

Chapter 1

A TOUR OF ROBUST LEARNING

Sanjay Jain*

School of Computing, National University of Singapore, Singapore 119260

sanjay@comp.nus.edu.sg

Frank Stephan†

Mathematisches Institut, INF 294, Universität Heidelberg, 69121 Heidelberg, Germany

fstefhan@math.uni-heidelberg.de

Abstract Bärzdiņš conjectured that only recursively enumerable classes of functions can be learned robustly. This conjecture, which was finally refuted by Fulk, initiated the study of notions of robust learning. The present work surveys research on robust learning and focuses on the recently introduced variants of uniformly robust and hyperrobust learning. Proofs are included for the (already known) results that uniformly robust Ex-learning is more restrictive than robust Ex-learning, that uniformly robustly Ex-learnable classes are consistently learnable, that hyperrobustly Ex-learnable classes are in Num and that some hyperrobustly BC-learnable class is not in Num.

1. Introduction

The general setting of function learning is the following: Given a set S of functions, a learner simultaneously reads data $f(0), f(1), \dots$ on the function to be learned and outputs hypotheses e_0, e_1, \dots such that these descriptions converge in some abstract sense to an explanation of the observed function f , whenever $f \in S$. Gold [11] made this informal setting more precise by requiring syntactical convergence in the case that

*Sanjay Jain is supported in part by the NUS grant number RP3992710.

†Frank Stephan is supported by the Deutsche Forschungsgemeinschaft (DFG) Heisenberg grant Ste 967/1-1; part of the work was done while he visited the University of Wisconsin – Madison.

$f \in S$: Almost all e_k output by the learner are the same program e for the function f .

Taking this as a starting point, learning theorists gave a large variation of learning paradigms in order to capture more precisely the intuitive notion of learning. But while the formalization of the intuitive notion of computing by recursive functions and equivalent concepts like Turing machines has turned out to be unique, the development in learning theory is more parallel to the one in complexity theory: there is a large family of important notions including Gold's notion of "explanatory learning" (Ex-learning) discussed above. One main line of research in inductive inference is to compare and evaluate these various notions of learning.

One of Bärzdiņš' main concerns with respect to the development of inductive inference was the question, to which extent the notions and also the considered learning problems are natural. Most natural learning problems S have an enumeration f_0, f_1, \dots such that $S \subseteq \{f_0, f_1, \dots\}$ and the mapping $e, x \rightarrow f_e(x)$ is total and recursive in both parameters. Such classes can be learned by enumeration: the learner searches for the first e such that f_e is consistent with the input and outputs a canonical program for f_e .

Of course, not all learnable classes are of this form. It is almost a standard homework for computer science students to produce a program which outputs its own code. This principle is used in the definition $SD = \{f : f = \varphi_{f(0)}\}$ of the class of self-describing functions. The class SD is learnable but not in Num, that is, SD is not contained in a recursively enumerable class of total functions. As the learning part consists only mapping the input $f(0) f(1) \dots f(n)$ to $f(0)$, this example looks a bit like cheating and it is obviously not very natural. One might ask, whether one can come up with a more natural example. Bärzdiņš thought that the answer is negative and that there are no naturally defined Ex-learnable classes outside Num.

Therefore, Bärzdiņš was looking for a framework of learning, which does not permit such obvious coding and enforces that learnable classes are naturally defined or at least subclasses of naturally defined learnable classes. His approach is to postulate that not only S itself but also all variants of S obtained by general recursive operators have to be Ex-learnable. The class SD does not satisfy this condition as $\Theta(SD) = REC$, where REC is the class of all total recursive functions and the operator Θ is given by the simple equation $\Theta(f)(x) = f(x + 1)$. So Bärzdiņš intention is that the notion of Robust Learning captures all naturally defined learnable classes and Bärzdiņš' Conjecture states: Ev-

ery robustly learnable class is already in Num. Although first incidence supported Bārzdīņš' Conjecture, Fulk [10] was able to disprove it – see Section 3 for more details.

On a philosophical side, Herman Weyl [26] 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.

Case [4] was concerned whether coding tricks really occur in nature. As it turns out that they occur quite frequent not only in inductive inference but also in recursion theory and other branches of mathematics. Case thinks, that perhaps the real world may actually have some of its parts coded in other of its parts. As he essentially pointed out, such a view is consistent with both certain Eastern metaphysical principles and Leibniz' Monadology. So, one might conclude that coding is not so unnatural as it seemed to be when Bārzdīņš formulated his conjecture.

Although Fulk [10] had settled the main question, the field of robust learning has received much more attention recently. The failure of Bārzdīņš' Conjecture motivated to introduce and investigate several modified approaches.

Uniformly robust learning, as outlined in Section 4, postulates that one can compute a learner for $\Theta(S)$ from any program for Θ . Jain, Smith and Wiehagen [14] introduced this notion and showed that also for this restrictive variant, Bārzdīņš' Conjecture fails. Furthermore, the counterexample for both notions, robust and uniformly robust learning, can be taken to consist of functions f which are topologically self-describing by having that the first non-zero value at a place $x > e$ where e is the minimal program of f . This result questioned the intention that robust learning makes coding tricks impossible. On the other hand, uniformly robustly Ex-learnable classes turned out to be natural in the way that they are always consistently learnable [6]. Note that a consistent learner might be partial, but for every $f \in S$ where S is the class to be learned, and for every n , a consistent learner outputs on input $f(0)f(1)\dots f(n)$ a function which is defined and coincides with f on the arguments $0, 1, \dots, n$. Taking this result as a starting point, Section 5 gives an overview on the connections between robustness and the various notions of consistency.

Hyperrobust learning, considered in Section 6, turned out to satisfy Bāzdiņš’ Conjecture for Ex-learning. Several further results support the naturalness of the definition. Hyperrobust BC-learning does not collapse to Num [20] and the corresponding class is an alternative example for Jain’s result [12] that robust BC-learning is more powerful than robust EX-learning.

Section 7 compares the team hierarchies for hyperrobust and standard versions of Ex-learning. On the one hand the Non-Union Theorem (which states that there are Ex-learnable classes whose union is even not BC-learnable) cannot be generalized to hyperrobustly Ex-learnable classes. On the other hand there is a class which can be hyperrobustly Ex-learned by a team of two learners but not by a single one.

Finally Section 8 gives some sample result from applying robustness notions to the scenario where a learner receives, in addition to the data on the function to be learned, also the data on a related function.

2. Function Identification

The basic learning model of inductive inference was introduced by Gold [11]: a class S of recursive functions is learnable iff there is a partial recursive machine M which maps strings of natural numbers to numbers such that

- Given any function $f \in S$, M is defined on input $f(0)f(1)\dots f(n)$ and outputs a number e_n ;
- Almost all e_n are the same number e ;
- The number e is an index for f , that is, $\varphi_e = f$.

Here the indices are chosen with respect to any fixed acceptable numbering φ . It does not matter which acceptable numbering is chosen since all acceptable numberings can be translated into another via total recursive functions. Examples of acceptable numberings are the numbering of all Turing machines and the numbering of all those programs in common programming languages with variables using natural numbers but not having a “maxint-bound” and using exactly one input and at most one output. Note that an acceptable numbering always contains also partial recursive functions; following a common convention, a “recursive function” is a function φ_e which is total and a “partial recursive function” is any function φ_e , whether total or not. So “partial recursive” indicates that the function is permitted to be undefined somewhere but it is not

required to be undefined somewhere.

Gold [11] showed that the class of all total recursive functions is not learnable, that is, for every recursive learner M there is a recursive function f not learned by M . So one of the central topics of inductive inference became the search for the boundaries of learning and for further learning criteria, which were compared to the already existing ones. The main criteria are the following.

Definition 2.1 (Learning Criteria) [1, 2, 3, 8, 11, 15, 17, 19, 28, 29]. Let S be a class of total recursive functions. S is learnable with respect to one of the following criteria iff there is a partial recursive machine M with the corresponding properties.

Ex-learning: if $f \in S$ then $M(f(0)f(1)\dots f(n))$ is defined for all n and there is an index e of f such that $M(f(0)f(1)\dots f(n)) = e$ for almost all n .

Ex_k-learning: if $f \in S$ then there is an n such that $M(f(0)f(1)\dots f(m))$ is defined iff $m > n$ and there are at most k numbers $m' > n$ with $M(f(0)f(1)\dots f(m')) \neq M(f(0)f(1)\dots f(m'+1))$. That is, there are at most k mind changes by M during the process of learning f .

Confident learning: M is a total Ex-learner for S and M converges on every, even non-recursive, function f . That is, given any total f , there is an index e with $M(f(0)f(1)\dots f(n)) = e$ for almost all n .

Consistent learning: M Ex-learns S and, for every $f \in S$ and for every n , $M(f(0)f(1)\dots f(n))$ is defined and outputs a program e such that, for $m = 0, 1, \dots, n$, $\varphi_e(m) \downarrow = f(m)$.

Conforming learning: M Ex-learns S and, for every $f \in S$ and for every n , $M(f(0)f(1)\dots f(n))$ is defined and outputs a program e such that, for $m = 0, 1, \dots, n$, either $\varphi_e(m) \uparrow$ or $\varphi_e(m) \downarrow = f(m)$.

Reliable learning: M is a total Ex-learner for S and whenever M converges on some function f to an index e , then e is an index for f ; note that reliable learners diverge on all nonrecursive functions.

BC-learning: if $f \in S$ then $M(f(0)f(1)\dots f(n))$ is defined for all n and $M(f(0)f(1)\dots f(n))$ is a program for f for almost all n ; note that these programs can be different for all n .

Definition 2.2 [24]. Let I be one of the notions from Definition 2.1. Then S is *I-learnable by a team of k learners* iff there are k learners M_1, M_2, \dots, M_k such that every $f \in S$ is *I-learned* by at least one of the learners M_1, M_2, \dots, M_k .

Note that one can postulate without loss of generality for the criteria of Ex-learning, confident learning and BC-learning that the learner M is even a total recursive function on the strings of natural numbers. As

one can make any hypothesis of a BC-learner to be consistent (and thus also conforming) with the data seen so far without loosing any learning power, a BC-learner is without loss of generality already consistent and conforming and therefore it does not make sense to introduce these notions for the BC-case. Furthermore, there are also alternative definitions of reliable learning which are not equivalent to the above one, see for example [20] for a brief discussion of these criteria in the context of robust and hyperrobust learning.

Case and Smith [8] published the following characterizations of a restricted version of Ex-learning and of BC-learning, the first characterization is due to Leeuwen and Bärzdiņš and the second is due to Podnieks [23].

Theorem 2.3 [8, 23]. *A class S is Ex-learnable via an M outputting (on all inputs) only indices of total functions iff there is a recursive N such that, for all $f \in S$ and for almost all n , $N(f(0)f(1)\dots f(n)) = f(n+1)$.*

A class S is BC-learnable iff there is a partial recursive N such that, for all $f \in S$ and for almost all n , $N(f(0)f(1)\dots f(n)) \downarrow = f(n+1)$. Note that even for any fixed $f \in S$ the predictor N is permitted to be undefined on finitely many inputs of the form $f(0)f(1)\dots f(n)$.

A class S is recursively enumerable iff $S = \{f_0, f_1, \dots\}$ for a family of total functions such that the mapping $e, x \rightarrow f_e(x)$ is recursive. A class S is defined to be in *Num* iff it is a subclass of a recursively enumerable class. These classes have the straight forward learning algorithm given below.

Remark 2.4 (Learning by Enumeration) [11, 23]. *Every class in Num is Ex-learnable by the following algorithm, called learning by enumeration: Given a recursive enumeration f_0, f_1, \dots of functions such that $S \subseteq \{f_0, f_1, \dots\}$, the learner searches on any input the first f_e which is consistent with the data seen so far and translates the e to an index of the given acceptable numbering. More precisely, a recursive function h is chosen such that $\varphi_{h(e')} = f_{e'}$ for all e' and the learner outputs, for any given input $f(0)f(1)\dots f(n)$, the value $h(e)$ where e is the first index satisfying $(\forall m \leq n) [f_e(m) = f(m)]$.*

Note that every class S in Num has a recursively enumerable superclass $S' = \{f_0, f_1, \dots\}$ which is also dense. Here a class S' is called *dense* iff every data σ is extended by some $f \in S'$, denoted $\sigma \preceq f$: $(\forall x \in \text{domain}(\sigma)) [\sigma(x) = f(x)]$. If learning by enumeration is based on this

dense superclass S' , then the resulting algorithm is total.

There are various learning-theoretic characterizations for S being in Num: S is Ex-learnable with a machine outputting only indices of total functions, S is learnable by enumeration, S has a total-recursive machine predicting every function in S almost always. This justifies to look upon Num as being a learning criterion and the study of robust learning was motivated by Bārzdiņš' question whether Num captures all natural learnable classes.

3. Bārzdiņš' Conjecture

Self-reference combined with coding-tricks gives an elegant way to prove many separation results in inductive inference as shown in the following example, in particular for the case of the criterion SD .

Example 3.1 (Classes SD and ASD). The following classes obtained by self-referential coding separate some of the learning criteria from Definition 2.1.

- The class $SD = \{f : \varphi_{f(0)} = f\}$ of all self-describing functions can be Ex_0 -learned and can be consistently learned. Thus SD is also conformingly learnable, confidently learnable, Ex-learnable and BC-learnable. But SD cannot be learned reliably and is also not in Num.
- The class $ASD = \{f : \varphi_{f(0)} =^* f\}$ of all almost self-describing functions f (these are computed at almost all x by the index $f(0)$) is BC-learnable but not Ex-learnable.
- The class $\{f : (\exists e)[\varphi_e = f \wedge e = \max\{f(x) : x = 0, 1, \dots\}]\}$ of all functions f bounded by a constant and computed by the maximum of its values is Ex-learnable and therefore also BC-learnable. But it cannot be learned according to any other learning criteria from Definition 2.1.

A proof for the first statement is provided as a sample for proofs using the Recursion Theorem. An Ex_0 -learner for SD just outputs on input $f(0)f(1)\dots f(n)$ the hypothesis $f(0)$. An Ex_0 -learner is obviously also an Ex-learner, BC-learner and confident learner as it makes at most one hypothesis. Consistency follows from the fact that the unique guess $f(0)$ is correct whenever the function f is in SD .

To prove that SD is not in Num, assume by way of contradiction that f_0, f_1, \dots is a recursive sequence of recursive functions which contains all the self-describing functions. Let $g(x) = f_x(x+1) + 1$. Now consider

the function h defined by

$$\varphi_{h(e)}(x) = \begin{cases} e & \text{if } x = 0; \\ g(x-1) & \text{if } x > 0. \end{cases}$$

By Kleene's recursion theorem, this function h has a fixed point e and $\varphi_e = \varphi_{h(e)} \in SD$. Let e' be such that $f_{e'} = \varphi_e$. But then $f_{e'}(e' + 1) = g(e') = f_{e'}(e' + 1) + 1$, a contradiction. Thus SD is not in Num.

Based on the results by Blum and Blum [3] and Minicozzi [17], one can indirectly prove that SD is not reliably Ex-learnable. The proof looks at the union $SD \cup AEZ$ where the class AEZ from Example 4.5 below is reliably Ex-learnable. On the one hand, $SD \cup AEZ$ is not Ex-learnable [3]. On the other hand, the union of any two reliably learnable classes is reliably learnable [3, 17] and thus also Ex-learnable. So, SD cannot be reliably learnable.

Case and Smith [8] study in detail many applications of self-reference to separate several learning criteria including the use of ASD to separate BC-learning from Ex-learning.

Although these proofs are elegant, they have the property of coming up with example classes which look very artificial. The main idea is that the coding permits the less pretentious learner to learn while the more pretentious one has to transform the data obtained in a way which would enforce the learner to solve some unsolvable problem. For example a hypothetical Ex-learner for the class ASD would fail because it cannot detect the finitely many undefined places which the index $f(0)$ might have. But the BC-learner would, on input $f(0)f(1)\dots f(n)$, output an index which looks up $f(m)$ in a table for input $m \leq n$ and computes $\varphi_{f(0)}(m)$ for $m > n$. After seeing sufficiently many examples, all places where $\varphi_{f(0)}$ is undefined or wrong are overwritten by entries in the table and from then on, each hypothesis is correct.

Besides the elegance of the proofs, a further reason to use self-describing function classes for separation results is that many candidates S for such a separation have already a self-describing subclass S' which witnesses the same separation but has the learning algorithm already "built in". This principle is just stated for the case of the second criterion being Num.

Proposition 3.2. *If S is closed under finite variants and if S is not in Num, then $S \cap SD$ is in Ex_0 but not in Num.*

Proof. Wiehagen [27] observed that for every function f there is a self-describing function g such that $f(x) = g(x)$ for all $x > 0$. Thus S

satisfies

$$S = \{f : (\exists g \in SD \cap S) (\forall x > 0) [f(x) = g(x)]\}$$

and whenever $SD \cap S$ is in Num, so is S . It follows that $SD \cap S$ is not in Num. But as SD is Ex_0 , so is its subclass $SD \cap S$. ■

Bārzdīņš' idea was to postulate that not only the class S itself but also every transformed class $\Theta(S)$ should be learnable. Taking the class of suitable permitted operators Θ , the resulting notion of robust learning was intended to meet the following requirements.

- Every class S in Num should also be robustly learnable.
- If the learnability of S depends essentially on coding, then $\Theta(S)$ should be non-learnable for a suitable Θ . That is, only natural classes should be robustly learnable.

Bārzdīņš conjectured that these two requirements would be met by permitting all general recursive operators Θ and that only classes in Num are robustly learnable.

Indeed one had also considered many other classes of operators, but none of these classes was able to satisfy both requirements. Clearly all permitted operators have to be recursive, but the question was whether to tolerate or not that some total functions are mapped to partial ones.

If one only requires that functions in S are mapped to total functions, then one could take the class S of all constant functions where the constant c is an index of a total function. If one then takes the operator defined by

$$\Theta(f)(x) = \varphi_{f(0)}(x)$$

one would have that $\Theta(S)$ is the class of all recursive functions which is not learnable. On the other hand, even this weak requirement does not destroy all coding tricks as the following example shows.

Example 3.3 (Class TSD). Jain, Smith and Wiehagen [14] introduced the class

$$TSD = \{f : (\exists e) [f = \varphi_e \wedge (\forall x \leq e) [f(x) = 0]]\}$$

of topologically self-describing functions. This class satisfies the following: Let Θ be a recursive operator mapping every function in TSD to a total one. Then $\Theta(TSD)$ is Ex -learnable as sketched below.

The class TSD contains 0^∞ , so the learner conjectures $\Theta(0^\infty)$ until some x is found with $\Theta(f)(x) \neq \Theta(0^\infty)(x)$. Now one can compute the

maximum argument y of the function 0^∞ evaluated by Θ in order to compute $\Theta(0^\infty)(x)$. It follows that $f(z) \neq 0$ for some $z \leq y$. Therefore, provided that $f \in S$, it holds that $f = \varphi_e$ for some $e \leq y$. From this information one can compute an upper bound $b_{\Theta,y}$ for the minimal index of $\Theta(f)$ and having this bound it is possible to find a program for $\Theta(f)$ in the limit.

It follows that it does not give any advantage to consider operators which fail to map some total (not necessarily recursive) functions to partial functions. So the main goals for a suitable notion of robust learning became the following one: classes in Num should be mapped to classes in Num and the concept should be as easy as possible, as the easiest concept is often the most natural one among a choice of several acceptable concepts to formalize a notion. Thus, one can define robust learning as below. Here a general recursive operator is an operator which is recursive and which maps every total function to a total one.

Definition 3.4 (Robust Learning) [31]. Fix a learning criterion I from Definition 2.1. A class S is robustly learnable with respect to the criterion I iff for every general recursive operator Θ the class $\Theta(S)$ is learnable with respect to the criterion I .

Since one can choose Θ to be the identity mapping every f to itself, every robustly learnable class is also learnable (for any criterion I of learning considered). As already mentioned in Example 3.3, the class TSD is robustly Ex-learnable but not in Num, therefore Bärzdiņš' Conjecture is refuted.

Theorem 3.5 [10, 14]. *There is a robustly Ex-learnable class which is not in Num.*

Furthermore, one can ask whether all naturally defined Ex-learnable classes are also robustly Ex-learnable. Blum and Blum [3] considered the following class which on the one hand has a natural definition but on the other hand is not robustly Ex-learnable in the case that the complexity measure used in the definition is a interpolating complexity measure. Recall that a complexity measure Φ associated to the given acceptable numbering φ is defined by the following properties.

- $\Phi_e(x)$ is defined iff $\varphi_e(x)$ is defined.
- The set $\{(e, x, t) : \Phi_e(x) \downarrow \leq t\}$ is recursive.

A complexity measure Φ is interpolating iff one can find for every i and every string σ a j such that $\Phi_j(x) = \sigma(x)$ for $x \in \text{domain}(\sigma)$ and $\Phi_j(x) = \Phi_i(x)$ for all other x .

Example 3.6 (Class BB) [25]. Let Φ be an interpolating complexity measure for the acceptable numbering φ . Using that $\Phi_e(e) < \infty$ iff e is in the diagonal halting problem K , Blum and Blum [3] defined the following approximations to K :

$$\psi_e(x) = \begin{cases} 1 & \text{if } \Phi_x(x) \leq \Phi_e(x) \text{ and } \Phi_e(x) < \infty; \\ 0 & \text{if } \Phi_x(x) > \Phi_e(x) \text{ and } \Phi_e(x) < \infty; \\ \uparrow & \text{if } \Phi_e(x) = \infty. \end{cases}$$

Note that ψ_e is total whenever Φ_e is total. Now they defined the class

$$BB = \{\psi_e : \Phi_e \text{ is total and increasing}\}$$

and showed that this class is Ex-learnable. Stephan and Zeugmann [25] reinvestigated the class and showed that BB is neither robustly Ex-learnable nor in Num.

One of the main topics of the research on robust learning was also the question whether Bärzdiņš' Conjecture holds for further criteria of learning, namely whether for a given criterion I every class learnable with respect to I is also in Num. Zeugmann [31] published Bärzdiņš' Conjecture and showed as an intermediate result that the notion of reliable learning satisfies this conjecture. The notion of reliable learning was introduced by Minicozzi [17] and intended to model that the learner is never misleading in the limit as it either converges to a correct program or diverges by outputting infinitely many different hypotheses.

Theorem 3.7 [31]. *If a class S is robustly reliably learnable then S is in Num.*

Proof. Let T be an infinite binary recursive tree without infinite recursive branches and let $\sigma_0, \sigma_1, \dots$ be a recursive enumeration of all of its leaves. Now one applies the general recursive operator Θ to S where Θ is given by

$$\Theta(f) = \sigma_{f(0)}\sigma_{f(1)}\sigma_{f(2)} \dots$$

which maps f to that $\{0, 1\}$ -valued functions whose course of values is given by the infinite string $\sigma_{f(0)}\sigma_{f(1)}\sigma_{f(2)} \dots$ and considers any reliable learner M for $\Theta(S)$. Recall that M is required to be total. Furthermore, without loss of generality, M is increasing, that is, whenever $M(\tau) \neq M(\eta)$ and $\tau \preceq \eta$, then $M(\tau) < M(\eta)$. Note that this property can

be obtained by using a padding function which is a recursive function p such that $p(e, n)$ is an index of the same function as e and satisfies $p(e, n) > n$. Now consider uniformly recursive binary trees

$$U_\tau = \{\eta \in \{0, 1\}^* : \tau \preceq \eta \wedge M(\tau) = M(\eta)\}$$

where τ ranges over all strings in $\{0, 1\}^*$. The set of all τ where U_τ is finite is recursively enumerable since whenever U_τ is finite then there is a level $l > |\tau|$ such that $U_\tau \cap \{0, 1\}^l = \emptyset$. Thus one can define the following functions g_τ by taking the value of that case which is found first to hold:

$$g_\tau(x) = \begin{cases} \varphi_{M(\tau)}(x) & \text{if } \varphi_{M(\tau)}(x) \downarrow \leq 1; \\ 0 & \text{if } U_\tau \text{ is finite.} \end{cases}$$

Every function g_τ is total. If U_τ is finite then the second condition guarantees g_τ to terminate. If U_τ is infinite then U_τ has an infinite branch g by König's Lemma. M converges on this infinite branch to $M(\tau)$ and since M is reliable, $M(\tau)$ is a program for this total function g . Furthermore, whenever M learns some function g , then there is a $\tau \preceq g$ such that $M(\tau)$ is already the final value for g and g is an infinite branch of U_τ , so $g = g_\tau$ for this τ . For every g_τ , consider the function f_τ where

$$f_\tau(x) = y_x \Leftrightarrow (\exists y_0, y_1, \dots, y_{x-1}) [\sigma_{y_0} \sigma_{y_1} \dots \sigma_{y_x} \preceq g_\tau].$$

Note that the definition cannot be ambiguous since $\sigma_i \not\preceq \sigma_j$ whenever $i \neq j$. Furthermore, T has no infinite recursive branch and thus every recursive function g_τ cannot be on any shifted version σT of the tree T . Thus one can show inductively that g_τ is not on $\sigma_{y_0} \sigma_{y_1} \dots \sigma_{y_x} T$ and therefore find an y_{x+1} with $\sigma_{y_0} \sigma_{y_1} \dots \sigma_{y_x} \sigma_{y_{x+1}} \preceq g_\tau$. One can furthermore verify that this process can be done uniformly for all g_τ and that thus the class S' of all f_τ is a recursively enumerable family. Since for every $f \in S$, M learns $\Theta(f)$, every $f \in S$ satisfies $\Theta(f) = g_\tau$ for some $\tau \in \{0, 1\}^*$. Thus $f = f_\tau$ for this τ . Thus $S \subseteq S'$ and S is in Num. ■

4. Uniformly Robust Learning

Quite often it may be useful to consider not only whether one can robustly identify a class, but whether one can effectively get a machine to learn every “transformed” class, using the description of the transformation. For example, if one can learn a certain type of geometric figure, one might expect to be able to learn its transformations via some operations such as rotation, scaling and so on, effectively from the parameters of the transformations. This motivates the consideration of uniform robust

learning.

In the context of uniform robust learning this means that the images (under general recursive operators) of S are not only learnable, but one can effectively find a machine to learn them. For this strengthened version of robust learning, one looks into two directions: (I) Are there nontrivial examples which are uniformly robustly learnable with respect to a given criterion I ? (II) Is uniformly robust learning with respect to I different from classical robust learning with respect to I ? Before investigating this, the formal definition is given.

Definition 4.1 [14]. Fix a learning criterion I from Definition 2.1. A class S is uniformly robustly learnable with respect to the criterion I iff there exists a recursive function g such that, for all e such that Θ_e is general recursive, $M_{g(e)}$ learns $\Theta_e(S)$ with respect to the criterion I .

Remark 4.2 [14]. *For the notion of learning with a constant bound on the number of mind changes, uniformly robust learning is very restrictive. If S is uniformly robustly Ex_k -learnable then S is finite and contains less than 2^{k+1} functions. In contrast to this, a class is robustly Ex_0 -learnable iff it is finite and some infinite class is even robustly Ex_1 -learnable.*

Remark 4.3 [6]. *The class of topologically self-describing functions TSD from Example 3.3 is uniformly robustly confidently learnable and thus also uniformly robustly Ex -learnable. But TSD is not in Num.*

So one knows that the uniformly robust version of confident learning is not trivial. The following result of Case, Jain, Stephan and Wiehagen [6] separates the uniform robust and robust versions of confident learning, this separation is much more involved than the previous results and the proof is therefore omitted.

Theorem 4.4 [6]. *There is a class S which is robustly confidently learnable but not uniformly robustly confidently learnable.*

Of course, the most interesting question is whether the robust and uniform robust versions of Ex -learning and BC -learning are different. Indeed, Case, Jain, Stephan and Wiehagen [6] constructed a class S which separates both notions simultaneously. Before constructing that class, the below variant of the Non-Union Theorem [2, 3] is given as it is a tool used in the proof.

Example 4.5 (Class AEZ). Let

$$AEZ = \{f : (\forall^\infty x) [f(x) = 0]\}$$

denote the class of all almost everywhere zero functions. This class is an example of a dense class in Num.

Example 4.6 (Non-Union Theorem). Let AEZ be as above and let TSD be the class of all topological self-describing functions from Example 3.1.

Both classes, AEZ and TSD , are uniformly robust Ex-learnable but their union $AEZ \cup TSD$ is not BC-learnable [2, 3, 14]. Therefore the Non-Union Theorem extends to the notions of robust and uniformly robust Ex-learning and BC-learning.

Concerning team-learning as defined in Definition 2.2, it should be noted that $AEZ \cup TSD$ is an example of a class robustly learnable by a team of two Ex-learners but not being in Ex.

Theorem 4.7 [6]. *There is a class S which is robustly Ex-learnable but not uniformly robustly BC-learnable. So the notions of robust Ex-learning and robust BC-learning are more general than their uniformly robust counterparts.*

Proof. A more detailed proof is found in the paper of Case, Jain, Stephan and Wiehagen [6].

One first constructs finite functions σ_x for all x such that one can compute the σ_x as strings in the limit from x . Then one takes a sequence a_0, a_1, \dots and chooses, for each n , finitely many recursive functions f extending σ_{a_n} and puts these functions into S . The basic idea is to take these functions in an adversary manner for uniformly robust BC-learners, while on the other hand the classes remain “almost uniformly Ex-learnable” which means that one can take an uniform Ex-learner which learns all but finitely many functions in $\Theta_e(S)$ and then makes a non-uniform upgrade in order to cover these finitely many functions as well.

The technical part in the construction of S is the inductive construction of the finite functions σ_x . The goal is that, for all $e \leq x$, Θ_e either maps all or no functions extending σ_x to $\Theta_e(0^\infty)$. The σ_x are constructed in stages s as follows.

- One uses two auxiliary partial-recursive two-place functions ψ, ϑ which are initialized as being undefined everywhere at the beginning and are used for book-keeping which Θ_e have already been addressed. Furthermore, $\sigma_{x,0} = 0^x 1$ before stage 0.
- At stage s : If there is an $e \leq x$, $y \leq s$ and $\tau \succeq \sigma_{x,s}$ such that $\tau \in \{0, 1\}^s$, $\psi(e, x)$ is still undefined and the computations $\Theta_e(\tau)(y)$, $\Theta_e(0^\infty)(y)$ converge within s steps to different values,

- Then take least such e and let $\psi(e, x) = y$ and $\sigma_{x,s+1} = \tau$ for corresponding y and τ . Furthermore let $\vartheta(e, x)$ be the largest argument z of the function 0^∞ queried during the computation of $\Theta_e(0^\infty)(\psi(e, x))$.
- Else all values $\psi(e, x)$ and $\vartheta(e, x)$ remain unchanged and also $\sigma_{x,s+1} = \sigma_{x,s}$.

Now the sequence of the a_n is defined inductively by letting $a_0 = 0$ and letting a_{n+1} be the sum of $a_n + 1$ and all values $\vartheta(e, x)$ where $e, x \leq a_n$ and $\vartheta(e, x)$ is defined. This is a finite sum and so a_{n+1} exists.

Now let h be a function dominating every total recursive function. For every n there is an operator, whose index is called e_n , doing the following.

$$\Theta_{e_n}(f) = \begin{cases} g & \text{if } f = \sigma_{a_{2n}} \cdot g \text{ or } f = \sigma_{a_{2n+1}} \cdot g \text{ for some } g; \\ 0^\infty & \text{otherwise.} \end{cases}$$

For every $e' \leq h(e_n)$ there is by the Non-Union Theorem a function $g_{n,e'} \in AEZ \cup TSD$ which the e' -th learner $M_{e'}$ does not BC-learn. Now put for each n and each $e' \leq h(e_n)$ the following function into S :

- If $g_{n,e'} \in AEZ$ then put $\sigma_{a_{2n}} \cdot g_{n,e'}$ into S ;
- If $g_{n,e'} \in TSD$ then put $\sigma_{a_{2n+1}} \cdot g_{n,e'}$ into S .

In the following it is shown that the resulting class is robustly Ex-learnable but not uniformly robustly BC-learnable.

Claim 1. S is not uniformly robust BC-learnable.

For every recursive function h' there is an n such that $h'(e_n) < h(e_n)$. It follows from the construction that $M_{h'(e_n)}$ does not BC-learn $g_{n,h'(e_n)}$ and that this function is in $\Theta_{e_n}(S)$. So there is no recursive function h' witnessing that S would be uniformly robustly BC-learnable, hence non-learnability is established.

Claim 2. The class $\Theta_e(S)$ is Ex-learnable whenever Θ_e is a general recursive operator.

Let Θ_e be a given general recursive operator and let $f \in S$ be a function such that $\Theta_e(f)$ should be learned.

Let $\{f_0, f_1, \dots, f_k\}$ be the set of all $\Theta_e(f')$ such that either $f' = 0^\infty$ or $f' \in S \wedge f' \succeq \sigma_{a_n}$ for some $n < e$. Finding this set is the non-uniform part of the learning algorithm. The learner searches for the least l such that f_l is consistent with the data seen so far and if this finite search terminates the learner outputs a hypothesis for f_l .

Otherwise, none of the functions $\{f_0, f_1, \dots, f_k\}$ is consistent with

the input. Let n be the number with $\sigma_{a_n} \preceq f$. Note that $n \geq e$, $\psi(a_n, e)$ is defined and $\Theta_e(f)(\psi(a_n, e)) \neq \Theta_e(0^\infty)(\psi(a_n, e))$ as $\Theta_e(f) \notin \{f_0, f_1, \dots, f_k\}$. If $e \leq m < n$ then either $\psi(a_m, e)$ is not defined or $\Theta_e(0^\infty)(\psi(a_m, e))$ is computed relative to 0^∞ by only asking queries up to $\vartheta(a_m, e)$, thus the values $\Theta_e(0^\infty)(\psi(a_m, e))$ and $\Theta_e(f)(\psi(a_m, e))$ coincide. Therefore one can find n in the limit by searching for the first number $m \geq e$ such that $\psi(a_m, e)$ is defined and witnesses that $\Theta_e(f)$ and $\Theta_e(0^\infty)$ are different.

As *AEZ* and *TSD* are uniformly robustly Ex-learnable and as one can compute an index for the general recursive operator mapping g to $\Theta_e(\tau \cdot g)$ from τ , one can, for every input $\Theta_e(f)(0) \Theta_e(f)(1) \dots \Theta_e(f)(m)$, work with approximations $n_m, a_{n,m}$ and $\sigma_{a_{n,m},m}$ to n, a_n, σ_{a_n} and run as a subalgorithm the algorithm to learn $\Theta_e(\sigma_{a_{n,m},m} \cdot \text{AEZ})$ if n_m is even and the algorithm to learn $\Theta_e(\sigma_{a_{n,m},m} \cdot \text{TSD})$ if n_m is odd. As these approximations coincide with the correct values for almost all m , the resulting algorithm coincides with the algorithm to learn $\Theta_e(\sigma_{a_n} \cdot \text{AEZ})$ or $\Theta_e(\sigma_{a_n} \cdot \text{TSD})$, respectively, for almost all m . It follows that $\Theta_e(f)$ is Ex-learned. ■

A further important question is whether the requirement of robustness is so strong that every robustly BC-learnable class is already (non-robustly) Ex-learnable. Jain [12] answered this question negatively, the class *BWT* from Example 6.13 is a witness for the following Theorem.

Theorem 4.8 [12]. *There is a class which is uniformly robustly BC-learnable but not Ex-learnable.*

5. Robust Learning and Consistency

This section deals with the relations between robust learning on the one hand and notions of consistency on the other hand. The main topic of this section is the surprising implication that every uniformly robustly Ex-learnable class is also consistently learnable. After that, a closer look will be given to the investigation of variants of the criterion of consistent learning.

Theorem 5.1 [6]. *If a class is uniformly robustly Ex-learnable, then it is also consistently learnable.*

Proof. Suppose that S is uniformly robustly Ex-learnable and that g is a recursive function witnessing this in the sense that whenever Θ_e is a general recursive operator then $M_{g(e)}$ is an Ex-learner for $\Theta_e(S)$. Without loss of generality one can assume that each $M_{g(e)}$ is total. Using

Kleene's recursion theorem [18], there exists an operator Θ_e such that $\Theta_e(f(0)f(1)\dots f(n))$ is defined inductively for $n = 0, 1, \dots$ by using the preceding string $\sigma = \lambda$ for $n = 0$ and $\sigma = \Theta_e(f(0)f(1)\dots f(n-1))$ for $n > 0$ and by taking $\Theta_e(f(0)f(1)\dots f(n))$ according of the first of the three cases below to apply.

- $\Theta_e(f(0)f(1)\dots f(n)) = \sigma$ if σ is already infinite.
- $\Theta_e(f(0)f(1)\dots f(n)) = \sigma \cdot (f(n)+1) \cdot 0^s$ for the first s such that, for $e' = M_{g(e)}(\sigma \cdot (f(n)+1) \cdot 0^s)$, for all m in the domain of $\sigma \cdot (f(n)+1)$, $\varphi_{e'}(m)$ outputs $(\sigma \cdot (f(n)+1))(m)$ within s steps.
- $\Theta_e(f(0)f(1)\dots f(n)) = \sigma \cdot (f(n)+1) \cdot 0^\infty$ if there is no such s in the previous case.

The dot is used here to denote string-concatenation in places where one might otherwise misunderstand it. Note that $\Theta_e(f(0)f(1)\dots f(n))$ being infinite does not mean that an infinite function is computed in finite time but it means that the values of the function $\Theta_e(f)$ does not depend on the values of f beyond n . It is easy to check that Θ_e is a general recursive operator.

Claim 1. If $f \in S$ then all $\Theta_e(f(0)f(1)\dots f(n))$ are defined according to the second case of the algorithm for Θ_e .

This is proven by induction. Assume that n is given such that all $\Theta_e(f(0)f(1)\dots f(m))$ with $m < n$ are defined according to the second case. Then $\Theta_e(f(0)f(1)\dots f(n))$ cannot be defined according to the first case since the previously defined strings are all finite.

If $\Theta_e(f(0)f(1)\dots f(n))$ would be defined according to the third case, this would produce an infinite string. So $M_{g(e)}$ would have to converge on this string to an index e' since $M_{g(e)}$ Ex-learns this infinite string and this e' would be an index for the function given by this infinite string. So there would be an s such that $M_{g(e)}(\Theta_e(f(0)f(1)\dots f(n-1))(f(n)+1)0^s)$ outputs e' and that the function $\varphi_{e'}$ terminates within s steps on the domain of the string $\Theta_e(f(0)f(1)\dots f(n-1))(f(n)+1)$.

Thus the second case of the algorithm applies in contradiction to the assumption and so for $f \in S$, all strings $\Theta_e(f(0)f(1)\dots f(n))$ are build according to the second case from the preceding strings.

Learner. Now one considers the following (consistent) learner N for S . $N(f(0)f(1)\dots f(n))$ simulates $\Theta_e(f)$ and $M_{g(e)}$. If $\Theta_e(f(0)f(1)\dots f(n))$ is defined by the second case in the definition of Θ_e , then N takes the index e' coming up in that second case and outputs $h(e')$ where h is defined such

$$\varphi_{h(e')}(k) = \varphi_{e'}(x_k) - 1$$

where x_k is the first x such that the set $\{y \leq x : \varphi_{e'}(y) \downarrow > 0\}$ has at least $k + 1$ elements.

Note that N is permitted to be undefined on input not belonging to any $f \in S$. The following Claims 2 and 3 establish that N is a consistent learner for S .

Claim 2. N is consistent, that is, if $f \in S$ then $N(f(0)f(1)\dots f(n))$ is defined and outputs an index $h(e')$ which is defined for $m \leq n$ and outputs the value $f(m)$ there.

The index e' used in the second case of the definition of Θ_e computes a function which extends a string of the form $(f(0) + 1)0^*(f(1) + 1)0^*\dots 0^*(f(n) + 1)$. Thus the function computed by the index $h(e')$ satisfies $\varphi_{h(e')}(m) = f(m)$ for $m = 0, 1, \dots, n$ and the consistency follows.

Claim 3. N Ex-learns every $f \in S$.

As $M_{g(e)}$ converges on $\Theta_e(f)$ to some index e'' , the learner N almost always outputs $h(e'')$. As N is consistent for $f \in S$, it follows that $h(e'')$ extends for infinitely many n the string $f(0)f(1)\dots f(n)$. This implies that $h(e'')$ is an index for f and N is an Ex-learner for f .

Conclusion. Claim 3 shows that N Ex-learns S and Claim 2 shows that all intermediate hypotheses for $f(0)f(1)\dots f(n)$ with $f \in S$ exist and are consistent with the data seen so far. Thus N is a consistent Ex-learner for S . ■

The above proof can even be made effective in the sense that one can construct for every $\Theta_d(S)$ a consistent learner $M_{g'(d)}$ and so one obtains that uniformly robustly Ex-learnable classes are also uniformly robustly consistently learnable. Furthermore, every consistent learner is conforming, so one has the following corollary.

Corollary 5.2. *If S is uniformly robustly Ex-learnable then S is conforming learnable.*

One of the main open questions in the theory of robust learning is whether the preceding result can be generalized to robust Ex-learning.

Problem 5.3. *Is every robustly Ex-learnable class also consistently learnable?*

A further interesting question is to study the relations between robust learning and the variants of consistent and conforming learning. The

most restrictive variants are those of globally conforming and globally consistent learners. These learners have to be total and for every input $y_0 y_1 \dots y_n$ they are defined and output an index e which satisfies for each $x \leq n$ either $\varphi_e(x) \uparrow$ or $\varphi_e(x) \downarrow = y_x$; for the criterion of global consistency the first case never and the second case always applies.

It is easy to adapt the proof of Theorem 3.7 to show that every robustly globally conforming learnable class is in Num. The key idea of this adaptation is the following. Given such a learner M for the $\Theta(S)$ defined there, every U_τ which contains for a given length more than two strings indicates that the corresponding index $M(\tau)$ does not compute a total function. Therefore one can define the g_τ by checking whether there is a unique string and outputting 0 if this fails:

$$g_\tau(x) = \begin{cases} \eta(x) & \text{if } \eta \text{ is the unique string of } U_\tau \text{ of length } |\tau| + x + 1; \\ 0 & \text{if such an } \eta \text{ does not exist or is not unique.} \end{cases}$$

The rest of the proof carries over directly. So one has the following.

Corollary 5.4 [6, 31]. *If S is robustly globally conforming learnable or robustly globally consistently learnable then S is in Num.*

Between these two extremes for consistent learning, there is the case where one adds to the standard definition the requirement that the learner has to be total although it might output inconsistent hypotheses for inputs not belonging to any function in the class S to be learned. A similar version can also be considered for conforming learning. These notions can be separated.

Theorem 5.5 [6]. *There is a class which is uniformly robustly Ex-learnable but which does not have a total conforming learner.*

Theorem 5.6 [6]. *There is a class S and a recursive function g such that $M_{g(e)}$ is a total conforming learner for every class $\Theta_e(S)$ where Θ_e is general recursive but S itself does not have a total consistent learner.*

Theorem 5.7 [6]. *There is a class S and a recursive function g such that $M_{g(e)}$ is a total consistent learner for every class $\Theta_e(S)$ where Θ_e is general recursive but S is not in Num.*

6. Hyperrobust Learning

As even uniformly robust learning does not rule out self-referential coding, Ott and Stephan [20] introduced the notion of hyperrobust learning. The key idea is that the learner has to deal not only with one image $\Theta(S)$

but with a lot of images at the same time without being able to distinguish among these. But one cannot take all general recursive operators at the same time, since every recursive function is the image of the function 0^∞ under a suitable general recursive operator. Therefore Ott and Stephan [20] defined that a class is hyperrobustly learnable iff there is *one* learner learning the image of *every* function in S under *every* primitive recursive operator.

It is shown in Theorem 6.3 below that the hyperrobustly learnable classes remain the same if one takes any larger recursively enumerable class of general recursive operators instead of the one above. Furthermore, hyperrobust learning is compatible to the standard notion of robust learning: if S is hyperrobustly learnable then S is also robustly learnable. Moreover, if S is closed under finite variants then both notions are equivalent. The set $[S]$ of all images of functions in S under primitive recursive operators is dense for every nonempty class S of functions. Thus, hyperrobust learning cannot respect any bounds on the number of mind changes. Therefore, it is not suitable to combine hyperrobust learning with the notions of Ex_k -learning and confident learning. However there are interesting results for Ex -learning, BC -learning and their team variants.

The notion of hyperrobust learning also has a more intuitive motivation: Assume that a learner M can learn all axis-parallel rectangles in the plane. Certainly, one assumes that from M one can build a learner which infers all rotated rectangles in addition. However, clearly one does not want to build, for every different rotation Θ , a learner succeeding just on rectangles mapped by the rotation Θ . Instead, one is interested in a learner which infers every image of any axis-parallel rectangle under *any* rotation Θ . The notion of hyperrobustness reflects this situation by requiring that *one* learning machine M learns every image of the functions in a class S under *all* primitive recursive operators. Now the informal definition is made more precise.

Definition 6.1. An operator Θ is called primitive recursive iff there are primitive recursive functions g_1, g_2 such that

$$(\forall \text{ total } f) [\Theta(f)(x) = g_1(x, f(0)f(1) \dots f(g_2(x)))].$$

Let I be one of the criteria Ex , BC or their team versions.

A class S is *hyperrobustly learnable* with respect to the criterion I if there is one learner M which learns every function in

$$[S] = \{\Theta(f) : \Theta \text{ is primitive recursive and } f \in S\}$$

with respect to the criterion I .

As the criteria Ex and BC permit total learners, learners to witness that $[S]$ is Ex-learnable or BC-learnable will be assumed to be total in the following. Hyperrobust learning satisfies two simple observations.

Fact 6.2. $[S]$ contains all primitive recursive functions since, for every primitive recursive function g , there is a primitive recursive operator Θ which maps every function f to g : $\Theta(f) = g$.

No class S is hyperrobustly learnable with any bound on the number of mind changes, since $[S]$ is dense and dense classes cannot satisfy mind change bounds.

These two facts establish a real difference between hyperrobust learning and robust learning because there are classes of recursive functions which are robustly learnable with at most one mind change [14]. On the other hand, for hyperrobust learning, the notions Ex, BC and their team-variants are the most interesting ones.

The definition of the mapping $S \rightarrow [S]$ and thus the definition of hyperrobustness are based on the class of primitive recursive operators. The decision to choose the class of primitive recursive operators may seem to be just arbitrary and one may wonder how other choices for the class of operators affect the notion of hyperrobustness. The next result justifies the definition: If in the definition of hyperrobust learning, the class of primitive recursive operators is replaced by a larger recursively enumerable class of general recursive operators, then one still gets the same learning notion. So the definition of hyperrobust learning does not depend on the actual choice of the class of operators as long as this class is “sufficiently rich” (for example, if the class contains all primitive recursive operators, or, all polynomial time computable operators). Clearly, if the class of operators contains only the identity operator, then hyperrobust and ordinary Ex-learning coincide and so, “sufficiently rich” is a necessary and natural postulate.

Theorem 6.3 [20]. *If S is hyperrobustly Ex-learnable or hyperrobustly BC-learnable and U is any recursively enumerable class of general recursive operators, then also the class*

$$\{\Theta(f) : f \in S \wedge \Theta \in U\}$$

is Ex-learnable or BC-learnable, respectively.

Corollary 6.4. *If S is hyperrobustly learnable with respect to some criterion I , then S is also robustly learnable with respect to the same criterion I .*

Theorem 6.5 [20]. *If S is closed under finite variants then S is hyperrobustly learnable with respect to a criterion I iff S is robustly learnable with respect to the same criterion I .*

Corollary 6.4 and Theorem 6.5 show that hyperrobust learning is a natural generalization of robust learning: it is equivalent to first taking the closure under all finite variants and then applying a suitable general recursive operator Θ .

Intuitively, the notion of robustness was designed to prevent coding tricks: for example, if $f(2x)$ is a program for f for almost all x , then the general recursive operator mapping f to $f(1)f(3)f(5)\dots$ destroys this coding trick. Such coding tricks are called *numerical* since the self-referential information is directly contained in the numerical values of the function.

Recall that Jain, Smith and Wiehagen [14] showed that the class TSD of the topologically self-describing functions is uniformly robustly Ex-learnable. So these topological coding cannot be destroyed by using a single general recursive operator. However, topological coding is destroyed by adding all finite variants of the functions in S to the class to be learned. Combining these two methods, that is, considering robust learning of classes closed under finite variants, one might hope that no coding tricks are left. Indeed, this hope is confirmed by the following characterization result which shows that the hyperrobustly Ex-learnable classes coincide with the classes in Num. So the next result gives a further learning-theoretic characterization for Num.

Theorem 6.6 [20]. *A class S is hyperrobustly Ex-learnable iff it is in Num.*

Proof. One direction is straightforward: If S is in Num, so is $[S]$. That is, if $S \subseteq \{f_0, f_1, \dots\}$ for an enumeration f_0, f_1, \dots of total functions, then $[S]$ is contained in the enumeration of all $g_{e,e'} = \Psi_e(f_{e'})$, where Ψ_0, Ψ_1, \dots is an enumeration of all primitive recursive operators.

For the converse direction, let M be an Ex-learner for $[S]$. Furthermore, one considers now the operator Θ_e from the proof of Theorem 5.1 where one uses M in place of the $M_{g(e)}$ there. The operator Θ_e constructed there is primitive recursive and as primitive recursive operators are closed under composition, it holds that $\Theta_e([S]) \subseteq [S]$. Thus M learns $\Theta_e([S])$ and one has that $[S]$ is consistently learnable via some N . As $[S]$ is dense, the learner N has to be total and to be consistent on every input. Therefore, N is a globally consistent learner. Furthermore, one can choose the tree T in the proof of Theorem 3.7 such that the resulting Θ is also primitive recursive. As a consequence, $\Theta([S]) \subseteq [S]$ and N is a

consistent Ex-learner for $\Theta([S])$. As a globally consistent learner is also reliable, it follows from the proof of Theorem 3.7 that $\Theta([S])$ and thus also $[S]$ and S are in Num. ■

As hyperrobust Ex-learning is equivalent to Num, it remains to investigate how hyperrobust BC-learning behaves. The next result shows that hyperrobustly BC-learnable classes share one property of classes in Num, namely that they are bounded in the following sense.

Definition 6.7. A class S is bounded if there is a total recursive function g which dominates every $f \in S$:

$$(\forall f \in S) (\exists x) (\forall y \geq x) [f(y) \leq g(y)].$$

Theorem 6.8 [20]. *If S is hyperrobustly BC-learnable, then S is bounded.*

Proof. It is convenient to represent the BC-learner as a machine M which predicts the function to be learned almost everywhere but which may, for each function in the class to be learned, be undefined at finitely many places, see Theorem 2.3.

Now one defines inductively, for every f , the following function $\Theta(f) = \lim_x \sigma_x$ starting with $\sigma_0 = \lambda$ and using $M_s(\sigma_x)$ as a notation for the result of $M(\sigma_x)$ after s computational steps (which is either undefined or $M(\sigma_x)$):

$$\sigma_{x+1} = \begin{cases} \sigma_x 1 & \text{if } M_s(\sigma_x) \downarrow = 0 \text{ for some } s \leq f(0) + f(1) + \dots + f(x); \\ \sigma_x 0 & \text{otherwise.} \end{cases}$$

Since Θ is a primitive recursive operator, M has to infer $\Theta(f)$ for every $f \in S$. But whenever $\Theta(f)(x)$ is 1, then M has made a prediction mistake and so, $\Theta(f)$ takes only finitely often a value different from 0. Since, by Fact 6.2, M has to infer every primitive recursive function, M learns, in particular, all functions of the form $\sigma 0^\infty$. Thus, the following function g is recursive:

$$g(x) = \max\{\min\{s : (\exists t < s) [M_s(\sigma 0^t) \downarrow = 0]\} : \sigma \in \{0, 1\}^x\}.$$

Whenever $f(x) > g(x)$ then one finds within $f(x)$ steps some $t < f(x)$ such that $M(\sigma_x 0^t) \downarrow = 0$. So, if $\sigma_{x+t} = \sigma_x 0^t$, then $\sigma_{x+t+1} = \sigma_x 0^t 1$, in particular $\sigma_{x+t+1} \neq \sigma_x 0^{t+1}$. Thus there is a $y \in \{x, x+1, \dots, x+t\}$ such that $\sigma_{y+1} = \sigma_y 1$. This happens only if $M(\sigma_y) \downarrow = 0$ and so M makes a false prediction on $\Theta(f)$ beyond x . But since M infers $\Theta(f)$, there exist

at most finitely many x with $f(x) > g(x)$. Therefore, g dominates f . Since the construction of g does not depend on the actual choice of f , g dominates every function in S . ■

Ott and Stephan [20] showed that there are classes S not in Num which are hyperrobustly BC-learnable. A major tool in this research is the use of recursively bounded recursive trees [18, I, page 509]; these trees are called *bounded recursive trees* from now on. These trees are a generalization of binary recursive trees: for a bounded recursive tree T one can compute for every $\sigma \in T$ a complete list of the immediate successors in T . In general, this is impossible even if σ has only finitely many successors. But it holds when some recursive function b bounds the size of the set of successors; that is, when $a \leq b(|\sigma|)$ for all $\sigma a \in T$. So, one can define a bounded recursive tree as a recursive function c which associates with every $\sigma \in T$ a finite and explicit list of all nodes $\sigma a \in T$. If c is primitive recursive then T is called a bounded primitive recursive tree.

Recall that a learning machine M is said to be *reliable* if, for any input function f , M either converges to a correct program for f or outputs infinitely often a signal for divergence, which, in the case of Ex-learning, can simply be a mind change. Producing semantic mind changes alone is not sufficient to get a reliable version of BC-learning that differs from ordinary BC, as the following fact shows. This fact is based on two observations: First, behavioural correct learners can be made consistent. That is, the new consistent learner outputs for every input a hypothesis which is correct on the data seen so far [2, 9]. Second, consistent learners either converge semantically to the desired function or make infinitely many semantic mind changes.

Fact 6.9. *For every BC-learnable class S there is a BC-learner for S which, on any total function f , either converges semantically to programs for f or makes infinitely many semantic mind changes.*

Since Fact 6.9 states that BC-learners can be made semantically divergent on functions not learned, the analogue of reliable learning for BC must signal divergence more explicitly. A suitable definition is the following: The reliable BC-learner indicates divergence either by outputting a special value like “?”, or, by making a definitely wrong prediction where the underlying BC-learner is given by a partial recursive predictor as in Theorem 2.3.

Definition 6.10. *M is a reliable BC-learner if, for every total function f , either M BC-learns f by predicting almost always the correct value (that is, for almost all x , $M(f(0)f(1)\dots f(x)) \downarrow = f(x+1)$) or M diverges*

on f by outputting infinitely often either “?” or a defined but wrong prediction.

Zeugmann [31] observed that robustly reliably Ex-learnable classes are just those in Num. Together with Theorem 6.6, one obtains the following equivalence.

Fact 6.11. *For a bounded class S the following three statements are equivalent.*

- S is in Num;
- S is reliably Ex-learnable;
- S is hyperrobustly Ex-learnable.

The central question of the following results in this section is: to what extent can the equivalence of the statements above be transferred to BC? The next characterization of bounded reliably BC-learnable classes is an important tool to attack this question.

Theorem 6.12 [20]. *A bounded class S is reliably BC-learnable iff there is a recursively enumerable family T_0, T_1, \dots of bounded recursive trees such that every tree has only finitely many infinite branches and every $f \in S$ is an infinite branch of such a tree.*

Proof. (\Rightarrow): Assume that g bounds S and M is a partial recursive predictor for S which in addition signals infinitely often divergence on every function f which M does not learn. Now let the tree T_σ contain all prefixes of σ plus all $\eta \succeq \sigma$ such that

- $\eta(x) \leq g(x)$ for all $x \in \text{domain}(\eta) - \text{domain}(\sigma)$ and
- there are no τ, a with $\sigma \preceq \tau a \prec \eta$ and $M_{|\eta|}(\tau) \downarrow \neq a$,

where, of course, the special symbol “?” is different from a .

Clearly, the trees T_σ form a recursively enumerable family of trees bounded by g . Assume now that T_σ has infinitely many infinite branches. As a consequence of König’s Lemma and the fact that T_σ is finitely branching, there is an infinite branch f which is not isolated. So for every x there is an $y > x$ and a further infinite branch h of T_σ such that f and h both agree on $0, 1, \dots, y$ but differ on $y + 1$. Then it cannot happen that $M(f(0)f(1)\dots f(y))$ predicts $f(y + 1)$ or $h(y + 1)$ since otherwise either f or h would not be an infinite branch of T_σ . Furthermore, M does also not signal divergence for any input $f(0)f(1)\dots f(x)$ with $x \geq |\sigma|$

since then f could not be an infinite branch of T_σ . This contradicts the assumption that M is a reliable BC-learner and therefore no tree T_σ has infinitely many infinite branches.

For every $f \in S$, there is a prefix $\sigma \preceq f$ such that M predicts f correctly after seeing σ and all x with $f(x) > g(x)$ are in $\text{domain}(\sigma)$. Then it follows from the definition that f is an infinite branch of T_σ . Direction (\Rightarrow) is completed.

(\Leftarrow) : Let T_0, T_1, \dots be a recursively enumerable family of bounded recursive trees such that every tree has only finitely many infinite branches and every function in S is a branch of such a tree. Without loss of generality, the family is dense in the sense that for every σ there is a tree containing σ . This can be achieved by adding all finite trees of the form $\{\tau : \tau \preceq \sigma\}$ to the list. The new family is still recursively enumerable and the class of functions on trees in the family remains the same. Let $T[\tau]$ denote all nodes of the tree T which are comparable to τ . Now the reliable BC-learner works as follows:

- $M(\sigma)$ finds the first tree T_e with $\sigma \in T_e$.
- If there is a recent change of the tree, that is, if there is $e' < e$ with $\tau \in T_{e'}$ for all $\tau \prec \sigma$, then $M(\sigma) = ?$ in order to signal divergence.
- Otherwise $M(\sigma)$ searches for an a such that $T_e[\sigma b]$ is finite for all $b \neq a$ and $M(\sigma) = a$, if such an a is found.

The first step of the algorithm is well-defined since every σ is a node of some tree T_e .

If f is not an infinite branch of any tree T_e , then, during the inference of f , M signals infinitely often divergence, since M has to change the trees infinitely often.

If f is an infinite branch of some tree then there is a first such tree T_e in the enumeration. For sufficiently large $\sigma = f(0)f(1)\dots f(x)$, f is the only infinite branch of $T_e[\sigma]$ and $\sigma \notin T_{e'}$ for any $e' < e$. Now $f(x+1)$ is the unique value a with $T_e[\sigma a]$ being infinite. Since the trees T_e are uniformly bounded recursive trees, a suitable search algorithm finds the value $f(x+1)$. Therefore, M correctly predicts f almost everywhere; that is, $M(f(0)f(1)\dots f(x)) \downarrow = f(x+1)$ for almost all x .

So, for every function f , M either BC-learns f (in the prediction model) or M signals infinitely often divergence. ■

Example 6.13 (Class BWT). Case, Kaufmann, Kinber and Kummer [7] showed that there is a recursively enumerable family $\{T_0, T_1, \dots\}$ of binary recursive trees of width two such that the class

$$BWT = \{f : (\exists e) [f \text{ is infinite branch of } T_e]\}$$

is not Ex-learnable. This class BWT is reliably BC-learnable. So BWT witnesses that for bounded classes, the concept of reliable BC-learning is also a proper generalization of Num.

As one can show that BWT is even uniformly robustly BC-learnable, one gets an alternative proof for Jain's separation of uniformly robust BC-learning from Ex-learning in Theorem 4.8.

Together with the next result one obtains that the three notions from Fact 6.11 become all different for BC: Num is properly included in bounded reliable BC, which is properly included in hyperrobust BC.

Theorem 6.14 [20]. *If S is a bounded and reliably BC-learnable class then S is also hyperrobustly BC-learnable. But the converse implication does not hold.*

So, the main theorems of this section showed that there is a class S which is hyperrobustly BC-learnable but not hyperrobustly Ex-learnable.

Theorem 6.15 [20]. *There is a hyperrobustly BC-learnable class which is not hyperrobustly Ex-learnable.*

Zeugmann [31] showed that a class S is robustly reliably Ex-learnable iff S is in Num. One can deduce from it that S is robustly reliably Ex-learnable iff S is bounded. Zeugmann's diagonalization strategy for this result can be transferred to the case where reliably BC-learnable classes are considered. That is, an unbounded reliably BC-learnable class is not robustly reliably learnable.

Theorem 6.16 [20]. *A reliably BC-learnable class S is also robustly reliably learnable iff S is bounded.*

7. Team-Learning and the Union-Theorem

Bärzdiņš [2, 3] showed that there are explanatorily learnable classes S_1, S_2 such that their union is not explanatorily learnable. This result can be easily generalized to the fact that there are unions of $n + 1$ learnable classes which are not contained in the union of n learnable classes [24]. Pitt and Smith [21, 22] showed that these unions can also be characterized in terms of probabilistic learners and teams of learners. That is, the following statements are equivalent for Ex-learning as well as for BC-learning.

- S is contained in the union of n learnable classes.

- Some probabilistic machine learns S with some probability p where $p > \frac{1}{n+1}$.
- A (h, k) -team with $\frac{h}{k} > \frac{1}{n+1}$ learns S in the sense that there are k learners such that, for every $f \in S$, at least h of them learn f .

Note that the probability and the fraction $\frac{h}{k}$ of successful machines in the team have to be greater than $\frac{1}{n+1}$, since a team of $k = h \cdot (n + 1)$ learners, where h learners have to succeed, can already infer the union of $n + 1$ learnable classes: the first h learners follow the algorithm to learn S_1 , the second h learners follow the algorithm to learn S_2 , ..., the last h learners follow the algorithm to learn S_{n+1} . Recall that Example 4.5 stated that the classes AEZ and TSD are both uniformly robustly Ex-learnable while their union $AEZ \cup TSD$ is not BC-learnable. Thus the Non-Union Theorem holds for the criteria of robust and uniformly robust Ex-learning and BC-learning.

For hyperrobust Ex-learning, one can show that this connection between team-learning on the one side and unions on the other side no longer holds. The hyperrobustly Ex-learnable classes are closed under union but teams of $n + 1$ hyperrobust Ex-learners are more powerful than teams of n learners. An intuitive explanation for this fact is that if $[S_1 \cup S_2]$ needs a team of two Ex-learners then so does $[S_1]$ or $[S_2]$. So, the closure operation does not permit to split a class of functions into two classes which are really easier to learn.

Fact 7.1. *If S_1 and S_2 are hyperrobustly Ex-learnable, so is $S_1 \cup S_2$.*

The result follows from the equivalence of hyperrobust Ex-learning and Num and from the fact that Num is closed under union. Note that the corresponding question for BC-learning is open.

Problem 7.2. *If S_1 and S_2 are hyperrobustly BC-learnable, is $S_1 \cup S_2$ also hyperrobustly BC-learnable?*

The next result establishes that the team hierarchies for hyperrobust Ex-learning and hyperrobust BC-learning are proper. This stands, for the Ex-case, in contrast to the collapse of the union-hierarchy.

Theorem 7.3 [20]. *The team hierarchies for hyperrobust Ex-learning and hyperrobust BC-learning are proper.*

Proof. This proof needs the notion of the rank of a tree T which measures the embeddability of complete binary trees into T as follows:

The *rank of a tree* T is the maximal k for which there is a finite function g with domain $\bigcup_{k' \leq k} \{0, 1\}^{k'}$ such that the range of g is a subset of T and $g(\sigma) \preceq g(\tau) \Leftrightarrow \sigma \preceq \tau$ for all strings σ, τ in the domain of g . If there is, for every k , such a function g then the rank of T is ∞ .

Let S_k be the set of all functions which are infinite branches of some bounded primitive recursive tree of rank up to k ; there is a uniformly recursive enumeration of these trees, in the sense that one can not only check whether $\sigma \in T_e$ but also compute an explicit list of the immediate successors of σ in T_e if $\sigma \in T_e$.

Given f , the learning algorithm first finds (in the limit) a tree T such that f is an infinite branch of T . Having found this tree T , one uses the algorithm of Case, Kaufmann, Kinber and Kummer [7]. Knowing an index of the tree and having a primitive recursive function majorizing all infinite branches, their algorithm learns the function by a team of $k + 1$ Ex-learners or k BC-learners, respectively. This team-size is optimal. The class S_k is closed; that is, $[S_k] = S_k$. So, it follows that S_k is learnable by a team of hyperrobust learners of size k (BC) and $k + 1$ (Ex), respectively, but not by a smaller team. ■

Note that the above classes S_k are also uniformly robustly Ex-learnable by teams of $k + 1$ Ex-learners and uniformly robustly BC-learnable by teams of k BC-learners. So these classes witness that the hierarchies are also proper for robust and uniformly robust Ex-learning and BC-learning.

Furthermore, for hyperrobust Ex-learning, one can even show that there exists a proper team hierarchy *within* the class of all hyperrobustly BC-learnable functions. The n -th level of this hierarchy is given by the class of all infinite branches of bounded primitive recursive trees of width up to n ; that is, of trees which have in every depth at most n nodes.

8. Learning Aided by Context

Learning aided by context means that the learner receives not only the data of the function to be learned but also the data of some further $f \in S$ and might try to exploit this further data-source for the learning process. Case, Jain, Ott, Sharma and Stephan [5] investigated robust notions of learning aided by context. The straight forward definition would be that S is Ex-learnable aided by context iff for every $f \in S$ and every $g \in S - \{f\}$ the learner M converges on input f, g to an index for f . But it turned out that this notion coincides with standard Ex-learning. Therefore the alternative version where the context is selected is the more interesting one.

Definition 8.1. A class S is learnable aided by context iff there is a mapping assigning to every $f \in S$ a context $g_f \in S$ and S has a total Ex-learner M (taking as input, two functions) such that for all $f \in S$, M converges on input f and context g_f to an index e for f :

$$(\forall^\infty n) [M(f(0)f(1)\dots f(n), g_f(0)g_f(1)\dots g_f(n)) = e \wedge f = \varphi_e].$$

Furthermore, S is robustly Ex-learnable aided by context iff for every general recursive operator Θ there is an M Ex-learning every $\Theta(f)$ from input $\Theta(f)$ and context $\Theta(g_f)$.

It was shown that many but not all classes are robustly Ex-learnable aided by context.

Theorem 8.2 [5]. *If S has a dense and recursively enumerable subclass then S can be robustly Ex-learned aided by context. In particular the class of all recursive functions is robustly Ex-learnable aided by context. But on the other hand there is an infinite class which is not robustly Ex-learnable aided by context.*

The further investigations about learning aided by context dealt with the question, how learnability is affected if some requirements are imposed on the mapping $f \rightarrow g_f$. The first result shows that for many classes the context becomes useless if it is required that the context is generated by a general recursive operator.

Theorem 8.3 [5]. *If S is closed under finite variants and S is robustly Ex-learnable aided by context via a mapping $f \rightarrow g_f$ computed via a general recursive operator Θ then S is already robustly Ex-learnable.*

In the following the hyperrobust variant of the notion of Ex-learning aided by context is introduced. Note that it is important that one requires Ex-learning only if f and g_f are transformed by the same primitive recursive operator.

Definition 8.4 [20]. A class S is hyperrobustly Ex-learnable aided by context if there exist a learner M and, for every $f \in S$, a context function $g_f \in S$ such that, for every primitive recursive operator Θ , M Ex-learns the function $\Theta(f)$ from the data-stream of the pair $(\Theta(f), \Theta(g_f))$. Similarly one defines the notion of hyperrobust BC-learning.

The next result shows that for Ex-learning this notion collapses to Num.

Theorem 8.5 [20]. *If a class S is hyperrobustly Ex-learnable from selected context then S is already hyperrobustly Ex-learnable and therefore S is in Num.*

Proof. The general idea of the proof is to split S into two classes S_1 and S_2 and to show that both $[S_1]$ and $[S_2]$ are in Num. This implies that S is in Num, too. Assume that M hyperrobustly learns S from selected context.

Let S_1 be the class of all functions $f \in S$ such that M learns f from the context f itself. Correspondingly, S_2 denotes the class of all $f \in S$ such that for all contexts $g \in S$ from which M infers f , it holds that $g \neq f$.

For any $f \in S$ and any primitive recursive operator Θ , the learner M Ex-infers $\Theta(f)$ from $\Theta(f)$ itself plus the image of the context under Θ . So, simulating M by adding the context $\Theta(f)$ to $\Theta(f)$ itself, one obtains an Ex-learner for $[S_1]$. As a consequence, $[S_1]$ and the generating class S_1 are both in Num.

For any $f \in S_2$, for the context g of f and for any primitive recursive operator Θ , there is a further primitive recursive operator Θ' such that $\Theta'(f) = \Theta(f)$ and $\Theta'(g) = 0^\infty$. This implies that M Ex-learns $\Theta(f)$ from the context 0^∞ . Again one can simulate the learner by adding just the context 0^∞ to the input function. Thus, it follows that $[S_2]$ and S_2 are in Num, too.

For any two classes in Num, their union is also in Num. In particular, the union S of S_1 and S_2 is in Num, which completes the proof. ■

For BC-learning, the corresponding question seems to be more difficult and is linked to Open Problem 7.2. If the answer to that problem is that hyperrobust BC is not closed under union, then selected context offers a real support for hyperrobust BC-learning.

Proposition 8.6 [20]. *A class is hyperrobustly BC-learnable aided by selected context iff it is contained in the union of two hyperrobustly BC-learnable classes.*

Proof. The proof of Theorem 8.5 for Ex can be transferred to BC such that $[S]$ is shown to be the union $[S_1] \cup [S_2]$ of two BC-learnable classes $[S_1]$ and $[S_2]$. Only the last argument that the union of classes in Num is again in Num cannot be transferred to the BC-case since Problem 7.2 is still open.

For the converse direction, let the hyperrobustly BC-learnable classes S_1 and S_2 be given. Now, one assigns to every $f \in S_1$ the context f itself and to every $f \in S_2 - S_1$ some fixed context $g \in S_1$. Let M_1 and M_2

be the BC-learners for the classes $[S_1]$ and $[S_2]$. Now the context-aided BC-learner M follows the output of M_1 as long as the function and the context are equal and changes to the output of M_2 when the context differs from the function itself. More precisely, for input σ, τ of the same length:

$$M(\sigma, \tau) = \begin{cases} M_1(\sigma) & \text{if } \sigma = \tau; \\ M_2(\sigma) & \text{otherwise.} \end{cases}$$

Now let $h = \Theta(f)$ be the function to be learned. If the context equals h then either $f \in S_1$ or Θ maps f and its context g to the same function. In the first case, the algorithm is correct since M_1 BC-learns $[S_1]$. In the second case, $h = \Theta(g)$ and h is again in $[S_1]$ since $g \in S_1$. Thus, M_1 is again a correct algorithm. If the context is different from h , then the original context must be different from f and $f \in S_2$. It follows that $h = \Theta(f)$ is in $[S_2]$. M outputs almost always the guesses of M_2 and thus M BC-learns h in this case, too. ■

9. Conclusion

Although self-referential coding is an elegant proof-method, the resulting classes used as witnesses to separate some learning criteria look artificial. So Bārzdīņš asked whether there are really naturally defined classes which are learnable but not already learnable by enumeration. To formalize this, Bārzdīņš proposed the notion of robust learning and conjectured that every robustly learnable class is already in Num; as a consequence every natural learnable class would already be in Num.

Zeugmann [31] published intermediate results on the way to solve Bārzdīņš' Conjecture and showed that it holds for the notion of reliable learning. Fulk [10] refuted Bārzdīņš' Conjecture and showed that there are robustly Ex-learnable classes which are not in Num. Later Jain, Smith and Wiehagen [14] found even a class defined with a topological version of self-referential coding which is robustly Ex-learnable. Jain [12] extended this line of research by giving an example of a robustly BC-learnable class which cannot be Ex-learned. Similar robust separations for several notions of consistency followed [6].

There were several attempts to tighten the notion of robust learning such that all types of coding were ruled out. In the case of uniformly robust learning the approach failed. In the case of hyperrobust learning, the approach succeeded for Ex-learning by collapsing this notion to Num, so that Bārzdīņš' Conjecture holds for this more restrictive variant of robustness. Furthermore, hyperrobust BC-learning turns out to contain some classes outside Num without applying self-reference. It has been

shown that uniformly robust learning and robust learning are different for the criteria of confident learning, Ex-learning and BC-learning.

Although much work has been done, several interesting open problems remain.

- (1) Is there a more appropriate way to model learnability in a natural way? On the one hand, the notions of robust and uniform robust Ex-learning cannot prevent all coding tricks. On the other hand, the three notions of robust, uniformly robust and hyperrobust learning all fail to learn the quite natural class BB from Example 3.6 which was introduced by Blum and Blum [3, 25].
- (2) Is every robustly Ex-learnable class also consistently learnable? It is a surprising result that this holds for uniformly robust Ex-learning [6] but all attempts to generalize it have failed so far.
- (3) Does the Union-Theorem hold for hyperrobust BC-learning? The characterization that the hyperrobust Ex-learnable classes are exactly the classes in Num implies as a corollary the Union-Theorem. As this proof-method cannot be applied in the case of hyperrobust BC-learning, there is a certain chance that the union of some hyperrobustly BC-learnable classes is not hyperrobustly BC-learnable.

Acknowledgments. The authors would like to thank the referee for useful comments and suggestions. Further thanks go to the co-authors (John Case, Wolfgang Merkle, Matthias Ott, Arun Sharma, Carl Smith, Rolf Wiehagen and Thomas Zeugmann) of joint research reported in this survey.

References

- [1] Janis Bārzdiņš. Inductive inference of automata, functions and programs. In *Int. Math. Congress, Vancouver*, pages 771–776, 1974.
- [2] Janis Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs*, Latvian State University, Riga, 210:82–88, 1974.
- [3] Lenore Blum and Manuel Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
- [4] John Case. Learning Machines. In Demopoulos, W. and Marras, A., editors, *Language Learning and Concept Acquisition*, pages 83–102, Ablex Publishing Company, 1986.
- [5] John Case, Sanjay Jain, Matthias Ott, Arun Sharma and Frank Stephan. Robust learning aided by context. *Journal of Computer and System Sciences (Special Issue COLT 1998)*, 60:234–257, 2000. Extended Abstract in *Proceedings of the Eleventh Annual ACM Conference on Computational Learning Theory – COLT 1998*, ACM-Press, 44–55, 1998.
- [6] John Case, Sanjay Jain, Frank Stephan and Rolf Wiehagen. Robust learning – rich and poor. *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory and the Fifth European Conference on Computational Learning Theory – COLT 2001 and EuroCOLT 2001*, Springer LNAI 2111:143–159, 2001. Also available as: Technical Report LSA-2000-03E, Centre for Learning Systems and Applications, Department of Computer Science, University of Kaiserslautern, 2000.
- [7] John Case, Susanne Kaufmann, Efim Kinber and Martin Kummer. Learning recursive functions from approximations. *Journal of Computer and System Sciences*, 55:183–196, 1997.

- [8] John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [9] Jerome Feldmann. Some decidability results on grammatical inference and complexity. *Information and Control*, 20:244–262, 1972.
- [10] Mark Fulk. Robust separations in inductive inference. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 405–410, St. Louis, Missouri, 1990.
- [11] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [12] Sanjay Jain. *Robust Behaviourally Correct Learning*. *Information and Computation*, 153:238–248, 1999.
- [13] Sanjay Jain, Daniel Osherson, James Royer and Arun Sharma: *Systems that Learn, Second Edition* [of 14]. The MIT Press, Cambridge, Massachusetts, 1999.
- [14] Sanjay Jain, Carl Smith and Rolf Wiehagen. Robust Learning is Rich. *Journal of Computer and System Sciences*, 62:178–212, 2001.
- [15] Klaus-Peter Jantke and Hans-Rainer Beick. Combining postulates of naturalness in inductive inference. *Journal of Information Processing and Cybernetics (EIK)*, 17:465–484, 1981.
- [16] Wolfgang Merkle and Frank Stephan. Trees and learning. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory (COLT)*, pages 270–279, ACM Press, New York, 1996.
- [17] Eliana Minicozzi. Some natural properties of strong-identification in inductive inference. *Theoretical Computer Science*, 2:345–360, 1976.
- [18] Piergiorgio Odifreddi. *Classical Recursion Theory I and II*. North-Holland, Amsterdam, 1989 and 1999.
- [19] Daniel Osherson, Michael Stob and Scott Weinstein. *Systems that Learn*. MIT Press, Cambridge, Massachusetts, 1986.
- [20] Matthias Ott and Frank Stephan. Avoiding coding tricks by hyper-robust learning. *Proceedings of the Fourth European Conference on Computational Learning Theory – EuroCOLT 1999*. Springer LNCS 1572, 183–197, 1999.

- [21] Lenny Pitt. Probabilistic inductive inference. *Journal of the Association of Computing Machinery*, 36:383–433, 1989.
- [22] Lenny Pitt and Carl Smith. Probability and plurality for aggregations of learning machines. *Information and Computation*, 77:77–92, 1998.
- [23] Karlis Podnieks. Comparing various concepts of function prediction, Part 1. *Theory of Algorithms and Programs, Latvian State University, Riga*, 210:68–81, 1974.
- [24] Carl Smith. The power of pluralism for automatic program synthesis. *Journal of the Association of Computing Machinery*, 29:1144–1165, 1982.
- [25] Frank Stephan and Thomas Zeugmann. On the uniform learnability of approximations to non-recursive functions. *Proceedings of the Ninth Annual Workshop on Algorithmic Learning Theory – ALT 1999*, Springer LNCS 1720, 276–290, 1999. Also available as *Learning Classes of Approximations to Non-Recursive Functions*. Department of Informatics, Report 166, Kyushu University, Fukuoka, 1999.
- [26] Hermann Weyl. *Symmetry*. Princeton University Press, 1952
- [27] Rolf Wiehagen. Identification of formal languages. In *Mathematical Foundations of Computer Science, Proceedings, 6th Symposium, Tatranska Lomnica*, pages 571–579. Springer-Verlag, 1977. Lecture Notes in Computer Science 53.
- [28] Rolf Wiehagen. *Zur Theorie der Algorithmischen Erkennung*. Dissertation B, Humboldt University of Berlin, 1978.
- [29] Rolf Wiehagen and Walter Liepe. Charakteristische Eigenschaften von erkennbaren Klassen rekursiver Funktionen. *Journal of Information Processing and Cybernetics (EIK)*, 12:421–438, 1976.
- [30] Thomas Zeugmann. On the nonboundability of total effective operators. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 30:169–172, 1984.
- [31] Thomas Zeugmann. On Bärzdiņš’ Conjecture. In K. P. Jantke, editor, *Proceedings of the International Workshop on Analogical and Inductive Inference (AII’86)*, volume 265 of LNCS, pages 220–227. Springer, 1986.