suppose by way of contradiction that \mathcal{C} is an infinite class in $(\mathbf{Ex}_0, \mathbf{Ex}_m)$ -recrobust. Suppose **M** witnesses that $\mathcal{C} \in \mathbf{Ex}_0$. Let $X = \{f[n+1] \mid \mathbf{M}(f[n]) = ? \wedge \mathbf{M}(f[n+1]) \neq ?\}$ (i.e. X denotes the initial segments on which **M** outputs its conjecture for the first time). Let $\sigma_0, \sigma_1, \dots$, be a 1–1 recursive enumeration of X . Let H_0, H_1, \dots , be a recursive enumeration of the class \mathcal{S} , as given by Proposition 15. Now define

$$\Theta(\eta) = \begin{cases} H_i, & \text{if } \sigma_i \subseteq \eta; \\ \Lambda, & \text{otherwise.} \end{cases}$$

It is easy to verify that $\Theta(\mathcal{C})$ is an infinite subset of \mathcal{S} . The theorem now follows from Proposition 15. ■

In fact the above theorem can be strengthened as follows. Let **I** be an identification type. **I** is called *diverse* iff for any infinite class \mathcal{C} in **I**, there is a computable functional Q such that both $\mathcal{C} \subseteq \text{domain}(Q)$, and $\{Q(f) \mid f \in \mathcal{C}\}$ is infinite (“diverse”). Then, for any diverse identification type **I**, for any infinite $\mathcal{C} \in \mathbf{I}$, one can define a recursive operator Θ as follows: $\Theta(f) = H_{Q(f)}$, where H_i is as above. Note that $\Theta(\mathcal{C})$ is an infinite subset of \mathcal{S} from Proposition 15. It thus

follows from Proposition 15 that no infinite class belongs to $(\mathbf{I}, \bigcup_{m \in \mathbb{N}} \mathbf{Ex}_m)$ -recrobust. As a further possible objection against recrobustness we mention that this notion of robustness is not closed under subset.

On the other hand, a close inspection of the proof of our main result on the robust mind change hierarchy (Theorem 24) shows that this result remains valid even for the extended notion of robustness. That is, the $(\mathbf{Ex}_n, \mathbf{Ex}_n)$ -recrobust hierarchy stands. Further, we could add the function Zero to the self-referential class \mathcal{C} of Theorem 36 to show that $(\mathbf{Ex}_1 - \mathbf{NUM}, \mathbf{Ex})$ -recrobust is non-empty (with essentially the same proof). Note that this is a strengthening of Fulk’s [Ful90] result who proved $(\mathbf{Ex} - \mathbf{NUM}, \mathbf{Ex})$ -recrobust non-empty. Moreover, our result leads to the non-expected consequence that even recursive operators are incapable of removing all self-referential coding! Recently, in [OS99] a notion of robustness, called hyperrobustness, has been defined which prevents robust learning from self-referential classes as used above. Actually, it is proved there that a class of recursive functions is hyperrobustly \mathbf{Ex} -learnable iff this class belongs to \mathbf{NUM} . Hence, on the one hand, for the notion of hyperrobustness, Bārzdīņš’ conjecture is provably true. On the other hand, in order to achieve this goal this notion of robustness had to be defined so “strong” that in a sense it loses much of its richness. It seems interesting to find out if there is a notion of robustness which both allows classes *outside* \mathbf{NUM} to be learnable robustly and prevents self-referential classes “as above” from being learnable robustly. Clearly, to answer this question in a rigorous way would require to give a formal definition of “self-referential” before. But, possibly, such a notion of robustness does not exist at all. The intuition behind this vague conjecture is that self-description may be an inherent and even natural property of all sufficiently rich concepts of computability. For example, recall that any polynomial (over the natural numbers as well as over the reals) is self-describing in that it is uniquely determined by every segment containing more points than the degree of the polynomial. Furthermore, one could argue that self-description is quite natural, as every cell of every organism contains a “program” that completely describes this organism.

Anyway, our work may be viewed as a further step to investigate robust learning. In our opinion, the results obtained so far give strong evidence that robust learning is really surprisingly rich and worth studying. Naturally, much remains to do to get a yet deeper understanding of the nature of robustness in learning and thereby, hopefully, of the nature of learning in general.

References

- [AF96] A. Ambainis and R. Freivalds. Transformations that preserve learnability. In S. Arikawa and A. Sharma, editors, *Algorithmic Learning Theory: Seventh International Workshop (ALT '96)*, volume 1160 of *Lecture Notes in Artificial Intelligence*, pages 299–311. Springer-Verlag, 1996.
- [AS83] D. Angluin and C. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–289, 1983.
- [Bār71] J. Bārzdīņš. *Complexity and Frequency Solution of Some Algorithmically Unsolvable Problems*. PhD thesis, Novosibirsk State University, 1971. In Russian.
- [Bār74] J. Bārzdīņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.

- [BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
- [Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.
- [CJO⁺98] J. Case, S. Jain, M. Ott, A. Sharma, and F. Stephan. Robust learning aided by context. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 44–55. ACM Press, 1998.
- [CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [Fre91] R. Freivalds. Inductive inference of recursive functions: Qualitative theory. In J. Bārzdīņš and D. Björner, editors, *Baltic Computer Science*, volume 502 of *Lecture Notes in Computer Science*, pages 77–110. Springer-Verlag, 1991.
- [Ful90] M. Fulk. Robust separations in inductive inference. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 405–410. IEEE Computer Society Press, 1990.
- [Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Mass., second edition, 1999.
- [JSW98] S. Jain, C. Smith, and R. Wiehagen. On the power of learning robustly. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 187–197. ACM Press, 1998.
- [JW97] S. Jain and R. Wiehagen. On the power of learning robustly. Technical Report LSA-97-06E, Centre for Learning Systems and Applications, Department of Computer Science, University of Kaiserslautern, Germany, 1997.
- [KS89] S. Kurtz and C. Smith. On the role of search for learning. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 303–311. Morgan Kaufmann, 1989.
- [KW80] R. Klette and R. Wiehagen. Research in the theory of inductive inference by GDR mathematicians – A survey. *Information Sciences*, 22:149–169, 1980.
- [Lin72] R. Lindner. *Algorithmische Erkennung*. Dissertation B, University of Jena, 1972. In German.
- [OS99] M. Ott and F. Stephan. Avoiding coding tricks by hyperrobust learning. In P. Vitányi, editor, *Fourth European Conference on Computational Learning Theory*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 183–197. Springer-Verlag, 1999.
- [OSW86] D. Osherson, M. Stob, and S. Weinstein. Aggregating inductive expertise. *Information and Control*, 70:69–95, 1986.

- [PS88] L. Pitt and C. Smith. Probability and plurality for aggregations of learning machines. *Information and Computation*, 77:77–92, 1988.
- [Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted, MIT Press 1987.
- [Smi82] C. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982.
- [Wey52] H. Weyl. *Symmetry*. Princeton University Press, 1952.
- [WZ95] R. Wiehagen and T. Zeugmann. Learning and consistency. In K. P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 1–24. Springer-Verlag, 1995.
- [Zeu86] T. Zeugmann. On Bārzdīņš’ conjecture. In K. P. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the International Workshop*, volume 265 of *Lecture Notes in Computer Science*, pages 220–227. Springer-Verlag, 1986.