Degrees of Computing and Learning

Habilitationsschrift von Frank Stephan

Überarbeitung der bei der

Fakultät für Mathematik

der

Ruprecht-Karls-Universität

Heidelberg

im Februar 1999

von Frank Stephan

aus der Bundesstadt Bonn

eingereichten Schrift

Degrees of Computing and Learning

Frank Stephan*

Zusammenfassung In dieser Arbeit wird die Gradstruktur von Orakeln, dargestellt als Mengen natürlicher Zahlen, hinsichtlich traditioneller rekursionstheoretischer sowie auch neuerer lerntheoretischer Reduktionen untersucht. Letztere sind definiert nach der Fähigkeit der Orakel, Klassen totaler und berechenbarer Funktionen unter einem vorgegebenen Lernkriterum zu lernen. Zu diesen Kriterien gehören das Lernen im Limes (Ex), das verhaltenskorrekte Lernen (BC), das endlichen Lernen (Fin) und die Vorhersage von Funktionswerten (NV). Die ersten beiden Kapitel geben eine Einleitung in das Thema und stellen die Nutzung von Orakeln fürs Lernen vor. Das dritte Kapitel untersucht jene Lernkriterien, die neben dem Orakel noch einen Lehrer vorsehen, der Auskunft über die zu lerndende Funktion gibt. Im vierten Kapitel wird das Lernen von Datenströmen untersucht, welche jede korrekte Information unendlich oft, aber auch zu einem gewissen Grad inkorrekte Störungen endlich oft enthalten. Das zentrale Ergebnis ist, daß das Funktions-Lernen im Limes von gestörten Daten genauso mächtig ist wie das endliche Funktions-Lernen von herkömmlichen, ungestörten Daten mit Orakel K. Im fünften Kapitel wird das Sprachlernen untersucht, insbesondere wird die Komplexität universeller Lerner bestimmt in Abhängigkeit von der Information, die über die Klasse S der zu lernenden Sprachen verfügbar ist. So gibt es zum Beispiel einen berechenbaren universellen Lerner, der jede prinzipiell lernbare Klasse S mit Hilfe ihrer Index-Menge B als Orakel lernt. Im sechsten Kapitel betrachtet man das Lernen bei robustem Zugriff auf die Orakel, das heißt, der Lerner muß mit jedem Orakel erfolgreich sein, das einer Spezifikation genügt. Es zeigt sich, daß Spezifikationen syntaktischer Natur dazu besonders gut geeignet sind. Wenn man dagegen einen Turing-Grad, zum Beispiel den vom Halteproblem K, als Spezifikation vorgibt, können nur solche Klassen gelernt werden, die auch ohne Orakel lernbar sind. Im siebten Kapitel wird die Klassifikation betrachtet, die irgendwo in der Mitte zwischen der Theorie des Lernens und der des Rechnens liegt. Das achte Kapitel betrachtet die Strukturen innerhalb von Truth-Table Graden. Es wird gezeigt, daß jeder Truth-Table Grad unendlich viele Bounded Truth-Table Grade enthält. Die Anzahl der positiven Grade innerhalb eines Truth-Table Grades kann einige ungerade Werte wie 3, 19, 219 annehmen oder auch unendlich sein, aber gerade Anzahlen und die Anzahl 1 sind nicht möglich. Des weiteren werden die Techniken dahingehend angewandt, daß man einen nicht-rekursiven Turing-Grad konstruiert, der nur aus 1-subjektiven und 2-subjektiven Mengen besteht.

*Mathematisches Institut, Universität Heidelberg, Im Neuenheimer Feld 294, 69120 Heidelberg, Germany, EU, Email: fstephan@math.uni-heidelberg.de.

Contents

1 Introduction

Computing and Learning are basic topics of computer science. Not only has there been a lot of practical and applicable work but there has also been much theoretical research on a very abstract level dedicated to study these phenomena.

Church's Thesis [33] states that all models for computation are equivalent. The computable functions have many natural formalizations, the best known one is the Turing machine [34, 119, 145]. Further approaches are those of the μ-recursive functions or abstract versions of programming languages where machine-dependent limitations like maximal values for variables are removed [111, Chapter I]. This uniqueness of the notion is lost, if resource bounds on time and space are introduced: then deterministic and nondeterministic computation might become different, also the power of algorithms depends on the exact nature of the bounds. Therefore it is still attractive to study the phenomenon of computing within the well-behaved framework of recursion theory.

Inductive inference is the counterpart to recursion theory within the world of learning. In contrast to the situation of computing, there is no unique model for learning. Already Gold [56] observed that learnability depends on the form of presenting the data, the number of revisions the learner may make and the quality which the final hypothesis should have: is it sufficient if the final hypothesis just generates the set to be learned by an enumeration procedure or is it required that the final hypothesis computes its characteristic function? So learning theorists consider a lot of different notions which capture various aspects of learning. Learning theorists look upon all these models as a family of similar but not identical phenomena which all have their own right of existence. They compare and classify the models in the same way as topologists deal with the large variety of topological spaces and their more or less restrictive additional axiomatizations.

There are many connections between the fields of computing and learning. The fact that there is no recursive enumeration containing exactly all total recursive functions is one of the reasons why it is impossible to learn all total recursive functions by a single computable machine. The fact that one cannot even find out whether two programs do the same thing on all data causes the difference between syntactic convergence (explanatory learning) and semantic convergence (behaviourally correct learning) where the learner can revise the hypothesis infinitely often but still has to output almost always a correct hypothesis.

Some basic learning models converge in the limit to a hypothesis. Similarly one can define computation in the limit: A function f is computable in the limit iff there is a computable function g such that, for each x, the sequence $g(x,0), g(x,1), \ldots$ converges to $f(x)$ in the sense that almost all values $g(x,y)$ are exactly $f(x)$. This concept of computation in the limit is more powerful than ordinary computation. But Shoenfield's Limit Lemma [130] states that there is still a connection: a function f is computable in the limit iff it is computable in the standard sense via a procedure which in addition can query the halting problem K as an oracle. Oracle queries allow to check whether some element belongs to the set representing the oracle and so the learner can determine, whether the e-th program terminates with some output for the input x or whether it ends up in an endless loop or undefined situation. One application of computing with help of an oracle is to model easily variants of computing like limiting recursive processes within the standard framework of recursion theory. A second application of oracles is to measure the degree of difficulty of some unsolvable problem by determining which oracles allow to solve this problem. For example, the oracle K is necessary and sufficient to compute functions which are, without any oracle, computable in the limit.

Post [120] started to compare oracles (in his case: enumerable sets) with respect to several reducibilities. His main question was whether there is some nonrecursive but enumerable set A such that the halting problem K is not computable relative to A. Post obtained several intermediate results and showed that such sets exist for many types of reducibilities, only for the most general Turing reducibility the question remained open for more than a decade. Finally, Friedberg [47] and Muchnik [107] solved the problem by constructing such a set A. So Post's problem was the beginning of the study of several types of degrees in computing, with some accent on enumerable Turing degrees. Many refinements of Turing reducibility have been considered in the sense that the computation of A relative to B has to satisfy further properties. Degree structures have been studied with respect to their whole structure and also with respect to the possible degree structures induced by a stronger reducibility within the degrees generated by a weaker reducibility.

Turing reducibility compares oracles with respect to their ability to perform any computational task. Thus, A is Turing reducible to B iff (the characteristic function of) A can be computed relative to B. Jockusch [71] analyzed the ability of the oracles with respect to one specific task. Namely Jockusch asked for prominent classes S whether they are "subuniform in A", that is, whether there is an A-recursive total function $e, x \to f_e(x)$ such that every function in S equals to some f_e. He showed that the class REC of all recursive functions is subuniform in an oracle A iff A is high ($A' \geq_T K'$). Learning theorists have an equivalent notation for classes S subuniform in A: S is NV-learnable relative to A where NV-learning is (the most restrictive form of) learning by predicting the next value from the previous ones. So Jockusch's main result, translated into the learning theoretic terminology, says that an oracle is omniscient for NV iff it is high.

Formally, Adleman and Blum [1] transferred the notion of oracles to learning theory. They showed that for the common notion of explanatory learning (Ex) a result very parallel to Jockusch's result for NV [71]: an oracle allows to learn the class REC iff this oracle is high. Gasarch and Pleszkoch [51] shaped the modern view of learning with oracles and started to investigate the ordering on the oracles induced by learning criteria: an oracle A is below B iff everything learnable with oracle A is also learnable with oracle B. This ordering has a minimum compatibility to the ordering induced by Turing reducibility in the sense that whenever A is computable relative to B then A is also below B: every class S learnable with oracle A is also learnable with oracle B. But there is also a difference: for many learning criteria like NV and Ex, there are omniscient oracles which allow to learn everything and which so form a greatest degree — the Turing degrees do not have such a greatest degree. For most criteria, the least and greatest degree, provided that the latter exists, have been determined.

1.1 Overview on the Results

Chapter 2 gives an introduction to the notion of oracles and their applications in learning theory. After an overview on various types of oracles, the most popular notions for learning computable functions are presented. It is analyzed to which extent certain types of oracles are useful to support learning functions under these learning criteria. Adleman and Blum [1] characterized the omniscient oracles for Ex-learning as the high ones, that is, as those relative to which one can compute a function dominating all recursive functions. This characterization is extended by analyzing the connections between the learnability of an oracle A and the quantity of total recursive functions which can be dominated by some function computable relative to A [89, 132]. For the notions of finite learning [42] and some other notions, it is shown that, comparing two oracles A and B, the oracle A can learn everything what B can learn iff B can be computed relative to A. The learning criterion BC turns out to be the only basic criterion for which it is possible to learn the class REC relative to some low oracle.

This chapter mainly serves as an introduction to learning with oracles and gives an overview on mostly known results, with some accent on the author's earlier work.

Chapter 3 is dedicated to the relations between oracles and teachers. An oracle is a fixed set which does not depend on the present function f to be learned, but a teacher knows the function f and answers questions on f provided that they are formulated in a given query language. So the amount of information about f accessible by the learner is controlled by the choice of this query language. Furthermore, the language might interfere with the oracle since one might deduce some nonrecursive knowledge if the query language either alone or in combination with f permits queries whose answers cannot be computed, even if one knows a program for f.

The three most common query languages $L[\mathrm{Succ}]$, $L[<]$ and $L[+]$ are considered and combined with oracles.

In the case of learning in the limit (Ex) the results for each of these three models of query-inference are the same: If an oracle is omniscient for query-inference then it is already omniscient for Ex. There is an oracle of trivial Ex-degree, which allows nontrivial query-inference. Furthermore, queries to a teacher cannot overcome differences between oracles and the query-inference degrees are a proper refinement of the Ex-degrees.

In the case of finite learning, the query-inference degrees coincide with the Turing degrees. Furthermore oracles cannot close the gap between the different types of queries to a teacher.

The main results of this chapter have been presented at the Eighth Annual Conference on Computational Learning Theory [137]. An improved version will appear in the Annals of Pure and Applied Logic.

Chapter 4 deals with several variants of inductive inference from noisy data. The notion of noise is based on the idea that the learner receives a sequence of data elements such that each correct element appears infinitely often and each incorrect element appears at most finitely often. The main result is that the concept of learning in the limit from noisy informant has the same power as finite learning using a K-oracle from noise-free informant. The analogous equality for text fails in general and holds only in one direction in the case of learning uniformly recursive families. Furthermore, learnability from noisy informant or text in presence of using oracles is investigated. It is shown that partial identification of all enumerable sets can also cope with noisy informant and text.

The results of this chapter have been presented at the Sixth Workshop on Algorithmic Learning Theory and appeared in the journal Theoretical Computer Science [139]. Case, Jain and Sharma [24, 27, 28] continued the research on the here presented model of noise.

Chapter 5 deals with language learning from text. A learner is universal if it succeeds to learn any given language from a suitable description of a principally learnable class containing this language. In particular, the following is shown, where S ranges over the class to be learned and L over the languages in S.

If the additional information is given by a set containing at least one index for each language from S and no index of any nonmember of S then there is a universal learner having the same Turing degree as the inclusion problem for enumerable sets. This result is optimal in the sense that any further universal learner has the same or higher Turing degree.

If the additional information is given as the set of all indices of languages in S then there is a computable universal learner.

Furthermore, if the additional information is presented as an upper bound on the size of some grammar that generates L then a high oracle is necessary and sufficient to learn every language L from such an upper bound without knowledge of S.

Finally, it is investigated for which classes there are universal learners with respect to finite learning and learning from good examples. These notions need the halting problem B' relative to the index set B as additional information.

This chapter is based on joint work with Sebastiaan Terwijn who presented it at the Eleventh International Symposium on the Foundations of Computation Theory [141].

Chapter 6 studies the use of oracles for learning functions under the assumption that the learner has to succeed with all oracles which meet a given specification. The advantage of this robust access to oracles is that the approach allows to analyze the usability of oracles with respect to the concrete information they provide on the family S to be learned.

The first main result considers oracles of the same Turing degree: Robust learning with any oracle from a given degree does not achieve more than learning without any additional information.

The further work considers learning from function oracles which describe the whole class of functions to be learned in one of the following five ways: as a list of all functions in this class, a predictor for this class, a one-sided classifier accepting just the functions in this class, an identifier for the class or a martingale succeeding on this class.

It is shown that for learning in the limit (Ex), lists are the most powerful additional information, the powers of predictors and classifiers are incomparable and identifiers and martingales are of no help at all. Similar results are obtained for the criteria of predicting the next value, finite, Popperian and finite Popperian learning. Lists are omniscient for the criterion of predicting the next value and also identifiers are helpful at this criterion. So it turns out that algorithms to predict the next value can much better exploit robustly oracles than algorithms which give explanations (Ex-learning). For Ex-learning none of these five types of help is omniscient, that is, some classes cannot be Ex-learned with any of these types of additional information. The class REC is Ex-learnable with the help of a list, a predictor or a classifier.

This chapter is based on joint work with Susanne Kaufmann. The work has been presented at the Third European Conference on Computational Learning Theory [77].

Chapter 7 deals with the world of two-sided classes ($= \Delta_2^0$ classes) within the one-sided classes ($= \Sigma_2^0$ classes). A one-sided classifier is a computable device which reads the characteristic function of a set and outputs a sequence of guesses which converges to 1 iff the set on the input belongs to the given class. Such a classifier is two-sided if the sequence of its output in addition converges to 0 on sets not belonging to the class. The present work obtains the below mentioned results for one-sided classes with respect to four areas: Turing complexity, 1-reductions, index sets and measure.

There are one-sided classes which are not two-sided. This fact may have either computational or topological reasons. Computational difficulties can be overcome by using a suitable powerful oracle and the present work determines for several one-sided classes those Turing degrees which provide a two-sided classifier for the given classes. Topological difficulties cannot be compensated by oracles and such classes also do not have nonrecursive two-sided classifiers.

The concepts of 1-reduction, 1-completeness and simple sets also exist for one-sided classes: There are 1-complete classes and simple classes, but no class is at the same time 1-complete and simple.

The one-sided classes have a natural numbering. Most of the common index sets relative to this numbering have complexity Π_1^1: the index sets of the class $\{0,1\}^\infty$, the index set of the equality problem and the index set of all two-sided classes. On the other side the index set of the empty class has complexity Π_2^0; Π_2^0 and Σ_2^0 are the least complexities any nontrivial index set can have.

Any one-sided class is measurable. It is shown that a one-sided class has effective measure 0 if it has measure 0, but that there are one-sided classes having measure 1 without having measure 1 effectively. The measure of a two-sided class can be computed in the limit.

Case, Kinber, Sharma and Stephan also considered the related model where one has to classify only the recursive sets correctly. This model was presented at the Symposium on Theoretical Aspects of Computer Science [30].

Chapter 8 is dedicated to the study of strong reducibilities inside truth-table degrees. The following theorems are established:

Dĕgtev's result that the number of bounded truth-table degrees inside a truth-table degree is at least two [36] is improved by showing that this number is infinite. There are even infinite chains and antichains of bounded truth-table degrees inside the truth-table degrees which implies an affirmative answer to a question of Jockusch [70] whether every truth-table degree contains an infinite antichain of many-one degrees.

Some but not all truth-table degrees have a least bounded truth-table degree. The technique to construct such a degree is used to solve an open problem of Beigel, Gasarch and Owings [15]: there are Turing degrees (constructed as hyperimmune-free truth-table degrees) which consist only of 2-subjective sets and do therefore not contain any objective set.

Furthermore, a truth-table degree consisting of three positive degrees is constructed where one positive degree consists of enumerable semirecursive sets, one of coenumerable semirecursive sets and one of sets, which are neither enumerable nor coenumerable nor semirecursive. So Jockusch's result that there are at least three positive degrees inside a truth-table degree is optimal [69]. The number of positive degrees inside a truth-table degree can also be some other odd integers as for example nineteen, but it is never an even finite number.

1.2 Acknowledgments

The author wants to thank Klaus Ambos-Spies for supervising the project "Recursion-Theoretic Aspects of Inductive Inference". Wolfram Menzel and Martin Kummer invested much time and work for the preceding project "Query-Complexity in Recursion Theory" and introduced the author into the fields of recursion theory and inductive inference. The DFG supports the author through these two projects since 1992. In 1996 and 1997, Arun Sharma invited the author to visit his group on learning-theory at the University of New South Wales. Chapters 5 and 6 are based on the fruitful collaboration with Susanne Kaufmann and Sebastiaan Terwijn. The author would also like to thank many further colleagues for helpful discussions and for proofreading parts or all of this work: Richard Beigel, Bernd Borchert, John Case, Alexander Dĕgtev, Rod Downey, Peter Fejer, Rūsiņš Freivalds, William Gasarch, Carl Jockusch, Dick de Jongh, Efim Kinber, Wolfgang Merkle, André Nies, Jim Owings, Piergiorgio Odifreddi, Matthias Ott, Victor Selivanov, Theodore Slaman, Carl Smith and Thomas Zeugmann.

2 Degrees of Inferability

One of the main companies offering, among other software, also database systems has the name "Oracle", a name which has a mathematical motivation. For mathematicians, an oracle is just an abstract representation of a data base with a minimum instruction set: the only type of data-base access is to ask whether a certain number x - representing any type of information numerically - is in the data-base or not. If $x \in A$ for the oracle A, then the return-value is 1, if $x \notin A$ then it is 0. So one writes $A(x)$ for the result of this query. The oracle is therefore nothing else then a subset of the set $\mathbb{N} = \{0, 1, \ldots\}$ of the natural numbers. It is used to retrieve nonrecursive information which cannot be obtained otherwise. The next section gives an overview on oracles and then the relationship between oracles and learning is investigated.

2.1 Various Types of Oracles

Oracles can be compared with each other with respect to certain abilities. The most common ability is that to compute. So an oracle A can be replaced by an oracle B, or more precisely, everything computable relative to A is also computable relative to B, iff every value $A(x)$ can be computed with some queries relative to B. For example if $A(x) = B(2x) \wedge B(2x+1)$ then $A(x)$ can be computed by making two queries to B at $2x$ and $2x+1$. If both values are in B, then $A(x) = 1$, otherwise $A(x) = 0$. One says that A *is computable relative to* B or — more formally — A *is Turing reducible to* B, written $A \leq_T B$. A Turing degree is a maximal set of oracles such that every oracle in this set is Turing reducible to every other oracle in this set. Now some examples for oracles and types of oracles are given. Indeed most definitions are not totally unique since they often depend on the chosen acceptable enumeration φ of the partial recursive functions. Furthermore, for some of the types defined below, the quantity of oracles satisfying the condition is uncountable.

Example 2.1.1 Some prominent oracles and classes of oracles.

- $P = \{x : x \text{ is a prime number}\}$. This oracle is recursive since one can test effectively whether there is some $y \in \{2, 3, \ldots, x-1\}$ which divides x. Therefore $P \leq_T A$ for every set A. Recursive oracle are therefore at the bottom end of the structure of all oracles. Normally the empty set $\emptyset$ is the default representative for the class of all recursive oracles.

- $K = \{x : \varphi_x(x)\downarrow\}$. This is the diagonal halting problem. It has the same Turing degree as the full halting problem $\{(x, y) : \varphi_x(y)\downarrow\}$, that is, K can be computed relative to this set and this set can be computed relative to K. K itself is not a recursive set.

- enumerable oracles: An oracle is enumerable iff it is the domain (or the range) of a partial recursive function. The domain of the function φ_e is denoted by W_e and every enumerable set equals to such a set W_e. An oracle has an enumerable Turing degree iff some of the oracles inside this Turing degree is enumerable. The enumerable oracles are traditionally called "recursively enumerable" and are now also known as "computably enumerable" or "computably generated" oracles.

- $K' = \{x : \varphi_x^K(x)\downarrow\}$ is the halting problem relativized to K. K is Turing reducible to K' but not vice versa. So K is strictly below K'. K' has the same Turing degree as the index set $\{e : W_e \text{ is finite}\}$ of the class of all finite sets.

- high oracles: An oracle A is high if the halting problem relative to A is at least as hard as K', that is, $A' \geq_T K'$. The high oracles have a characterization: A is high iff there is some function f computable relative to A which dominates every recursive function: for every $g \in \text{REC}$ there is an x such that $g(y) \leq f(y)$ for all $y \geq x$.

- low oracles: An oracle A is low if its halting problem is computable relative to K: $A' \equiv_T K$. Note that there are nonrecursive low oracles.

- hyperimmune oracles: An oracle A is hyperimmune if some hyperimmune set is computable relative to A. An alternative characterization is that a function f is computable relative to A which is not dominated by any recursive function. For example, high oracles are hyperimmune, but some hyperimmune oracles are not high.

- hyperimmune-free oracles: those which are not hyperimmune. These oracles have also alternative characterizations: An oracle A is hyperimmune-free iff every function f computable relative to A is majorized by some $g \in \text{REC}$: $(\forall x)\,[f(x) \leq g(x)]$. Also A is hyperimmune-free iff its tt-degree and Turing degree coincide [70].

- 1-generic oracles: An oracle A is 1-generic if it either meets or strongly avoids every recursive set of strings. That is, given a recursive set W of strings, there is an $\sigma \preceq A$ such that either $\sigma \in W$ or no string $\tau \succeq \sigma$ is in W. Note that for every $B \geq_T K$ there is a 1-generic set A such that $A \oplus K$, A' and B have the same Turing degree. There are 1-generic oracles below K and these are low [72, Lemma 2.6 (a)].

- PA-complete oracles: An oracle A is called PA-complete if there is a complete extension of Peano Arithmetic which is computable relative to this oracle [111, Section V.5]. Alternative equivalent definitions are that every recursive tree has a branch computable relative to A and that every partial-recursive $\{0,1\}$-valued function has an A-recursive total extension. Jockusch and Soare [73] showed that there is a low PA-complete set, therefore it is much easier to extend the $\{0,1\}$-valued partial-recursive functions than those without any limitations on the range.

- An oracle $A = \{a_0, a_1, \ldots\}$ is retraceable iff there is a partial recursive function f with $f(a_{n+1})\downarrow = a_n$ for all n. Without loss of generality, one can define that in addition $f(a_0)\downarrow = a_0$ holds.

The Turing degrees have an order induced by Turing reduction. This order has a least degree, given by the recursive oracles, but no greatest degree, since every oracle A is strictly below A'. For any two oracles A and B, the join $A \oplus B$ is the least common upper bound; formally $2x + y \in A \oplus B$ iff $x \in A \wedge y = 0$ or $x \in B \wedge y = 1$. Friedberg [47] and Muchnik [107] showed that there are enumerable incomparable Turing degrees. If one considers only enumerable degrees, then there is also a greatest one, namely that of K.

Dekker and Myhill [38] introduced the retraceable oracles. They are useful to simplify proofs due to the following two points: every Turing degree contains a retraceable set; if a retraceable set A is enumerable relative to B then A is already computable relative to B. It is often more convenient to show that an oracle is enumerable in B rather than computable in B. Hence the following fact will be useful throughout the present work.

Fact 2.1.2 [38] *In every Turing degree there is a set A such that whenever an infinite subset of A is enumerable relative to some set B then A is already Turing reducible to B.*

Proof Given a set C representing some Turing degree, let A be the set of all strings $a_n = C(0)C(1)\ldots C(n)$ which are identified with the numbers $2^{n+1} + C(0)2^n + C(1)2^{n-1} + \ldots + C(n)2^0$. A is clearly retraceable because a_n can be obtained from a_{n+1} by just omitting the last bit in its binary representation.

If now an infinite subset of A is enumerable relative to B then one can transform an enumeration uniformly into a decision procedure of A relative to B as follows: Given x, wait until the first a_n with $a_n > x$ is enumerated and then compute all values a_m for $m \leq n$ using the retracing function. If $x = a_m$ for some m then $A(x) = 1$ else $A(x) = 0$. ∎

2.2 Degrees of Ex-Learning

If A is Turing reducible to B, then one can state it in the form that everything, which is computable relative to A, is also computable relative to B. The next sections deal with the analogous statement for learning: A has Ex-degree below B if everything, which is Ex-learnable relative to A is also Ex-learnable relative to B. Before investigating this notion it is suitable to introduce Ex-learning, that is learning in the limit, formally.

Definition 2.2.1 [56] A machine M *explanatorily learns* (Ex-learns) a function f iff M is total, $M(f(0)f(1)\ldots f(n))$ outputs the same program e for almost all n and e computes f. The machine M learns a class $S \subseteq \mathrm{REC}$ iff M learns every function $f \in S$.

Now one writes $S \in \mathrm{Ex}$ iff there is a recursive machine M which Ex-learns S. In the case, that M in only A-recursive, one writes $S \in \mathrm{Ex}[A]$. Gasarch and Pleszkoch [51] introduced the following inference degrees.

- A has inference degree below B iff $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$. The notions "below" and "above" include equality, otherwise one says "strictly below" and "strictly above", respectively.
- A and B have the same inference degree iff $\mathrm{Ex}[A] = \mathrm{Ex}[B]$.
- A has least inference degree iff $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$ for all oracles B. A has trivial inference degree iff $\mathrm{Ex}[A] = \mathrm{Ex}$. By definition, every recursive oracle belongs to the trivial degree. This trivial degree is then also the least inference degree.
- A has greatest inference degree iff $\mathrm{Ex}[B] \subseteq \mathrm{Ex}[A]$ for all oracles B. A has omniscient inference degree iff every class S can be learned relative to A. Theoretically, a greatest degree is not automatically omniscient. Indeed for criteria like the variants of learning uniformly recursive families of languages from text considered in Corollary 4.6.8, there are greatest degrees which are not omniscient; the class of all finite sets plus $\mathbb{N}$ is not learnable under any of the learning-criteria considered there. But the more standard notions of inference either have an omniscient degree or no greatest one. For example, the notion Ex considered here has a greatest and also omniscient degree.
- A and B have incomparable inference degree iff $\mathrm{Ex}[A]$ and $\mathrm{Ex}[B]$ are incomparable viewed as sets. That is, some class S is $\mathrm{Ex}[A]$-learnable but not $\mathrm{Ex}[B]$-learnable and some class $\tilde{S}$ is $\mathrm{Ex}[B]$-learnable but not $\mathrm{Ex}[A]$-learnable.

There are also further notions for learning. If one knows for such a notion, when a learner M learns a function f, then one can deduce all further definitions analogously to the way it is presented here for Ex.

Any structure of inference-degrees (like the ones for Ex) is coarser than the structure of Turing degrees. This is due to the following observation: Let $A \leq_T B$ and M be an A-recursive learner for some class S. There is an algorithm translating every query to A into queries to B. Concatenating the learner M and this algorithm gives a B-recursive learner with exactly the same input- and output-behaviour as M. So S is also B-learnable and $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$. It follows that every inference degree is the union of some Turing degrees.

The degree-structure of the Ex-degrees is closely related to the capability to dominate certain classes of recursive functions. So one introduces the notion $F[A]$ for the collection of all classes $S \subseteq \mathrm{REC}$ which are dominated by some $f \leq_T A$. That is,

$$S \in \mathrm{F}[A] \;\Leftrightarrow\; (\exists f \leq_T A)\,(\forall g \in S)\,(\exists x)\,(\forall y \geq x)\,[f(y) \geq g(y)].$$

Adleman and Blum [1] showed that an oracle is omniscient iff it is high. Thus A is omniscient iff $\mathrm{REC} \in \mathrm{F}[A]$. Kummer and Stephan [89] showed that the Ex-degrees of oracles below K can be characterized by domination properties. On the other hand this characterization does not carry over to all degrees.

Theorem 2.2.2 *For all oracles A, B, C with $A \leq_T K$ and $C' \not\geq_T K'$ the following holds:*
(a) $\mathrm{REC} \in \mathrm{Ex}[B] \Leftrightarrow \mathrm{REC} \in \mathrm{F}[B]$ [1, §IV].
(b) $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B] \Leftrightarrow \mathrm{F}[A] \subseteq \mathrm{F}[B]$ [89, Theorem 9.4].
(c) $\mathrm{Ex}[B] \subseteq \mathrm{Ex}[C] \Rightarrow B' \leq_T C' \wedge \mathrm{F}[B] \subseteq \mathrm{F}[C]$ [89, Theorem 9.1].

Kummer and Stephan [89] showed that $R_A \in \mathrm{F}[B]$ is in the second condition enough to imply $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$ where $R_A = \{f \in \mathrm{REC} : (\forall x)\,[f(x) \leq c_A(x)]\}$, $c_A(x) = \min\{s \geq x : (\forall y \leq x)\,[A_s(y) = A(y)]\}$ and A_s is a recursive approximation for A. So R_A is a typical member of $\mathrm{F}[A]$. The function c_A is also linked to Turing degrees: $A \leq_T B$ iff there is a B-recursive function dominating c_A [105]. The last result also implies that in general the structures induced by domination and learning differ: If B is hyperimmune-free then $\mathrm{F}[B] = \mathrm{F}[\emptyset]$ by definition but $\mathrm{Ex}[B] \not\subseteq \mathrm{Ex}[\emptyset]$ since $B' \not\leq_T K$ [111, Exercise V.5.3.(d)].

Based on a result of Haught [60], Kummer and Stephan [89] showed, that for every $A \leq_T K$ which is not below a 1-generic set, the class R_A is not dominated by any recursive function, that is, $R_A \notin \mathrm{F}[\emptyset]$. Therefore one can learn with this oracle A some classes not learnable by a recursive learner. Together with Theorem 2.2.2 (c), this gives an alternative proof for the hard direction of Slaman and Solovay's characterization of the trivial oracles for Ex.

Theorem 2.2.3 [132] $\mathrm{Ex}[A] = \mathrm{Ex}$ *iff there is a 1-generic set B with $A \leq_T B \leq_T K$.*

Kummer and Stephan [89] have two further results on the structure of the Ex-degrees.

Theorem 2.2.4 (a) *If A is enumerable then* $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$ *iff B is high or $A \leq_T B$.*
(b) *If A, B are 1-generic then* $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$ *iff B is high or $A \leq_T B \oplus K$.*

2.3 Finite Learning

A finite learner cannot withdraw a hypothesis. So the finite learner reads data and outputs the special symbol "?" until enough data is collected to come up with the then true hypothesis. Now the formal definition follows.

Definition 2.3.1 [56] M finitely learns (Fin-learns) a function f iff there is a unique index e of f ($f = \varphi_e$) such that $M(\sigma) = e$ for some $\sigma \preceq f$ and $M(\tau) \in \{e, ?\}$ for all other $\tau \preceq f$.

Clearly if M learns f finitely, then M learns f also in the limit. So the class Fin of all S which have a finite recursive learner is contained in Ex; this containment is even proper since the class of all almost everywhere 0 functions is in Ex but not in Fin.

Fortnow et al. [43] showed that the structure induced by Turing reducibility and that induced by the capacity of an oracle to support finite learning are the same. Let Fin[A] denote the collection of all classes $S \subseteq$ REC which are learnable relative to oracle A.

Theorem 2.3.2 [43] *$A \leq_T B$ iff* Fin[A] $\subseteq$ Fin[B].

Proof Clearly, if $A \leq_T B$ then Fin[A] $\subseteq$ Fin[B]. So only the converse direction is interesting. According to Fact 2.1.2, one can chose A such that $A \leq_T B$ whenever A is enumerable relative to B.

Now one takes the class of all functions whose course of values is first a (probably void) sequence of nonelements from A then followed by a constant sequence given by some fixed $b \in A$:

$$f \in S \;\Leftrightarrow\; (\exists a_0, a_1, \ldots, a_n \notin A)\,(\exists b \in A)\,[f = a_0 a_1 \ldots a_n b^\infty] \tag{1}$$

Assume now that there is a B-recursive learner M for S. Now if $x \in A$ then $M(x^n) \neq ?$ for some n since M must output on the input x^∞ some index for f. If $x \notin A$ then $M(x^n) = ?$ for all n since every function $x^n y^\infty$ with $y \in A$ is in S and the learner M must not make any output before the function to be learned is specified uniquely by the data seen so far. Thus

$$x \in A \;\Leftrightarrow\; (\exists n)\,[M(x^n) \neq ?] \tag{2}$$

and so A is enumerable and even computable relative to B. ∎

So the structure of the degrees of finite learning is just the intensively studied structure of the Turing degrees [111, Chapter V]. The criterion Fin[K] is quite prominent in learning theory. The following statements either directly characterize Fin[K] or show that K is the least or greatest degree among the oracles A such that Fin[A] satisfies some condition.

Remark 2.3.3 (a) $S \in$ Fin[K] iff S is learnable from noisy data in the limit. Chapter 4 and Theorem 4.2.1 provide more details.

(b) If $S = \{f_0, f_1, \ldots\}$ is a uniformly recursive family (also called an "indexed family") in the sense that the function $e, x \to f_e(x)$ is recursive in both parameters e and x then $S \in$ Fin[K] iff $S \in$ Fin[A] for some A iff S is discrete where S is called *discrete* iff, for every $f \in S$, there is a $\sigma \preceq f$ such that no function $g \in S - \{f\}$ extends σ. Furthermore, Fin[K] $\not\subseteq$ Fin[A] for all oracles $A \not\geq_T K$, so the choice of the oracle K is optimal in this context. — The proof of Corollary 4.6.5 is actually for set-learning, but can be easily adapted to the model of learning functions considered here.

(c) $S \in$ Fin[K] iff there is an enumerable tree such that all $f \in S$ are fully isolated on T [104, Theorem 5.2], where a branch of a tree T is called fully isolated iff there is some n such that the nodes in T above $f(0)f(1)\ldots f(n)$ are exactly the nodes $f(0)f(1)\ldots f(m)$ with $m \geq n$.

(d) Fin[A] $\subseteq$ Ex iff $A \leq_T K$. So K is the greatest oracle A such that Fin[A] is contained in Ex. The inclusion is quite obvious and follows the ideas outlined in Theorem 4.2.1. The noninclusion is more difficult and thus proven in the next theorem.

Theorem 2.3.4 *If* Fin[A] $\subseteq$ Ex *then* $A \leq_T K$.

Proof It is possible to define a family $\varphi_{g(i,j)}$ which satisfies the conditions below — this family is just that one from [89, Theorem 8.1], equipped with a further parameter j.

- $1^i 0 1^j 0 \preceq \varphi_{g(i,j)}$ for all i and j.
- $\varphi_{g(i,j)}(x)$ is defined for all x with at most one exception $a_{i,j} > i + j$.
- If, for some fixed j, there exists a machine M inferring all total functions among the $\varphi_{g(i,j)}$, $i \in \mathbb{N}$, then there exists an i such that $\varphi_{g(i,j)}$ is partial and M converges to the same index $e_{i,j}$ on the two functions f_0 and f_1 which extend $\varphi_{g(i,j)}$ with 0 and 1 at $a_{i,j}$, respectively.

The third condition is just a corollary from the one presented in [89, Theorem 8.1]. Now let S be the collection of all total functions $\varphi_{g(i,j)}$ where $j \in A$ and of all functions f extending some $\varphi_{g(i,j)}$ with one undefined place where $j \notin A$. One has that

> j is in A iff there are i, s, a, e such that
> (a) $\varphi_{g(i,j)}(a)\uparrow$ and
> (b) $M(\varphi_{g(i,j)}(0)\varphi_{g(i,j)}(1) \ldots \varphi_{g(i,j)}(a-1)b\varphi_{g(i,j)}(a+1) \ldots \varphi_{g(i,j)}(x)) = e$
> for all $x > a + s$ and $b \in \{0, 1\}$.

It follows that A is enumerable relative to K: the conditions (a) and (b) are K-recursive and the whole term is an existential quantification over the conjunction of these K-recursive terms. To see that (b) is really K-recursive, note that $\varphi_{g(i,j)}(x)\downarrow$ whenever $x \neq a$ and $\varphi_{g(i,j)}(a)\uparrow$, so the term inside the universal quantification is recursive and these quantifiers can be resolved using the oracle K. Thus A is enumerable in K.

By Fact 2.1.2, one can chose the oracle A within the given Turing degree such that A is enumerable relative to some oracle B iff A is recursive to B. So one obtains that A is recursive relative to K. ∎

2.4 Learning Criteria Closed Under Union

Many results on Fin and Ex use that there are two learnable classes whose union is not learnable. This enables to code the oracle into a parameterized mixture of the two classes. The notions considered within this section are closed under union so that this method does not work. The five notions considered are predicting the next value, learning in the limit by total guesses only, two variants of reliable learning in the limit and learning via a machine which converges on all functions, also on the nonrecursive ones. Here the formal definitions.

Definition 2.4.1 [11, 19, 31, 106] Let M be a learner and f be a function.
(a) *Next Value learning*: M NV-learns f iff M is total and $M(f(0)f(1)\ldots f(x)) = f(x+1)$ for almost all x.
(b) *Reliable Explanatory learning*: M REx-learns f iff M Ex-learns f and in addition M makes infinitely many mind changes on every recursive function g which M does not Ex-learn.
(c) *Reliable Explanatory learning with respect to all functions*: M R_{all}Ex-learns f iff M Ex-learns f and in addition M makes infinitely many mind changes on every function g which M does not Ex-learn, including nonrecursive functions g.
(d) *Popperian Explanatory learning*: M PEx-learns f iff M Ex-learns f and every guess

on any data (even if not belonging to any valid concept) is a total program.
(e) *Explanatory learning with total convergence on recursive functions*: M TEx-learns f iff M Ex-learns f and M in addition converges on every recursive function to some index.
(f) *Explanatory learning with total convergence*: M T_{all}Ex-learns f iff M Ex-learns f and M in addition converges on every function to some index, even on the nonrecursive ones.

It is quite well-known that these criteria are closed under union, nevertheless some basic ideas are presented how to construct a learner for the union from two learners M_1 and M_2 for the given classes.

In the case of NV, the learner M just follows that learner M_i which has made less errors on the preceding data. In the case of REx and R_{all}Ex, M takes the value of that learner M_i which has made less mind changes.

In the case of PEx, the closure under union can be deduced from the following characterization: A class S is PEx$[A]$-learnable iff there is an A-recursive set E of indices of total recursive functions such that every function in S has an index in E. Having two classes S and $\tilde{S}$ in PEx$[A]$, one just takes the corresponding A-enumerable index sets E and $\tilde{E}$. Their union is also enumerable relative to A and so $S \cup \tilde{S}$ is also in PEx$[A]$.

In the last case of TEx and T_{all}Ex, the algorithm is more complicated. A learner $M_i(\sigma)$ is called *currently faulty* iff there is some x such that $\varphi_{M_i(\sigma),|\sigma|}(x)\downarrow \neq \sigma(x)\downarrow$. If one learner, say M_1, is currently faulty, then M takes the output of the other learner, in this case M_2. Otherwise, that is, if no learner is currently faulty, then M just outputs the amalgamation of the programs $M_1(\sigma)$ and $M_2(\sigma)$, where the *amalgamation* of a set I of programs computes for each input x that y such that there is some $i \in I$ for which the computation $\varphi_i(x)$ outputs y before the other computations have converged. That is, the amalgamation just runs all programs in parallel and then takes the first output generated by any of them.

So the common property of all these notions is that they are closed under union. Many diagonalization techniques are based on the nonunion property of Ex or BC. These techniques are therefore not applicable here. Nevertheless it turns out that in spite of this common property the degree structures induced by these learning-criteria are quite different.

Bārzdins and Freivalds [12] showed that $S \in$ NV iff there is a uniformly recursive family $f_0, f_1, \ldots$ such that every $f \in S$ equals to some f_e. This result also holds relatived to any oracle A. Jockusch [71] called such a class S subuniform in A [135, Section V.5] and showed that $\mathrm{REC}_{0,1}$ is subuniform for all high and all PA-complete oracles and REC is A-subuniform iff A is high. So only the high oracles are omniscient for NV. The same holds also for REx and R_{all}Ex since these criteria are restrictions of the criterion Ex, which is also omniscient only for high oracles. As the next theorem shows, are high oracles also sufficient to learn REC under the criteria NV, REx and R_{all}Ex.

Theorem 2.4.2 [1, 43] *The omniscient degrees of* NV*,* R_{all}Ex *and* REx *consist of the high oracles. The omniscient degree of* PEx *consists of all oracles A which satisfy $A \oplus K \geq_T K'$.*

The trivial degree of these notions is more complicated. There is no recursion-theoretic characterization but it could be shown that many oracles are trivial for PEx, in particular the cardinality of this trivial degree is not countable.

Theorem 2.4.3 [43, Theorem 6.40] *If $A \leq_T K$ or if A has hyperimmune-free Turing degree then A is in the trivial degree for* PEx*-learning.*

For the degrees below K, there is a partial characterization in the following sense: If $\mathrm{Ex}[A] \subseteq \mathrm{Ex}[B]$ then the same inclusion holds for the notions REx and NV of inference. So the REx and NV degrees are courser than the Ex-degrees below K. Note that the inclusion structure of Ex and F coincide below K so that the following theorem is stated in terms of the domination classes F. The central idea of the proof follows that one of [89, Theorem 9.4].

Theorem 2.4.4 *If $A \leq_T K$ and $\mathrm{F}[A] \subseteq \mathrm{F}[B]$ then $\mathrm{NV}[A] \subseteq \mathrm{NV}[B]$, $\mathrm{REx}[A] \subseteq \mathrm{REx}[B]$, $\mathrm{R}_{all}\mathrm{Ex}[A] \subseteq \mathrm{R}_{all}\mathrm{Ex}[B]$ and $\mathrm{TEx}[A] \subseteq \mathrm{TEx}[B]$.*

Proof Let S in $\mathrm{NV}[A]$, $\mathrm{REx}[A]$, $\mathrm{R}_{all}\mathrm{Ex}[A]$ or $\mathrm{TEx}[A]$, respectively. The learner M is recursive relative to A but might be nonrecursive. Nevertheless for each $f \in S$ in the first three cases and for each $f \in \mathrm{REC}$ in the case of TEx, the function

$$x \to M(f(0)f(1)\dots f(x))$$

is still recursive: In the case NV, it is a finite variant of f. In the other cases, M converges to an index e and the function is almost constant. In these last three cases one can use the Padding Lemma to make M increasing in the sense that $M(\sigma\tau) \geq M(\sigma)$ for all σ and τ and on every function f, M either converges to an index of f or grows unboundedly. Since $A \leq_T K$, M has a recursive approximation M_s which in the cases REx and $\mathrm{R}_{all}\mathrm{Ex}$ again satisfies $M_s(\sigma\tau) \geq M_s(\sigma)$ for all s, σ and τ. Furthermore, let $\sigma_0, \sigma_1, \dots$ be an enumeration of all strings. Now one defines for M and for every $f \in S$ and in the case of TEx for every $f \in \mathrm{REC}$ the functions

$$\begin{aligned} c_M(x) &= \min\{s \geq x : M_s(\sigma_y) = M(\sigma_y) \text{ for all } y \leq x\}, \\ c_f(x) &= \min\{s \geq x : M_s(\sigma_y) = M(\sigma_y) \text{ for all } y \leq x \text{ with } \sigma_y \preceq f\}. \end{aligned}$$

The function c_M is A-recursive, the functions c_f are all recursive. By definition, c_M dominates every function c_f. Therefore there is a B-recursive function g which dominates every c_f. Using this g it is possible to give the following inference algorithm:

$$N(\sigma_y) = M_x(\sigma_y) \text{ for the first } x \geq y \text{ with } M_x(\sigma_y) = M_s(\sigma_y) \text{ for } s = x, x{+}1, \dots, g(x). \quad (3)$$

Clearly the algorithm is B-recursive. Since furthermore $M_s(\sigma_y)$ converges to $M(\sigma_y)$ at some t, any $x \geq y + t$ satisfies the condition in (3) and N is total. Given f, the function c_f is dominated by g. So there is some x_f such that $c_f(x) \leq g(x)$ for all $x \geq x_f$. Almost all $\sigma_y \preceq f$ satisfy $y \geq x_f$, for each of them the algorithm produces an $x \geq x_f$ such that $M_s(\sigma_y) = M_x(\sigma_y)$ for $s = x, x + 1, \dots, g(x)$. By the choice of c_f, one of these $M_s(\sigma_y)$ coincides with $M(\sigma_y)$. So the output of M and N coincide for almost all $\sigma_y \preceq f$. In particular, if M NV-learns f, so does N, if M Ex-learns f, so does N, and if M converges on $f \in \mathrm{REC}$, so does N.

For the cases of REx-learning and $\mathrm{R}_{all}\mathrm{Ex}$-learning, it is shown that whenever M diverges on a function so does N. Given such an f and a constant k there is an y such that $\sigma_y \preceq f$ and $M(\sigma_y) \geq k$. Then there is a stage t such that $M_s(\sigma_y) \geq k$ for all $s \geq t$. If now $\sigma_z \preceq f$ is sufficiently large, that is, if $\sigma_z \succeq \sigma_y$ and $z \geq t$ then the algorithm converges to some value $M_x(\sigma_z)$ with $x \geq z$. Since $z \geq t$, it follows that $M_x(\sigma_y) \geq k$ and by the monotonicity of M_x it follows that also $M_x(\sigma_z) \geq k$. Now $N(\sigma_z) \geq k$ and N takes on f arbitrary large values, in particular N does not converge on f. ∎

The same proof would also work with Ex, LimEx and Ex* in place of REx since the exact nature of the indices is not important and the additional requirement on the divergence could be dropped. A LimEx-learner converges to a program which computes f in the limit and an Ex*-learner to a program which computes f at all but finitely many places. For LimEx and oracles $A, B \leq_T K$ one has that the implication

$$\mathrm{Ex}[A] = \mathrm{Ex}[B] \;\Rightarrow\; \mathrm{LimEx}[A] = \mathrm{LimEx}[B]$$

holds but not its converse. There is a low oracle A which is omniscient for LimEx [43]. So $\mathrm{LimEx}[A] = \mathrm{LimEx}[K]$ but $\mathrm{Ex}[A] \neq \mathrm{Ex}[K]$. This example gives some (small) incidence that the degree-structures below K of the criteria NV, REx and $\mathrm{R}_{all}\mathrm{Ex}$ might differ from that of F and Ex. Nevertheless one can at least for the PA-completes oracle show that the degrees of F and NV coincide.

Theorem 2.4.5 *Let A and B be PA-complete. Then* $\mathrm{NV}[A] = \mathrm{F}[A]$. *In particular the inclusion structures are the same:* $\mathrm{NV}[A] \subseteq \mathrm{NV}[B]$ *iff* $\mathrm{F}[A] \subseteq \mathrm{F}[B]$.

Proof The equivalence of the inclusion structure is a direct corollary of $\mathrm{NV}[A] = \mathrm{F}[A]$ and the corresponding result for B. So it is sufficient to show the equality of the notions $\mathrm{NV}[A]$ and $\mathrm{F}[A]$ for any PA-complete oracle A.

Let S be in $\mathrm{NV}[A]$. Then there is a uniformly A-recursive array $f_0, f_1, \ldots$ such that every $f \in S$ equals to some f_e. It follows that the A-recursive function

$$x \to f_0(x) + f_1(x) + \ldots + f_x(x)$$

dominates every function f_e and thus also every in S. It follows that $S \in \mathrm{F}[A]$.

For the converse let $S \in \mathrm{F}[A]$ and g be an A-recursive function which dominates the class S. Since A is PA-complete there is a total A-recursive function h such that $h(e,x) = \varphi_e(x)$ whenever $\varphi_e(x)\downarrow \leq g(x) + e$. If $f \in S$ then g dominates f and there is a sufficiently large index e of f such that $f(x) \leq g(x) + e$ for all x. Using h it is possible to learn all functions dominated by g by enumeration. Let σ abbreviate $f(0)f(1)\ldots f(x)$ and

$$M(\sigma) = h(e, x+1) \text{ for the first } e \text{ with } h(e,y) = \sigma(y) \text{ for all } y \in dom(\sigma).$$

It is well-known that "learning by enumeration" is correct, but it remains to show that this particular M terminates for every input σ. To see this, note that the function $\sigma 0^\infty$ has some index e greater than any of its values. Now $\varphi_e(y) \leq e \leq g(y) + e$ for all y and thus $h(e,y) = \varphi_e(y)$ for all y. So if the algorithm had not found an index below this e, it will take this e, predict 0 and terminate. So the learner M is total and this completes the proof. ∎

Recall that $\mathrm{TEx}[A]$ and $\mathrm{T}_{all}\mathrm{Ex}[A]$ denote the classes of the functions which are $\mathrm{Ex}[A]$-learnable via a learner converging on every recursive and on every, also nonrecursive, function, respectively. Ambainis, Jain and Sharma [4] showed that the second learning criterion is equivalent to the model of learning with ordinal mind change bounds introduced by Freivalds and Smith [45]. The easiest way to realize such an ordinal bound is to consider a well-ordering $\sqsubseteq$ on the natural numbers and a marker placed initially on 0. Now the learner can make a mind change only if the marker moves at the same time for its current position i to some $j \sqsubset i$. A learner M learns now a class S iff there is such a recursive well-ordering $\sqsubset$ which can be used to permit the mind changes. Since it is possible to move downward in a well-ordered set only finitely often, the learner converges on every function f to some index. The difficult direction of this characterization is the other one.

The next theorem shows that the $\mathrm{T}_{all}\mathrm{Ex}$-degrees and Turing degrees coincide and that, in particular, the trivial degree consists only of the recursive oracles and no greatest degree exists. This shows that there is a clear difference, whether convergence on all or only on recursive functions is required: An omniscient Ex-learner converges on every recursive function and thus all high oracles are omniscient for TEx. Also the implication $\mathrm{F}[A] = \mathrm{F}[B] \Rightarrow \mathrm{TEx}[A] = \mathrm{TEx}[B]$ for $A, B \leq_T K$ from Theorem 2.4.4 does not transfer to $\mathrm{T}_{all}\mathrm{Ex}$.

Theorem 2.4.6 *$\mathrm{T}_{all}\mathrm{Ex}[A] \subseteq \mathrm{T}_{all}\mathrm{Ex}[B]$ iff $A \leq_T B$.*

Proof According to Fact 2.1.2 one can take A such that $A \leq_T B$ whenever A is enumerable relative to B. If $A \leq_T B$ then $\mathrm{T}_{all}\mathrm{Ex}[A] \subseteq \mathrm{T}_{all}\mathrm{Ex}[B]$. For the converse direction, let S be the class in $\mathrm{Fin}[A]$ from Theorem 2.3.2, Equation (1):

$$f \in S \;\Leftrightarrow\; (\exists a_0, a_1, \ldots, a_n \notin A)\,(\exists b \in A)\,[f = a_0 a_1 \ldots a_n b^\infty].$$

This class is clearly also in $\mathrm{T}_{all}\mathrm{Ex}[A]$. By assumption there is a B-recursive learner M which Ex-learns every $f \in S$ and in addition converges on every function. So there is a $\sigma \in \overline{A}^*$ such that $M(\sigma\tau) = M(\sigma)$ for every $\tau \in \overline{A}^*$ — otherwise one could repeatedly extend any string by a further string from $\overline{A}^*$ such that a mind change occurs and so get a function on which M diverges. Having this σ there is an $a \notin A$ such that $a \neq \varphi_{M(\sigma)}(x)$ for the first $x \notin dom(\sigma)$. So one knows that $M(\sigma)$ is not the index of any function extending σa. Given any b, one knows by the choice of σ that there is no mind change on $\sigma a b^\infty$ for $b \notin A$ and that there is a mind change on $\sigma a b^\infty$ for $b \in A$ since $\sigma a b^\infty \in S$ and M infers this function. So one obtains the following condition which corresponds to (2):

$$b \in A \;\Leftrightarrow\; (\exists n)\,[M(\sigma a b^n) \neq M(\sigma)] \tag{4}$$

and A is enumerable relative to B by this formula. By the choice of A, A is computable relative to B. ▮

This result, as already indicated, does not go through for TEx in place of $\mathrm{T}_{all}\mathrm{Ex}$. But there is a subcase where it can be saved.

Theorem 2.4.7 *Let A, B have enumerable and nonhigh Turing degree. Then* $\mathrm{TEx}[A] \subseteq \mathrm{TEx}[B]$ *iff $A \leq_T B$.*

Proof If $A \leq_T B$ then $\mathrm{TEx}[A] \subseteq \mathrm{TEx}[B]$. For the converse direction, let $\mathrm{TEx}[A] \subseteq \mathrm{TEx}[B]$ and B be enumerable but not high. Without loss of generality, A can be taken as a coenumerable set within the given degree. Recall the class S from Theorem 2.3.2, Equation 1. One can now use the proof of the preceding theorem to show that $S \in \mathrm{TEx}[B]$ only if $B \geq_T A$. Only the Equation (4) needs an explicit proof and fails for some nonenumerable oracles — due to this fact, the present theorem is restricted to enumerable and nonhigh oracles. So assume by way of contradiction that

$$(\forall \sigma \in \overline{A}^*)\,(\exists \tau \in \overline{A}^*)\,[M^B(\sigma\tau) \neq M^B(\sigma)]. \tag{5}$$

Let $\sigma_0, \sigma_1, \ldots$ be an enumeration of $\overline{A}^*$ where σ_0 is the empty string λ. There is a B-recursive function g such that for every n there is a n' such that $n \leq n' \leq g(n)$ and, for every $m \leq n$, there is a $m' \leq n'$ with $\sigma_{m'} \succeq \sigma_m$ and $M^B(\sigma_{m'}) \neq M^B(\sigma_m)$ and all elements of B queried by some computation $M^B(\sigma_{m'})$ with $m' \leq n'$ are enumerated into B within time $g(n)$. Since B is not high, there is a recursive function h not dominated by the function $n \to g(g(n))$. Now let $\tau_0 = \sigma_0$ and τ_{n+1} be the first string $\sigma_{m'} \succeq \tau_n$ such that there is an $s \geq h(m)$ with $M^{B_s}(\sigma_{m'}) \neq M^{B_s}(\tau_n)$ and $s \geq m'$ where m is the index with $\sigma_m = \tau_n$.

The sequence $\tau_0, \tau_1, \ldots$ has a limit f which is a recursive function. By construction, the range of f is contained in $\overline{A}$.

Now it is shown that M^B diverges on f. Let $h(e) > g(g(e))$ for some e and let n be the maximal index of some $\tau_n = \sigma_m$ with $m \leq e$. Let k be the index of τ_{n+1}, that is, $\sigma_k = \tau_{n+1}$. Now either $k \geq g(m)$ or $g(k) \leq h(k)$.

In the first case $k \geq g(m)$, one has that there is an $s \geq k$ — which then also satisfies $s \geq g(m)$ — such that τ_{n+1} is the first string extending τ_n with $M^{B_s}(\tau_{n+1}) \neq M^{B_s}(\tau_n)$. Furthermore, by assumption there is a smallest index m' such that $\sigma_{m'}$ extends τ_n and $M^B(\sigma_{m'}) \neq M^B(\tau_n)$. Since $s \geq g(m)$ it follows that $M^{B_s}(\sigma_{m''}) = M^B(\sigma_{m''})$ for all $m'' \leq m'$ and one has that $k \geq m'$. On the other hand $k \leq m'$ since $M^{B_s}(\sigma_{m'}) = M^B(\sigma_{m'}) \neq M^B(\tau_n) = M^{B_s}(\tau_n)$ and τ_{n+1} is the first string $\sigma_{m'}$ with this property. So $M^B(\tau_{n+1}) \neq M^B(\tau_n)$ for this n.

In the second case $g(k) \leq h(k)$ one has that τ_{n+2} is the first string $\sigma_{m'}$ extending τ_{n+1} such that $M^{B_s}(\tau_{n+2}) \neq M^{B_s}(\tau_{n+1})$. Here now m is replaced by k and $n+1$ by n compared to the first case, but the argumentation is again the same since $s \geq h(k) \geq g(k)$. So $M^B(\tau_{n+2}) \neq M^B(\tau_{n+1})$ in this case.

Since there are infinitely many e such that $h(e) > g(g(e))$, it follows that there are infinitely many n such that τ_{n+1} has an index larger than e and that either $M^B(\tau_n)$ or $M^B(\tau_{n+2})$ differs from $M^B(\tau_{n+1})$. So M makes infinitely many mind changes on the function f. Since M^B converges on every recursive function, such an f cannot exist and the assumption (5) is false. This completes the proof. ▮

2.5 Behavioural Correct Learning

The problem, whether two hypotheses are equivalent, is undecidable. Sometimes a learner cannot detect whether the momentary hypothesis is consistent with the data seen so far and therefore, the learner does not know when to make a mind change. To overcome this problem, Bārzdins [11] proposed to require only semantic and not syntactic convergence.

Definition 2.5.1 *M learns behaviourally correct (BC) a function f iff $M(\sigma)$ is a program for f for almost all $\sigma \preceq f$.*

By definition, BC is more general than Ex. Bārzdins [11] showed that the inclusion is proper. The following example S of the almost self-describing functions [31] witnesses this fact:

$$f \in S \quad \Leftrightarrow \quad (\exists x)\,(\forall y > x)\,[\varphi_{f(0)}(y){\downarrow} = f(y)]. \tag{6}$$

A BC-learner for this class S produces for every input $\sigma = f(0)f(1)\ldots f(x)$ an hypothesis ψ_σ such that $\psi_\sigma(y) = \sigma(y)$ if $y \in dom(\sigma)$ and $\psi_\sigma(y) = \varphi_{f(0)}(y)$ if $y \notin dom(\sigma)$. If $x \in dom(\sigma)$ for the x in Equation (6) then the guess is total and correct: for arguments $y \in dom(\sigma)$, in particular for arguments $y \leq x$, the function $\psi_\sigma(y)$ takes the correct data from the input and for arguments $y \notin dom(\sigma)$ the function $\psi_\sigma(y)$ takes the value $\varphi_{f(0)}(y)$ which is correct by Equation (6). On the other hand, one cannot learn S under the criterion Ex since one cannot identify the places of diverging computations in the limit [31]. Kummer and Stephan [89, Theorem 8.1] showed that the class S is not even contained in Ex[A] for any nonhigh oracle A.

The inference degrees of BC behave on the enumerable and 1-generic oracles like Ex; also the trivial degree is the same. But on the PA-complete oracles, they are different:

every PA-complete oracle allows to learn the whole class REC under the criterion BC [43, 89].

Theorem 2.5.2 (a) REC $\in$ BC$[A]$ *for every* PA-*complete oracle* A [43, 89].
(b) BC$[A]$ = BC *iff* $A \leq_T B \leq_T K$ *for some 1-generic oracle* B [43].
(c) *If* A, B *are enumerable and not high then* BC$[A] \subseteq$ BC$[B] \Leftrightarrow A \leq_T B$ [89].
(d) *If* A, B *are 1-generic and not high then* BC$[A] \subseteq$ BC$[B] \Leftrightarrow A' \leq_T B'$ [89].

Proof Items (b), (c) and (d) are proved in a similar way as the corresponding results for Ex. A proof of (a) is given in [43] and included here for the readers convenience as one of the most relevant contributions of the author to this work. Since K is omniscient for Ex and therefore also for BC, the remaining interesting case is that $A \not\geq_T K$.

Let α and β be partial functions. α is *compatible with* β iff $(\forall x \in dom(\alpha) \cap dom(\beta))$ $[\alpha(x) = \beta(x)]$. α is *incompatible with* β otherwise. α *extends* β iff α is compatible with β and $dom(\beta) \subseteq dom(\alpha)$.

Let I be a finite subset of $\mathbb{N} \times \mathbb{N}$. Let $mam(I)$ be a program for the modified amalgamation of I which computes the function $\varphi_{mam(I)}(x)$ as follows: for all $(i, a) \in I$, run (by dovetailing) $\varphi_i(x)$; if an (i, a, t) is found such that $\varphi_{i,t}(x)\downarrow = y$ and $a \notin K_t$ then output y for the first such (i, a, t) found.

Assume $(i, a_1), \ldots, (i, a_m) \in I$, $x \in \mathbb{N}$ and $\varphi_i(x)\downarrow$. If $\{a_1, \ldots, a_m\} \cap \overline{K} \neq \emptyset$ then $\varphi_{mam(I)}(x)$ might be $\varphi_i(x)$. If $\{a_1, \ldots, a_m\} \subseteq K$ then $\varphi_{mam(I)}(x)$ can be $\varphi_i(x)$ only if at least one of the elements of $\{a_1, \ldots, a_m\}$ enters K after $\varphi_i(x)$ has converged. Consider now the following partial recursive function:

$$\gamma(i, j, a) = \begin{cases} 0 & \text{if } (\exists t)[a \in K_t \ \wedge \varphi_i \text{ extends } \varphi_{j,t}]; \\ 1 & \text{if } (\exists t)[a \notin K_t \ \wedge \\ & \qquad \varphi_i \text{ is incompatible with } \varphi_{j,t}]; \\ \uparrow & \text{otherwise.} \end{cases}$$

Assume $a \in K$. Then $\gamma(i, j, a)$ is trying to test if φ_i and φ_j are incompatible, but only allowing the test to use part of φ_j. In particular it can only use the part of φ_j that has been computed by the time a enters K. Assume $a \notin K$. Then $\gamma(i, j, a)$ will halt and output 1 if φ_i and φ_j are incompatible and diverge otherwise.

Since A is PA-complete, there is a $\{0, 1\}$-valued total A-recursive function g extending γ. This function g yields the following informations.

- If $g(i, j, a) = 1$ then $\gamma(i, j, a) \neq 0$. In the subcase $a \in K$, φ_i does not extend φ_j. In the subcase $a \notin K$ no information can be obtained.

- If $g(i, j, a) = 0$ then $\gamma(i, j, a) \neq 1$. Hence, for all t such that $a \notin K_t$, φ_i and $\varphi_{j,t}$ are compatible.

The inference-algorithm M works as follows:

To infer $f \in REC$, initialize $c(0) = 0$.
For all stages s and input $f_s = f(0)f(1)\ldots f(s)$ do the following:

- Let $I(s) = \{(j, a) : c(s) \leq j \leq s \wedge a \leq s \wedge g(c(s), j, a) = 0\}$.
- Output $M(f_s) = mam(I(s))$.

- If one of the conditions

  (I) $(\exists s' < s)[c(s') = c(s) \wedge \varphi_{M(f_{s'}),s}$ is incompatible with $f_s]$

  (II) $(\exists j, a, t \leq s)\,[a \in K_t \wedge f_s$ extends $\varphi_{j,t} \wedge g(c(s), j, a) = 1]$

  hold then let $c(s+1) = c(s) + 1$ else $c(s+1) = c(s)$.

For a given recursive f there is a least index i such that $\varphi_i = f$.

Claim 1 $(\forall s)[c(s) \leq i]$.

Assume there is a stage s such that $c(s) = i$. Now it is shown that the conditions (I) and (II) of the third item are not satisfied and therefore, $c(s+1) = i$:

(I): Assume, that $c(s') = c(s)$ and $\varphi_{M(f_{s'}),s}(x) \downarrow \neq f(x)$ for some $s', x \leq s$. Then there are j, a, y, t such that $\varphi_{j,t}(x) = y$, $a \notin K_t$, $y \neq f(x)$ and $g(i, j, a) = 0$. Since $\varphi_i(x) = f(x) \neq y$, φ_i is incompatible with $\varphi_{j,t}$ and so $\gamma(i, j, a) = 1$, contradicting that g extends γ.

(II): Let $a \in K_t$ and f_s extend $\varphi_{j,t}$. Then φ_i extends $\varphi_{j,t}$ as well and $g(i, j, a) = \gamma(i, j, a) = 0$. Thus $c(s)$ converges to some limit $k \leq i$. Say, $c(s) = k$ for all $s \geq s_0$.

Claim 2 $\varphi_{M(f_s)} = f$ for almost all stages s.

Since $c(s+1) = c(s)$ for all stages $s > s_0$, conditions (I) and (II) are never satisfied. Hence f extends $\varphi_{M(f_s)}$ for all these s.

Since $g(k, i, a) = 0$ for $a \in K$ by condition (II) and since $K \not\leq_T A$, there is some $a_0 \notin K$ such that $g(k, i, a_0) = 0$. For $s > s_0 + a_0$, $(i, a_0) \in I(s)$. For any x, there is some stage t and some y such that $\varphi_{i,t}(x) = y$ and $a_0 \notin K_t$. So (i, a_0, t, y) enforces that $\varphi_{M(f_s)} = \varphi_{mam(I(s))}$ is defined at x. Thus $\varphi_{M(f_s)}$ is total and $\varphi_{M(f_s)} = f$. ∎

Harrington [31] showed that the whole REC is learnable under the criterion BC* where M BC*-learns f iff $M(\sigma)$ computes a finite variant of f for almost all $\sigma \preceq f$. This result can also be proved using the fact of Adleman and Blum that REC is Ex-learnable with K-oracle. There is an approximation M_s to the Ex[K]-learner M and using this approximation one defines the BC*-learner N implicitly by

$$\varphi_{N(\sigma)}(x) = \varphi_{M_x(\sigma)}(x).$$

Now $\varphi_{N(\sigma)}(x)$ coincides with $\varphi_{M(\sigma)}(x)$ for almost all x since $M_x(\sigma) = M(\sigma)$ for almost all x. Given any $f \in$ REC, the Ex[K]-learner M outputs for almost all $\sigma \preceq f$ a program for f; then $N(\sigma)$ coincides with that program on all but finitely many places and thus N succeeds under the criterion BC* for all recursive functions.

2.6 Branch-Learning

Kummer and Ott [88, 116] studied a problem related to inductive inference: given data on a recursive infinite tree, find an infinite recursive branch of it. The most direct approach would be to learn a program for the tree from the data on the input — which presents for every node σ the information whether σ belongs to the tree or not — and then to compute a program for an infinite branch from the program for the tree. This direct approach does not always work as the following example shows.

Example 2.6.1 [88] *Let the class S contain every tree which has a unique root of form e in the sense that only nodes σ with $\sigma(0) = e$ are in the tree and for which φ_e is a program for an infinite branch. Then S is* BranchFin*-learnable but the task to find programs for the trees in S is as hard as to learn REC.*

Proof A BranchFin-learner has to output exactly one hypothesis which is an infinite branch of the tree whose data the learner sees. This is done by waiting for the first nonvoid node σ in the tree and then taking $\sigma(0)$ which is an index for an infinite branch of the tree by the choice of the trees in S.

The hardness follows from the fact, that, given a fixed self-describing function φ_e with $\varphi_e(0) = e$ and $\varphi_e(1) = 1$, one can now add for every function $f \in$ REC to the nodes $\varphi_e(0)\varphi_e(1)\ldots\varphi_e(x)$ the nodes $e0$ and $e0f(0)f(1)\ldots f(x)$ for all x. Given a learner M for the trees, one can design a learner N for REC which for any given f constructs the "joint tree of φ_e and f" and then extracts the guesses for f from those of programs for the joint trees. ∎

Now let BranchEx and BranchBC denote the learning criteria corresponding to Ex and BC in the sense that the learner converges syntactically or behaviourally correct to a branch of the given tree. BranchBC can be generalized to BranchWBC where the learner outputs almost always programs for branches of the given tree, but these branches are not required to be the same.

Ott and Stephan [117] studied for these notions also the amount of help, oracles provide during the learning process. High oracles are omniscient for BranchEx, since they allow to learn a program for the tree in the limit and then to find an infinite branch of them. The second step can be executed with the oracle K' by looking for the first index e of a total function such that $\varphi_e(0)\varphi_e(1)\ldots\varphi_e(x)$ is on the tree for all x. Since this search may be executed in the limit, any oracle A with $A' \geq_T K$ is sufficient to do this, so the second step also needs only a high oracle. Now let S contain for each recursive function f the tree containing exactly the nodes $f(0)f(1)\ldots f(x)$; then this class is in BranchEx[A] iff REC is in Ex[A] and therefore high oracles are also necessary for BranchEx-learning the given class S — so the oracles omniscient for BranchEx are exactly the high oracles.

More interesting is the question whether BranchBC and BranchWBC have nonhigh omniscient oracles. It turns out that it is impossible to generalize Theorem 2.5.2 (a) to branch-learning since the second step of the learning algorithm requires a high oracle as the following example shows.

Example 2.6.2 [117] *Let the e-th tree in the class S contain every node σ such that either $\sigma(0) = 0 \wedge (\forall n \in dom(\sigma))\,[\,|W_{e,\sigma(n)}| \geq n]$ or $\sigma(0) > |W_{e,|\sigma|}|$. Then S is in* BranchWBC[A] *iff A is high.*

Proof Since S is in BranchEx[A] for high oracles A, only the other direction is interesting.

If W_e has the finite cardinality n, then all infinite branches f of the e-th tree satisfy $f(0) > n$. If W_e is infinite then all infinite branches f of the e-th tree satisfy $f(0) = 0$. Furthermore, in both cases some infinite branches of the e-th tree are recursive.

If now M^A is a BranchWBC[A]-learner, then one can compute relative to A whether W_e is finite by simulating M^A on the e-th tree as input: Let $i_0, i_1, \ldots$ be the sequence of the output, almost all i_n are indices of some infinite branch of the e-th tree. Therefore $\varphi_{i_n}(0)\downarrow$ for almost i_n. Now one computes at stage s the maximum $k \leq s$ such that the program i_k converges within s computation steps at 0 to some value a_s. If $a_s = 0$ then one takes $b_s = 0$, otherwise $b_s = 1$. It follows from the construction that b_s converges to 0 if W_e is infinite and to 1 if W_e is finite. Thus one can compute relative to A in the limit which W_e are finite and which are infinite, therefore A is high. ∎

3 Oracles and Teachers

An oracle gives some general information which is independent of the current concept presented to the learner. This is not adequate for those learning situations where the learner has access to some interactive help which knows the current problem to be learned, for example for the situation of pupils and teachers in school. Angluin [3] as well as Gasarch and Smith [55] formalized this situation and introduced the concept of a teacher into learning theory: the teacher knows the concept and answers specific queries on it. The basic ideas behind both concepts can be illustrated using the famous example of the game Mastermind.

The teacher first selects the code — a quadruple of colours — that should be learned. Then the learner tries to figure out the code. In each round, the learner makes one guess what the code might be. This guess implicitly codes two questions: how many colours and how many positions of the guess are correct? The teacher supplies the answers to these two questions and then the learner starts the next round with a new guess based on the previous questions and answers. The learner succeeds if the correct code is found after a given number of rounds.

The learner may in addition consult an oracle, for example, a book or a database. Such a book is of course ignorant of the current code to be learned since it was pressed before the code was elected. But nevertheless the book may be helpful if it contains an algorithm of how to generate the queries. The power of this help depends on the quality of the algorithm. Now both phenomena, the teacher and the oracle, are combined and the power of the oracles is measured with respect to the different concepts of learning from a teacher.

As the example showed, the communication between learner and teacher follows a specific protocol. In the framework of inductive inference and in the absence of restrictions on the number of queries, one can restrict the query language to questions, which can be answered by "yes" or "no" and which have always a unique answer.

This removes nondeterminism from the behaviour of the teacher, so that some problems of Angluin's model [3] can be avoided directly: the teacher might use the ability to choose among different answers for either helping the learner by encoding syntactically the solution into the answers or spoil the learner by giving lengthy answers to destroy resource bounds. By the way, Angluin overcame the problem by requiring that the learner succeeds with all legal answers of the teacher and by computing the resource bounds relative to the size of the input and the size of the answers of the teacher.

Gasarch and Smith [55] studied learning from queries within inductive inference and considered query languages which used logical queries about functions like $(\exists x, y)\,[f(x) = 0 \wedge f(y) = 1]$ or $(\exists x)\,(\forall y)\,[f(y) = x]$.

Formally, a *query language* is a language that has the usual logical symbols (equality, constants, variables, quantifiers and logical operations to link the atom) and a special symbol f. Query languages may contain further symbols, in particular Succ, $<$ and $+$ which denote the successor-function $\mathrm{Succ}(x) = x+1$ on $\mathbb{N}$, the less-than relation $<$ on $\mathbb{N}$ and the addition $+$ on $\mathbb{N}$. These three languages — with one of the additional symbols — are denoted by $L[\mathrm{Succ}]$, $L[<]$ and $L[+]$. A *query* in the language is a formula without free variables; such formulas without free variables are called sentences. The intention is that a query $\phi(f)$ is asking about f. For example in the language $L[<]$ the query

$$(\exists x)\,(\forall y)\,[y < x \vee f(y) = 0]$$

is asking if at some point the function becomes the constant 0. $S \in \mathrm{QEx}[\star]$ means, that some machine infers S using the query-language $L[\star]$ where $\star$ denotes — in contrast to other papers on this field — one of the symbols Succ, $<$, $+$. Note that by introducing a new variable and one quantifier, Succ and $<$ can be expressed by $<$ and $+$, respectively:

$$\begin{aligned} x = \mathrm{Succ}(y) \quad &\Leftrightarrow \quad y < x \wedge (\forall z)\,[z < y \vee z = y \vee z = x \vee z > x]; \\ x < y \quad &\Leftrightarrow \quad (\exists z)\,[y = x + z + 1]. \end{aligned}$$

Indeed $\mathrm{Ex} \subset \mathrm{QEx}[\mathrm{Succ}] \subset \mathrm{QEx}[<,\mathrm{Succ}] = \mathrm{QEx}[<] \subset \mathrm{QEx}[+,<] = \mathrm{QEx}[+]$ holds. In principal, every learning criterion like Ex and Fin can be combined with queries from a given language $L[\star]$, but the main accent lies on the concepts $\mathrm{QEx}[\star]$ and $\mathrm{QFin}[\star]$ where in the first case finitely many revisions of the hypothesis are allowed in and the second case none.

The present work addresses the combination of *both* types of queries, those to oracles and those to teachers. Given some language $L[\star]$ and an oracle A, the learner may query either the teacher in the language $L[\star]$ about f or make a membership query to the oracle A. But the learner must not make combined queries; so queries like $(\exists x)\,[f(x) = 0 \wedge x \in A]$ are not permitted. The so obtained criteria induce in the same way as the traditional learning criteria an ordering on the oracles, for example, the QEx[Succ] degree of an oracle A contains all oracles B such that every class $S \subseteq \mathrm{REC}$ is learnable under the criterion $\mathrm{QEx}[\mathrm{Succ}; A]$ iff S is learnable under the criterion $\mathrm{QEx}[\mathrm{Succ}; B]$.

Gasarch was in particular interested to obtain characterizations for the trivial and omniscient degrees of query inference, which had been left open while the same questions where solved for most other learning criteria [43]. Within the next sections, the omniscient degree of $\mathrm{QEx}[\star]$ is shown to consist exactly of the high degrees and so to coincide with that one of Ex. The trivial degree is a proper subset of the trivial degree of Ex. After that noninclusions like $\mathrm{QEx}[<] \not\subseteq \mathrm{QEx}[\mathrm{Succ}]$ are strengthened in the way that there is a family $S \in \mathrm{QEx}[<]$ which is not contained in $\mathrm{QEx}[\mathrm{Succ}; B]$ for any nonomniscient oracle B. Further results in this chapter deal with generalizations of QEx like QBC to one direction and QFin to the other directions; also it is investigated to which extent one can replace Succ by some other fixed function F.

3.1 Some Useful Facts

The separation results within this chapter use the notion of "$k_+(n)$-good functions" which had been introduced by Gasarch, Pleszkoch and Solovay [52]. They showed that for an $k_+(n)$-good function the truth-value $\phi(f)$ can be computed from some suitable initial segment of f. This behaviour is, that f is $\{0,1\}$-valued and takes infinitely often both values but f takes them only on a very thin sequence of arguments. The intuitive idea is that it is then almost impossible to distinguish which x take $f(x) = 1$ with small formulas.

Definition 3.1.1 A function f is $k_+(n)$-good if the following holds for $m = k_+(n)$:

(I) $(\exists^\infty x)\,[f(x) = 1]$;
(II) $(\forall x > n)\,[f(x) = 1 \Rightarrow m! \text{ divides } x]$;
(III) $(\forall x > n)\,(\forall y > x)\,[f(x) = 1 \wedge f(y) = 1 \Rightarrow x \cdot m \leq y]$.

The formula $m!$ denotes the product $1 \cdot 2 \cdot \ldots \cdot m$.

The function k_+ can be chosen such that it is possible to decide any queries above a function f with the knowledge of only finitely many values of f.

Fact 3.1.2 [52, Theorem 11] *There are recursive functions k_+ and $truth_+$ such that for every sentence $\phi \in L[+]$ and every $\{0,1\}$-valued function f: If f is $k_+(n)$-good and $n \geq |\phi|$ then $\phi(f) \Leftrightarrow truth_+(n, \phi, f(0), f(1), \ldots, f(n))$.*

Gasarch, Pleszkoch and Solovay [52] formulated Fact 3.1.2 a little bit different; they computed for each $\sigma = f(0), f(1), \ldots, f(n)$ and for each sentence ϕ a value $k(\phi, \sigma)$ depending on ϕ and σ instead of n. So the function k_+ used here is derived from their function k by the formula $k_+(n) = \max\{k(\phi, \sigma) : |\phi| \leq n \wedge |\sigma| \leq n+1\}$ where ϕ ranges over all sentences in $L[+]$ and σ over all strings in $\{0,1\}^*$. It is easy to see that every $k_+(n)$-good function f is also $k(\phi, \sigma)$-good for all sentences ϕ and strings σ with $|\phi| \leq n$ and $|\sigma| \leq n+1$; the formulation of Fact 3.1.2 given above is more suitable for the present work.

Sometimes it is necessary to make functions "inaccessable" via formulas in the language $L[<]$ while they can be "accessed" via formulas in $L[+]$. Such functions are the $k_<(n)$-good functions which will be used in Theorems 3.3.4, 3.4.1 and 3.6.4.

Definition 3.1.3 A function f is $k_<(n)$-good if the following holds for $m = k_<(n)$:

(IV) $(\exists^\infty x)\,[f(x) = 1]$;
(V) $(\forall x > n)\,[f(x) = 1 \Rightarrow m < x]$;
(VI) $(\forall x > n)\,(\forall y > x)\,[f(x) = 1 \wedge f(y) = 1 \Rightarrow x + m \leq y]$.

Fixing the function $k_<$ in a suitable way it is possible to compute the truth-value for any $k_<(n)$-good function and any formula in $L[<]$ of size at most n only from the first $n+1$ values of f.

Fact 3.1.4 *There are recursive functions $k_<$ and $truth_<$ such that for every sentence $\phi \in L[<]$ and every $\{0,1\}$-valued function f: If f is $k_<(n)$-good and $n \geq |\phi|$ then $\phi(f) \Leftrightarrow truth_<(n, \phi, f(0), f(1), \ldots, f(n))$.*

These recursive functions $k_+, k_<, truth_+, truth_<$ are used in the following sections. Without loss of generality, $k_<(n) < k_<(n+1)$, $k_+(n) < k_+(n+1)$ and $n < k_<(n) \leq k_+(n)$ for all n.

3.2 The Omniscient Degree

Fortnow et al. [43] left open the problem to determine the omniscient inference degree of the criteria QEx[Succ], QEx[<] and QEx[+]. Since all Ex-omniscient oracles are QEx[$\star$]-omniscient, only the other direction is interesting: whether there are QEx[$\star$]-omniscient oracles, which are not Ex-omniscient. The following theorem gives a negative answer: if $\mathrm{Ex}[A] \not\subseteq \mathrm{Ex}[B]$ then also $\mathrm{QEx}[\star; A] \not\subseteq \mathrm{QEx}[\star; B]$. Thus the QEx[$\star$]-degrees are a refinement of the Ex-degrees and their omniscient degree is a subset of the omniscient Ex-degree; indeed these omniscient degrees are equal.

Theorem 3.2.1 *If* $\mathrm{Ex}[A] \not\subseteq \mathrm{Ex}[B]$ *then* $\mathrm{Ex}[A] \not\subseteq \mathrm{QEx}[+; B]$.

Proof The basic idea of this proof is to define a recursive transformation Γ such that for every $S \subseteq \mathrm{REC}_{0,1}$ the transformed set $\tilde{S} = \{\Gamma(g) : g \in S\}$ has the following properties:

$$\tilde{S} \subseteq \mathrm{REC}_{0,1}. \tag{7}$$
$$S \in \mathrm{Ex}[A] \Rightarrow \tilde{S} \in \mathrm{Ex}[A]. \tag{8}$$
$$S \notin \mathrm{Ex}[B] \Rightarrow \tilde{S} \notin \mathrm{Ex}[B]. \tag{9}$$
$$\tilde{S} \notin \mathrm{Ex}[B] \Rightarrow \tilde{S} \notin \mathrm{QEx}[+; B]. \tag{10}$$

Choosing $S \in \mathrm{Ex}[A] - \mathrm{Ex}[B]$ gives directly a witness $\tilde{S} \in \mathrm{Ex}[A] - \mathrm{QEx}[+;B]$.

The transformation makes use of Fact 3.1.2 in the way that on one hand there is an increasing recursive sequence $a_0, a_1, \ldots$ such that the transformed function $\Gamma(g)$ is $k_+(a_n)$-good for all n and on the other hand $\Gamma(g)$ codes g in a very easy way so that (8) and (9) can be guaranteed. The sequence $a_0, a_1, \ldots$ and the transformation Γ are defined as follows:

$$\begin{aligned} a_0 &= 0; \\ a_{n+1} &= (k_+(a_n))!; \\ f(x) &= \Gamma(g)(x) = \begin{cases} g(m) & \text{if } x = a_{2m} \text{ for some } m; \\ 1 & \text{if } x = a_{2m+1} \text{ for some } m; \\ 0 & \text{otherwise, that is, } x \notin \{a_0, a_1, a_2, \ldots\}. \end{cases} \end{aligned}$$

Let $S \subseteq \mathrm{REC}_{0,1}$. Obviously $\tilde{S} \subseteq \mathrm{REC}_{0,1}$ so (7) holds. (8) follows since g can be recovered from $f = \Gamma(g)$ by $g(m) = f(a_{2m})$: Let M witness $S \in \mathrm{Ex}[A]$. The new learner N to witness $\tilde{S} \in \mathrm{Ex}[A]$ computes on input $f(0), f(1), \ldots, f(a_{2m})$ the values $g(0), g(1), \ldots, g(m)$, then N simulates M to compute M's current guess $e = M(g(0), g(1), \ldots, g(m))$ and finally N outputs an index $h(e)$ for $\Gamma(\varphi_e)$; such a recursive function h exists since the sequence $a_0, a_1, \ldots$ is recursive. It is easy to see that N converges on $\Gamma(g)$ to $h(e)$ iff M converges on g to e; thus if M infers S then N infers $\tilde{S}$. So (8) holds. Similarly one can show that there is also an inverse translation of the inductive inference machines and (9) follows from its contrapositive. It remains to show (10):

First it is shown that every $f = \Gamma(g)$ is $k_+(a_n)$-good for all n: Since $f(a_{2m+1}) = 1$ for all m, (I) holds. If $x > a_n$ and $f(x) = 1$ then $x = (k_+(a_m))!$ for some $m \geq n$, $(k_+(a_n))!$ divides x (since $a_m \geq a_n$) and $k_+(a_m) \geq k_+(a_n)$. Thus (II) holds. If $m \geq n$ then $a_{m+1} = (k_+(a_m))! \geq k_+(a_m) \cdot (k_+(a_m) - 1) \geq k_+(a_m) \cdot a_m \geq k_+(a_n) \cdot a_m$. Thus also (III) is satisfied. So f is $k_+(a_n)$-good for all n. Since $a_{|\phi|} \geq |\phi|$, the relation

$$\phi(f) \;\Leftrightarrow\; \mathit{truth}_+(a_{|\phi|}, \phi, f(0), f(1), \ldots, f(a_{|\phi|}))$$

enables to answer any query ϕ to f by analyzing the prefix $f(0)f(1)\ldots f(a_{|\phi|})$ of f. So the $\mathrm{QEx}[+;B]$ inference algorithm can be translated into an $\mathrm{Ex}[B]$ algorithm, that is, $\tilde{S} \in \mathrm{QEx}[+;B] \Rightarrow \tilde{S} \in \mathrm{Ex}[B]$ and $\tilde{S} \notin \mathrm{Ex}[B] \Rightarrow \tilde{S} \notin \mathrm{QEx}[+;B]$. This finishes the proof of (10).

If $\mathrm{Ex}[A] \not\subseteq \mathrm{Ex}[B]$ then this is witnessed by some $S \subseteq \mathrm{REC}_{0,1}$. By (8) and (9), also $\tilde{S} \in \mathrm{Ex}[A] - \mathrm{Ex}[B]$. From (10) it follows that $\tilde{S} \notin \mathrm{QEx}[+;B]$. Therefore the noninclusion $\mathrm{Ex}[A] \not\subseteq \mathrm{QEx}[+;B]$ holds. ▮

The inclusions $\mathrm{Ex}[C] \subseteq \mathrm{QEx}[\mathrm{Succ};C] \subseteq \mathrm{QEx}[<;C] \subseteq \mathrm{QEx}[+;C]$ hold for all oracles C. Therefore every noninclusion in the Ex-degrees induces a noninclusion in the $\mathrm{QEx}[\star]$-degrees. Also if $\mathrm{REC} \in \mathrm{Ex}[C]$ then $\mathrm{REC} \in \mathrm{QEx}[\star;C]$ and the omniscient degree of the criteria Ex, QEx[Succ], QEx[<], QEx[+] coincides:

Theorem 3.2.2 *If* $\mathrm{Ex}[A] \not\subseteq \mathrm{Ex}[B]$ *then*
(a) $\mathrm{QEx}[+;A] \not\subseteq \mathrm{QEx}[+;B]$;
(b) $\mathrm{QEx}[<;A] \not\subseteq \mathrm{QEx}[<;B]$;
(c) $\mathrm{QEx}[\mathrm{Succ};A] \not\subseteq \mathrm{QEx}[\mathrm{Succ};B]$.
Furthermore $\mathrm{QEx}[+;A]$, $\mathrm{QEx}[<;A]$ *and* $\mathrm{QEx}[\mathrm{Succ};A]$ *are omniscient iff* A *is high.*

3.3 The Trivial Degree

To characterize the trivial QEx[$\star$]-degrees is still an open problem. However it is shown that these trivial QEx[$\star$]-degrees are different from the trivial Ex-degree, indeed they are a proper subset of it.

This result is based on the idea of coding some nonrecursive enumerable set B uniformly into the graph of each $f \in S$ for some $S \notin \text{QEx}[\star; B]$. From the fact that any learner can recover B with queries about f it follows that $\text{QEx}[\star; A] = \text{QEx}[\star; A \oplus B]$ for all A. Now there is a 1-generic set $A \leq_T K$ such that $A \oplus B \equiv_T K$. This A has nontrivial QEx[$\star$]-degree since $S \in \text{Ex}[K] \subseteq \text{QEx}[\star; A \oplus B]$, but A has trivial Ex-degree. So it follows using Theorem 3.2.2, that the trivial QEx[$\star$]-degree is properly contained in the trivial Ex-degree.

Again the basic idea of the construction is defining a transformation Γ^B depending on an enumerable nonhigh and nonrecursive oracle B. Starting with the set $\text{REC}_{0,1}$ of all recursive $\{0,1\}$-valued functions, the transformed set $S = \{\Gamma^B(g) : g \in \text{REC}_{0,1}\}$ has the following properties:

$$S \subseteq \text{REC}_{0,1}. \tag{11}$$

$$A \oplus B \geq_T K \Rightarrow S \in \text{QEx}[\star; A]. \tag{12}$$

$$S \notin \text{Ex}[B]. \tag{13}$$

$$S \notin \text{QEx}[\star; B] \text{ and } S \notin \text{QEx}[\star]. \tag{14}$$

Posner and Robinson [110] showed that every B below K can be lifted to K by joining a 1-generic set.

Fact 3.3.1 *Given any nonrecursive set $B \leq_T K$ there is a 1-generic set $A \leq_T K$ such that $A \oplus B \equiv_T K$.*

Using this 1-generic set A, it will come out that on one hand $A \oplus B \equiv_T K$ and therefore $S \in \text{QEx}[\star; A]$ but on the other hand $S \notin \text{QEx}[\star]$, so A does not have trivial QEx[$\star$]-degree, but A has trivial Ex-degree.

Now the proof in detail. Lemma 3.3.2 is the first step in obtaining such a set B and transformation Γ^B. B is chosen as some kind of deficiency set of nonhigh nonrecursive degree; h is an auxiliary function used below. This special form of B makes it easier to define Γ^B in the proofs of Theorems 3.3.3 and 3.3.4.

Lemma 3.3.2 *In every enumerable Turing degree there is an enumerable set B and a recursive function h such that*
(a) $B = \{n : (\exists m > n)\,[h(m) \leq n]\}$ *and*
(b) $(\forall n)\,[h(n) \leq n]$.

Proof Let f be a recursive 1-1 enumeration of some enumerable set C representing the given enumerable Turing degree. For each s define

$$\begin{aligned} g(s) &= \text{the } f(s)\text{-th element of } \overline{B_s}; \\ h(s) &= \min\{g(s), s\}; \\ B_s &= \{n \leq s : (\exists m > n)\,[m \leq s \wedge h(m) \leq n]\}. \end{aligned}$$

Now it is shown that $B = \bigcup_s B_s$ has the desired properties. It is easy to see that $B = \{n : (\exists m > n)\,[h(m) \leq n]\}$ and $(\forall n)\,[h(n) \leq n]$ but it remains the more difficult part to prove that $B \equiv_T C$.

$B \leq_T C$ since B is permitted by C: Any new element n enters B at stage s only if $h(s) \leq n < s$ and $h(s)$ is the $f(s)$-th element of $\overline{B_s}$, that is, n is permitted by the new element $f(s)$ of C with $f(s) \leq h(s) \leq n$.

$C \leq_T B$: This part needs first to show that $\overline{B}$ is infinite. For any c there is some s with $f(n) \geq c$ for all $n \geq s$. Since B_s is finite, $\overline{B_s}$ is infinite and its least c elements do never enter B, so it follows that $\overline{B}$ is infinite. Now it is shown that

$$c \in C \quad \Leftrightarrow \quad c \in \{f(0), f(1), \ldots, f(s+c+1)\} \tag{15}$$

where s is the first stage such that the $c+1$ least elements of $\overline{B}$ and $\overline{B_s}$ are identical: If $c \notin C$ then also $c \notin \{f(0), f(1), \ldots, f(s+c+1)\}$ since f is an enumeration of C. If $c \in C$ then there is some t with $c = f(t)$. Let d be the $c+1$-st element of $\overline{B}$ and $\overline{B_s}$. Since B_s contains only elements below s, $d \leq s+c+1$. If $t > s+c+1$ then $h(t) = g(t) < d < t$ and d would be enumerated to B in contradiction to the choice of d. Thus $t \leq s+c+1$. So the equivalence (15) follows. Since s is computed using the oracle B and since f is computable without any oracle, the question whether $c \in C$ is decided using the oracle B. So $C \leq_T B$. ∎

Using Lemma 3.3.2 and Theorems 3.3.3 and 3.3.4, it is possible to show that there is a set A of nontrivial query inference degree but of trivial Ex-degree. Theorem 3.3.3 has a full proof for the case of $L[+]$ while Theorem 3.3.4 deals with the remaining cases $L[\text{Succ}]$ and $L[<]$ and its proof shows only the differences to the proof of Theorem 3.3.3.

Theorem 3.3.3 *There is a 1-generic set $A \leq_T K$ such that* $\text{QEx}[+; A] \not\subseteq \text{QEx}[+]$.

Proof Let B and h be as in Lemma 3.3.2 and let B have nonhigh and nonrecursive Turing degree. Further let k be as in Theorem 3.2.1 and let A be as in Fact 3.3.1. The modification from Γ to Γ^B is implemented implicitly by modifying the sequence $a_0, a_1, \ldots$ from Theorem 3.2.1; the way Γ^B is derived from the modified sequence is exactly analogous to the way Γ is derived from the original sequence.

$$\begin{aligned}
a_0 &= 0; \\
a_{n+1} &= \begin{cases} (k_+(a_n))! + a_{2h(m)} & \text{if } n = 2m \text{ and } h(m) < m; \\ (k_+(a_n))! & \text{otherwise.} \end{cases} \\
f(x) &= \Gamma^B(g)(x) = \begin{cases} g(m) & \text{if } x = a_{2m} \text{ for some } m; \\ 1 & \text{if } x = a_{2m+1} \text{ for some } m; \\ 0 & \text{otherwise, that is, } x \notin \{a_0, a_1, a_2, \ldots\}. \end{cases}
\end{aligned}$$

The basic idea of the construction is that the queries in $L[+]$ to f — independent of g — have the same Turing degree as B. For each sentence ϕ there is some $m > |\phi|$ such that $m \notin B$. For these $m \notin B$ (but not for all m), f is $k_+(a_{2m})$-good and $\phi(f)$ is equivalent to $truth_+(a_{2m}, \phi, f(0), f(1), f(2), \ldots, f(a_{2m}))$. On the other hand every membership query to B can be answered via logical queries about any $f = \Gamma^B(g)$. Recall the four conditions from the beginning of the section, now applied for $L[\star] = L[+]$:

$$S \subseteq \text{REC}_{0,1}. \tag{11}$$

$$A \oplus B \geq_T K \Rightarrow S \in \text{QEx}[+; A]. \tag{12}$$

$$S \notin \text{Ex}[B]. \tag{13}$$

$$S \notin \text{QEx}[+; B] \text{ and } S \notin \text{QEx}[+]. \tag{14}$$

Now it is shown that $S = \{\Gamma^B(g) : g \in \text{REC}_{0,1}\}$ satisfies these four conditions and thus witnesses the noninclusion: $S \in \text{QEx}[A; +] - \text{QEx}[+]$. (11): This item follows directly from the construction.

(12): $A \oplus B \geq_T K \Rightarrow S \in \mathrm{Ex}[A \oplus B]$: $S \in \mathrm{Ex}[K]$ since K is an omniscient oracle. Now any query to K can be computed via some membership queries to A and B. Those to A can be executed directly, but those to B have to be translated into logical queries about f. This is possible without any a priori knowledge on f beyond the fact that $f \in S$, that is, the queries use only the fact that $f \in S$ but not which particular function f is:

$$\begin{aligned} m \in B &\Leftrightarrow (\exists n > m)\,[h(n) \leq m] \\ &\Leftrightarrow (\exists n > m)\,[a_{2n+1} \not\equiv 0 \text{ modulo } a_{2m+1}] \\ &\Leftrightarrow (\exists x > a_{2m+1})\,[f(x) = 1 \text{ and } x \not\equiv 0 \text{ modulo } a_{2m+1}] \\ &\Leftrightarrow (\exists y, z)\,[f(a_{2m+1} \cdot y + z) = 1 \wedge y \neq 0 \wedge \\ &\qquad\qquad (z = 1 \vee z = 2 \vee \ldots \vee z = a_{2m+1} - 1)]. \end{aligned}$$

Multiplications with constants can be expressed in the language $L[+]$, for example, $5 \cdot x$ as $x + x + x + x + x$. The constant a_{2m+1} can be computed from m. Thus the whole query is effectively equivalent to a sentence in $L[+]$ and the $\mathrm{Ex}[A \oplus B]$ inference process can be simulated by a $\mathrm{QEx}[+; A]$ inference machine. Therefore $A \oplus B \geq_T K \Rightarrow S \in \mathrm{QEx}[+; A]$.

(13): Assume by way of contradiction that $S \in \mathrm{Ex}[B]$ via M. Then a new $\mathrm{Ex}[B]$-learner N translates the input function g to $f = \Gamma^B(g)$, computes $e = M(f(0), f(1), \ldots, f(a_{2m}))$ and outputs an index $u(e)$ for $\varphi_{u(e)}(x) = \varphi_e(a_{2x})$. It is easy to see that N witnesses $\mathrm{REC}_{0,1} \in \mathrm{Ex}[B]$ since $\Gamma^B(g) \in S$ for every $g \in \mathrm{REC}_{0,1}$. Then B has to be high in contradiction to the choice of B, thus $S \notin \mathrm{Ex}[B]$.

(14): This part is based on the following observation: If $m \notin B$ then $\Gamma(g)$ is $k_+(a_{2m+1})$-good. Since $f(a_n) = 1$ for all odd n, f takes infinitely often the value 1 and satisfies (I). Since $h(n) > m$ for all $n > m$, all $n \geq 2m + 1$ either satisfy $a_{n+1} = (k_+(a_n))!$ or $a_{n+1} = (k_+(a_n))! + a_{2l}!$ for some $l > m$; note that $a_{2l} = (k_+(a_{2l-1}))!$. In both cases $(k_+(a_{2m+1}))!$ divides a_{n+1} and (II) holds. Furthermore, $a_{n+1} \geq (k_+(a_n))! \geq k_+(a_{2m+1}) \cdot a_n$ for all $n \geq 2m + 1$, thus also (III) holds.

Therefore for any sentence $\phi \in L[+]$ there is some $m \notin B$ such that $|\phi| \leq a_{2m+1}$. Since $f = \Gamma^B(g)$ is $k_+(a_{2m+1})$-good, the value $\phi(f)$ is equivalent to $truth_+(a_{2m+1}, \phi, f(0), \ldots, f(a_{2m+1}))$ and the query can be answered from $f(0), \ldots, f(a_{2m+1})$. Thus $S \in \mathrm{QEx}[+] \Rightarrow S \in \mathrm{Ex}[B]$. Note that the oracle B cannot be removed since it is necessary to find the index $2m + 1$.

So all 4 items hold. By Fact 3.3.1 it is possible to select a 1-generic set $A \leq_T K$ such that $A \oplus B \equiv_T K$. Then $S \in \mathrm{QEx}[+; A] - \mathrm{QEx}[+]$ and A does not have trivial $\mathrm{QEx}[+]$-degree. ∎

Theorem 3.3.4 *There is a 1-generic set $A \leq_T K$ such that* $\mathrm{QEx}[\mathrm{Succ}; A] \not\subseteq \mathrm{QEx}[<]$.

Proof This proof is similar to that of Theorem 3.3.3. The sets A and B as well as the function h are the same, but it is necessary to use Fact 3.1.4 instead of Fact 3.1.2 and to adapt Γ^B.

$$\begin{aligned} a_0 &= 0; \\ a_{n+1} &= \begin{cases} a_n + k_<(a_{3h(m)+1}) & \text{if } n = 3m + 1 \text{ for some } m; \\ a_n + k_<(a_n) & \text{otherwise, that is, } n = 3m \text{ or } n = 3m + 2. \end{cases} \\ f(x) &= \Gamma^B(g)(x) = \begin{cases} g(m) & \text{if } x = a_{3m} \text{ for some } m; \\ 1 & \text{if } x = a_{3m+1} \text{ or } x = a_{3m+2} \text{ for some } m; \\ 0 & \text{otherwise, that is, } x \notin \{a_0, a_1, a_2, \ldots\}. \end{cases} \end{aligned}$$

Again f is $k_<(a_{3m})$-good whenever $m \notin B$. So any $L[<]$-query to f can be answered using B-oracle and a sufficiently long prefix of f.

On the other hand, $m \in B$ iff there is some $c \in \{k_<(a_0), k_<(a_1), \ldots, k_<(a_{3m+2})\}$ such that $(\exists x > a_{3m+1})\,[f(x) = 1 \wedge f(x+c) = 1]$ holds. So B can be recovered via $3m$ queries in $L[\mathrm{Succ}]$ to f. The rest of the proof follows the lines of Theorem 3.3.3. ∎

The following Theorem summarizes the results of this section:

Theorem 3.3.5 *There is a set A such that*
(a) $\mathrm{Ex}[A] = \mathrm{Ex}$;
(b) $\mathrm{QEx}[+; A] \not\subseteq \mathrm{QEx}[+]$;
(c) $\mathrm{QEx}[<; A] \not\subseteq \mathrm{QEx}[<]$;
(d) $\mathrm{QEx}[\mathrm{Succ}; A] \not\subseteq \mathrm{QEx}[\mathrm{Succ}]$.

3.4 Complete Families

The notion of a complete family S is normally used with respect to some notion of reduction such that all families inferable under a given criterion are reducible to S. The complete families considered here are not complete with respect to an inference criterion but with respect to a noninclusion, that is, a family S is called complete with respect to the noninclusion $\mathrm{QEx}[+] \not\subseteq \mathrm{QEx}[<]$ iff $S \in \mathrm{QEx}[+]$ and $S \notin \mathrm{QEx}[<; A]$ for every oracle A of nonomniscient $\mathrm{QEx}[<]$-degree. In this section it is shown that there are complete families with respect to to the noninclusions $\mathrm{QEx}[+] \not\subseteq \mathrm{QEx}[<]$, $\mathrm{QEx}[<] \not\subseteq \mathrm{QEx}[\mathrm{Succ}]$ and $\mathrm{QEx}[\mathrm{Succ}] \not\subseteq \mathrm{Ex}$. As a corollary one obtains that $\mathrm{Ex}[A] \subset \mathrm{QEx}[\mathrm{Succ}; A] \subset \mathrm{QEx}[<; A] \subset \mathrm{QEx}[+; A]$ for all nonhigh oracles A.

Theorem 3.4.1 *There is a family $S \in \mathrm{QEx}[+]$ which is not $\mathrm{QEx}[<; A]$ learnable for any nonomniscient A.*

Proof This family S is the same as in Theorem 3.3.3 with the only difference that B is chosen to have the same degree as K. The proof of (12) does not require that B is nonhigh and therefore can be adapted to show $S \in \mathrm{QEx}[+]$. The oracle B can be recovered by the same existential queries about f as in Theorem 3.3.3:

$$\begin{aligned} m \in B \;\Leftrightarrow\; (\exists x, y)\; [\,& f(x \cdot a_{2m+1} + y) = 1 \;\wedge x \neq 0 \;\wedge \\ & (y = 1 \vee y = 2 \vee \ldots \vee y = a_{2m+1} - 1)\,] \end{aligned}$$

Note that a_{2m+1} in this sentence is a constant. So inferring any $f \in S$, the learner has access to B via the above coded membership queries and since B belongs to the omniscient Ex-degree, the learner identifies f. Thus $S \in \mathrm{QEx}[+]$.

Now it is shown that every $f \in S$ has the property $k_<(a_n)$-good from Fact 3.1.4 for all n: The functions $f \in S$ take the value 1 exactly on some infinite subset of $\{a_0, a_1, \ldots\}$. So (IV) holds, for (V) and (VI) it is necessary to look at the sequence $a_0, a_1, \ldots$; by construction all $m \geq n$ satisfy: $a_{m+1} \geq (k_+(a_m))! \geq (k_+(a_m) - 1) \cdot k_+(a_m) \geq a_m \cdot k_+(a_m) \geq a_m + k_+(a_m) \geq a_m + k_<(a_m) \geq a_m + k_<(a_n)$. This implies (IV) since $a_m \geq a_{n+1} \geq a_n + k_<(a_n)$ and (V) since for all $x > a_n$ and $y > x$ with $f(x) = 1 \wedge f(y) = 1$ there is an m with $x = a_m$ and $y \geq a_{m+1} \geq x + k_<(x)$.

So each $f \in S$ is $k_<(a_n)$-good for all n and therefore each $L[<]$-query ϕ to f can be decided after reading $f(0), \ldots, f(a_{|\phi|})$. Thus $S \in \mathrm{QEx}[<; A] \Rightarrow S \in \mathrm{Ex}[A]$. Furthermore, $S \in \mathrm{Ex}[A] \Rightarrow \mathrm{REC}_{0,1} \in \mathrm{Ex}[A]$ and if $S \in \mathrm{QEx}[<; A]$ then A is high, that is, A has omniscient $\mathrm{QEx}[<]$-degree. ∎

The next theorem separates the class QEx[<] from the class QEx[Succ; A]. It has a more complicated proof. In Theorem 3.4.1 only existential queries are necessary to recover B from f. But every existential query in $L[<]$ to a $\{0,1\}$-valued function f can be replaced by a sequence of existential queries in L[Succ] to f, or in other words, if $S \in REC_{0,1} \cap \mathrm{Q_1Ex}[<]$ then $S \in \mathrm{Q_1Ex[Succ]}$ where $\mathrm{Q_1Ex}[\star]$ means that queries of the inference process may only contain existential quantifiers but not universal quantifiers. There are two ways to overcome this difficulty: either by dropping the requirement that $S \subseteq \mathrm{REC}_{0,1}$ and considering a family like

$$\{f : f = \varphi_{f(0)}\} \cup \{f : f = \varphi_x \text{ where } x = \max\{y : f(y) > y\}\}$$

or by coding the oracle K in f such that it cannot be recovered via existential queries in $L[<]$. The proof of Theorem 3.4.2 goes the second way.

Theorem 3.4.2 *There is a set $S \in$ QEx[<] with $S \notin$ QEx[Succ; A] for all nonhigh oracles A.*

Proof Let K_s be an enumeration of K. Using K_s the following approximation h to $\overline{K}$ is defined: Let $h(x,s) = 0$ if $x \in K_s$ and $h(x,s) = 1$ otherwise ($x \notin K_s$). Given any function $g \in \mathrm{REC}_{0,1}$ the function $f = \Gamma(g)$ is inductively defined as $\sigma_0\sigma_1\sigma_2\ldots$ with

$$\sigma_s = 0^s 1 0^s g(s) 0^s\ 1 0^0 h(0,s) 0^s\ 1 0^1 h(1,s) 0^s\ 1 0^2 h(2,s) 0^s\ \ldots\ 1 0^s h(s,s) 0^s.$$

This construction is used to code g and K into $f = \Gamma(g)$. Since the length of the σ_s does not depend on g but only on s and since $g(s) = \sigma_s(2s{+}1)$, the σ_s — and therefore also g — can be recovered from f. Furthermore, each σ_s contains for all $x \leq s$ the substring $10^x h(x,s)$: Thus if all $h(x,s) = 1$, that is, if $x \notin K$, then 10^s1 occurs in all strings σ_s with $s \geq x$. Otherwise $h(x,s) = 0$ for almost all s and 10^x1 is not a substring of σ_s for almost all s. So $x \in K$ iff the substring 10^x1 appears only finitely often in f iff $f(y) = 0 \vee f(y+x+1) = 0$ for almost all y. In short

$$x \in K \ \Leftrightarrow\ (\exists z)\,(\forall y > z)\,[f(y) = 0 \vee f(y+x+1) = 0].$$

This query is in L[Succ, <] since x is a constant, for example, if $x = 3$ then $y + x + 1$ is an abbreviation for Succ(Succ(Succ(Succ(y)))). Now let

$$S = \{\Gamma(g) : g \in \mathrm{REC}_{0,1}\}.$$

$S \in$ QEx[Succ, <] = QEx[<] since queries to K can be coded as queries in L[Succ; <] about f. So it remains to show that $S \notin$ QEx[Succ; A] for every nonhigh oracle A. This is done by showing that if $S \in$ QEx[Succ; A] then $S \in$ Ex[A]. In particular it is sufficient to show that every L[Succ]-query ϕ to any $f \in S$ can be decided effectively by analyzing a sufficiently long prefix of f.

The algorithm step-wise eliminates all quantifiers from the formula. It proceeds as follows, where ψ, ψ_1, ψ_2 always denote quantifier-free subformulas of ϕ.

(a) $(\forall u)\,[\psi(u, x_1, \ldots, x_n)]$ occurs in ϕ:
Then $(\forall u)\,[\phi(u, x_1, \ldots, x_n)]$ is replaced by $\neg((\exists u)\,[\neg\phi(u, x_1, \ldots, x_n)])$.

(b) $(\exists u)\,[\psi(u, x_1, \ldots, x_n)]$ occurs in ϕ and ψ is not in disjunctive normal form:
ψ is transformed into disjunctive normal form.

(c) $(\exists u)\,[\psi_1(u, x_1, \ldots, x_n) \vee \psi_2(u, x_1, \ldots, x_n)]$ occurs in ϕ:
Then the subformula is replaced by $(\exists u)[\psi_1(u, x_1, ..., x_n)] \vee (\exists u)[\psi_2(u, x_1, ..., x_n)]$.

(d) $(\exists u)\,[u = x_1 + c \wedge \psi(u, x_1, \ldots, x_n)]$ occurs in ϕ:
If $c \geq 0$ then the subformula is replaced by $\psi(x_1{+}c, x_1, \ldots, x_n)$. Otherwise ($c < 0$) the subformula contains the additional information $x_1 \geq -c$. Thus the subformula becomes $(x_1 \neq 0 \wedge x_1 \neq 1 \wedge \ldots \wedge x_1 \neq -c{-}1 \wedge \psi(x_1{+}c, x_1, ..., x_n))$.

(e) $(\exists u)\,[\psi(u, x_1, \ldots, x_n)]$ occurs in ϕ, but none of the above cases holds:
Without loss of generality $\psi(u, x_1, \ldots, x_n) = (f(u+c_0) = d_0) \wedge \ldots \wedge (f(u+c_i) = d_i)$ $\wedge\ (u \neq c_{i+1}) \wedge \ldots \wedge (u \neq c_j) \wedge (u \neq x_{l_{j+1}} + c_{j+1}) \wedge \ldots \wedge (u \neq x_{l_k} + c_k) \wedge \eta(x_1, \ldots, x_n)$. Let $c = 2 + k + |c_0| + \ldots + |c_k|$ and $d = |\sigma_0\sigma_1 \ldots \sigma_c|$. Then $(\exists u)\,[\psi(u, x_1, \ldots, x_n)]$ is replaced by $\psi(0, x_1, \ldots, x_n) \vee \psi(1, x_1, \ldots, x_n) \vee \ldots \vee \psi(d, x_1, \ldots, x_n)$.

If several of the above rules are applicable then the first one is executed. Thus ψ is always in disjunctive normal form, if rule (c), (d) or (e) is applied. Furthermore, ψ does not contain any $\vee$ if rule (d) or (e) is applied. It is easy to verify that the substitutions (a), (b), (c) and (d) work properly, that is, do not change the truth-value of the formula. The last operation (e) is more complex and so its correctness is now proven explicitly:

For ease of argumentation assume that $c_0 = 0, c_1 = 1, \ldots, c_i = i$. Let τ denote $d_0 d_1 \ldots d_i$. If τ contains more than two 1s then τ is a substring of $\sigma_0\sigma_1 \ldots \sigma_i$. So the formula is correct in this case. Otherwise τ is of the form 0^i or 0^*10^* and τ occurs in $\sigma_i, \sigma_{i+1}, \ldots, \sigma_c$. So there are more than k occurrences of τ in f before d and since there are only $k - i$ inequalities in ψ, the subformula $(f(u) = d_0) \wedge (f(u+1) = d_1) \wedge \ldots \wedge (f(u+i) = d_i) \wedge (u \neq c_{i+1}) \wedge \ldots \wedge (u \neq c_j) \wedge (u \neq x_{l_{j+1}} + c_{j+1}) \wedge \ldots \wedge (u \neq x_{l_k} + c_k)$ holds. In the last case τ has the form $0^*10^a10^*$ with $i > a + 2$. Either $a \notin K_c$ and then τ occurs in each of the strings $\sigma_i, \sigma_{i+1}, \ldots, \sigma_c$, so the new formula is correct by the same argument as before. Or $a \in K_c$ and then τ does not occur in f beyond d.

So the formula is iteratively transformed into an equivalent one without variables and quantifiers. This formula consists only of atoms of the form $f(c) = d$ and can be decided by knowing a sufficiently long prefix of f. ∎

Let $S_0 = \{f : (\exists e)\,[\varphi_e = f \wedge 0^e 1 \preceq f]\}$ and $S_1 = \{f : (\forall^\infty x)\,[f(x) = 0]\}$. The family $S_0 \cup S_1$ is not Ex[A] learnable for any nonomniscient A [89, Proof of Theorem 8.1]. But $S_0 \cup S_1$ is QEx[Succ] learnable [55, Theorem 6]: A learner conjectures φ_e but continuously checks for $c = 0, 1, \ldots$ whether $(\exists x > c)\,[f(x) \neq 0]$. These queries can be formulated without the symbol $<$ since c is a constant, so if $f = \sigma 0^\infty$ then the learner will discover this fact after making $c = |\sigma|$ queries. The learner makes a mind change to $\sigma 0^\infty$ and terminates. Thus one obtains the following fact:

Fact 3.4.3 $S_0 \cup S_1 \in \text{QEx[Succ]} - \text{Ex}[A]$ *for all nonhigh oracles A.*

So the families given in Theorem 3.4.1, Theorem 3.4.2 and Fact 3.4.3 show that the following corollary is true:

Corollary 3.4.4 $\text{Ex}[A] \subset \text{QEx}[\text{Succ}; A] \subset \text{QEx}[<; A] \subset \text{QEx}[+; A]$ *for all nonhigh oracles A.*

3.5 Variants of Learning in the Limit

The results of the previous sections can be transferred to other concepts of learning. This section gives a summary for three important notions and the next one will give a more detailed analysis of the situation for the case of finite learning.

The first notion to be looked at is that of team learning. A class S can be learned by a team of n machines ($S \in [1,n]\mathrm{QEx}[\star]$) iff there are n machines $M_1, \ldots, M_n$ such that for each $f \in S$ at least one machine M_m of these infers f under the criterion $\mathrm{QEx}[\star]$. Theorem 3.2.1 generalizes to the fact that $[1,n]\mathrm{QEx}[\star; A] \not\subseteq [1,n]\mathrm{QEx}[\star; B]$ whenever $[1,n]\mathrm{Ex}[A]$ is not contained in $[1,n]\mathrm{Ex}[B]$. Since only high oracles are omniscient for $[1,n]\mathrm{Ex}[A]$ [43, Theorem 6.19] the same holds for $[1,n]\mathrm{QEx}[\star]$. The transformations of $\mathrm{REC}_{0,1}$ considered in Theorems 3.3.3 and 3.3.4 are for no n and nonhigh A in $[1,n]\mathrm{Ex}[A]$, thus $\mathrm{QEx}[\star; A]$ is not trivial for some oracle A in the trivial $[1,n]\mathrm{Ex}$-degree. Also the families witnessing the noninclusions in Theorems 3.4.1 and 3.4.2 witness also the corresponding noninclusions for teams. Only in family in Fact 3.4.3 has to be replace by an other suitable family, for example, $\{f \in \mathrm{REC} : \mathit{range}(f) = K\}$. Theorem 3.5.1 summarizes the results.

Theorem 3.5.1 *The inference degrees of* $[1,n]\mathrm{QEx}[\star]$ *are a refinement of the* $[1,n]\mathrm{Ex}$*-degrees; the omniscient degree of* $[1,n]\mathrm{QEx}[\star]$ *and* $[1,n]\mathrm{Ex}$ *coincide but the trivial degree of* $[1,n]\mathrm{QEx}[\star]$ *is a proper subset of the trivial* $[1,n]\mathrm{Ex}$*-degree. Furthermore,* $\mathrm{Ex}[\star] \subset [1,n]\mathrm{QEx}[\mathrm{Succ}] \subset [1,n]\mathrm{QEx}[<] \subset [1,n]\mathrm{QEx}[+]$ *and these witnesses for the properness of the inclusions are also not learnable by the weaker criterion if this receives in addition a nonhigh oracle.*

The second notion to be looked at is that of learning with only finitely many queries to the oracle, that is, a family S is in $\mathrm{QEx}[\star; A*]$ iff there is a machine M which $\mathrm{QEx}[\star; A]$ learns S and in addition while inferring any $f \in S$ makes only finitely many queries to f. Surprisingly the structure of these degrees is the same as that of Ex-degrees with finitely many queries.

Theorem 3.5.2 $\mathrm{QEx}[\star; A*] \subseteq \mathrm{QEx}[\star; B*] \Leftrightarrow \mathrm{Ex}[A*] \subseteq \mathrm{Ex}[B*]$.

Proof This theorem is a combination of three facts, whose proofs transfer directly.

First, $\mathrm{QEx}[\star; (A \oplus K)*] = \mathrm{QEx}[\star; A*]$ in the same way as $\mathrm{Ex}[(A \oplus K)*] = \mathrm{Ex}[A*]$ [43, Lemma 4.9]: the finitely many queries to K are computed in the limit and for each value $M(\sigma)$ the approximation $K_{|\sigma|}$ is used instead of K.

Second, if $A \oplus K \geq_T K'$ then $\mathrm{REC} \in \mathrm{Ex}[A*]$ [43, Theorem 5.8]. Also the more general notion $\mathrm{QEx}[\star; A*]$ is omniscient for these A.

Third, if $A \oplus K \not\leq_T B \oplus K$ and $B \oplus K \not\geq_T K'$ then there is a family $S_A \in \mathrm{Ex}[A*] - \mathrm{Ex}[B*]$ [89, Theorem 5.5, $(2 \Rightarrow 1)$]. Since this set S_A consists only of $\{0,1\}$-valued functions, its transformed set $\tilde{S}_A$ using the transformation Γ from Theorem 3.2.1 witnesses the noninclusion: $\tilde{S}_A \in \mathrm{QEx}[\star; A*] - \mathrm{QEx}[\star; B*]$. ∎

Note that one result does not hold for $\mathrm{QEx}[\star; A]$ learning: queries to a 1-generic oracle cannot be replaced by finitely many queries to the same 1-generic oracle as in the case of $\mathrm{Ex}[A*]$ [43, Lemma 4.19]: By Theorems 3.3.3 and 3.3.4 there is a 1-generic $A \leq_T K$ such that $\mathrm{QEx}[\star; A] \neq \mathrm{QEx}[\star]$. Since this $A \leq_T K$, Theorem 3.5.2 gives that $\mathrm{QEx}[\star; A*] = \mathrm{QEx}[\star]$, in particular that $\mathrm{QEx}[\star; A] \neq \mathrm{QEx}[\star; A*]$ for this 1-generic oracle A.

The third notion considered in this section is the well-studied concept of behaviourally correct inference. A machine M learns a function f under the criterion $\mathrm{QBC}[\star; A]$ if M computes using queries to the oracle A and the teacher in the language $L[\star]$ and infinite sequence of indices such that almost all of them compute f. Furthermore, $S \in \mathrm{QBC}[\star; A]$ iff there is one machine M which learns every $f \in S$ under the criterion $\mathrm{QBC}[\star; A]$. Theorem 3.5.3 gives a summary of the main results.

Theorem 3.5.3 *The inference degrees of* $\mathrm{QBC}[\star]$ *are a refinement of the* BC*-degrees; the omniscient degree of* $\mathrm{QBC}[\star]$ *and* BC *coincide but the trivial degree of* $\mathrm{QBC}[\star]$ *is a proper subset of the trivial* BC*-degree.*

The results for QBC[$\star$] correspond very much to those for QEx[$\star$] in the previous sections. The next items go into some details of this correspondence.

- If $\mathrm{BC}[A] \not\subseteq \mathrm{BC}[B]$ then $\mathrm{QBC}[\star; A] \not\subseteq \mathrm{QBC}[\star; B]$. In particular, if S witnesses the noninclusion for BC then the transformed set $\tilde{S}$ of Theorem 3.2.1 witnesses the noninclusion for QBC.

- $\mathrm{REC} \in \mathrm{QBC}[\star; A]$ iff $\mathrm{REC} \in \mathrm{BC}[A]$, that is, the omniscient degree for learning with or without queries to a teacher coincide. This result is a direct application of the previous one combined with the fact that $\mathrm{REC} \in \mathrm{BC}[A] \Rightarrow \mathrm{REC} \in \mathrm{QBC}[\star; A]$, that is, that every BC-omniscient degree is also QBC[$\star$]-omniscient. Note that the omniscient BC-degree does not coincide with the omniscient Ex-degree [43, Theorem 5.18].

- There is an oracle A which is trivial for BC but not for QBC, that is, $\mathrm{QBC}[\star; A] \neq \mathrm{QBC}[\star]$ while $\mathrm{BC}[A] = \mathrm{BC}$. The construction is the same as in Theorems 3.3.3 and 3.3.4.

- Theorems 3.4.1 and 3.4.2 state that there are families witnessing the noninclusions $\mathrm{QEx}[<] \not\subseteq \mathrm{QEx}[\mathrm{Succ}; A]$ and $\mathrm{QEx}[+] \not\subseteq \mathrm{QEx}[<; A]$ for all oracles A which are not omniscient for Ex. The same languages witness even the noninclusions $\mathrm{QBC}[<] \not\subseteq \mathrm{QBC}[\mathrm{Succ}; A]$ and $\mathrm{QBC}[+] \not\subseteq \mathrm{QBC}[<; A]$, respectively, where A is any oracle which is not omniscient for BC, so Theorems 3.4.1 and 3.4.2 transfer from QEx to QBC if the condition "not high" is replaced by "not omniscient for BC".

- It is unknown whether $S_0 \cup S_1 \in \mathrm{BC}[A]$ holds for some non-BC-omniscient oracle A. But Fact 3.4.3 can be transferred using the family $\{f \in \mathrm{REC} : range(f) = K\}$.

Furthermore, the degree structure of $\mathrm{QBC}[\star; A*]$ is the same as that of $\mathrm{QEx}[\star; A*]$, $\mathrm{BC}[A*]$ and $\mathrm{Ex}[A*]$.

3.6 The Finite Case

Recall that finite learning (Fin) means learning without mind changes, that is, the learner first outputs "?" until at some time the only guess e is output and never withdrawn. Theorem 2.3.2 by Fortnow et al. [43] states that the inference degrees of finite inference coincide with the Turing degrees. This result can be extended to finite inference with queries:

Theorem 3.6.1 *The following is equivalent for any oracles A and B:*
(a) $A \leq_T B$;
(b) $\mathrm{Fin}[A] \subseteq \mathrm{Fin}[B]$;
(c) $\mathrm{QFin}[\mathrm{Succ}; A] \subseteq \mathrm{QFin}[\mathrm{Succ}; B]$;
(d) $\mathrm{QFin}[<; A] \subseteq \mathrm{QFin}[<; B]$;
(e) $\mathrm{QFin}[+; A] \subseteq \mathrm{QFin}[+; B]$.

Proof The whole theorem can be reduced to the fact

$$\mathrm{Fin}[A] \subseteq \mathrm{QFin}[+; B] \Rightarrow A \leq_T B$$

whose contrapositive is now shown: Let $A \not\leq_T B$. The transformation Γ from Theorem 3.2.1 satisfies also the following properties for every set $S \subseteq \mathrm{REC}_{0,1}$:

$$\Gamma(S) = \{\Gamma(g) : g \in S\} \subseteq \mathrm{REC}_{0,1}. \tag{16}$$
$$S \in \mathrm{Fin}[A] \Rightarrow \Gamma(S) \in \mathrm{Fin}[A]. \tag{17}$$
$$S \notin \mathrm{Fin}[B] \Rightarrow \Gamma(S) \notin \mathrm{Fin}[B]. \tag{18}$$
$$\Gamma(S) \notin \mathrm{Fin}[B] \Rightarrow \Gamma(S) \notin \mathrm{QFin}[+;B]. \tag{19}$$

By [43, Theorem 6.36] there is a set $S \in \mathrm{Fin}[A] - \mathrm{Fin}[B]$, for example, the set

$$S = \{0^n1^\infty : n \in A \oplus \overline{A}\} \cup \{0^n1^m0^\infty : n \in \overline{A} \oplus A \wedge m > 0\}$$

is a witness for this noninclusion $\mathrm{Fin}[A] \not\subseteq \mathrm{Fin}[B]$. So $\Gamma(S) \in \mathrm{Fin}[A] - \mathrm{QFin}[+;B]$ and $\mathrm{Fin}[A] \not\subseteq \mathrm{QFin}[+;B]$.

Also the separation of the query-hierarchy easily transfers from the Ex-case to the case of finite inference. Here the sets are easier to define:

Theorem 3.6.2 $S = \{0^\infty\} \cup \{0^n1^\infty : n \geq 0\} \in \mathrm{QFin}[\mathrm{Succ}] - \mathrm{Fin}[A]$ *for all oracles* A.

Proof $S \in \mathrm{QFin}[\mathrm{Succ}]$ since the query $(\exists x)\,[f(x) = 1]$ tells the learner whether either $f = 0^\infty$ or $f = 0^n1^\infty$ for some n. In the first case the learner outputs an index for 0^∞, in the second case the learner reads the input so long until the first 1 appears and therefore n is known. Then the learner outputs an index of 0^n1^∞.

If a learner with A-oracle but without teacher infers 0^∞ then the learner must output an index for 0^∞ on some input 0^n. Thus the learner fails to identify 0^n1^∞ without making a mind change. ∎

Theorem 3.6.3 *There is a family* $S \in \mathrm{QFin}[<] - \mathrm{QFin}[\mathrm{Succ};A]$ *for all oracles* A.

Proof Let S consist of the function $g_0 = 10^010^110^210^310^4\ldots$ and of all functions $g_n = 10^010^1\ 10^210^310^4\ldots10^n10^\infty$ for all $n > 0$. In the same way as in Theorem 5.2 it can be shown that every $L[\mathrm{Succ}]$ query to any $f \in S$ can be decided after reading a sufficiently long prefix of f; thus $S \in \mathrm{QFin}[\mathrm{Succ};A] \Rightarrow S \in \mathrm{Fin}[A]$. The function g_0 is a limit-point of the sequence $g_1, g_2, g_3, \ldots$ and therefore $S \notin \mathrm{Fin}[A]$ by an argument similar to that in Theorem 3.6.2. It follows that $S \notin \mathrm{QFin}[\mathrm{Succ};A]$.

On the other hand the query $(\forall x)\,(\exists y > x)\,[f(y) = 1]$ separates g_0 from the other functions: if the answer is yes the function inferred is g_0, otherwise there is an n such that the inferred function has the prefix $10^010^110^210^310^4\ldots10^n10^{n+2}$: g_n is the only function in S with this prefix and so g_n is inferred after seeing a sufficiently long prefix of f. ∎

Further $S_0 \cup S_1 \in \mathrm{QFin}[<] - \mathrm{QFin}[\mathrm{Succ};A]$ witnesses the same noninclusion for all oracles A where $S_0 = \{f : (\exists e)\,[\varphi_e = f \wedge 0^e1 \preceq f]\}$ and $S_1 = \{f : (\forall^\infty x)\,[f(x) = 0]\}$. This set $S_0 \cup S_1$ is a very popular example in many fields of inductive inference, but the proof for this fact is omitted since Theorem 3.6.3 already shows the noninclusion.

Theorem 3.6.4 *Let* $S = \{0^\infty\} \cup \{0^n1^\infty : n \in \mathbb{N}\}$. $\Gamma(S) \in \mathrm{QFin}[+] - \mathrm{QFin}[<;A]$ *for some suitable transformation* Γ *and all oracles* A.

Proof The sequence $a_0, a_1, \ldots$ and the transformation Γ are defined as

$$\begin{aligned} a_0 &= 0; \\ a_{2n+1} &= 2 \cdot (a_{2n} + k_<(a_{2n})) + 1; \end{aligned}$$

$$
\begin{aligned}
a_{2n+2} &= 2 \cdot (a_{2n+1} + k_<(a_{2n+1})); \\
\Gamma(g)(x) &= \begin{cases} g(m) & \text{if } x = a_{2m}; \\ 1 & \text{if } x \in \{a_1, a_3, a_5, \ldots\}; \\ 0 & \text{otherwise, that is, } x \notin \{a_0, a_1, a_2, \ldots\}. \end{cases}
\end{aligned}
$$

By Fact 3.1.4 there is a procedure that answers every $L[<]$-query to any $f \in \Gamma(S)$ after reading a sufficient long segment of the input, thus $\Gamma(S) \in \mathrm{QFin}[<;A]$ iff $\Gamma(S) \in \mathrm{Fin}[A]$ iff $S \in \mathrm{Fin}[A]$. So $\Gamma(S) \notin \mathrm{QFin}[<;A]$ since $S \notin \mathrm{Fin}[A]$ by Theorem 3.6.2.

For the other direction note that a_m is even iff m is even. So only the values of the even arguments are interesting and the query

$$(\forall x)\,[f(x+x) = 0]$$

checks whether f takes only the value 0 on the even arguments. If so, $f = \Gamma(0^\infty)$ and the learner outputs an index for this function $\Gamma(0^\infty)$. Otherwise there is a minimal n such that $f(a_{2n}) = 1$. This n can be found by looking at $f(a_0), f(a_2), f(a_4), \ldots$ and then the learner outputs an index for $\Gamma(0^n 1^\infty)$. Such an index for $\Gamma(0^n 1^\infty)$ can be computed effectively from n. ∎

3.7 Successor and Other Functions

It was pointed out to the author that some results on the classes QEx[Succ] generalize when the Successor is replaced by a suitable function F. In particular Theorems 3.4.2 and 3.6.2 hold also for functions like $F(x) = x^2$ or $F(x) = 2x + 5$. Within the query-language, "F" represents the fixed built in function while "f" represents the function from S to be learned.

Theorem 3.7.1 *Let F be computable and strictly increasing: $(\forall x)\,[F(x) < F(x+1)]$. Then $\mathrm{QFin}[<] \not\subseteq \mathrm{QFin}[F;A]$ and $\mathrm{QEx}[<] \not\subseteq \mathrm{QEx}[F;B]$ for all oracles A and all nonhigh oracles B.*

The proof of this theorem can be directly constructed from the proofs of the Theorems 3.4.2 and 3.6.2. Somehow there are limitations so that F cannot be taken of a more general form. If queries to F, which do not contain any reference to the function f to be learned, allow to decide a high set, say the halting problem K, then it is possible to QEx[F]-learn REC. So the following conditions are not sufficient for Theorem 3.7.1.

- F strictly increasing: Take $F(x) = 2x + K(x)$. Then $K(x)$ can be computed by simple asking whether the value of $F(x)$ is $2x$ or $2x+1$ — where the values x, $2x$, $2x+1$ in this query are constants. So the query language gives access to the halting-problem K.

- F recursive: There is a recursive enumeration $a_0, a_1, \ldots$ of K and by letting $F(x) = a_x$ every query to K can be translated into the query whether $(\exists y)\,[F(y) = x]$ which then gives again the K-oracle.

- F increasing and recursive: Let $a_0, a_1, \ldots$ be again a recursive enumeration of K and take $F = 0^{a_0} 1^{a_1} 2^{a_2} \ldots$; that means the first a_0 numbers are mapped to 0, the next a_1 to 1, the next a_2 to 2 and so on. Then for each x the query whether there are exactly x numbers $y_1, y_2, \ldots, y_x$ which take the same value c_x is equivalent to the query whether $x \in K$:

$$x \in K \Leftrightarrow (\exists c)\,(\exists^x y)\,[F(y) = c].$$

Speaking more verbosely, for any fixed x, say for $x = 3$, the $\exists^x$-quantifier can be restated: $(\exists^x y)\,[F(y) = x]$ is equivalent to $(\exists y_1, y_2, y_3)\,(\forall z)\,[y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \wedge (f(z) = c \Leftrightarrow z = y_1 \vee z = y_2 \vee z = y_3)]$.

While Theorem 3.7.1 still holds for $F(x) = x^2$ or $F(x) = 2x+5$, other results depend much more on the special form of Succ than this one.

Theorem 3.7.2 *The following is equivalent for every function F:*
(a) $\mathrm{QFin}[F,<] = \mathrm{QFin}[<]$.
(b) *F is definable in $L[<]$.*
(c) *F is definable in $L[\mathrm{Succ}]$.*
(d) *Either F takes almost everywhere the same value or $F(x) = x + c$ for some integer c and almost all x.*

Proof The proof is (d $\Rightarrow$ c $\Rightarrow$ b $\Rightarrow$ a $\Rightarrow$ d) where the last implication is the most difficult one.

(d $\Rightarrow$ c): If (d) holds then either f is almost constant or $F(x)$ is almost everywhere equal to $x + d - c$. In these cases, F is definable in $L[\mathrm{Succ}]$. For example, let $F(x) = 4$ for $x = 0, 1, 2, 8$ and $F(x) = x - 1$ otherwise. Now one can construct a formula for F which splits of the finitely many cases via a table and uses the successor to define the rest:

$$F(x) = y \Leftrightarrow (x \in \{0,1,2,8\} \wedge y = 4) \vee (x \notin \{0,1,2,8\} \wedge \mathrm{Succ}(y) = x)$$

where the membership of the finite set $\{0, 1, 2, 8\}$ is equivalent to a disjunction on finitely many equalities: $x = 0 \vee x = 1 \vee x = 2 \vee x = 8$. It is easy to see how to apply this system for the other functions F of this type. So F is definable in $L[\mathrm{Succ}]$.

(c $\Rightarrow$ b $\Rightarrow$ a): The implication (c $\Rightarrow$ b) is due to the fact that the function Succ is definable in $L[<]$. The implication (b $\Rightarrow$ a) to due to the following observation: if F is definable in $L[<]$ then the languages $L[F,<]$ and $L[<]$ can express the same things and so the additional feature to use F in the query language does not give more learning power.

(a $\Rightarrow$ d): Claim (a) states that $\mathrm{QFin}[F,<] = \mathrm{QFin}[<]$. Let $x \sim y$ mean that x and y are neighbours with respect to F, that is, that either $x = F(y)$ or $y = F(x)$. For the following distinction of four cases, one chooses the first one to hold, for example, if F is not computable and each x is mapped either to $x + 1$ or $x + 2$ then case (I) is taken and not case (IV).

(I) Some nonrecursive set A is definable in $L[F,<]$, in particular F is not recursive itself: Then there is a class $S \in \mathrm{Fin}[A] - \mathrm{Fin}$. Using the transformation Γ from Theorem 3.6.1 it follows that $\Gamma(S) \in \mathrm{Fin}[A] - \mathrm{QFin}[<]$ and thus $\mathrm{QFin}[F] \not\subseteq \mathrm{QFin}[<]$. In particular $\mathrm{QFin}[F,<] \neq \mathrm{QFin}[<]$ in contrary to the choice of F. So this case does not occur and F is recursive in the cases (II), (III) and (IV) below.

(II) $(\exists x)\,(\exists^\infty y)\,[x \sim y]$: Fix this x with infinitely many neighbours. The set $B = \{y : x \sim y\} = \{F(x)\} \cup \{y : F(y) = x\}$ is definable in $L[F]$. If B is cofinite then $F(y) = x$ for almost all y and (d) holds. So it remains to look at the subcase where B is coinfinite. By choice B is also infinite. Now B can take the role of the set of even numbers in Theorem 3.6.4 and the sequence $a_0, a_1, \ldots$ is defined as follows:

$$\begin{aligned} a_0 &= \text{least } x \in B; \\ a_{2n+1} &= \text{least } x \notin B \text{ with } x > 2 \cdot (a_{2n} + k_<(a_{2n})); \\ a_{2n+2} &= \text{least } x \in B \text{ with } x > 2 \cdot (a_{2n+1} + k_<(a_{2n+1})). \end{aligned}$$

Γ and S are defined analogously to those in the proof of Theorem 3.6.4. Note that Γ is computable and translates computable functions into computable ones since B is computable. Now on one hand $\Gamma(S) \notin \mathrm{QFin}[<]$ and on the other hand $\Gamma(S) \in \mathrm{QFin}[F]$ where the only query is $(\forall y \in B)\,[f(y) = 0]$ or more precisely $(\forall y)\,[F(y) = x \vee F(x) = y \Rightarrow f(y) = 0]$. By (I), such a class $\Gamma(S)$ does not exist and in the case (II) only the subcase occurs where F is takes almost everywhere the same constant value.

(III) $(\forall x, c)\,(\exists y, d)\,[x < y \wedge c < d \wedge y \sim y + d]$: Since case (II) does not hold, for each x there are only finitely many y with $x \sim y$. Starting with $a_0 = 0$ it is possible to construct the sequence $a_0, a_1, \ldots$ as follows:

> a_{2n+1} and a_{2n+2} are the least numbers y and $y + d$ such that $y, d > 2 \cdot (a_{2n} + k_<(a_{2n}))$, $y \sim y + d$ and $x \not\sim y \wedge x \not\sim y + d$ for all $x \leq a_{2n}$.

This sequence defines again an operator Γ such that $\Gamma(S) \notin \mathrm{QFin}[<]$ where the proof to show $\Gamma(S) \notin \mathrm{Fin} \Rightarrow \Gamma(S) \notin \mathrm{QFin}[<]$ has to uses the fact that the functions in $\Gamma(S)$ are $k_<(a_n)$-good for even n. So on one hand the classes $S = \{0^\infty\} \cup \{0^n 1^\infty : n \in \mathbb{N}\}$ and $\Gamma(S)$ are not finitely learnable and thus $\Gamma(S) \notin \mathrm{QFin}[<]$. On the other hand $S \in \mathrm{QFin}[F]$ since the query whether no two neighbouring numbers are mapped to 1, that is, whether

$$(\forall x, y)\,[F(x) = y \Rightarrow f(x) = 0 \vee f(y) = 0],$$

is answered with yes if $f = \Gamma(0^\infty)$ and with no if $f = \Gamma(0^n 1^\infty)$. In the first case f is already found and in the second case the parameter n for f is found by inspecting $\Gamma(f)$ at the places $a_1, a_3, a_5, \ldots$ until $\Gamma(f)(a_m) = 1$ for some odd m. It follows that $\Gamma(S) \in \mathrm{QFin}[F] - \mathrm{QFin}[<]$ in contradiction to (a) and so case (III) does not occur.

(IV) Otherwise, that is, $(\exists d)\,(\forall x)\,[x - d \leq F(x) \leq x + d]$: That this formula indeed holds is due to the fact that (II) is considered if F takes some value infinitely often and that (III) is considered if the tradeoff between x and $F(x)$ is not bounded. So fix now this d. For $c = -d, 1 - d, \ldots, d - 1, d$, each set $B_c = \{x : F(x) = x + c\}$ is definable using Succ and F, for example $B_{-1} = \{x : \mathrm{Succ}(F(x)) = x\}$. Since the successor is definable in $L[<]$, it follows that each set B_c is definable in $L[F, <]$. Arguing as in case (II), all these sets B_c are either finite or cofinite. Since they cover $\mathbb{N}$, exactly one of them is cofinite and so $F(x) = x + c$ for almost all x and the index c of this cofinite set B_c.

So cases (I) and (III) do not occur, case (II) states that F takes almost everywhere the same number and case (IV) states that F equals $x + d$ for some integer d on all but finitely many places. Thus (a $\Rightarrow$ d) holds and the Theorem follows. ∎

In cases (I), (II) and (III) it is even shown that $\mathrm{QFin}[F]$ without the additonal symbol is not covered by $\mathrm{QFin}[<]$.

Corollary 3.7.3 *If* $\mathrm{QFin}[F] \subseteq \mathrm{QFin}[<]$ *then there is a constant c such that either $F(x) = c$ for almost all x or $y - c \leq F(y) \leq y + c$ for all y.*

So functions like $F(x) = x^2$ or $F(x) = 2x + 5$ define classes $\mathrm{QFin}[F]$ which are not contained in $\mathrm{QFin}[<]$. But the function F given by $F(0) = 1$, $F(2x + 2) = 2x$ and $F(2x + 1) = 2x + 3$ is somehow in between: on one hand $\mathrm{QFin}[F] \subseteq \mathrm{QFin}[<]$ and on the other hand $\mathrm{QFin}[F, <] \not\subseteq \mathrm{QFin}[<]$.

So one sees that the successor (or equivalently predecessor) is somehow the most natural example of a function F with $\mathrm{QFin}[F] \subseteq \mathrm{QFin}[<]$. Furthermore, the successor defines up to equivalence the most powerful language of the type $L[F]$ among all F with $L[F, <] = L[<]$.

4 Noisy Data

Many scientific problems are only solved numerically and the correctness of the solution depends on the computing power and equipment available. This situation forces scientists to base their theories on data, which might be inaccurate. Modelling the development of science has therefore to take into account inaccuracies in experimental data.

The present model of inaccuracies guarantees that these inaccuracies are detected at some future point, that is, succeeding generations of researchers, who have better simulation techniques and computing equipment, can each remove some of the worst inaccuracies in their data. Therefore they can replace the old theories by new improved hypotheses.

This — of course a bit simplified — view on scientific progress motivates the following abstract model of learning from noisy data: The learner wants to design a program for a given function via outputting better and better hypotheses based on more and more data. The learner's input describes the function via giving for each x an infinite sequence of pairs (x, y) representing the more and more precise simulations which intend to compute $f(x)$ from x. The x-th sequence converges in the sense that almost all pairs (x, y) are identical with $(x, f(x))$. For the ease of notation, the different sequences are merged into one for all x which contains each pair $(x, f(x))$ infinitely often and for each x only finitely many additional pairs of the form (x, y). The learner's output is a sequence of hypotheses which converge either syntactically to a program for f or behaviourally correct to f. This notion of noise is also transferred to language learning from informant and text in a suitable way.

There are various approaches to inference from faulty data [10, 50, 61, 113]; Jain [61] distinguishes three basic types of models for learning in the limit from faulty data: (a) the learner receives some spurious data together with all correct data on the concept to be learned, (b) all data presented is correct but some data about the concept is never presented and (c) the combination of both kinds of inaccuracies.

Many models of learning from faulty data have the disadvantage, that it is impossible to define the object to be learned only from the input to the learner. If for example in case (a) information $(0, 0)$ and $(0, 1)$ are both supplied to the learner, then it is impossible to know which one is correct, that is, whether $f(0) = 0$ or $f(0) = 1$. The same holds if according to (b) no statement of the form $(0, y)$ is made at all. The learner therefore has to overcome this gap by a priori knowledge about f, for example, that always $f(0) = f(1)$ and $f(1)$ is specified uniquely on the information supplied to the learner.

The present model solves this problem by presenting the correct information infinitely often while the incorrect one occurs only finitely often, that is, $f(x) = y$ iff (x, y) occurs infinitely often on the learner's input-tape. During the inference process, the learner still has the problem not to know whether the current input is correct, but in the limit it turns out which data is correct and which is incorrect; thus the learner needs less a priori knowledge for learning in the limit.

So the noisy inference considered here can be put into Jain's first category (a), that is, learning with additional spurious information. The noisy texts in this context are a combination of the intrusion texts as defined by Osherson, Stob and Weinstein [113, Exercise 5.4.1E] which may contain finite additional false data-elements and the fat texts [113, Section 5.5.4] in which each data-element appears infinitely often. Learning from texts and learning from fat texts are equivalent models [113, Proposition 5.5.4A]. But the intrusion texts are more restrictive than noisy texts as defined below since the class $\{\{0\}, \{1\}\}$ can be learned from noisy text but not from intrusion texts.

One main topic of this chapter is to look at relationships of noisy learning in the limit on one side and noise-free finite learning on the other side. It turns out that removing noise allow to bring down learning in the limit to finite learning, provided that some "additional computing power" is added in form of the oracle K.

- Learning Functions or Languages from Noisy Informant
  The power of this learning criterion turned out to be equal to that of finite learning (from informant) with K-oracle.
- Learning Languages from Noisy Text
  This notion turned out to be incomparable to finite language-learning from text with any oracle. So the result above does not transfer.
- Learning Uniformly Recursive Families of Languages from Noisy Text
  In this context, learning from noisy text is more powerful than finite learning with K-oracle from noise-free text. Here the inclusion holds in one direction and is proper, therefore the situation is somewhere between those of the previously mentioned scenarios.

Furthermore relativizations and variants of these results are studied. The last section deals with weaker forms of identification such as behavioural correct and partial identification [19, 113]. In the setting of partial identification, a single machine can learn all languages from text [113]. This result is transferred to the setting of noisy text, which may contain only finitely many incorrect data-elements. But the result cannot be transferred to very noisy texts T whose only requirement is that a data-element w occurs infinitely often in T iff it belongs to the language L to be learned.

4.1 Language Learning

Languages are enumerable sets. Since there characteristic functions is in general not computable, one normally describes them by programs which generate all their elements. So it is quite natural, to learn them from a data presentation related to this description: a text is just a list of all elements of a language in an arbitrary order. A learner then learns a language L iff it succeeds with every legal description, in this case the learner has to find a program generating the language from every text of the language. In addition to this new form of input, one also transferred the traditional input $f(0)f(1)\ldots$ for functions f to languages, which is there called an informant.

Besides these two basic notions, this section also considers noisy versions of these notions. Texts are already more difficult to deal with than informants since they provide only information on the elements in the language, but no direct information on those not in the language. Noisy texts which contain in addition s to the elements also some noise are of course even more difficult to deal with. All notions of noisy input have here in common, that they determine in the limit which elements belong to L and which not. The definitions of informant and noisy informant are stated in the versions for functions since the characteristic function is — formally — also nothing else than a (nonrecursive) $\{0,1\}$-valued function.

Definition 4.1.1 The input T is always an infinite sequence of data-elements describing the object to be learned. T can be specified in six different ways. In the definitions L stands for a language and f stands either for a recursive function or the characteristic function of a language L.

- An *informant* for f is an infinite sequence of pairs such that the pair $(x, f(x))$ occurs for every x. In this case of learning from noise-free informant it is convenient to present the data in the default ordering and to give the values $f(0)$, $f(1)$, ... instead of the pairs $(0, f(0))$, $(1, f(1))$, ...; in particular a string $\alpha = a_0 a_1 \ldots a_n$ represents the first $n+1$ values $f(0)$, $f(1)$, ..., $f(n)$.

- A *noisy informant* for f is an infinite sequence such that every pair $(x, f(x))$ occurs infinitely often in this sequence while for each x only finitely often some data-element (x, y) with $y \neq f(x)$ occurs.

- A *very noisy informant* for f is an infinite sequence such that every pair $(x, f(x))$ occurs infinitely often in this sequence while for each x and y with $y \neq f(x)$ the pair (x, y) occurs only finitely often. But in contrast to the noisy informant, it is legal that for fixed x the total amount of all occurrences of pairs (x, y) with $y \neq f(x)$ is infinite.

- A *text* for L is an infinite sequence of numbers such that every $x \in L$ occurs in the text and no $x \notin L$. In texts may also occur the symbol # which stands for void information — this symbol is necessary since otherwise the empty set would not have a text.

- A *noisy text* for L is an infinite sequence of numbers such that every $x \in L$ occurs infinitely often in the text and the total amount of occurrences of $x \notin L$ is finite.

- A *very noisy text* for L is an infinite sequence of numbers such that every $x \in L$ occurs infinitely often in the text and every $x \notin L$ occurs only finitely often.

Informant and noisy informant for sets are defined by using the characteristic function of L as the function f in the first two definitions. In the third definition, any very noisy informant T for L can be turned into a noisy one by removing all data-elements (x, y) with $y > 1$ from T. Thus the first three definitions apply for functions and the first two together with the last three apply for sets.

All these forms of input can be combined with traditional learning criteria for functions and one obtains the corresponding variants for languages learning from informant or text as well as for learning functions or languages from the corresponding noisy form of input. The definition shows how this is done for learning languages in the limit from text or informant.

Definition 4.1.2 A learner M learns L in the limit from some given sequence T iff $M(\sigma) = e$ and e generates L for almost all $\sigma \preceq T$. If this sequence T is just $L(0)L(1)\ldots$ then one just says that M *learns* L *in the limit from informant.* If T is a text and if M succeeds on every text T for L, then one says that M *learns* L *in the limit from text.* So the notion of Ex-learning functions has two generalizations for Ex-learning sets:

$S \in \mathrm{Ex}$	iff	some recursive machine M learns every $f \in S$ from the informant for f in the limit.
$S \in \mathrm{ExInf}$	iff	some recursive machine M learns every $L \in S$ from the informant for L in the limit.
$S \in \mathrm{ExTxt}$	iff	some recursive machine M learns every $L \in S$ from every text for L in the limit.

Similarly one can define the corresponding versions for FinInf, FinTxt, BCInf and BCTxt of finite and behaviourally correct language learning. Besides them, the following three learning criteria are considered.

- Noisy Inference (Noisy). $S \in$ Noisy via M means that S is a class of recursive functions such that M from any noisy informant of some function $f \in S$ computes a finite sequence of guesses such that the last guess is a program for f. NoisyInf and NoisyTxt are the corresponding notions for learning languages from noisy informant or text in the limit. Related criteria as learning from very noisy informant or text are defined analogously but do not have own symbols.

- Dual Strong Monotonic ($\mathrm{SMon}^{\mathrm{d}}$) learning is considered here only with respect to learning languages from text. M infers L under the criterion $\mathrm{SMon}^{\mathrm{d}}\mathrm{Txt}$ from a text T for L iff M guesses on T a finite sequence $e_0, e_1, \ldots, e_n$ of grammars such that $W_{e_0} \supseteq W_{e_1} \supseteq \ldots \supseteq W_{e_n} = L$ [75, 94].

- Partial Identification. A machine identifies an object partially iff it while reading a description outputs an infinite sequence of guesses such that exactly one index e appears infinitely often in the output and this e is an index for the object.

Finite and dual strong monotonic learning are not combined with noisy data.

Language learning from (nonnoisy) texts and informants has been studied from the beginning. Gold [56] showed already, that not all classes of languages are learnable from text. In particular the class consisting of $\mathbb{N}$ and all finite sets is not learnable from text, even not under the criterion BC and with access to an arbitrary powerful oracle A. The structure of the inference degrees induced by the criterion TxtEx does not have any greatest or maximal element [64, 113]. An oracle A is trivial iff it is Turing reducible to some 1-generic oracle B below K which can be obtained by generalizing the techniques from the case of function learning [43, 132]. The degrees induced by finite learning from text or from informant coincide with the Turing degrees. Sharma [129] observed that finite learning from informant can be turned into Ex-learning from text so that one has the hierarchy FinTxt $\subset$ FinInf $\subset$ ExTxt $\subset$ ExInf where the inclusions are proper.

Angluin [3] started to study the learnability of those classes $S = \{L_0, L_1, \ldots\}$ which have a uniformly recursive decision procedure $i, x \to L_i(x)$. She gave a characterization in terms of tell-tale sets: There is an ExTxt-learner for S finding for every L_i and every text T of L_i an index j with $L_i = L_j$ iff there is a uniformly enumerable array of finite sets $W_{f(i)}$ such that, for every i and j, the relation $W_{f(i)} \subseteq L_j \subseteq L_i$ holds only if $L_j = L_i$. This concept was studied extensively, in particular after Jantke [68] introduced a variety of learning concepts where the learner has to satisfy certain monotonicity constraints as for example the concept of strong monotonic learning where the learner M can only increase its hypothesis but not decrease it: $L_{M(\sigma)} \subseteq L_{M(\tau)}$ for all σ, τ. Section 4.6 is dedicated to the study of learning uniformly recursive families from noisy data.

4.2 Inference From Informant

The main result of this section is, that finite learning from informant with K-oracle equals learning from noisy informant. This relation motivates the study of connections between noisy inference and finite inference with oracles.

Theorem 4.2.1 NoisyInf = FinInf$[K]$.

Proof The basic idea of the proof is for the first direction is that using the oracle K it is possible to find some kind of "locking sequence" for the NoisyInf-learner which then is output by the FinInf$[K]$-learner. The idea for the reverse direction is, that — even from noisy informant — it is possible to find in the limit the input first α which the FinInf$[K]$-learner needs to make a guess and second the index e which the learner outputs on input α.

In this proof α, β are strings of numbers ranging over prefixes of noise-free informants while σ, τ, η are strings of pairs ranging over prefixes of noisy texts. Let $\beta_0, \beta_1, \ldots$ be an enumeration of all binary strings of numbers and $\eta_0, \eta_1, \ldots$ an enumeration of all strings of pairs (x, y) with The main idea of the proof is that the FinInf$[K]$-learner emulates the NoisyInf-learner and vice versa. It will turn out that the each learner converges to the same output as the other one.

NoisyInf $\subseteq$ FinInf$[K]$: Assume that M is a recursive machine which learns a class S from noisy informant. A string σ is called $L(0)L(1)\ldots L(k)$-consistent iff

$$(\forall x \leq k)\,[\text{ if } (x, y) \text{ occurs in } \sigma \text{ then } y = L(x)].$$

Now the FinInf$[K]$-learner N searches — using K-oracle — some kind of "locking sequence" η_m and then outputs $e = M(\eta_m)$. Formally N is defined as follows:

> N outputs $M(\eta_m)$ for first pair (m, k) which satisfies the equality $M(\eta_m\tau) = M(\eta_m)$ for every $L(0)L(1)\ldots L(k)$-consistent string τ.

Note that the query "$(\forall\, L(0)L(1)\ldots L(k)\text{-consistent } \tau)\,[M(\eta_m\tau) = M(\eta_m)]$?" is indeed recursive in K.

For the verification assume now that on the inference of $L \in S$, N outputs $e = M(\eta_m)$ for some pair (m, k). Let $w_{\langle x,y\rangle} = (x, L(x))$ for all x, y where $\langle x, y\rangle = \frac{1}{2}\cdot(x+y)\cdot(x+y+1)+x$. Now $\eta_m w_0 w_1 w_2 \ldots$ is a noisy informant for L and thus M has to converge on this informant to a correct index. Since all strings $\tau_n = w_0 w_1 \ldots w_n$ are $L(0)L(1)\ldots L(k)$-consistent, $e = M(\eta_m\tau_n)$ and e is an index for L. Thus if N outputs a guess then this guess is correct.

So it remains to verify that N always finds a pair (m, k). Assume that N does not converge, that is, for every $\sigma = \eta_m$ and every k there is a $L(0)L(1)\ldots L(k)$-consistent string τ with $M(\sigma\tau) \neq M(\sigma)$. Let $\sigma_0 = (0, 0)$. For $n = 1, 2, \ldots$, there are $L(0)\,L(1)\ldots L(n)$-consistent strings τ_n such that

$$\begin{aligned} M(\sigma_n) &\neq M(\sigma_{n-1}) \text{ where} \\ \sigma_n &= \sigma_{n-1}\,(0, L(0))\,(1, L(1))\,\ldots\,(n, L(n))\,\tau_n. \end{aligned}$$

It follows that $T = \lim_n \sigma_n$ is a noisy informant for L and that M diverges on T, a contradiction. Thus N infers every $L \in S$ and NoisyInf $\subseteq$ FinInf$[K]$.

FinInf$[K]$ $\subseteq$ NoisyInf: For any oracle X let N^X denote that N is equipped with the oracle X. During the proof K is sometimes replaced by a recursive approximation $K_s = \{a_0, a_1, \ldots, a_s\}$ where $a_0, a_1, \ldots$ is a recursive enumeration of K; therefore it is necessary to denote explicitly the oracle supplied to N at the different situations. So let N^K be a FinInf$[K]$-learner for some class S.

A string α is called σ-consistent iff for all $x \in dom(\alpha)$ the pair $(x, \alpha(x))$ occurs in σ at least as often as any other pair (x, y). Now

$$M(\sigma) = \begin{cases} N^{K_{|\sigma|}}(\alpha) & \text{for the first } \sigma\text{-consistent } \alpha \in \{\beta_0, \beta_1, \ldots, \beta_{|\sigma|}\} \\ & \text{which satisfies } N^{K_{|\sigma|}}(\alpha)\downarrow \neq ? \text{ within } |\sigma| \text{ steps;} \\ ? & \text{otherwise, that is, there is no such } \alpha. \end{cases}$$

M NoisyInf infers S: Let T be a noisy informant for L and let $\alpha = \beta_i$ be the first string such that $\alpha \preceq L$, that is, $\alpha = L(0)L(1)\ldots L(n)$ for some n, and $N^K(\alpha) = e \neq ?$. Then α is σ-consistent for almost all $\sigma \preceq T$. Let $j < i$. Either $\beta_j \not\preceq L$ or $N^K(\beta_j) = ?$. In the first case β_j is not σ-consistent for almost all $\sigma \preceq T$, in the second case $N^{K_{|\sigma|}}(\beta_j) = ?$ for almost all $\sigma \preceq T$. So these β_j are considered only by finitely many $\sigma \preceq T$ and the output of M is $e = N^K(\alpha)$ for almost all $\sigma \preceq T$. Thus M converges on every noisy informant T to the index e of L and so M infers S from noisy informant. ∎

It is easy to see that the proof holds as well for learning functions as well for learning enumerable sets; the only major changes in the proof are to replace $\{0, 1\}$ by $\mathbb{N}$ and L by f.

Corollary 4.2.2 *A class $S \subseteq$ REC of functions can be learned from noisy informant in the limit iff S can be learned from noise-free informant finitely using the oracle K, that is,* Noisy = Fin[K].

Furthermore the proof of Theorem 4.2.1 relativizes. Since the Turing degrees coincide with the inference degrees of finite learning, that is, since FinInf[A] $\subseteq$ FinInf[B] $\Leftrightarrow A \leq_T B$ [43, Theorem 6.36], the relativized version of this theorem also characterizes the inference degrees for noisy informant. In the nonrelativized world, NoisyInf is between FinInf and ExInf.

Corollary 4.2.3
(a) NoisyInf[A] = FinInf[A'].
(b) NoisyInf[A] $\subseteq$ NoisyInf[B] *iff* $A' \leq_T B'$.
(c) FinInf $\subset$ NoisyInf $\subset$ ExInf.

While for sets the definitions of noisy informant and very noisy informant are equivalent, this equivalence does not hold in the field of inferring functions. But there remains a connection:

Theorem 4.2.4 *If $S \subseteq$ REC can be learned from noisy informant and some K-recursive function f bounds all functions $g \in S$, then S can also be learned from very noisy informant.*

Proof Let M infer S from noisy informant and let f_s be a uniform recursive sequence of functions which approximate f in the limit: $(\forall x)\,(\forall^\infty s)\,[f(x) = f_s(x)]$. Since f only has to be an upper bound, without loss of generality the f_s approximate f from below.

Every very noisy informant $T = w_0 w_1 \ldots$ for $g \in S$ can be translated into a new noisy informant $T' = v_0 v_1 \ldots$ as follows:

$$v_s = \begin{cases} w_s & \text{if } w_s = (x, y) \text{ and } y \leq f_s(x); \\ \# & \text{otherwise.} \end{cases}$$

Also in T' every pair $(x, g(x))$ occurs infinitely often since $(x, g(x))$ occurs infinitely often in T, $g(x) \leq f(x)$ and therefore $g(x) \leq f_s(x)$ for almost all s. On the other hand, if $y > f(x)$, then $y > f_s(x)$ for all x and therefore (x, y) never occurs in T'. Further if $y \leq f(x)$ and $y \neq g(x)$, then (x, y) occurs only finitely often in T and therefore also only finitely often in T'. Thus T' is a noisy informant for g. Since this translation is computable and can be done on all finite initial segments of T, M can infer g from T'. ▮

The converse does not hold. For example the class $\{e1^e0^\infty : e \in \mathbb{N}\}$ can be learned from very noisy informant, but it has no bound on $f(0)$ at all. On the other hand, the condition, that f is K-recursive cannot be weakened, since the class

$$\{0^x y 0^\infty : x \in \mathbb{N} \wedge 1 \leq y \leq f(x)\}$$

can be learned from very noisy informant iff some K-recursive function majorizes f.

4.3 Inference From Text

Comparing the definition for learning from noisy informant with those for learning from noisy text and from very noisy text, the second seems more to fit to its counterpart than the first one. But it turns out that learning from very noisy text is a very restrictive concept since here two restrictions add - that of texts (compared to informant) and that of severe noise. Indeed the class of all singleton sets can only be learned from noisy text and not from very noisy text. This result has a mirror-image: the class of all constant functions can only be learned from noisy informant but not from very noisy informant. So the next theorem indicates why noisy text is more interesting than very noisy text and noisy informant is more interesting than very noisy informant.

Theorem 4.3.1 *The class S containing all singleton sets $\{x\}$ can be learned from noisy text but not from very noisy text.*

Proof There is an easy algorithm to infer S from noisy text: For each input σw the learner just guesses $\{w\}$. Since for each given noisy text $w_0 w_1 \ldots$ almost all w_i are the single data-element x of the singleton language $\{x\}$ to be learned, this algorithm is correct.

Now the assumption that some machine M learns the sets $\{1\}, \{2\}, \ldots$ from very noisy text is used to construct a very noisy text T for the set $\{0\}$ on which M even does not converge to a guess. Let σ_0 be the empty string and $\sigma_{n+1} = \sigma_n 0 n^k$ for the first k with $M(\sigma_n 0 n^k)$ outputting an index for $\{n\}$. Such a k must exist since $\sigma_n 0 n^\infty$ is a very noisy text for $\{n\}$. The limit T of all these σ_n is a very noisy text for $\{0\}$ since 0 occurs infinitely often (in each σ_n exactly n times) and each n occurs only the k times for the k in the definition of σ_{n+1}. Since M outputs infinitely many different indices on T (for each set $\{n\}$ at least one), M does not converge on T. So there is no machine which infers S from very noisy text. ▮

Locking sequences are an important tool in understanding learning from text. Therefore it is useful to define them also for inference from noisy text. Let M infer L. σ is called a locking sequence for L iff

- $M(\sigma) = e$ with $W_e = L$ and
- $M(\sigma\tau) = M(\sigma)$ for all $\tau \in L^*$.

Since $L = W_e$, σ is also called a locking sequence for the index e. The proof, that a locking sequence exists, is almost identical to the one in the case of learning from noise-free text and is therefore omitted. Using the concept of locking sequences, the next theorem shows, that it is impossible to learn a class of sets from noisy text, if some of the sets is a proper subset of some other.

Theorem 4.3.2 *If $L' \subset L$ then $\{L', L\} \notin$ NoisyTxt.*

Proof Let M infer at least L and has a locking sequence σ for L. Further let $w_0w_1\ldots$ be an enumeration of L' in which every element of L' occurs infinitely often. Now $\sigma w_0 w_1 \ldots$ is a noisy text for L', but since $e = M(\sigma)$ is an index for L and $M(\sigma w_0 w_1 \ldots w_n) = e$ for all n (by $w_0, w_1, \ldots \in L$), M does not infer L' from noisy text. ∎

Case, Jain and Sharma [27, 28] generalized this theorem to learning with errors in the sense that the learner outputs in the limit only grammars which generate a finite variant of the set to be learned. The result is now that whenever $L' \subseteq L$ and $L - L'$ is infinite then $\{L, L'\}$ cannot be learned by a learner permitted to make finitely many errors.

They also showed that for BC-learning from noisy text with finitely many errors (that is, the learner outputs on every noisy text T for an $L \in S$ almost always some grammar which differs from L only at finitely many places) is always possible if S is uniformly recursive. In particular if an index is given for this uniform enumeration, it can be turned effectively into a machine which learns S from noisy texts with finitely many errors behaviourally correct.

The nonlearnability of these classes depends besides noise on a second factor: the noisy text may be uncomputable. Case and Jain [24] also analyzed the case, where noise is present but the noisy text must be computable. Within this setting, classes like $\{\emptyset, \mathbb{N}\}$ can be BC-learned, but syntactic convergence is still impossible: For Ex-style learning, computable noisy texts are as hard as noisy texts in general.

The severe restriction from Theorem 4.3.2 contrasts with the fact, that if the sets to be learned are the graphs $\mathcal{G}$ of a set of functions, then there is no difference between noisy and noise-free text, so learning from noisy text is in general not so restrictive as learning from noisy informant.

Theorem 4.3.3 *Let S_{graph} be the set of the graphs of some set of total recursive functions. Then $S_{\text{graph}} \in$ NoisyTxt $\Leftrightarrow S_{\text{graph}} \in$ ExTxt.*

Proof It is sufficient to proof the direction "$S_{\text{graph}} \in \text{ExTxt} \Rightarrow S_{\text{graph}} \in \text{NoisyTxt}$" since the other one follows directly from the definitions. So let $S_{\text{graph}} \in$ ExTxt via M and $T = w_0w_1\ldots$ be a noisy text for the graph of a function g. T contains only finitely many (x, y) with $y \neq g(x)$ while each pair $(x, g(x))$ occurs infinitely often in T. There is a first k such that the data-elements $w_k, w_{k+1}, \ldots$ are all correct, that is, are of the form $(x, g(x))$ for some x. So $w_k w_{k+1} w_{k+2} \ldots$ is a noise-free text for $graph(g)$ and M converges on this text to an index of $graph(g)$. The idea is now that the NoisyTxt learner N approximates k from below and then simulates M on the data-elements starting with w_k. This can be done since each incorrect element is discovered in the limit. Formally N is given as follows.

> On input $w_0w_1\ldots w_n$, N searches the least $m \leq n$ such that the information $w_m, w_{m+1}, \ldots, w_n$ is not contradictory, that is,
>
> $$(\forall i, j, x, y, z)\,[m \leq i \leq j \leq n \wedge w_i = (x, y) \wedge w_j = (x, z) \Rightarrow y = z],$$
>
> and then N outputs $M(w_m w_{m+1} \ldots w_n)$.

For almost all n, this m (depending on n) coincides with k and therefore

$$(\forall^\infty n)\,[N(w_0w_1\ldots w_n) = M(w_k w_{k+1} \ldots w_n)\,].$$

Thus N on the noisy text $w_0 w_1 \dots$ and M on the text $w_k w_{k+1} \dots$, both converge to the same index for $graph(g)$. So $S_{\mathrm{graph}} \in \mathrm{NoisyTxt}$ via N. ∎

Some classes of functions can be inferred in the limit, but are not in $\mathrm{FinInf}[A]$ for any oracle A. The class

$$\{f : (\forall^\infty x)\,[f(x) = 0]\} \in \mathrm{ExInf} - \mathrm{FinInf}[A]$$

is an example. So their graphs are ExTxt and NoisyTxt learnable, but not $\mathrm{FinInf}[A]$ and $\mathrm{FinTxt}[A]$ learnable for any oracle A. Therefore inference from noisy text is not contained in finite inference relative to any oracle:

Corollary 4.3.4 $\mathrm{NoisyTxt} \not\subseteq \mathrm{FinTxt}[A]$ *for all oracles* A.

So in contrary to the case of the informant, the classes $\mathrm{FinTxt}[K]$ and NoisyTxt do not coincide. Indeed Theorem 4.3.7 will show, that $\mathrm{FinTxt}[A]$ and NoisyTxt are incomparable for all oracles A. Since Theorem 4.3.7 also studies the connections $\mathrm{FinTxt}[A] \subseteq \mathrm{NoisyTxt}[B]$ it is worth to look first at the inference degrees with respect to learning from noisy text:

Theorem 4.3.5 *The following holds for all oracles A and B:*
(a) *If A is enumerable then* $\mathrm{NoisyTxt}[A] \subseteq \mathrm{NoisyTxt}[B] \Leftrightarrow A \leq_T B$.
(b) *If $A, B \geq_T K$ then* $\mathrm{NoisyTxt}[A] \subseteq \mathrm{NoisyTxt}[B] \Leftrightarrow A' \leq_T B'$.
(c) $\mathrm{NoisyTxt}[A] = \mathrm{NoisyTxt}$ *iff $A \leq_T K$ and A has recursive or 1-generic degree.*

Proof (a): Obviously $A \leq_T B \Rightarrow \mathrm{NoisyTxt}[A] \subseteq \mathrm{NoisyTxt}[B]$ holds. For the converse consider the class S consisting of the enumerable set A and all sets $\{x\}$ with $x \notin A$. It will be shown that $S \in \mathrm{NoisyTxt}[A]$ and $S \in \mathrm{NoisyTxt}[B]$ only for $B \geq_T A$.

The $\mathrm{NoisyTxt}[A]$-learner M for S is given as follows: $M(w_0 w_1 \dots w_n)$ outputs a canonical index of the set $\{w_n\}$ if $w_n \notin A$ and a fixed index of A iff $w_n \in A$. M is obviously A-recursive; further if $w_0 w_1 \dots$ is a noisy text for $\{x\}$, then $w_i = x$ for almost all i and M converges to an index for $\{x\}$. If $w_0 w_1 \dots$ is a noisy text for A then $w_i \in A$ for almost all i and M almost always outputs the same index for A.

On the other hand, assume that $S \in \mathrm{NoisyTxt}[B]$ via some B-recursive N. A has a locking sequence σ. If $x \notin A$, then N converges on σx^∞ to an index of $\{x\}$, thus $N(\sigma x^n) \neq N(\sigma)$ for some n. If $x \in A$, then $N(\sigma x^n) = N(\sigma)$ for all n since σ is a locking sequence for A. In short

$$x \in A \;\Leftrightarrow\; (\forall n)\,[N(\sigma x^n) = N(\sigma)]$$

and the enumerable set A is coenumerable relative to B. Thus $A \leq_T B$.

(b): Let $A, B \geq_T K$, $A' \leq_T B'$ and $S \in \mathrm{NoisyTxt}[A]$ via M. It is shown that M can be translated into a $\mathrm{NoisyTxt}[B]$-learner N for the same class S. Let $\eta_0, \eta_1, \dots$ be an 1-1 enumeration of all strings in $\mathbb{N}^*$. The set of all (e, m) such that η_m is locking sequence for W_e is recursive in A' by the formula

$$E = \{(e, m) : (\forall \tau \in {W_e}^*)\,[M(\eta_m \tau) = M(\eta_m)]\}.$$

Thus E has a B-recursive approximation E_n such that without loss of generality no E_n is void. The new B-recursive machine N infers $L \in S$ from the text $w_0 w_1 \dots$ as follows:

For input $w_0 w_1 \dots w_n$ find the first pair $(e, m) \in E_n$ such that the norm

$$e + m + |\{i \leq n : w_i \notin W_e\}|$$

of (e, m) with respect to the current input $w_0 w_1 \dots w_n$ is minimal among the norms of all $(e', m') \in E_n$ with respect to the same input $w_0 w_1 \dots w_n$. Then output e.

Since $B \geq_T K$ the W_e are uniformly decidable relative to B. Since for each n there is a pair $(e, m) \in E_n$, the algorithm finds at least one e. Furthermore N has to compare the pair (e, m) only with a finite number of other pairs (e', m') since almost all pairs (e', m') have a higher norm than (e, m). Thus the algorithm terminates using the B-oracle.

Since for every set $L \in S$ there is a pair $(m, e) \in E$ with $W_e = L$, either the algorithm finds this pair for sufficient long n and converges to e or the algorithms converges to e' for some other pair (m', e'). Assume by way of contradiction, that the algorithm takes the second case for some e' with $W_{e'} \neq L$. If there is some $w \in L - W_{e'}$, then this w occurs infinitely often. While the norm of (m, e) with respect to each input $w_0 w_1 \dots w_n$ is bounded by a constant c, the norm of (m', e') is greater than the number of occurrences of w in the input seen so far and so the norm of (m', e') is almost always greater than c and greater than the norm of (m, e). From this contradiction it follows that the algorithm takes e' only if $L \subset W_{e'}$. Since $(m', e') \in E$, it follows that $M(\eta_{m'} \tau) = e'$ for all $\tau \in W_{e'}$ and in particular $M(\sigma' \tau) = e'$ for all $\tau \in L^*$. Since M converges to e' on some noisy text $T \in \sigma L^\infty$, e' must be an index for L, a contradiction. So this case also fails and N infers S.

For the converse direction, let C be a retraceable set of degree A', which is coenumerable in A. Using this set C as a parameter, the following class $S \in \text{NoisyTxt}[A]$ is constructed coding the set C such that every NoisyTxt$[B]$-learner N allows to enumerate C in the limit and thus to compute A in the limit. The class S consists of the sets

$$\begin{array}{lll} \{x, 0\} & \text{iff} & x > 0 \text{ and } x \in C, \\ \{x\} & \text{iff} & x > 0 \text{ and } x \notin C. \end{array}$$

Further C_s denotes an A-recursive approximation of C. Now given any input σ, let $x(\sigma)$ denote the last $y > 0$ which occurs in σ, that is, $x(\sigma) = y \Leftrightarrow \sigma \in \mathbb{N}^* y 0^*$. If $\sigma \in 0^*$ then $x(\sigma) = 0$. Now $M(\sigma)$ outputs some index of the set

$$\begin{array}{ll} \{x(\sigma), 0\} & \text{if } x(\sigma) \in C_{|\sigma|} \\ \{x(\sigma)\} & \text{otherwise.} \end{array}$$

If T is a noisy text for $\{0, x\}$ or $\{x\}$, then $x(\sigma) = x$ for almost all $\sigma \preceq T$. Further $x \in C_{|\sigma|}$ iff $x \in C$ for almost all $\sigma \preceq T$. So $S \in \text{NoisyTxt}[A]$ via M.

Thus $S \in \text{NoisyTxt}[B]$ via some B-recursive N. If $x \in C$ then there is a locking sequence σ such that $N(\sigma\tau) = e$ for some index e of $\{0, x\}$ and all $\tau \in \{0, x\}^*$. On the other hand if $x \notin C$ then N converges on every text σx^∞ to an index for $\{x\}$. Thus

$$x \in C \;\Leftrightarrow\; (\exists \sigma)\,(\exists e)\,(\forall n)\,[0 \in W_e \wedge N(\sigma x^n) = e].$$

Therefore C is enumerable in B'; since C is retraceable, C is even recursive in B' and $A' \leq_T B'$ follows.

(c): The proof of this fact is similar to that of [89, Theorem 10.5] concerning ExTxt inference degrees. ▮

Theorem 4.3.5 also holds with ExTxt instead of NoisyTxt [89, Theorems 10.2, 10.4 and 10.5]. So it is likely, that the structures of the ExTxt and NoisyTxt inference degrees coincide and the following conjecture holds:

Conjecture 4.3.6 $\mathrm{NoisyTxt}[A] \subseteq \mathrm{NoisyTxt}[B] \Leftrightarrow \mathrm{ExTxt}[A] \subseteq \mathrm{ExTxt}[B]$.

The next result deals with the relation between $\mathrm{FinTxt}[A]$ and $\mathrm{NoisyTxt}[B]$.

Theorem 4.3.7 $\mathrm{FinTxt}[A] \subseteq \mathrm{NoisyTxt}[B] \Leftrightarrow K \leq_T B \wedge (A \oplus K)' \leq_T B'$.

Proof The proof consists of three parts:
(a) If $\mathrm{FinTxt} \subseteq \mathrm{NoisyTxt}[B]$ then $K \leq_T B$.
(b) If $\mathrm{FinTxt}[A] \subseteq \mathrm{NoisyTxt}[B]$ then $(A \oplus K)' \leq_T B'$.
(c) If $(A \oplus K)' \leq_T B'$ and $K \leq_T B$ then $\mathrm{FinTxt}[A] \subseteq \mathrm{NoisyTxt}[B]$.

(a): Let S contain K plus all singletons $\{x\}$ with $x \notin K$. There is a recursive function f such that

$$W_{f(x)} = \begin{cases} K & \text{if } x \in K; \\ \{x\} & \text{if } x \notin K. \end{cases}$$

Now the FinTxt-learner waits for the first x to appear on the input, outputs the guess $W_{f(x)}$ and terminates. The proof of Theorem 4.3.5 (a) shows that $S \in \mathrm{NoisyTxt}[B]$ only if $K \leq_T B$.

(b): Let $\mathrm{FinTxt}[A] \subseteq \mathrm{NoisyTxt}[B]$. Let C be a retraceable set of degree $(A \oplus K)'$ which is coenumerable in $A \oplus K$. So $\overline{C}$ is the domain of the partial function $\psi^{A \oplus K}$. Let U_y contain all x such that the computation of $\psi^{A \oplus K_y}(x)$ terminates within y steps and equals to that relative to $A \oplus K$: whenever an odd number $2z+1$ is queried, then either $z \in K_y$ or $z \notin K$. Furthermore, all queries are made to numbers below y. Note that $U_y \subseteq \overline{C}$. Further each $x \notin C$ is in almost all sets U_y. The sets U_y are uniformly coenumerable in A and there is a recursive function h such that $\overline{U_y} = W_{h(y)}^{\{z \in A : z < y\}}$. Now let S consist of the sets

$$\begin{array}{lll} \{2x, 1, 3, 5, 7, \ldots\} & \text{iff} & x \in C, \\ \{2x, 2y+1\} & \text{iff} & x \in U_y. \end{array}$$

First $S \in \mathrm{FinTxt}[A]$ is shown. The learner waits until the even number $2x$ and an odd number $2y+1$ are in the input. Then it outputs the index $f(x,y)$ where

$$W_{f(x,y)} = \begin{cases} \{2x, 2y+1\} & \text{if } x \in U_y, \text{ that is, if } x \notin W_{h(y)}^{\{z \in A : z < y\}}; \\ \{2x, 1, 3, 5, 7, \ldots\} & \text{otherwise, that is, if } x \in W_{h(y)}^{\{z \in A : z < y\}}. \end{cases}$$

The function f is A-recursive and queries A only below y. $f(x,y)$ contains a table of $A(0), A(1), \ldots, A(y)$ and first enumerates $2x$ and $2y+1$ into $W_{f(x,y)}$. Then the machine emulates the enumeration of $W_{h(y)}^{\{z \in A : z < y\}}$ until x is enumerated into this set; if this happens then all odd numbers are enumerated into $W_{f(x,y)}$. So the learner guesses $\{2x, 2y+1\}$ if $x \in U_y$ and guesses $\{2x, 1, 3, 5, 7, \ldots\}$ if $x \notin U_y$, in particular if $x \in C$. Thus $S \in \mathrm{FinTxt}[A]$ and $S \in \mathrm{NoisyTxt}[B]$ via some B-recursive M.

If $x \in C$ then M infers $V_x = \{2x, 1, 3, 5, 7, \ldots\}$ and V_x has a locking sequence. If $x \notin C$, then $x \in U_y$ for some y. Then M infers $\{2x, 2y+1\}$ and V_x has no locking sequence since $\{2x, 2y+1\} \subset V_x$. So the equivalences

$$\begin{array}{rcl} x \in C & \Leftrightarrow & V_x \text{ has a locking sequence} \\ & \Leftrightarrow & (\exists \sigma)\,(\exists e)\,(\forall \tau \in {V_x}^*)\,[M(\sigma\tau) = e \wedge |W_e| > 2] \end{array}$$

hold. C is enumerable in B'. Since C is retraceable, $C \leq_T B'$ and $(A \oplus K)' \leq_T B'$.

(c): If $B \geq_T K$ and $(A \oplus K)' \leq_T B'$, then $\mathrm{NoisyTxt}[A \oplus K] \subseteq \mathrm{NoisyTxt}[B]$. So it remains to show that $\mathrm{FinTxt}[A] \subseteq \mathrm{NoisyTxt}[A \oplus K]$.

Let $S \in \mathrm{FinTxt}[A]$. Form the definition of finite learning follows, that there are A-recursive functions f, g such that for every $L \in S$:

- If $D_{f(i)} \subseteq L$ then $W_{g(i)} = L$;
- There is some i with $D_{f(i)} \subseteq L$.

Such a sequence can be obtained by A-recursively enumerating all strings σ on which a given FinTxt[A]-learner M outputs some $e \neq ?$. Then for the i-th such string σ_i, let $D_{f(i)} = range(\sigma_i)$ and $g(i) = M(\sigma_i)$. Without loss of generality $S \neq \{\emptyset\}$ and therefore $D_{f(i)} \neq \emptyset$ for all i. Now the following machine N infers S from noisy text:

- For all $i \leq |\sigma|$, N calculates c_i which is the maximal number y such that every $x \in D_{f(i)}$ occurs y times in σ.
- N finds the least i with $c_i \geq c_j$ for all $j \leq |\sigma|$.
- N outputs $g(j)$ for the least j with $D_{f(j)} \subseteq W_{g(i)}$ and $D_{f(i)} \subseteq W_{g(j)}$.

In a given text T for L, only finitely often, say k times, occurs some $x \notin L$. On the other hand each $x \in L$ occurs infinitely often in T. There is a minimal j with $D_{f(j)} \subseteq L$ and $W_{g(j)} = L$. Every $x \in D_{f(j)}$ occurs at least $k+1$ times in almost all $\sigma \preceq T$, thus for almost all $\sigma \preceq T$, the i computed in the second step satisfies $W_{g(i)} = L$. Then $D_{f(j)} \subseteq W_{g(i)}$ and $D_{f(i)} \subseteq W_{g(j)}$. Furthermore, j is the smallest index with this property and N outputs $g(j)$. It follows that N converges on T to $g(j)$ and that N infers S from noisy text. ∎

So the only relation is FinTxt[K] $\subset$ NoisyTxt[K] and there is no equivalent statement to FinInf[K] $=$ NoisyInf. The class

$$\{\mathbb{N} - \{i\} : i \in \mathbb{N}\}$$

is learnable from very noisy text but not FinTxt[A] learnable for any oracle A.

4.4 Characterizing Finite Learning From Text With K-Oracle

Since the equivalence NoisyInf $=$ FinInf[K] did not transfer to learning from text, the question arrizes whether there is an alternative characterization for the class FinTxt[K]. Indeed such a characterization can be found using monotonicity notions.

Kapur [75] introduced (in the restricted context of section 4.6) the notion of strongly dual monotonic inference, that is, whenever the learner makes a mind change from e to e', then the new language must be more special: $W_{e'} \subseteq W_e$. Jain and Sharma [65] and Kinber and Stephan [80] generalized this and other notions of monotonic inference to learning enumerable languages. While the class FinTxt[K] cannot be characterized in terms of noisy inference, it turned out to be equivalent with strongly dual monotonic inference without oracle. The reader may find more information on the field of monotonic learning in [68, 75, 94, 146, 150].

Theorem 4.4.1 $S \in$ FinTxt[K] *iff S can be learned via a recursive and strongly dual monotonic machine.*

Proof SMon$^{\mathrm{d}}$Txt $\subseteq$ FinTxt[K]: Assume that M SMon$^{\mathrm{d}}$Txt infers S. Then a new FinTxt[K]-learner N for S can be defined as follows:

$$N(\sigma) = \begin{cases} e & \text{if there is a locking sequence } \tau \text{ for } W_e \text{ with } M(\tau) = e, \\ & |\tau| \leq |\sigma| \text{ and } range(\tau) \subseteq range(\sigma); \\ ? & \text{otherwise.} \end{cases}$$

Further N is required to make no further mind change if it once has made a guess. Since N has only to check the strings τ in a finite set whether they are locking sequences for $W_{M(\tau)}$ or not, this can be done with K-oracle: τ is a locking sequence iff $M(\tau\eta) = M(\tau)$ for all $\eta \in {W_{M(\tau)}}^*$. By the strong dual monotonicity and the construction it holds that $range(\tau) \subseteq range(\sigma) \subseteq L \subseteq W_{M(\tau)}$. Since some text for L starts with τ and since M makes no mind change on this text after guessing $M(\tau)$, $M(\tau)$ is an index for L. On the other hand there is a locking sequence τ and whenever σ is long enough, that is, whenever $range(\sigma) \supseteq range(\tau)$ and $|\sigma| \geq |\tau|$, the locking sequence is discovered.

$\mathrm{FinTxt}[K] \subseteq \mathrm{SMon^dTxt}$: This proof is similar to the corresponding part of Theorem 4.2.1. Given the $\mathrm{FinTxt}[K]$-learner M, the guess $N(\sigma)$ of the new $\mathrm{SMon^dTxt}$-learner is calculated as follows:

- Let $s = |\sigma|$. N searches for the shortest $\tau \preceq \sigma$ with $M^{K_s}(\tau) \neq ?$.
- If there is no such τ, then N outputs some index of $\mathbb{N}$.
- Otherwise N computes $e = M^{K_s}(\tau)$ and outputs some index $f(e)$ of the set

$$W_{f(e)} = \begin{cases} W_e & \text{if } M^{K_t}(\eta) = M^{K_s}(\eta) \text{ for all } \eta \preceq \tau \text{ and } t \geq s; \\ \mathbb{N} & \text{otherwise, that is, if } M^{K_t}(\eta) \neq M^{K_s}(\eta) \\ & \qquad \text{for some } \eta \preceq \tau \text{ and } t \geq s. \end{cases}$$

  The condition in the "otherwise"-case is enumerable, thus an uniform algorithm for $W_{f(e)}$ first enumerates W_e until it discovers that the condition in the "otherwise"-case holds and then enumerates the whole set $\mathbb{N}$. So f is recursive.

The inference process converges to the guess e of M and all previously guessed grammars enumerate $\mathbb{N}$ – if not directly then at the moment that an error in the estimation M^{K_s} is discovered. ▮

One might ask, if this theorem relativizes. It does not relativize in the obvious way; the relativization needs the concept of inferring with finitely many queries. A machine M $\mathrm{SMon^dTxt}[A*]$ infers L iff M is strongly dual monotonic and L has a locking sequence σ such that $M(\sigma\tau) = M(\sigma)$ for all $\tau \in {W_{M(\sigma)}}^*$ and M makes the same oracle queries while calculating $M(\sigma)$ and $M(\sigma\tau)$. An equivalent definition is that M on every text for L makes only finitely many queries to A. See [43, Definition 2.23 and Section 5.2] for more information. Now the relativizations are:

Theorem 4.4.2
(a) $\mathrm{SMon^dTxt}[A*] = \mathrm{FinTxt}[A \oplus K]$.
(b) $\mathrm{SMon^dTxt}[A] \subseteq \mathrm{FinTxt}[A']$.
(c) $\mathrm{SMon^dTxt}[A] = \mathrm{FinTxt}[A']$ *for 1-generic sets A.*
(d) $\mathrm{SMon^dTxt}[K] \subset \mathrm{FinTxt}[K']$.

Proof The proofs of (a) and (b) follow the corresponding parts of Theorem 4.4.1. (c) follows from the fact, that $A' \equiv_T A \oplus K$ for every 1-generic set A. The inclusion in (d) follows from (b) and the class S containing the sets

$$\begin{array}{rll} \{x\} & \text{iff} & x > 0 \text{ and } x \in K''; \\ \{0, x\} & \text{iff} & x > 0 \text{ and } x \notin K''; \end{array}$$

witnesses that the inclusion $\mathrm{SMon^dTxt}[K] \subset \mathrm{FinTxt}[K']$ is proper: The proof of Theorem 4.3.7 shows, that $S \in \mathrm{FinTxt}[K']$ since K'' is enumerable in K'. To show that

$S \notin \mathrm{SMon}^{\mathrm{d}}\mathrm{Txt}[K]$ assume by way of contradiction, that a K-recursive machine M infers S dual strong monotonically from text. If $x \in K''$, then M infers $\{x\}$ from the text x^∞ and there is an n such that $M(x^n)$ outputs some index for $\{x\}$. Otherwise ($x \notin K''$) the learner M must identify $\{0, x\}$ on each text $x^{n+1}0^\infty$ and therefore $M(x^n)$ always outputs a language which not only contains x but also 0. Thus

$$x \in K'' \;\Leftrightarrow\; (\exists n)\,[0 \notin W_{M(x^n)}].$$

Since the computation of $M(x^n)$ and the test, whether $0 \notin W_{M(x^n)}$, are recursive in K, K'' would be enumerable in K, which is obviously not possible. Thus such an M does not exist and the inclusion is proper. ∎

4.5 Informant Versus Text

It follows immediately from the definition that every class of enumerable sets, which is learnable from text, is also learnable from informant. But this does not hold in the case of noisy inference, since the definitions of noisy text and noisy informant do not match so good as in the standard case. So the following holds:

Theorem 4.5.1 NoisyInf$[A]$ *and* NoisyTxt$[B]$ *are incomparable for all oracles A and B.*

Proof The class $\{\emptyset, \mathbb{N}\}$ is finitely learnable from noisy informant, but not learnable from noisy text by Theorem 4.3.2. The class mentioned to prove Corollary 4.3.4 is in NoisyTxt$[B]$ for all oracles B, but not in FinInf$[A']$ for any oracle A, in particular not in NoisyInf$[A]$. ∎

So it is better to look for inclusions which hold under additional constraints. The first is to consider very noisy text versus (very) noisy informant; note that in the case of characteristic functions of sets, there is no difference between noisy and very noisy informant. Given a noisy informant $T = (w_0, b_0),\ (w_1, b_1),\ \ldots$ for a set L, the sequence T containing all w_i with $b_i = 1$ is a very noisy text for L: w_i occurs in T' infinitely often iff $(w_i, 1)$ occurs in T infinitely often iff $w_i \in L$. Thus one can translate every noisy informant into a very noisy text and simulate the machine learning from very noisy text. Thus the following theorem holds (and also relativizes to every oracle):

Theorem 4.5.2 *Every class of sets learnable from very noisy text is also learnable from noisy informant.*

While NoisyInf$[A] \not\subseteq$ NoisyTxt$[B]$ for all oracles A and B, there is a connection if the machine learns from text without any noise:

Theorem 4.5.3 NoisyInf$[A] \subseteq$ ExTxt$[B]$ $\Leftrightarrow$ *$A' \leq_T B'$.*

Proof ($\Rightarrow$): Let NoisyInf$[A] \subseteq$ ExTxt$[B]$. Further let C be a retraceable set of degree A' and let the class S contain the sets

$$\begin{array}{lcll} X_x & = & \{x, x{+}1, x{+}2, \ldots\} & \text{iff} \quad x \in C; \\ X_{x,y} & = & \{x, x{+}1, x{+}2, \ldots, x{+}y\} & \text{iff} \quad x \notin C \ \text{ and } \ y \in \mathbb{N}. \end{array}$$

The class has a FinInf$[A']$-learner which on input σ outputs indices of the following sets:

$$\begin{array}{ll} X_x & \text{if } x \in C \ \text{ and } \ \sigma \succeq 0^x1; \\ X_{x,y} & \text{if } x \notin C \ \text{ and } \ \sigma \succeq 0^x1\cdot1^y0; \\ ? & \text{otherwise.} \end{array}$$

The learner makes only one guess and is recursive in C, that is, recursive in A'. From $\text{FinInf}[A'] = \text{NoisyInf}[A]$ follows, that $S \in \text{ExTxt}[B]$ via some N.

If $x \in C$ then N has a locking sequence σ for the set X_x. If $x \notin C$ then there is no locking sequence $\sigma \in X_x^*$: The range of σ is finite and there is some $y > \max(range(\sigma))$ such that $M(\sigma)$ is not an index for $X_{x,y}$. Therefore there is some $\tau \in X_{x,y}^*$ with $M(\sigma\tau) \neq M(\sigma)$. So the equivalences

$$\begin{aligned} x \in C \quad &\Leftrightarrow \quad N \text{ has a locking sequence on the set } X_x \\ &\Leftrightarrow \quad (\exists \sigma \in X_x^*)\,(\forall \tau \in X_x^*)\,[N(\sigma\tau) = N(\sigma)] \end{aligned}$$

hold and show that C is enumerable in B'. Since C is retraceable, C is recursive in B' and $A' \leq_T B'$.

($\Leftarrow$): From $S \in \text{NoisyInf}[A]$ and $A' \leq_T B'$, it follows by Corollary 4.2.3 (a) that $S \in \text{NoisyInf}[B]$ via some B-recursive learner M. Now a B-recursive $\text{ExTxt}[B]$ learner N just translates the given text $w_0\, w_1 \ldots$ into a noisy informant $v_0\, v_1 \ldots$ for M and emulates M:

From input $w_0\, w_1 \ldots w_n$ compute $v_0, v_1, \ldots, v_n$ via

$$v_{\langle i,j\rangle} = \begin{cases} (i,1) & \text{if } i \in \{w_0, w_1, \ldots, w_j\}; \\ (i,0) & \text{otherwise } (i \notin \{w_0, w_1, \ldots, w_j\}); \end{cases}$$

and output $M(v_0\, v_1 \ldots v_n)$.

Since $j \leq \langle i,j\rangle$ the values $v_0, v_1, \ldots, v_n$ are computed without accessing the input-text beyond w_n, thus the computation is well-defined. Furthermore the whole sequence $v_0\, v_1 \ldots$ is a noisy informant for L: if $i \notin L$ then i does not occur in the sequence $w_0\, w_1 \ldots$ and thus only $(i,0)$ occurs in the informant. If $i \in L$ then $w_n = i$ for some n and $v_{\langle i,j\rangle} = (i,1)$ for all $j \geq n$, that is, $(i,1)$ occurs infinitely often in the noisy informant and $(i,0)$ only finitely often (at most n times). So N behaves on the text $w_0\, w_1 \ldots$ exactly as M on the noisy informant $v_0\, v_1 \ldots$ and thus $S \in \text{ExTxt}[B]$ via N. ∎

4.6 Learning Uniformly Recursive Families

Angluin [2] introduced the concept of learning, where the class S to be learned must have a uniformly recursive representation $L_0, L_1, \ldots$, that is, $S = \{L_0, L_1, \ldots\}$ and the function giving the characteristic function $L_k(x)$ for an index k and an input x is effective in both parameters k and x. This 4.6-th section is dedicated to this model of learning uniform recursive families of languages. Zeugmann's Habilitationsschrift [150] gives an overview on this field. There are three well-known forms of learning uniformly recursive family $\{L_0, L_1, \ldots\}$ of languages:

- **Exact Learning:** The learner outputs indices of the original uniformly recursive family $\{L_0, L_1, \ldots\}$.
- **Class Preserving Learning:** The learner outputs indices of some uniformly recursive family $\{H_0, H_1, \ldots\}$ with $\{L_0, L_1, \ldots\} = \{H_0, H_1, \ldots\}$.
- **Class Comprising Learning:** The learner outputs indices of some uniformly recursive family $\{H_0, H_1, \ldots\}$ with $\{L_0, L_1, \ldots\} \subseteq \{H_0, H_1, \ldots\}$.

In the context of noisy inference these three notions turn out to be equivalent. Furthermore, they are very restrictive, therefore the results, in particular with respect to relativization, are different from those in section 4.3

Theorem 4.6.1 *For a uniformly recursive family $S = \{L_i\}$ the following are equivalent:*
(a) $(\forall i, j)\,[L_i \subseteq L_j \Rightarrow L_i = L_j]$.
(b) *S is exactly learnable from noisy text.*
(c) *S is class preserving learnable from noisy text.*
(d) *S is class comprising learnable from noisy text.*

Proof (b $\Rightarrow$ c) and (c $\Rightarrow$ d) are obvious. Further (d $\Rightarrow$ a) follows from Theorem 4.3.2 which states that any class learnable from noisy text does not contain two languages such that one is a proper subset of the other.

(a $\Rightarrow$ b): Let S fulfill the requirement from (a) and let $w_0 w_1 \ldots$ be a noisy text for some $L \in S$.

M converges to that i for which the norm $i + |\{m : w_m \notin L_i\}|$ is minimal.

For each i, M approximates this norm $i + |\{m : w_m \notin L_i\}|$ from below via looking at larger and larger parts of the input. If $L_i \neq L$ then $L_i \not\subseteq L$, in particular there is some $w \in L - L_i$. This w occurs infinitely often in the input and $i + |\{m : w_m \notin L_i\}| \geq i + |\{m : w_m = w\}| = \infty$. Otherwise $L_i = L$ and $i + |\{m : w_m \notin L_i\}| = i + k$ where $k = |\{m : w_m \notin L\}|$ is finite. The language L has a smallest index i. M converges to this minimal index i of L since its norm $i + k$ is finite, less than the norm of every other index and M can exploit the implicit knowledge that any $j > i + k$ has a norm greater than $i + k$. ∎

It is easy to see that Theorem 4.3.2 holds also in a relativized world, that is, that for any oracle A, $L \subset L' \Rightarrow \{L, L'\} \notin \mathrm{NoisyTxt}[A]$. Since avoiding inclusions is the only restriction to S and this restriction cannot be overcome, oracles do not help to increase the learning power:

Theorem 4.6.2 *If S is a uniformly recursive family which is* NoisyTxt[A] *learnable for some oracle A, then S is already learnable from noisy text without any oracle.*

The theorem needs that S is uniformly recursive. Note that this is totally different in the case of learning arbitrary families of enumerable languages since by Theorem 4.3.7, there is even no greatest inference degree and the jump of an oracle always supplies more learning power: $\mathrm{NoisyTxt}[A] \subset \mathrm{NoisyTxt}[A']$.

Case, Jain and Sharma [27] extended the result by showing that a NoisyTxt-learner for S can be computed from an index of a function $i, x \to L_i(x)$ whenever such a learner exists, that is, whenever $L = L'$ for all $L, L' \in S$ with $L \subseteq L'$. Similarly, a behaviourally correct learner dealing with recursive noisy texts can be synthesized from an index of a uniformly recursive family S whenever S is learnable from text in the limit [24].

Theorem 4.6.3 *For a uniformly recursive family $S = \{L_i\}$ the following are equivalent:*
(a) $(\forall i)\,(\exists$ *finite set* $D)\,(\forall j)\,[D \subseteq L_j \Leftrightarrow L_i = L_j]$.
(b) *S is exactly* FinTxt[K] *learnable.*
(c) *S is class preserving* FinTxt[K] *learnable.*
(d) *S is class comprising* FinTxt[K] *learnable.*

Proof (b $\Rightarrow$ c) and (c $\Rightarrow$ d) are obvious.

(a $\Rightarrow$ b): Let S fulfill the requirement from (a). The FinTxt[K]-learner asks on input σ with range D always iff D has two incomparable extensions in S. Or more formally, the learner asks the query

$$(\exists i, j, x)\,[D \subseteq L_i \wedge D \subseteq L_j \wedge (x \in L_i - L_j \vee x \in L_j - L_i)].$$

Since D is a fixed finite set, the query is K-recursive. By condition (a) after finite time the query receives a negative answer. Then the learner has only to output the first index i with $D \subseteq L_i$; this index exists since σ is part of a text of some language L_i.

(d $\Rightarrow$ a): If S is class comprising FinTxt[K] learnable, then for each L_i there is some string σ such that the FinTxt[K]-learner M makes a guess, which of course is correct, that is, $M(\sigma)$ guesses L_i. Assume that $D = range(\sigma) \subseteq L_j$. Then on one hand σ is also a prefix of some text for L_j and on the other hand M does not change its mind after guessing L_i on input σ. It follows that $L_j = L_i$ and for each i there is a D satisfying condition (a). ▮

This result has an effective variant [152, Theorem 7] which states that a uniformly recursive family $\{L_0, L_1, \ldots\}$ is FinTxt-learnable iff there is a recursive procedure which assigns to each i a finite set T_i such that $(\forall i)\,(\forall j)\,[T_i \subseteq L_j \Leftrightarrow L_i = L_j]$.

The degree-structure of the FinTxt and FinInf inference degrees relative to learning uniform recursive families of sets is different from the degree structure of learning arbitrary families of enumerable sets. Fortnow et al. [43, Theorem 6.36] showed that the latter coincides with the Turing degrees.

Let F[A] be the set of all functions f which are majorized by an A-recursive function and for which the set $\{(x, y) : y < f(x)\}$ is enumerable. Note that the last condition is not present in Section 2.2, so the definition of F[A] here does not exactly coincide with the notion given there.

Theorem 4.6.4 *In the context of learning uniformly recursive families, the following three statements are equivalent for all oracles A and B:*
(a) F[A] $\subseteq$ F[B].
(b) FinTxt[A] $\subseteq$ FinTxt[B].
(c) FinInf[A] $\subseteq$ FinInf[B].

Proof (a $\Rightarrow$ *b*): Let F[A] $\subseteq$ F[B] and $S = \{L_i\} \in$ FinTxt[A] be a uniformly recursive family. Without loss of generality if $i \neq j$ then $L_i \neq L_j$. Now for each i let $w_{i,x} = x$ if $x \in L_i$ and $w_{i,x} = \#$ otherwise ($x \notin L_i$). Further let $f_A(i)$ be the first x such that $M(w_{i,0}w_{i,1}\ldots w_{i,x}) \neq ?$. Certainly $range(w_{i,0}w_{i,1}\ldots w_{i,x}) = \{y \in L_i : y \leq x\} \not\subseteq L_j$ for every set $L_j \neq L_i$. Thus f_A dominates the function f_S given by

$$f_S(i) = \min\{x : (\forall j \neq i)\,(\exists y \leq x)\,[y \in L_i - L_j]\}.$$

The set $\{(i, y) : y < f_S(i)\}$ is enumerable and $f_S \in$ F[A]. From the hypothesis (a) follows, that a B-recursive function f_B majorizes f_S. The new learner M works as follows:

$$M(\sigma) = \begin{cases} i & \text{if } i \leq |\sigma| \text{ and } (\forall x \leq f_B(i))\,[x \in L_i \Leftrightarrow x \in range(\sigma)]; \\ ? & \text{otherwise.} \end{cases}$$

Since $\{x \in L_i : x \leq f_B(i)\} \not\subseteq L_j$ for all $j \neq i$, the i in the expression is unique and M is well-defined. Whenever M infers L_i then M outputs the symbol "?" until it has seen all elements in $\{x \in L_i : x \leq f_B(i)\}$; then it begins to output its only guess i. So $S \in$ FinTxt[B] via M.

(b $\Rightarrow$ c): Note that $S \in \text{FinInf}[A] \Leftrightarrow S' = \{L \oplus \overline{L} : L \in S\} \in \text{FinTxt}[A]$. Thus $S \in \text{FinInf}[A] \Rightarrow S' \in \text{FinTxt}[A] \Rightarrow S' \in \text{FinTxt}[B] \Rightarrow S \in \text{FinInf}[B]$ and therefore $\text{FinInf}[A] \subseteq \text{FinInf}[B]$.

(c $\Rightarrow$ a): This is done by showing the contrapositive, let $f \in \text{F}[A] - \text{F}[B]$ and let the A-recursive function f_A majorize f. Since f has a recursive approximation from below, the family

$$S = \{\text{finite and nonempty } D : \max(D) \leq \min(D) + f(\min(D))\}$$

is uniformly recursive. M finitely infers S relative to A as follows:

- If $\sigma = 0^i 1\tau$ and $|\tau| > f_A(i)$ then M outputs an index for $\{x \in dom(\sigma) : \sigma(x) = 1\}$.
- Otherwise M makes no guess, that is, $M(\sigma) = ?$.

On the other hand assume that $S \in \text{FinInf}[B]$ via N and let

$$f_B(i) = \min\{|\tau| : N(0^i 1\tau) \neq ?\}.$$

Since no B-recursive function majorizes f, there is some i with $f_B(i) < f(i)$. Thus there is $D \in S$ such that $i = \min(D)$ and inferring D, N makes its guess before seeing whether $i + f(i) \in D$ or not. N fails to infer either $D \cup \{i + f(i)\}$ or $D - \{i + f(i)\}$, but both sets are in S. ∎

Corollary 4.6.5 *For the inference degrees of* FinTxt *or* FinInf *learning uniformly recursive families, the following holds:*
(a) *All oracles of hyperimmune-free degree are in the least inference degree.*
(b) *All 1-generic oracles are in the least inference degree.*
(c) *If A is enumerable, then A's inference degree is below that of B iff $A \leq_T B$.*
(d) *$\{A : A \geq_T K\}$ is the greatest inference degree.*

Proof (a): If A is of hyperimmune-free degree then $\text{F}[A] = \text{F}[\emptyset]$ since any A-recursive function is majorized by a recursive one. Thus all sets of hyperimmune-free degrees belong to the least inference degree.

(b): Let A be a 1-generic set. Consider any $f \in \text{F}[A]$ and let the A-recursive function $f_A = \{e\}^A$ majorize f. The set

$$B = \{\eta : (\exists x)\,[\{e\}^\eta(x)\!\downarrow < f(x)\,]\,\}$$

is enumerable; since $\{e\}^A(x)\!\downarrow \geq f(x)$, no string in B is a prefix of A. Since A is 1-generic, there is a string $\sigma \preceq A$ such that no extension of σ is in B. Now let

$$g(x) = \{e\}^\eta(x) \text{ for the first } \eta \succeq \sigma \text{ such that } \{e\}^\eta(x)\!\downarrow \text{ within } |\eta| \text{ steps.}$$

g is recursive and majorizes f. Thus $f \in \text{F}[\emptyset]$, that is, $\text{F}[A] = \text{F}[\emptyset]$.

(c): Let A_s be a recursive enumeration of A and $\text{F}[A] \subseteq \text{F}[B]$. Now

$$f(x) = \begin{cases} s & \text{for the first } s \text{ with } x \in A_s; \\ 0 & \text{otherwise } (x \notin A, \text{ that is, there is no such } s); \end{cases}$$

is a function in $\text{F}[A]$ and some B-recursive function g majorizes f. Then $x \in A \Leftrightarrow x \in A_{g(x)}$ and $A \leq_T B$.

(d): The greatest degree can only contain oracles $A \geq_T K$ since K is enumerable; so it remains to show that $\mathrm{F}[A] \subseteq \mathrm{F}[K]$ for all oracles A. But this follows from the fact, that each function $f \in \mathrm{F}[A]$ is already K-recursive since $\{(x, y) : y < f(x)\}$ is an enumerable set. ∎

Theorem 4.6.6 $\mathrm{FinTxt}[K] \subset \mathrm{NoisyTxt}$ *in the context of uniformly recursive families.*

Proof Assume that S satisfies the condition (a) of Theorem 4.6.3. Then S also satisfies condition (a) of Theorem 4.6.1: If $L_i \subseteq L_j$ then there is some $D \subseteq L_i$ such that all sets $L \in S$ which contain D are equal to L_i. Then in particular, $L_i = L_j$.

The family $S = \{\mathbb{N} - \{i\} : i \in \mathbb{N}\}$ of all sets whose complement has cardinality 1 witnesses that the inclusion is proper. ∎

Sometimes the addition of an oracle allows to overcome the difference between two concepts. For example Theorem 4.4.1 showed, that in the general context the K-oracle closes the gap between finite and $\mathrm{SMon}^{\mathrm{d}}$ learning from text. The next result states for the setting of uniformly recursive families, that K closes the gap between conservative learning and learning in the limit. Angluin [2] introduced the notion *conservative*: A learner is defined to be conservative iff every mind change is motivated by a counterexample to the previous conjecture, that is, j is guessed on input $\sigma\tau$ after i was guessed on input σ only if $range(\sigma\tau) \not\subseteq L_i$.

Theorem 4.6.7 $\mathrm{ConsvTxt}[K] = \mathrm{ExTxt}$ *in the context of uniformly recursive families.*

Proof ($\Rightarrow$): Let $S = \{L_0, L_1, \ldots\} \in \mathrm{ConsvTxt}[K]$ via M^K and T be a text for some $L \in S$. M can be taken conservative for all oracles, that is, the following holds for every A: If $i = M^A(\sigma) \neq j = M^A(\sigma\tau)$ then $range(\sigma\tau) \not\subseteq L_i$. That means that M regardless of the oracle postpones any mind change until a witness is seen that makes it necessary. Since the sets $L_0, L_1, \ldots$ are uniformly recursive, this postponing does not require access to the oracle. On each input σ the ExTxt learner guesses $N(\sigma) = M^{K_{|\sigma|}}(\sigma)$. M converges on some $\tau \preceq T$ to an index i for the language to be learned. Now for sufficient long $\sigma \in \tau \cdot L_i^*$ it holds that $M^{K_{|\sigma|}}(\tau) = M^K(\tau) = i$ and therefore also $M^{K_{|\sigma|}}(\sigma) = i$ by the conservativeness of the machine $M^{K_{|\sigma|}}$. So $N(\sigma) = i$ for all sufficient long σ and N infers S.

($\Leftarrow$): Let $S = \{L_0, L_1, \ldots\} \in \mathrm{ExTxt}$ via N and $w_0 w_1 \ldots$ be a text for some $L \in S$. With K-oracle it is possible to test whether a given sequence σ is a locking sequence for N. The $\mathrm{ConsvTxt}[K]$-algorithm defines inductively (using the K-oracle) a new text $w_0\sigma_0 w_1\sigma_1 \ldots$ and emulates N on this text:

If there are i and τ such that

- $i + |\tau| \leq n$ and $\tau \in \{w_0, w_1, \ldots, w_n\}^*$;
- $w_0, w_1, \ldots, w_n \in L_i$;
- $(\forall \eta \in L_i^*)\,[N(w_0\sigma_0 w_1\sigma_1 \ldots w_n\tau\eta) = i]$;

then let $\sigma_n = \tau$, $M^K(w_0 w_1 \ldots w_n) = N(w_0\sigma_0 w_1\sigma_1 \ldots w_n\sigma_n) = i$;
else let $\sigma_n = \lambda$, $M^K(w_0 w_1 \ldots w_n) = ?$.

The algorithm works with K-oracle, since the search for the τ is bounded. If $w_0 w_1 \ldots$ is a text for L_i then $w_0\sigma_0 w_1\sigma_1 \ldots$ is also a text for L_i. N converges on this text to i and so M^K converges on the text $w_0 w_1 \ldots$ also to i. Furthermore if $M^K(w_0 w_1 \ldots w_n) = j \neq M^K(w_0 w_1 \ldots w_m) = i$ with $m > n$ then $N(w_0\sigma_0 w_1\sigma_1 \ldots w_n\sigma_n) = j$, $N(w_0\sigma_0 w_1\sigma_1 \ldots w_m\sigma_m)$

$= i$ and $N(w_0\sigma_0 w_1\sigma_1 \ldots w_n\sigma_n\eta) = j$ for all $\eta \in L_j^*$. Thus $w_0\sigma_0 w_1\sigma_1 \ldots w_m\sigma_m \notin L_j^*$ and since $\sigma_k \in \{w_0, w_1, \ldots, w_m\}^*$ for $k \leq m$ it follows that some $w_k \notin L_j$ for $k \leq m$. So the mind change from j to i was induced by a counterexample and M^K is conservative (using the definition that outputting the symbol "?" does not count as a mind change). ∎

The proof even relativizes to ExTxt$[A]$ = ConsvTxt$[A']$ which shows that the inference-degrees with respect to learning uniform recursive families is quite different to the degree-structure with respect to learning arbitrary families of enumerable sets: In the latter case the low enumerable oracles all belong to different inference-degrees. Furthermore if $L_i \not\subset L_j$ for all i, j then the family S can be learned conservatively: On input σ the learner just guesses the first index i with $range(\sigma) \subseteq L_i$. So in the context of learning uniformly recursive sets the following holds for all oracles A and B:

Corollary 4.6.8 FinTxt $\subset$ FinTxt$[K]$ = FinTxt$[A \oplus K]$ $\subset$ NoisyTxt $\subset$ ConsvTxt $\subset$ ConsvTxt$[K]$ = ExTxt $\subset$ ExTxt$[K]$ = ExTxt$[B \oplus K]$.

4.7 Behaviourally Correct and Partial Identification

Behavioural correct identification means that the learner outputs an infinite sequence of hypotheses which almost all compute the correct function or generate the correct set. It turns out that learning functions from noisy informant, there is no difference between behaviourally correct and explanatory convergence:

Theorem 4.7.1 *The following three statements are equivalent for any class $S \subseteq$ REC:*
(a) *S can be learned finitely from informant using K-oracle.*
(b) *S can be learned in the limit from noisy informant.*
(c) *S can be learned behaviourally correct from noisy informant.*

Proof Since convergence in the limit always implies behaviourally correct convergence, obviously (b $\Rightarrow$ c) holds. (a $\Rightarrow$ b) is shown in Theorem 4.2.1, part FinInf$[K]$ $\subseteq$ NoisyInf. The remaining implication (c $\Rightarrow$ a) is an adapted version of Theorem 4.2.1, part NoisyInf $\subseteq$ FinInf$[K]$:

Assume that M is recursive and learns the class S of functions behaviourally correct from noisy informant. A string σ is called α-consistent iff all pairs (x, y) occurring in σ satisfy either $x \notin dom(\alpha)$ or $y = \alpha(x)$. Now the following Fin$[K]$-learner N infers S:

> On input α, N checks using the K-oracle whether there is a string σ of length up to $|\alpha|$ such that, for all α-consistent strings τ and τ', the relation
>
> $$(\forall x)\,[\varphi_{M(\sigma\tau)}(x)\!\downarrow \wedge\, \varphi_{M(\sigma\tau')}(x)\!\downarrow \Rightarrow \varphi_{M(\sigma\tau)}(x) = \varphi_{M(\sigma\tau')}(x)]$$
>
> holds. If yes, then no two guesses $M(\sigma\tau)$ and $M(\sigma\tau')$ contradict each other and $N(\alpha)$ converges to an index e of the amalgamation of all functions $\varphi_{M(\sigma\tau)}$ with τ ranging over all α-consistent strings. If not, then $N(\alpha) = ?$.

Let $f \in S$ and M behaviourally correct infer f. Then there is some $\alpha \preceq f$ and some string σ such that $M(\sigma\tau)$ is an index for f for all α-consistent strings τ — otherwise it could be shown as in Theorem 4.2.1 that there is a noisy informant from which M does not learn f behaviourally correct. Without loss of generality assume that $|\sigma| \leq |\alpha|$. Then $N(\alpha)$ outputs some index e of the amalgamation of the functions $\varphi_{M(\sigma\tau)}$; it is easy to see that

$\varphi_e = f$.

So it remains to show that N does not output an other false index before finding e, that is, that already the first index output by N is correct. So let $\alpha \preceq f$ satisfy $N(\alpha) = e \neq ?$. Take the σ from the definition of $N(\alpha)$. Let T enumerate all pairs $(x, f(x))$ infinitely often without any noise. Now σT is obviously a noisy informant for f and there is a $\tau \preceq T$ such that $f = \varphi_{M(\sigma\tau)}$. By choice, τ is α-consistent. So $\varphi_e(x){\downarrow} = \varphi_{M(\sigma\tau)}(x){\downarrow}$ for all x and $\varphi_e = f$. It follows that inferring any function $f \in S$ the first guess of N is already correct and without loss of generality N makes no mind changes. ∎

While for learning functions from noisy informant the concepts of behaviourally correct learning and learning in the limit coincide, this is not longer true for learning languages. Case, Jain and Stephan [28] showed this for learning languages from noisy informant and the theorems below show it for learning languages from text: While FinTxt is not included in NoisyTxt, FinTxt is included in the criterion of behaviourally correct learning from noisy text.

Theorem 4.7.2 *If S can be learned finitely from text then S can also be learned behaviourally correct from noisy text.*

Proof Let M infer finitely a class S of languages from text, in particular M guesses the symbol "?" until it outputs a guess e and then keeps this output e for ever. Now consider N given by

$$N(w_0 w_1 \ldots w_n) = \begin{array}{l} M(w_m w_{m+1} \ldots w_n) \text{ for the maximal } m \leq n \\ \text{with } M(w_m w_{m+1} \ldots w_n) \neq ? \end{array}$$

and let $w_0 w_1 \ldots$ be a noisy text for L. Since there is a maximal k with $w_k \notin L$, each sequence $w_m w_{m+1} \ldots$ with $m > k$ is a text for L. In particular for all $n \geq m$, $M(w_m w_{m+1} \ldots w_n)$ is either the symbol "?" or an index for L. Since N outputs $M(w_m w_{m+1} \ldots w_n)$ for the maximal m such that $M(w_m w_{m+1} \ldots w_n) \neq ?$, these m satisfy $m > k$ for almost all input $w_0 w_1 \ldots w_n$; then $w_m, w_{m+1}, \ldots, w_n \in L$ and since M finitely learns L, $M(w_m w_{m+1} \ldots w_n)$ is always an index for L.

The properness of the inclusion follows from NoisyTxt $\not\subseteq$ FinTxt (Corollary 4.3.4) and the obvious fact that any S which can be learned in the limit from noisy text can also be learned behaviourally correct from noisy text. ∎

Osherson, Stob and Weinstein [113, Exercise 7.5A] introduced the notion of partial identification from text and showed that the class of all enumerable languages can be learned from text under this criterion. This identification criterion is the mirror image of noisy input since the learner outputs the correct guess infinitely often and each other guess only finitely often.

Definition 4.7.3 [113] A machine M partially identifies S from noisy text iff for every $L \in S$ and every noisy text T for L there is a unique index e such that M outputs e infinitely often on input T and this e is an index for L: $W_e = L$. Partial identification from very noisy text and noisy informant is defined analogously. The concept also transfers easily to learning functions from noisy or very noisy informant.

Let REC denote the class of all total recursive functions and RE that of all enumerable sets. The result of Osherson, Stob and Weinstein generalizes for learning from very noisy informant and from noisy text:

Theorem 4.7.4 REC *is partially identifiable from very noisy informant.*
RE *is partially identifiable from noisy informant.*
RE *is partially identifiable from noisy text.*

Proof REC is partially identifiable from very noisy informant:

Let $\{\varphi_{h(e)}\}_{e \in \mathbb{N}}$ be a Friedberg numbering of all partial recursive functions, h is total recursive. Further let T be a very noisy informant for f. M may be specified only by stating how often M outputs an index $h(e)$ on text T since it does not matter when these outputs occur and identification only depends on how often M outputs an index.

> M outputs $h(e)$ at least n times iff for $x = 0, 1, \ldots, n$ the following two conditions are satisfied:
> - $\varphi_{h(e)}(x)\downarrow$,
> - $(x, \varphi_{h(e)}(x))$ occurs at least n times in T.

So M reads longer and longer initial segments and whenever M notices that it has put out less than n times $h(e)$ while the conditions above demand to output $h(e)$ at least n times, M's next output is $h(e)$.

There is an unique index e with $f = \varphi_{h(e)}$. For each x, the pairs $(x, f(x))$ occur infinitely often in T and furthermore, $\varphi_{h(e)}(x)\downarrow = f(x)$ for all x. Thus the conditions are satisfied for each n and M outputs $h(e)$ infinitely often.

Now consider any $e' \neq e$. There is some x such that either $\varphi_{h(e')}(x)\uparrow$ or $\varphi_{h(e')}(x) \neq f(x)$. In the latter case, $(x, \varphi_{h(e')}(x))$ occurs only finitely often, say m times in T. Thus for all $n > x$ — with additionally $n > m$ in the second case — M outputs the index $h(e')$ less than n times, in particularly only finitely often. Therefore M partially identifies REC from very noisy informant.

RE is partially identifiable from noisy informant:

Note that for characteristic functions, the notions noisy informant and very noisy informant are the same. So the statement is equivalent to saying that RE can be partially identified from very noisy informant. Now let $W_{h(0)}, W_{h(1)}, \ldots$ be a Friedberg numbering of all enumerable sets and let T be a noisy informant for some enumerable set L. This inference process is similar to the previous one.

> M outputs $h(e)$ at least n times iff there is some $s \geq n$ such that the pairs $(x, W_{h(e),s}(x))$ occur at least n times in T for $x = 0, 1, \ldots, n$.

Let e be the index of L, that is, $L = W_{h(e)}$. For each n there is $s \geq n$ such that $W_{h(e)}(x) = W_{h(e),s}(x)$ for all $x \leq n$. Thus $(x, W_{h(e),s}(x))$ occurs in T infinitely many times for these x and M outputs $h(e)$ at least n times, therefore even infinitely often.

Let $e' \neq e$. There is some x with $W_{h(e)}(x) \neq W_{h(e')}(x)$. There is some m such that $(x, W_{h(e')}(x))$ does not occur in T more than m times and $W_{h(e'),s}(x) = W_{h(e')}(x)$ for all $s \geq m$. Then M does not output $h(e')$ for any $n > x + m$. Thus M partially identifies L from T.

RE is partially identifiable from noisy text:

To proof this, one needs a padded version of the Friedberg numbering. So let $W_{g(e,k)} = W_{h(e)}$ for an injective recursive function g and the Friedberg numbering h of all enumerable sets from the second part. Let $T = w_0 w_1 w_2 \ldots$ be a noisy text for the enumerable language L.

M outputs $g(e,k)$ at least n times iff the following three enumerable conditions are satisfied:
- $w_k, w_{k+1}, \ldots, w_{k+n} \in W_{h(e)}$;
- $k = 0$ or $w_{k-1} \notin W_{h(e),n}$;
- Each $x \in W_{h(e),n}$ occurs at least n times in T.

There are e and k with $W_{g(e,k)} = L$ and $k = \min\{l : (\forall m \geq l)\,[w_m \in L]\}$. The number k exists since T is a noisy text for L and so almost all w_m are in L.

By the choice of k, $w_k, w_{k+1}, \ldots, w_{k+n} \in W_{h(e)}$ holds for all n. Either $k = 0$ or $w_{k-1} \notin W_{h(e)}$ (and therefore $w_{k-1} \notin W_{h(e),n}$). Each $x \in W_{h(e)}$ occurs infinitely often in T. So all three conditions are satisfied for each n and M outputs $g(e,k)$ infinitely often.

It remains to show that whenever M outputs $g(e',k')$ infinitely often on this text T then $e = e'$ and $k = k'$. Each $x \in W_{g(e',k')}$ occurs infinitely often in T since each such x is enumerated into $W_{g(e',k')}$ at some stage s and for all $n > x + s$, if M outputs $g(e',k)$ at least n times then x occurs in T at least n times. Thus $x \in L$. If $x \notin W_{g(e',k')}$ then x must not occur in T beyond the k'-th position, in particular only finitely often. Therefore $x \notin L$. So $W_{g(e',k')} = L$ and $e' = e$. By the second condition in the algorithm, $g(e',k')$ occurs only then infinitely often if either $k' = 0$ or $w_{k'-1} \notin W_{h(e)}$. It follows that $k' \leq k$. On the other hand, $w_{k'}, w_{k'+1}, \ldots \in W_{h(e)}$, so $k' \geq k$. Thus M outputs $g(e',k')$ infinitely often iff $g(e',k') = g(e,k)$ and M identifies RE partially from noisy text. ∎

While RE is partially identifiable from noisy text, RE is not partially identifiable from very noisy text as the following example shows:

Example 4.7.5 *Let S contain all sets $\{x, x+1, x+2, x+3, \ldots\}$.*
S is partially identifiable from very noisy text.
$S \cup \{\emptyset\}$ is not partially identifiable from very noisy text.

Proof Since each set in S is cofinite, every very noisy text for some $L \in S$ is already a noisy text: each number not in L occurs only finitely often and since there are only finitely many numbers outside L, only finitely many data-elements of a very noisy text for L are not in L: Thus the text is already noisy. Since every class of languages can be partially identified from noisy text, S can be identified from very noisy text.

Assume by way of contradiction that M partially identifies $S \cup \{\emptyset\}$. Using a list $e_0, e_1, e_2, \ldots$ of all indices of the empty set, a very noisy text $T = \sigma_0\sigma_1\sigma_2\ldots$ is constructed inductively on which M fails.

For each n select a string $\sigma_n \in \{n, n+1, n+2, \ldots\}^*$ such that

$$(\forall \tau \in \{n, n+1, n+2, \ldots\}^*)\,[M(\eta_n\sigma_n\tau) \neq e_n],$$

where $\eta_0 = \lambda$ and $\eta_n = \sigma_0\sigma_1\ldots\sigma_{n-1}$ for $n > 0$.

This construction works, because if σ_n would not exist there would be a noisy text $T_n \in \eta_n\{n, n+1, n+2, \ldots\}^\infty$ for $\{n, n+1, n+2, \ldots\}$ on which M infinitely often outputs e_n and then M would not partially identify $\{n, n+1, n+2, \ldots\}$ since e_n is an index of $\emptyset$.

So by construction, $M(\tau) \neq e_n$ whenever $\eta_n\sigma_n \preceq \tau \preceq T$, thus M outputs e_n on input T only finitely often. Further each number n occurs only in the strings σ_m for $m \leq n$, thus each number n occurs only finitely often in T. So T is a very noisy text for $\emptyset$ but M does not partially identify $\emptyset$ from T. ∎

Since S is learnable in the limit from text by guessing $\emptyset$ if $range(\sigma) = \emptyset$ and guessing the set $\{n, n+1, n+2, \ldots\}$ if $range(\sigma)$ is not empty and has minimum n, S is a witness for the fact, that ExTxt does not imply partially identifiability from very noisy text. On the other hand the class of all graphs of recursive functions is partial identifiable from very noisy text without being learnable in the limit from text or informant.

Corollary 4.7.6 *Learning in the limit from text and partially identification from very noisy text are incomparable concepts.*

4.8 Alternative Models and Further Work

The notions are robust in the sense, that similar notions can be translated into one of them. For example, the following variants to present the input-data for the learner are equivalent to noisy informant:

- for each x, finitely many pairs (x, y) occur in the input such that the last one has the form $(x, f(x))$;
- for each x, finitely many pairs (x, y) occur in the input such that the majority has the form $(x, f(x))$;
- for each x, the proportion of the number of pairs $(x, f(x))$ within the first n data-elements with respect to the number of all pairs (x, y) within the first n data-elements converges to 1 for $n \to \infty$.

The translation procedures between these three notions of noisy informant and that one from Definition 4.1.1 are recursive and defined on finite strings of data-elements. For example, using a sequence $x_0, x_1, x_2, \ldots$ in which every number of $\mathbb{N}$ occurs infinitely often, one can define a translation of a noisy informant according to the first condition to be an infinite sequence $(x_0, y_0)\,(x_1, y_1)\,\ldots$ such that $y_n = y$ for the last pair of the form (x_n, y) appearing in T within the first n pairs and $y_n = 0$ if there is no such pair within the first n elements of T of this form. The disadvantage of these alternatives is that they do not have corresponding notions for noisy text. Furthermore, the nice property that a data-item is correct iff it appears infinitely often in the data-stream holds for all notions of noisyness considered here, but would fail for these three alternative definitions of noisy informant.

Case, Jain and Stephan [28] investigated two learning criteria between explanatory learning and behaviourally correct learning from noisy text: Bārzdins and Podnieks [13] introduced FEx-convergence where the learner converges semantically as in the case BC with the additional constraint that the learner outputs in total only finitely many programs. Case and Smith [31] introduced the notion of FEx-convergence where the learner outputs finitely many programs such that one of them is correct. It is no longer required to detect the some subset of correct programs in the limit. Case, Jain and Stephan [28] obtained the following results for these criteria as well as some further, parameterized results (where the parameters count the number of programs, mind changes, errors and so on).

Fact 4.8.1 [28] *The following inclusions follow directly from the definition for learning from noisy text or informant:* Ex*-convergence implies* FEx*-convergence;* FEx*-convergence implies* OEx*-convergence and* BC*-convergence.*

For learning from noisy text, there are no further inclusions between these parameters.

Theorem 4.8.2 [28] *The class containing K and all sets $\{x\}$ for $x \notin K$ is* BC*-learnable from noisy text but not* OEx*-learnable from noisy text. The class containing $\emptyset$ and $\mathbb{N}$ is* OEx*-learnable from noisy text but not* BC*-learnable from noisy text.*

The situation for learning from informant is between the previous one and the one for learning from noise-free informant where the learning criteria FEx and OEx coincide with Ex [13, 31].

Theorem 4.8.3 [28] *For learning from noisy informant, the notions of* FEx*-convergence and* OEx*-convergence are equivalent and properly situated between those of* Ex*-convergence and* BC*-convergence.*

Case, Jain and Sharma [27] extended this work by studying the following question: given a program for a uniform enumeration or decision procedure, to which extent is it possible to synthesize learners which are able to cope with noisy data. Furthermore, Case and Jain [24] addressed the same question for the case of learning from recursive noisy texts only. It is easier to synthesize BC-learners which have only to deal with recursive texts than to synthesize those which have to deal with every text. But for the synthesis of Ex-learners, the restriction to recursive texts does not give any advantage.

5 Universal Language Learners

The basic concept considered in this section is learning from (noise-free) text as defined in Definition 4.1.2. There are no omniscient learners for the model of learning from text: The class $\{\mathbb{N}\} \cup \{D \subseteq \mathbb{N} : D \text{ is finite}\}$ cannot be learned from text relative to any oracle [56]. Let Λ denote the collection of all classes S of languages which are learnable by some nonrecursive machine. Jain and Sharma [64] showed that no Turing degree suffices to learn all classes in Λ: for any oracle A, there exists a class $S \in \Lambda$ which is not learnable relative to A. An alternative proof for this fact is given by Osherson, Stob and Weinstein [113, Proposition 4.1A] who showed that no denumerable set of learners can learn every class from Λ — this implies the just mentioned fact directly since there are only countably many learners computable relative to a given Turing degree.

It is a natural question to ask what resources are needed to learn all the learnable in a uniform way. Although no fixed resource allows to learn all classes in Λ, one can inquire whether there exists a uniform learning procedure that is given as a parameter extra information about the class of which the current target language is an element. The question is asked whether there is a learner M which succeeds for every class $S \in \Lambda$ when M receives as additional information an oracle B which describes S in some specified way.

It is shown that such a learner exists if B contains an index for all languages in S but for no language outside of S. The Turing degrees of such learners are exactly the degrees above $\mathbf{0}''$, that is, every universal learner must be able to solve the inclusion problem for recursively enumerable sets. While in this general case the learner is inherently nonrecursive, it is shown that for the more restricted case where $B = \{e : W_e \in S\}$ there exists already a computable universal learner. After presenting these results in Section 5.1, they are adapted to the world of learning recursive languages in Section 5.2. Section 5.3 deals with the case where an upper bound on the size of some grammar for each language L is provided to the learner instead of information on the whole class S. In this setting, which was introduced by Freivalds and Wiehagen [46] and explored by Jain and Sharma [63], there is a single learner for the whole class of all enumerable languages. Such a learner exists in a Turing degree **a** iff **a** is high. In Section 5.4 it is investigated to which extent it is possible to transfer the results of the previous sections to the concept of finite learning. Even if B is an index set of S, it might in some cases be necessary to work with B' instead of B since a finite learner cannot investigate the whole set B in finite time.

Osherson, Stob and Weinstein [115] proved a result similar to those in Sections 5.1 and 5.2 in a model-theoretic context. They constructed a universal inductive inference machine that learns in the limit from data about a model of a set of sentences T whether a given sentence θ holds in this model, provided that T is given as an oracle and that both θ and $\neg\theta$ are equivalent under T to an existential-universal sentence. Further related work considers the case where uniformly enumerable classes are given by a single index e where the class to be learned is of the form $S = \{W_{e'} : e' \in W_e\}$. Osherson, Stob and Weinstein [114] introduced this concept and Baliga, Case and Jain [9] extended the study. One fundamental result is that on the one hand there is a computable learner which identifies every finite class S provided that S is given via a set W_e which contains for every $L \in S$ exactly one index but that on the other hand this fails if W_e may contain up to 2 indices per set in S. Kapur and Bilardi [76] considered the case where the family to be learned is uniformly recursive. They showed that it is impossible to learn these families universally if the only information supplied is an index of a uniformly recursive enumeration of the family. Nevertheless they give some natural subcollections of Λ which have universal learners using an index of the families to be learned as the only additional information.

5.1 Universal Learning From Text With Index Sets

This section contains the two main results: in the general scenario the Turing degrees of the universal learners are just the cone above $\mathbf{0}''$, that is, the Turing degree $\mathbf{0}''$ is necessary and sufficient. Furthermore, in the special case where the oracle B contains exactly those indices e where $W_e \in S$ there is already a computable universal learner, that is a universal learner of Turing degree $\mathbf{0}$.

Definition 5.1.1 A set B is *for* S if $S = \{W_e : e \in B\}$. Note that this set B is not fully determined by S, that is, B can be a proper subset of the index set $\{e : W_e \in S\}$ of S. A *universal* (*text*) *learner* is a (not necessarily computable) machine M which learns every class $S \in \Lambda$ using a set B for S as additional information.

M is not requested to be computable but M still has a Turing degree. The main result of this paper is that such a machine can have any Turing degree above or equal to $\mathbf{0}''$ but no other one. Note that the condition $S \in \Lambda$ is still necessary: whenever M learns a class S then S is learnable relative to every oracle encoding M and B; thus S is in Λ.

Theorem 5.1.2 *There is a universal learner M of Turing degree $\mathbf{0}''$.*

Proof The learner M works as follows:

On input σ, M looks for the first $i \leq |\sigma|$ such that

$$range(\sigma) \subseteq W_i; \tag{20}$$

$$range(\sigma) \subseteq W_j \subset W_i \text{ for no } j \in B \text{ with } j \leq |\sigma|. \tag{21}$$

If such i are found then M outputs the smallest one of them. Otherwise M abstains from guessing by outputting "?".

For the verification, let i be the minimal index within B of a language $L \in S$. Angluin [2, Theorem 1] and, for the noneffective case, Osherson, Stob and Weinstein [113, Proposition 2.4A] showed that for the given W_i there is a finite set F such that F is a subset of W_i and no language in S is between F and W_i:

$$F \subseteq W_i \wedge (\forall W_j \in S)\,[F \not\subseteq W_j \vee W_j \not\subset W_i] \tag{22}$$

and so no $j \in B$ satisfies $F \subseteq W_j \subset W_i$. Any text T for L has a sufficiently long prefix σ such that $F \subseteq range(\sigma)$ and $i \leq |\sigma|$. It is easy to see that the algorithm for this and any longer prefix outputs i.

So it remains to show that the algorithm can be executed by a machine having Turing degree $\mathbf{0}''$. The only two uncomputable operations required in the algorithm, besides the access to the oracle B, are to compare whether one language is a subset of another and to check whether the range of a finite string is contained in some language. Both operations can be performed by a machine whose Turing degree is $\mathbf{0}''$. ∎

The next theorem shows that it is necessary to have the ability to check subset-conditions. It shows that every universal learner has the computational power to decide whether $W_i \subseteq W_j$ or not.

Theorem 5.1.3 *The Turing degree of every universal learner is at least* $\mathbf{0}''$.

Proof Let M be a universal learner and define $S = \{L : L = K \vee (L \not\subseteq K \wedge L \text{ is finite})\}$, that is, S contains K and all finite sets which are not subsets of K. This class is in Λ since it is learnable with oracle K: the learner guesses K if $range(\sigma) \subseteq K$ and guesses $range(\sigma)$ otherwise.

The expression $use(M, B, \sigma)$ denotes all oracle queries made by the learner M^B to compute $M^B(\tau)$ for some $\tau \preceq \sigma$. This use is always a finite set. Now the following statement holds by adapting the construction of a locking sequence to the general case of universal learners:

> There exists a finite set C of indices of languages in S including some index of K and there is a string $\sigma \in K^*$ such that $M^B(\sigma\tau) = M^C(\sigma)$ for all $\tau \in K^*$ whenever B contains only indices of languages in S and $B(x) = C(x)$ for all $x \in use(M, B, \sigma)$. (23)

If (23) fails, then one can construct inductively a sequence of finite sets C_i and strings $\sigma_i \in K^*$ such that $M^{C_{i+1}}(\sigma_{i+1}) \neq M^{C_i}(\sigma_i)$, each C_i contains the minimal index of K and perhaps finitely many other indices of languages in S, $C_{i+1}(x) = C_i(x)$ for all $x \in use(M, C_i, \sigma_i)$ and $\sigma_{i+1} \succeq \sigma_i a_i$ where $a_0 a_1 \ldots$ is an enumeration of K. It follows that $T = \lim_i \sigma_i$ is a text for K and the set $B = \bigcup_i C_i$ contains only indices of languages in S. Furthermore, $M^B(\sigma_i) = M^{C_i}(\sigma_i)$ for all i and so M^B diverges on the text T. But since B is a set for a class in Λ containing K, M^B has to learn K from the text T and from this contradiction it follows that (23) holds.

Now the condition (23) is used in order to decide the inclusion problem. Let C and σ be as in (23). Let g be a number in $K - range(\sigma)$. There is a one-one recursive function g such that $W_e(x) = K(g(e, x))$ and g has a recursive range disjoint from $\{d\} \cup range(\sigma)$. Now one defines

$$W_{h(e,e')} = \{g(e', x) : x \in W_e\} \cup (K - \{d\} - \{g(e', x) : x \in \mathbb{N}\})$$

where the sets $\{g(e', x) : x \in \mathbb{N}\}$ are uniformly recursive for $e' = 0, 1, 2, \ldots$ since the range of g is recursive and g is one-one. So the equivalences

$$W_e \subseteq W_{e'} \;\Leftrightarrow\; \{g(e', x) : x \in W_e\} \subseteq \{g(e', x) : x \in W_{e'}\}$$

$$\begin{aligned} &\Leftrightarrow \; \{g(e',x) : x \in W_e\} \subseteq K \\ &\Leftrightarrow \; W_{h(e,e')} \subseteq K \\ &\Leftrightarrow \; W_{h(e,e')} \subset K \end{aligned}$$

hold where the last equivalence is due to the fact that $d \notin W_{h(e,e')}$ and therefore $K \neq W_{h(e,e')}$ for all e, e'. Furthermore, one can slow down the enumeration of $W_{h(e,e')}$ by enumerating a number into the new set $W_{f(e,e')}$ iff all numbers previously enumerated into $W_{h(e,e')}$ have also appeared in K. So $W_{f(e,e')} = W_{h(e,e')}$ if $W_{h(e,e')} \subseteq K$ and $W_{f(e,e')}$ is a finite set not contained in K otherwise. Furthermore, one can use padding to define f such that the range of f is disjoint from $use(M, \sigma, C)$.

Let $B = C \cup \{f(e,e')\}$. Since B is finite, the class of all languages with indices in B is learnable and M^B has to identify $W_{f(e,e')}$. Furthermore, if $W_{f(e,e')}$ is a proper subset of K, then $M^B(\sigma\tau) \neq M^C(\sigma)$ for some $\tau \in K^*$. Otherwise $W_{f(e,e')}$ is in S and $M^B(\sigma\tau) = M^C(\sigma)$ for all $\tau \in K^*$. So one obtains

$$W_e \subseteq W_{e'} \;\Leftrightarrow\; (\exists \tau \in K^*)\,[M^B(\sigma\tau) \neq M^C(\sigma)]$$

where the characteristic function of B is uniformly recursive with parameters e and e'. The inclusion problem $\{(e,e') : W_e \subseteq W_{e'}\}$ is enumerable relative to the learner M as an oracle.

Since $\overline{K}$ is many-one-reducible to the inclusion problem, $\overline{K}$ is enumerable relative to M. Hence K is computable relative to M. Since the complement $\{(e,e') : W_e \not\subseteq W_{e'}\}$ of the inclusion problem is enumerable relative to K, it is also enumerable relative to M. So the inclusion problem is computable relative to M. Hence, the Turing degree of M is $\mathbf{0}''$ or above. ∎

If B contains all indices of the languages to be learned and not only some then there exist computable universal learners. This is chiefly due to the fact that index sets have a high complexity in terms of Turing degree. Rice [121] showed that every nontrivial index set (and those of learnable classes are always nontrivial) has at least the Turing degree $\mathbf{0}'$. The construction exploits that whenever a learnable class contains an infinite language then its index set B has even degree $\mathbf{0}''$, in particular, one can find in the limit an algorithm which computes relative to B the inclusion problem $\{(e,e') : W_e \subseteq W_{e'}\}$.

Theorem 5.1.4 *There is a computable universal learner M such that M^B learns every class $S \in \Lambda$ provided that B is the index set of S.*

Proof A canonical encoding D_i of a finite set allows to compute the cardinality and a list of all elements of D_i from the index i. Since canonical indices can be translated effectively into enumerable indices, not only the question whether $W_i \in S$ but also whether $D_i \in S$ can be answered effectively by inspecting the index set B of S.

A pair (D_i, W_j) (or better said, its indices) are a potential candidate to compute the inclusion problem if

$$D_i \subseteq W_j,\; D_i \notin S \text{ and } W_j \in S; \tag{24}$$

$$D_k \notin S \text{ for all } k \text{ with } D_i \subseteq D_k \subseteq W_j. \tag{25}$$

The set A of (the indices of) these candidates is the difference of two sets which are recursively enumerable relative to B: the first one enumerates the pairs (D_i, W_j) which satisfy the condition (24) and the second one all pairs which fail to satisfy (25). So A has a B-recursive approximation: $A = \lim_s A_s$.

Recall that whenever S contains an infinite language W_j then this W_j has a finite subset

$D_i \notin S$ such that no set in S, in particular no finite set $D_k \in S$, is between D_i and W_j [2, 113]. So whenever S contains an infinite language then there is also a pair (D_i, W_j) satisfying (24) and (25). Furthermore, whenever a pair (D_i, W_j) belongs to A, then W_j is infinite: If W_j is finite then it has a canonical index k. By $D_i \subseteq D_k = W_j \wedge D_k \in S$ it follows that no subset D_i satisfies (25) together with W_j. The inclusion problem whether $W_e \subseteq W_{e'}$ can be computed relative to B under the assumption that a given pair (D_i, W_j) is in A.

From the index e one can compute an enumeration $a_0, a_1, \ldots$ of W_e and define

$$W_{f(e,e')} = D_i \cup \{x \in W_j : (\forall y \leq x)\,[a_y \in W_{e'}]\}.$$

Now $W_e \subseteq W_{e'} \Leftrightarrow W_{f(e,e')} = W_j \Leftrightarrow W_{f(e,e')} \in S \Leftrightarrow f(e,e') \in B$.

For the verification note that $W_{f(e,e')} = W_j$ in the case that $W_e \subseteq W_{e'}$, that is, in the case that all a_y are enumerated into W_e'. In order to deal with $\emptyset$ the enumeration $a_0, a_1, \ldots$ might also contain repeatedly the symbol # for void information which a priori is in every set. Otherwise some a_x is not in $W_{e'}$ and $W_{f(e,e')}$ is a finite set: it contains just the elements of D_i and those of W_j below x. So $W_{f(e,e')}$ is finite and between D_i and W_j, therefore $W_{f(e,e')}$ is not contained in S.

All pairs (D_i, W_j) can be put into an ordering equivalent to that of the natural numbers, so that it is possible to speak of a first pair, second pair and so on. Recall that it is computable relative to B in the limit as to which of these pairs belong to A — $A_0, A_1, \ldots$ denotes this B-recursive approximation of A. Having this approximation, it is possible to describe a computable universal learner which learns every class S from text using the index set B for S provided that $S \in \Lambda$.

On input σ check whether there is a pair in $A_{|\sigma|}$ among the first $|range(\sigma)|$ pairs.

If so, take the least such pair $(D_i, W_j) \in A_{|\sigma|}$ and emulate the algorithm from Theorem 5.1.2 using this pair to answer the inclusion queries at positions (20) and (21).

If not, just output the canonical index for $range(\sigma)$.

For the verification of the algorithm it is necessary to consider two cases where L denotes the actual language $L \in S$ whose text is fed into the learner. Recall that $|L|$ is the cardinality of L and note that $n \leq |L|$ for all n if L is infinite.

First, the case that, for all $n \leq |L|$, the n-th pair does not belong to A. Then L has to be finite since otherwise such a pair must exist and must have an index below the cardinality ∞ of L. So for sufficiently long prefixes σ of a given text, $range(\sigma) = L$ and none of the first $|L|$ pairs is in the current approximation $A_{|\sigma|}$. So for all these sufficiently long prefixes of the text, M outputs the canonical index of L and so converges to a correct index.

Second, there is a least n-th pair $(D_i, W_j) \in A$ and $n \leq |L|$. Then for all sufficiently long prefixes σ of a given text of L, $|range(\sigma)| > n$, (D_i, W_j) belongs to $A_{|\sigma|}$ but none of the pairs before (D_i, W_j). So M goes into the first case and uses the pair (D_i, W_j) for deciding the subset queries of type (20) and (21). Therefore the algorithm produces for almost all prefixes σ of the given text the same output as the algorithm in Theorem 5.1.2 and converges to an index of L. ∎

5.2 Learning From Recursive Indices

In the case of learning classes of recursive languages, one may consider the situation where one or more *programs* are given for each $L \in S$, rather than just one or more grammars generating it. An index e is a program for L if φ_e computes the characteristic function of L: φ_e is total and $L(x) = \varphi_e(x)$ for all x. In the following every total function φ_e is identified with the set $\{x : \varphi_e(x) > 0\}$. Furthermore, B is *a set of programs for* S iff B contains only total programs (which converge on every input) and $S = \{\varphi_e : e \in B\}$. A machine M BC-learns a class S from text iff on every text for some $L \in S$, M outputs an infinite sequence $e_0, e_1, \ldots$ of hypotheses such that almost all e_n are grammars for L.

The next result is quite parallel to the a result of Baliga, Case and Jain [9, Theorem 11], who showed that there is an algorithm which translates every index of a uniform decision procedure of a class $S \in \Lambda$ into a program of a BC-learner for S. The essential ideas of the proof of that result and the one below are the same.

Theorem 5.2.1 *There is a computable universal* BC*-learner M such that M^B learns a class $S \in \Lambda$ whenever S consists of recursive languages and B is a set of programs for S.*

Proof The BC-learner M is constructed as follows: M computes on input σ first the set I_0 of all indices $i \leq |\sigma|$ such that $i \in B$ and $range(\sigma) \subseteq \varphi_i$. The guess $f(I_0)$ then does two processes in parallel: First it throws out all indices from I_0 which are believed to be incorrect. Second it enumerates all elements which are in all remaining sets φ_i with $i \in I$ into $W_{f(I_0)}$. Formally the set of the "correct indices" is given as an intersection $I = I_0 \cap I_1 \cap I_2 \cap \ldots$ where the sets I_t are defined inductively by

$$i \in I_{t+1} \;\Leftrightarrow\; i \in I_t \wedge (\forall j < i)\,[j \notin I_t \vee \varphi_i(t) \leq \varphi_j(t)].$$

Now $W_{f(I_0)} = \bigcap_{i \in I} \varphi_i$ is enumerable by the following formula:

$$W_{f(I_0)} = \{x : (\exists t)\,(\forall i \in I_t)\,[x \in \varphi_i]\}.$$

For the verification note that, for every $i \in B$, there is a finite subset F of φ_i such that, for all $j \in B$ with $F \subseteq \varphi_j$, the following holds:

- if $j < i$ then $\varphi_j \supseteq \varphi_i$;
- if $j > i$ then $\varphi_j \not\subset \varphi_i$.

Therefore whenever $F \subseteq range(\sigma) \subseteq \varphi_i$ and $|\sigma| > i$ then $i \in I$ and every $j \in I$ is a program of a superset of φ_i. So $W_{f(I_0)} = \varphi_i$ for almost all prefixes σ of a text for L. ∎

If a learner having Turing degree $\mathbf{a} \geq \mathbf{0}'$ is chosen then even Ex-convergence is possible, that is, the learner makes on every text of a language in S only finitely many mind changes. In addition to that, such a learner can take characteristic indices. So given a BC-learner M, the new Ex-learner N is obtained by

$$N(\sigma) = \min\{e : (\forall x \leq |\sigma|)\,[\varphi_e(x)\!\downarrow = W_{M(\sigma)}(x)\,]\,\}.$$

On the other hand, a universal Ex-learner which succeeds on every family $S \in \Lambda$ of recursive languages given as a set of programs for S needs at least Turing degree $\mathbf{0}'$.

Theorem 5.2.2 *There is a universal* Ex*-learner M such that M^B learns a class $S \in \Lambda$ whenever S consists of recursive languages and B is a set of programs for S. The Turing degrees of such learners are just those above $\mathbf{0}'$.*

Proof As just mentioned, any universal computable BC-learner can be transferred into a $\mathbf{0}'$-recursive Ex-learner. So the converse direction is the interesting one. Its proof is obtained by adapting the one of Theorem 5.1.3.

The role of K is replaced by the role of the set E of even numbers, any other infinite and coinfinite recursive set could also be used. Now S just consists of the set E and every finite set containing an odd number, that is, a nonelement of E. One can again show that for every universal learner M there is a finite set C of characteristic indices for sets in S including one for E, a constant c and $\sigma \in E^*$ such that, for all sets B containing only characteristic indices for languages in S and agreeing with C below c and for all $\tau \in E^*$, the equation $M^B(\sigma\tau) = M^C(\sigma)$ holds. $M^C(\sigma)$ is then a characteristic index for E. Having this, one can enumerate $\overline{K}$ relative to C:

There is a recursive function f with range $\{c+1, c+2, \ldots\}$ which assigns to any x the characteristic index of the language $range(\sigma)$ in the case $x \notin K$ and $range(\sigma) \cup \{2s+1\}$ in the case that x is enumerated into K exactly at stage s. Then

$$x \notin K \;\Leftrightarrow\; (\exists \tau \in E^*)\,[M^{C \cup \{f(x)\}}(\sigma\tau) \neq M^C(\sigma)]$$

is an existentially quantified formula for $\overline{K}$ relative to the Turing degree of M. It follows that K is computable relative to M, that is, the Turing degree of M is at least $\mathbf{0}'$. ∎

The above result uses the fact that the learner M must also succeed with nonrecursive sets B for some languages. If one requires that B is recursive, one obtains a further restriction. These restricted classes are then exactly the uniformly recursive classes. The next result shows that the Turing degrees of universal learners for these classes are exactly the high ones.

Theorem 5.2.3 *There are universal learners who learn those classes $S \in \Lambda$ from every recursive set B of characteristic indices for the languages in S which have such a set B. The Turing degrees of these universal learners are just the high degrees.*

Proof In a high Turing degree $\mathbf{a}$, the learner can first identify an index for the set B in the limit. Whenever the learner makes a mind change concerning B, the learning algorithm for the language learner is restarted. So it is sufficient to formulate the algorithm for the case that a program for B is known to the universal learner M. The learner M uses a list of pairs (i, τ) such that all pairs of indices and strings occur exactly once in the list. The set

$$A = \{(i,\tau) : (\forall j \in B)\,(\forall x)\,(\exists y)\,[range(\tau) \subseteq \varphi_j \wedge x \in \varphi_i - \varphi_j \Rightarrow y \in \varphi_j - \varphi_i]\}$$

contains all (i, τ) such that $W_i \in S$ and $(range(\tau), W_i)$ satisfy Angluin's condition (22) on page 68 for the class given by B. The set A is in Π_2 and thus membership in A can be computed relative to $\mathbf{a}$ in the limit, let A_s be the corresponding $\mathbf{a}$-recursive approximation. Now the learning algorithm is the following:

$M(\sigma)$ searches the first pair (i, τ) such that

- $i \in B$, $i \leq |\sigma|$ and $\tau \preceq \sigma$,
- $\varphi_i(x) = 1$ for all $x \in range(\sigma)$,
- $(i, \tau) \in A_{|\sigma|}$.

If the pair (i, τ) is found then $M(\sigma) = i$ else $M(\sigma) = ?$.

For any text T for a language $L \in S$ there is a first pair $(i, \tau) \in A$ satisfying $\tau \preceq T$, $i \in B$ and $\varphi_i = L$. The learner converges either to this pair or to some previous one, thus also the output of the learner converges to some i'.

Assume now by way of contradiction, that i' is not an index for L. Then i' must be an index of a proper superset and there must be a τ' such that the leaner converges to (i', τ'). $range(\tau')$ is a subset of L and therefore $(i', \tau') \notin A$ since the relation $range(\tau') \subseteq \varphi_i \subset \varphi_{i'}$ holds. This contradiction gives the correctness of M.

For the converse direction let $\mathbf{a}$ be the Turing degree of some learner M for the class S containing all languages $\{2x, 2x+2, \ldots\}$, all finite sets with at least one odd element and all sets $\{2x, 2x+2, \ldots, 2x+2y\}$ whenever W_x has at least y elements but is finite. There is a recursive set B for S: Let B contain standard indices for the languages $\{2x, 2x+2, 2x+4, \ldots\}$ and the finite languages with odd elements. Furthermore, let $f(x, y, t)$ be an index of the set having the elements $2x, 2x+2, \ldots, 2x+2y$ plus the element $2s+1$ in the case that $s > t$ and s is the first stage where a new element not yet in $W_{x,t}$ is enumerated into W_x. Let B contain those $f(x, y, t)$ where $W_{x,t}$ has at least y elements.

Now one enumerates relative to $\mathbf{a}'$ the locking sequences σ_x for the sets $\{2x, 2x+2, \ldots\}$ and defines that $g(x) = \max(range(\sigma_x))$. Since each set $\{2x, 2x+2, \ldots\}$ has at least one locking sequence, $g(x)$ is defined for all x. One has that $g(x) > 2x + 2|W_x|$ whenever W_x is finite and it follows that W_x is finite iff W_x has at most $g(x)$ elements. The condition $|W_x| > g(x)$ can be checked relative to $\mathbf{a}'$ and one can compute relative to $\mathbf{a}'$ which sets W_x are finite. Therefore $\mathbf{a}$ is high. ∎

Kapur and Bilardi [76, Theorem 3] showed that there is no computable learner which is universal for enumerable families which are learnable from text by a computable learner. Indeed they showed that the Turing degree $\mathbf{a}$ of such a learner satisfies $\mathbf{a}'' \geq \mathbf{0}'''$, that is, $\mathbf{a}$ is high$_2$. The above proof uses a single family such that this family is not learnable without a high oracle. So the previous theorem does not imply the result of Kapur and Bilardi, but one can adapt the above proof by considering the parameterized classes

$$S_x = \{L \in S : \min(L) \in \{2x, 2x+1\}\}$$

which have uniformly recursive decision procedures whose index can be computed from x. These classes contain the set $\{2x, 2x+2, \ldots\}$, some finite sets with at least one odd element and perhaps finitely many subsets of $\{2x, 2x+2, \ldots\}$, so they are all learnable by a recursive learner. But a universal learner for all of them can be translated into a learner for S by waiting until some first data-item appears in the text and then emulating always the learning procedure for that S_x where x is the minimal number such that some number $y \leq 2x+1$ has occurred in the input text so far.

So any learner which is universal for those enumerable families that are learnable by a computable learner must have high Turing degree. This improves the result of Kapur and Bilardi quoted above.

5.3 Bounds on the Grammar Size

Freivalds and Wiehagen [46] introduced the model where the learner receives in addition to the data $f(0), f(1), \ldots$ of the function to be learned some upper bound b on the size of some program e of f, that is, some number b such that $b > e$ for at least one of the programs for f. They showed that in this case there exists a computable universal learner which is able to learn all computable functions. Jain and Sharma [63] transferred this model to the scenario of language learning from text and showed that the result does not hold in this setting. This kind of nonlearnability is not a principal one but is only caused by the limited computational abilities of a recursive machine. Using more complex machines it is possible to learn the class of all enumerable languages with one machine whose input is a text for a language L to be learned and an upper bound b on the size of some grammar for L. The Turing degrees of these machines are exactly the high degrees and so the result is very similar to those of Adleman and Blum [1] and Fortnow et al. [43] for many other learning criteria.

Theorem 5.3.1 *Let M be a learner which can infer every enumerable language L from a text for L and from an upper bound b on some grammar for L. Then M has a high Turing degree. Furthermore, there exists such a learner in every high Turing degree.*

Proof Let $\mathbf{a}$ be a high Turing degree. Now a learner M as specified in the theorem is constructed which is computable relative to $\mathbf{a}$. Consider the following function f:

$$f(i,j) = \begin{cases} x & \text{for the smallest number } x \text{ with } W_i(x) \neq W_j(x); \\ 0 & \text{if there is no such } x, \text{ that is, if } W_i = W_j. \end{cases}$$

This function is computable relative to $\mathbf{0}''$. The high degrees are those which can compute in the limit every $\mathbf{0}''$-recursive function. So it follows that for each pair i, j, $f(i,j)$ can be computed in the limit by some machine of Turing degree $\mathbf{a}$. In particular, for every b, the value $c(b) = \max\{f(i,j) : i < j \leq b\}$ can be approximated by a sequence $c_s(b)$ which is $\mathbf{a}$-recursive in both parameters s and b. The learner M uses the approximation $c_s(b)$:

$$M(\sigma, b) = i \text{ for the smallest } i \text{ with } W_{i,|\sigma|}(x) = range(\sigma)(x) \text{ for all } x \leq c_{|\sigma|}(b).$$

Note that $M(\sigma, b)$ is always defined since every finite set $range(\sigma)$ has a canonical index and one might define that, for canonical indices, every element appears already at stage 0.

Now, for the verification, let T be a text and s be so large that $c_s(b)$ already has converged to $c(b)$, that every element x of L which is smaller than $c(b)$ has already appeared in the text and that every $y \in W_j$ with $j \leq b$ and $y \leq c(b)$ has already been enumerated to W_j. Then $M(\sigma, b)$ outputs the least index i of L for every prefix $\sigma \preceq T$ of length at least s: W_i and $range(\sigma)$ coincide below $c_{|\sigma|}(b)$ since $W_i = L$ and the conditions above are satisfied. For $j < i$ the values of W_j and W_i differ below $c(b)$ and $W_{j,|\sigma|}$ and W_j coincide below $c(b)$. Thus $W_{j,|\sigma|}$ and $range(\sigma)$ disagree below $c_{|\sigma|}(b)$. The algorithm outputs i and therefore converges to the minimal index of L.

It remains to show that such a learner has high Turing degree. Let M learn every enumerable language from text with an upper bound b on a grammar of this language and let $\mathbf{a}$ be the Turing degree of M. Now it is shown that the problem whether a set equals $\mathbb{N}$ can be computed relative to $\mathbf{a}'$ and thus $\mathbf{a}$ is high.

Let a be an index for the language $\mathbb{N}$ and consider the behaviour of M with the additional information $e + a$, which is an upper bound for the indices of both languages, W_e and $\mathbb{N}$. So M must learn them both using this upper bound. The language $\mathbb{N}$ has a locking sequence σ in the sense that $M(e + a, \sigma\tau) = M(e + a, \sigma)$ for all strings τ. Such a σ can be found by a suitable algorithm of Turing degree $\mathbf{a}'$. If $W_e \neq \mathbb{N}$ then the difference must occur in $range(\sigma)$ since otherwise M would fail to learn W_e. So

$$W_e = \mathbb{N} \;\Leftrightarrow\; range(\sigma) \subseteq W_e.$$

This test whether $range(\sigma) \subseteq W_e$ is recursive in $\mathbf{0}'$ and in particular recursive in $\mathbf{a}'$. So it can be computed within Turing degree $\mathbf{a}'$ whether $W_e = \mathbb{N}$. This problem has the complexity $\mathbf{0}''$ and thus the Turing degree $\mathbf{a}$ is high. ▮

The algorithm to learn all languages from an upper bound needs nonrecursive information for exactly one part: the computation of $c(b)$. Taking now b as the minimal index of a language L, the Ex-learner can be made recursive by supplying an upper bound $c \geq c(b)$ instead of b itself. So one obtains an (unpublished) result of Jain, that an Ex-learner identifies all enumerable languages with a sufficiently large upper bound as additional information.

Corollary 5.3.2 *There is a computable learner M which infers every enumerable language L from text, given a bound c such that for the minimal index i of L and every $j < i$ there is some $x < c$ such that W_i and W_j differ on x.*

The difficulty for learning with upper bounds on the size of a grammar is due to the fact that it is impossible to know whether two languages are equal or not. To overcome this problem, Bārzdins and Podnieks [13] have introduced the slightly weaker criterion called FEx: Here the learner is not required to converge syntactically but is allowed to alternate between finitely many correct indices infinitely often. Jain and Sharma [63, Proposition 16] showed that there is a universal FEx-learner which succeeds on every enumerable language L provided that an upper bound b on some grammar for L is given to the learner.

The algorithm is quite easy: For every string σ the learner takes just that index e below the given bound b for which the value

$$x(e,\sigma) = \max\{y \leq |\sigma| : (\forall z \leq y)\,[range(\sigma)(z) = W_{e,|\sigma|}(z)]\}$$

is maximal. On a text for the language L, $x(e,\sigma)$ is bounded uniformly for all prefixes σ of the text if $W_e \neq L$ and converges to ∞ if $W_e = L$. Thus the learner outputs from some certain stage only correct indices.

So weakening from Ex to FEx brings down the complexities of universal learners from the high Turing degrees to computable.

5.4 Finite Learning With Additional Information

Smith proposed to study topics related to those in the previous sections also for finite learning. The classes which are finitely learnable, even relative to oracles, are more restricted than for the other learning criteria. Therefore, besides Λ, also the collections Λ_{fin} and Λ_{if} are considered, where Λ_{fin} contains all classes of languages learnable by some finite learner with access to an oracle and Λ_{if} all inclusion-free classes. A class S is inclusion-free iff any two distinct languages $L, H \in S$ satisfy $L \not\subseteq H$ and $H \not\subseteq L$. Osherson, Stob and Weinstein [113, Exercise 1.5.2C] characterized Λ_{fin}; Mukouchi [98, Theorem 7] did the same for uniformly recursive families. A further related result is Theorem 4.6.3 which characterizes the uniformly recursive families which are finitely learnable from text with help of the oracle K.

Fact 5.4.1 [113] *A class S is in Λ_{fin} iff every $L \in S$ has a finite subset $F \subseteq L$ such that $F \not\subseteq H$ for every $H \in S$ different from L.*

The proper inclusions $\Lambda_{\text{fin}} \subset \Lambda_{\text{if}} \subset \Lambda$ hold: Clearly every finitely learnable class is inclusion-free and the class S containing the set E of all even numbers and every set $\{0, 2, 4, \ldots, 2x, 2x+1\}$ witnesses the properness of the first inclusion since S is inclusion-free but E has no finite subset F such that E is the only superset of F within S. The algorithm "Learning by Enumeration" which outputs an index of the first L_i to satisfy

$range(\sigma) \subseteq L_i$ witnesses that every class $\{L_0, L_1, \ldots\} \in \Lambda_{\text{if}}$ is also in Λ. The class $\{\emptyset, \mathbb{N}\}$ witnesses the properness of this second inclusion $\Lambda_{\text{if}} \subset \Lambda$.

Behaviourally correct and explanatory learning can take into account the whole set B since they have the right to withdraw or update a hypothesis if some assumption on B turns out to be false. This is no longer true for finite learning, therefore a finite universal learner cannot succeed if it has access only to the index set. Some method to obtain infinite information on B is necessary.

Theorem 5.4.2 *There is no universal learner M (of arbitrarily high Turing degree) which learns every $S \in \Lambda_{\text{fin}}$ finitely from text with the index set $B = \{e : W_e \in S\}$ of S as the only additional information about S.*

Proof Let S contain the set $E = \{0, 2, 4, \ldots\}$ of all even numbers and perhaps also a finite set L which contains an odd number. The learner M^B has to output after reading some part $0\,2\,4\,\ldots\,2x$ of the canonical text for E a guess, this guess must compute E. Since M^B does not know whether there is any language in S besides E, M^B outputs this hypothesis after having queried only elements $B(b)$ with $b \leq a$ and each such b is in B iff b is an index for E.

Now there is another set $L = \{0, 2, 4, \ldots, 2y, 2y+1\}$ with $y \geq x$ which has no index below a — this can be obtained by taking a sufficiently large y. So L can be added to the class S without changing B at the queried places. The language L has a text which starts with $0\,2\,4\,\ldots\,2x$ and therefore M^B fails to identify it. So M is not a finite universal learner for all languages in Λ_{fin}. ▮

A direct consequence of the proof is that there is no finite learner — with an arbitrarily high oracle — which learns the class containing E and all sets of the form $\{0, 2, 4, \ldots, 2x, 2x+1\}$ from text. Thus the inclusion $\Lambda_{\text{fin}} \subset \Lambda_{\text{if}}$ is proper.

It is possible to find an uniform learner for S if more information than an index set is supplied to M. This information is the halting problem relative to the index set which then also allows to derive some facts of the structure of the whole set by one query. The learner can be taken to be recursive. This fact is not very surprising, since $K \leq_m B'$ in a uniform way and by Rice's Theorem [121] even $K' \leq_m B'$.

Theorem 5.4.3 *There is a machine M such that $M^{B'}$ finitely learns every class $S \in \Lambda_{\text{fin}}$ where the oracle B' is the halting problem relativized to the index set B of S.*

Proof If $i, j \in B$ and $W_i \neq W_j$, then the union $W_i \cup W_j$ does not belong to S since for no proper superset of a set in S is also in S; but if $W_i = W_j$ then the union equals to W_i and is still in S. Thus the index $f(i,j)$ of the union is in B iff $W_i = W_j$. The finite learner has to read new data-items until there is a unique superset of the data seen so far in S and so the learner can check this condition by asking to B' whether

$$(\exists i, j \in B)\,[range(\sigma) \subseteq W_i \cap W_j \wedge f(i,j) \notin B] \tag{26}$$

and outputs the symbol "?" for no guess as long as (26) is satisfied. Then the learner checks using B' whether there is an $i \in B$ with $range(\sigma) \subseteq W_i$. If so, the learner outputs the smallest such i, otherwise the data is incorrect and the learner continues to output the special symbol "?". ▮

Instead of going from B to B' one might ask whether there are other ways to improve learnability. Indeed one can use the concept of using an upper bound on the size of the smallest grammar to generate a concept. This still does not work for the class containing

$\{0\}$ and $\{0,1\}$ since an upper bound on the size of both programs does not help to decide whether a given text starting with a lot of 0's will eventually have also a 1. But it works for all inclusion-free classes, so this learning criterion is more powerful than finite learning alone. Progress is made in two directions: the collection of learnable classes is increased from Λ_{fin} to Λ_{if} and the complexity of the additional input for the universal learner is decreased from B' to B.

Theorem 5.4.4 *There is a computable learner M which learns every class $S \in \Lambda_{\text{if}}$ from the additional information consisting of the index set B for S and an upper bound a on the size of a grammar generating the language L to be learned.*

Proof M executes the following algorithm.

> $M^B(a,\sigma)$ computes the finite sets
> $B_0 = \{i \in B : \text{the size of } i \text{ is below } a\}$ and
> $B_1 = \{j \in B_0 : (\forall i \in B_0)\,[i < j \Rightarrow W_i \neq W_j]\}$.
> If there is a unique $i \in B_1$ with $range(\sigma) \subseteq W_i$
> Then $M^B(a,\sigma)$ outputs this i else $M^B(a,\sigma)$ outputs the symbol "?".

First, it is shown that all steps of the algorithm can be computed using the data given. Since S is inclusion-free one knows that, for $i,j \in B$ with $W_i \neq W_j$, the union $W_i \cup W_j$ is a proper superset of W_i and W_j and thus not in S. As in the previous proof, $f(i,j)$ computes an index of $W_i \cup W_j$ and, for $i,j \in B$, one has that $W_i = W_j \Leftrightarrow f(i,j) \in B$. Similarly it can be checked using B whether $range(\sigma) \subseteq W_i$ for any $i \in B$ and σ: If so, then $W_i \cup range(\sigma)$ is in S, otherwise $W_i \cup range(\sigma)$ is a proper superset of W_i and not in S.

Second, one verifies that the algorithm never outputs a wrong index. Whenever T is a text for some $L \in S$ and a is an upper bound on the size of some grammar generating L then the minimal index i of L is in B_1. For any $\sigma \preceq T$ it holds that $range(\sigma) \subseteq W_i$ and therefore the output can either be this i or the symbol "?". So the output cannot be an incorrect index.

Third, one verifies that the correct index is output eventually. Each two indices in B_1 are minimal indices of different languages in S. They enumerate sets which are incomparable according to the choice of S. Thus for every $j \in B_1$ different from i there is an x_j in $W_i - W_j$. After sufficient long time, for the finitely many $j \in B_1 - \{i\}$, the corresponding x_j have shown up in the text and thus $range(\sigma) \not\subseteq W_j$. It follows that from this time on, the index i is unique. The learner M outputs this index i. ∎

The nonlearnability of the class containing $\{0\}$ and $\{0,1\}$ cannot be overcome by using powerful oracles combined with upper bounds on programs. Freivalds, Kinber and Wiehagen [44] introduced the concept of learning from good examples. Lange, Nessel and Wiehagen [92] transferred the concepts to learning from text and showed that the learning power can be increased to that of conservative learning [2] from text for uniformly recursive families [92, Theorem 2]. Using sufficiently powerful oracles, one can turn every text learner into a conservative learner, thus one knows that every class from Λ is learnable from good examples with a sufficiently powerful learner. So good examples are a variant of finite learning having the advantage of covering all classes in Λ. Goldman and Mathias [57] defined the same notion and addressed the role of a teacher (that is, the algorithm to compute F from e in the definition below) in learning concrete classes like Horn formulas and decision lists.

Definition 5.4.5 [57, 92] A class S is *learnable from good examples* iff there are a partial learner M and a partial function ψ such that, for every e with $W_e \in S$, $\psi(e)$ is the canonical index of a finite subset $D_{\psi(e)}$ of W_e such that, for all finite sets D_d with $D_{\psi(e)} \subseteq D_d \subseteq W_e$, $M(d)$ is defined and an index for W_e.

Again it is not possible to generate the learner M and the partial function ψ from the index set B alone. The proof of Theorem 5.4.2 can be adapted by taking $\mathbb{N}$ instead of E and any finite superset L of some prefix $0\,1\,\ldots\,x$ of the canonical text for $\mathbb{N}$. Therefore the next result uses B' instead of B.

Definition 5.4.5 omitted any constraints on the computability of the mappings related to the learning process. Therefore it was possible to define the process without caring about the enumeration on which the concept is based — Lange, Nessel and Wiehagen [92] used uniformly recursive families which makes it easier to compute $\psi(e)$ than in the case where e is taken from some acceptable numbering of all enumerable sets — but the next result shows that besides the set B' no extra source of nonrecursive computation power is necessary for computing M and ψ.

Theorem 5.4.6 *There is a universal learner which learns every S in Λ from good examples using the oracle B' where B is the index set of S. Furthermore, no fixed $S \notin \Lambda$ can be learned from good examples, even when no bound in terms of Turing degrees is placed on the complexity of the learner.*

Proof Let M be the universal learner from Theorem 5.1.4. For every $W_e \in S$, M has a locking sequence σ satisfying $M^B(\sigma\tau) = M^B(\sigma)$ for all $\tau \in W_e^*$. This definition can be checked using oracle B' and so one can define the following B'-recursive algorithm to compute $\psi(e)$:

Enumerate W_e^* until a $\sigma \in W_e^*$ with $(\forall \tau \in W_e^*)\,[M^B(\sigma\tau) = M^B(\sigma)]$ is found. Then let $\psi(e)$ be the canonical index for $range(\sigma)$. (27)

Since M^B learns all sets in S and B is the index set for S, $\psi(e)$ is defined for all $e \in B$.

Given M^B, the following machine $N^{B'}$ is a universal learner for the criterion of learning from good examples.

$N^{B'}(d)$ enumerates all strings in D_d^* until a $\eta \in D_d^*$ is found such that $D_d \subseteq W_{M^B(\eta)}$ and $(\forall \tau \in W_{M^B(\eta)}^*)\,[M^B(\eta\tau) = M^B(\eta)]$. Then $N^{B'}(d)$ outputs the index $M^B(\eta)$. (28)

For the verification of $N^{B'}$, assume that $W_e \in S$ and $D_{\psi(e)} \subseteq D_d \subseteq W_e$. Let $\sigma \in D_{\psi(e)}^*$ be the string from (27).

The algorithm for $N^{B'}(d)$ is defined since it could take σ for the value η in its definition (28). For the case that the η taken there is different from σ, consider the strings $\sigma\eta$ and $\eta\sigma$. Recall that M^B was constructed in Theorem 5.1.4 such that M^B outputs for strings of the same length and range the same index, thus $M^B(\sigma\eta) = M^B(\eta\sigma)$. Furthermore, η is in $W_{M^B(\sigma)}^*$ since the set $W_{M^B(\sigma)}$ equals to W_e and contains therefore D_d and $range(\eta)$. It follows that $M^B(\sigma\eta) = M^B(\eta)$. Since σ is in $D_{\psi(e)}^*$, $W_{M^B(\eta)}$ contains D_d and D_d contains $D_{\psi(e)}$, the equality $M^B(\eta\sigma) = M^B(\eta)$ holds as well. Putting these observations together, one obtains that $N^{B'}(d)$ outputs also in the case $\eta \neq \sigma$ the hypothesis $M^B(\sigma)$: $N^{B'}(d) = M^B(\eta) = M^B(\eta\sigma) = M^B(\sigma\eta) = M^B(\sigma)$.

Since there is a text for W_e starting with σ and since M^B does not withdraw the hypothesis $M^B(\sigma)$ on this text, it follows that $M^B(\sigma)$ is a recursively enumerable index for W_e and the output $N^{B'}(d)$ is correct. So $N^{B'}$ witnesses together with the auxiliary function $\psi^{B'}$ that the languages in Λ are universally learnable from good examples.

The second result can be obtained by just transferring the learnability result [92, Section 3] from the world of uniformly recursive classes to arbitrary classes: the usage of learners of higher Turing degrees compensates the loss by giving up the uniform decision procedure. The direct proof is nevertheless shorter and therefore included here.

Let M and ψ witness that S is learnable from good examples. Since constraints on computability are absent, M and ψ are without loss of generality total. A new learner N which infers S from text in the limit, is defined as follows:

> $N(\sigma)$ is the minimal index e of the language generated by $M(d)$ where d is the canonical index for $range(\sigma)$.

For every $W_e \in S$, $D_{\psi(e)}$ is a finite subset of W_e and whenever $D_{\psi(e)} \subseteq range(\sigma) \subseteq W_e$, then $N(\sigma)$ outputs the minimal index of W_e. All the finitely many elements of $D_{\psi(e)}$ show up eventually on every text for W_e, thus the learner N converges on every text of W_e to an index for W_e. So N learns S in the limit from text and $S \in \Lambda$. ∎

6 Robust Access to Oracles

In the previous chapters, the oracle A was always fixed and the learner was always constructed exactly for one given oracle. Now it is investigated what happens if the learner cannot specialize on one specific oracle but has to be robust in the sense that learning has to succeed with every oracle from a given set of oracles. So one considers oracles which meet certain specifications in the way they present information on the class S to be learned: (a) the oracle is a list of all functions in S; (b) the oracle is a predictor which predicts every $f \in S$ under the model "next value"; (c) the oracle is a one-sided classifier which converges on a function f to 1 iff $f \in S$; (d) the oracle is an identifier which converges on every function $f \in S$ to some value e_f which is unique for f; (e) the oracle is a martingale which succeeds on every function in S.

Two types of additional information where already presented in the previous chapters: the standard use of oracles and the bounds on the size of programs [46, 63] discussed in Section 5.3. The latter was generalized by considering other additional finite information as indices for higher order programs which compute the function f to be learned relative to K [8], programs which agree with f on sufficiently many arguments [62] and indices of certain trees on which f is an infinite branch [104]. Learning with infinite additional information has a further root in the literature, namely Angluin's model of the "minimal adequate teacher" [3]: Here the learner may ask queries about f to a teacher which serves as additional information on the function f or the language to be learned. This additional information is infinte in the sense, that it provides answers to infinitely many queries in some fixed query language. Furthermore, the answers to the queries are not always unique; for example, there may be several ways to select counterexamples to a learner's hypothesis. So the learner has in her model to be robust in the sense that learning has to succeed with every teacher which meets the specification. Similarly robust learning in the present work is modelled by the infinite concept of a "minimal adequate oracle".

Five basic learning notions are considered, namely Ex, PEx, Fin, PFin and NV where PFin is a finite learner whose output is on any data either the symbol "?" for "no hypothesis yet" or an index of a total recursive function. For these notions, it is investigated to which extent the types of oracles defined below support learning classes $S \subseteq$ REC. The learner M accesses the oracle O via queries for $O(x)$ at certain numbers or strings x. M

has to learn every $f \in S$ with any "minimal adequate" oracle meeting the specification. Note that M has to be a learner only for these permitted oracles, if the oracle O does not meet the specification then M may also violate the specifications of the learning process, in particular, M may be partial in the case of NV-learning or output programs not being total in the case of PEx-learning or PFin-learning. The requirement, that the learner meets its specifications only for minimal adequate oracles, is quite reasonable: otherwise, for example in the case of PEx-learning, the notion would be too restrictive and not permit any nontrivial inference; that is, a PEx-learner which outputs total programs for all oracles can be replaced by a PEx-learner operating without any oracle.

Together with the definitions of the required relation between S and the oracles, an overview on the results is given. The oracles are always function oracles; one can access them either by analizing their graph or by giving the arguments of the function and receiving the output. Mathematically, both ways are equivalent for total functions, but the second definition is much more convenient. So, a function oracle like a list F is accessed by querying x, y and receiving the value $F(x, y)$ and not by repeatedly asking whether $F(x, y) = z$ for some given x, y, z.

List: A class S is in Ex[List] iff there is a machine M such that M equipped with a function oracle F Ex-learns every $f \in S$ whenever F is a list of S, that is, $S = \{F_0, F_1, \ldots\}$ where F_x is the function given by $F_x(y) = F(x, y)$. The machine M accesses the oracle by querying x, y and receiving $F(x, y)$.

Most prominent classes of recursive functions like the class REC of all recursive functions and the class $\mathrm{REC}_{0,1}$ of all $\{0,1\}$-valued recursive functions are in Ex[List], but there is also some $S \notin$ Ex[List]. Furthermore, every class is in NV[List] but PEx[List], Fin[List] and PFin[List] are weaker than PEx[List].

Predictor: A class S is in Ex[Predictor] iff there is a machine M such that M equipped with a function oracle P Ex-learns every $f \in S$ whenever P is a device which NV-learns all $f \in S$. The machine M accesses P by giving as input a finite sequence σ and receiving the prediction $P(\sigma)$ of P.

Predictors are strictly weaker than lists, for example, $\mathrm{REC}_{0,1}$ is in Ex[List] but not in Ex[Predictor]. Interestingly this is one of the few cases in inductive inference where a criterion behaves differently for $\mathrm{REC}_{0,1}$ and REC: REC $\in$ Ex[Predictor] but $\mathrm{REC}_{0,1} \notin$ Ex[Predictor]; this observation is a corollary of the fact that $\mathrm{REC}_{0,1}$ but not REC can be NV-learned relative to a suitable low oracle in the classic setting of oracle-use. By definition, predictors are omniscient for the criterion NV but on the other hand they are useless for the criteria PFin, Fin and PEx, that is, anything learned with a predictor under one of these criteria can also be learned without any additional help under the same criterion.

Classifier: A class S is in Ex[Classifier] iff there is a machine M such that M equipped with a function oracle C Ex-learns every $f \in S$ whenever C is a one-sided classifier for S. A one-sided classifier C converges on all $f \in S$ to 1, that is, $(\forall^\infty \sigma \preceq f)\,[C(\sigma) = 1]$, and does not converge to 1 on every (also nonrecursive) $f \notin S$, that is, $(\exists^\infty \sigma \preceq f)$ $[C(\sigma) = 0]$. M accesses C by querying some finite sequence σ and receiving the value $C(\sigma)$.

Classifiers allow to Ex-learn and NV-learn the classes REC and $\mathrm{REC}_{0,1}$ but they are not omniscient for these criteria. For Fin, PEx and PFin they are useless.

Identifier: A class S is in Ex[Identifier] iff there is a machine M such that M equipped with a function oracle I Ex-learns every $f \in S$ whenever I identifies S in a very abstract manner: The sequence $I(f(0)f(1)\ldots f(n))$ converges for every $f \in S$ to a value i which is unique for f within S: I converges on every $g \in S$ with $g \neq f$ to some value $j \neq i$. M accesses I by receiving $I(\sigma)$ for a query σ.

Such an identifier is a generalization of the criterion of learning higher-order programs [8, 26]. In the abstract way defined above it is equivalent to the concept where the identifier chooses some oracle A and converges to a program computing f with help of A on every $f \in S$. It turns out that it is very difficult to use these abstract algorithms: Identifiers are useless for Ex, PEx, Fin and PFin. But they are still helpful for certain classes during NV-learning, but these classes have to be dominated by a recursive function. So REC, the class S_0 of self-describing functions and the class S_5 of all total step-counting functions, cannot be NV-learned using identifiers.

Martingale: A class S is in Ex[Martingale] iff there is a machine M such that M equipped with a function oracle m Ex-learns every $f \in S$ whenever m is a martingale succeeding on S. A martingale is a total function with positive rational values such that $m(\lambda) = 1$ and for each σ there is a rational number q with $0 \leq q < m(\sigma)$ and a prediction a such that $m(\sigma a) = m(\sigma) + q$ and $m(\sigma b) = m(\sigma) - q$ for all $b \neq a$. M accesses m by receiving the rational in a suitable coding after querying σ.

It turns out that martingales are useless for all considered learning criteria.

Inside Given Degrees: The oracles in this notion are (other than the previous ones) independent of S. A class S is robustly learnable inside a given degree $\mathbf{a}$ of oracles iff there is a machine M which Ex-learns every $f \in S$ with any oracle $A \in \mathbf{a}$. M accesses the oracle A in the traditional way: on input x the value $A(x)$ is returned (0 for $x \notin A$, 1 for $x \in A$).

It is shown that for most common notions of degrees (Turing, tt, wtt, btt, m) this kind of additional information allows only to learn classes which can already be learned without access to any oracle. This result justifies the approach to investigate more syntactical qualities of the oracle than their Turing degrees with respect to the ability of oracles to support learning robustly. Only for 1-degrees the world looks different: there are some omniscient oracles like the 1-degrees of maximal sets and also some intermediate degrees.

Furthermore, one could, following the ideas of Chapter 5, consider a set B for S as additional information. Such a set is omniscient for Ex, PEx and NV since in the case of function learning these criteria might use the algorithm "learning by enumeration". For the criteria Fin and PFin, one can adapt the results from Section 5.4 and obtains that it is possible to learn all discrete classes from the jump B' of any set B for S but some discrete classes cannot be learned from B alone. The same results hold for index sets. But both notions, sets B for S and the index set of S, need to fix an acceptable enumeration of the partial-recursive functions in contrast to the main idea of this chapter where the sources of additional information are not linked to such a fixed enumeration in order to make such a source a bit more unpredictable for the learner. Therefore, these concepts are not studied within this chapter beyond the summary just given.

6.1 Robust Learning Inside Given Degrees

For a given oracle A, the Turing degree of A is the collection of all oracles B which have the same computational complexity as A, that is, which are Turing equivalent to A. There are refinements of the notion of a Turing degree such as m-degree and 1-degree: A set A is m-reducible to B iff there is a recursive function f such that $A(x) = B(f(x))$ for all x. If this f is furthermore one-one, then A is 1-reducible to B. A and B are called m-equivalent, that is, A and B have the same m-degree, if A is m-reducible to B and B is m-reducible to A. Similarly 1-equivalence and 1-degrees are defined. Odifreddi [111, Chapter VI] gives an overview on these and other degrees. The following theorem states that robust learning from an m-degree does not help. The same result also holds for the degrees given by the reductions btt, tt, wtt and Turing as defined in [111] since each such degree is the union of several m-degrees.

Theorem 6.1.1 *Assume that a single machine M* Ex$[B]$*-learns* (NV$[B]$*-learns*) *S via access to oracle B for any B in the m-degree of A. Then S can be* Ex*-learned* (NV*-learned*) *without any oracle.*

(a) Proof for Ex-Learning Let $S \in \text{Ex}[B]$ via a machine M which succeeds using any of the oracles B in the m-degree of A. Without loss of generality, M is total also for the oracles outside the m-degree of A and $M(\sigma)$ is computed with oracle access only below $|\sigma|$ — these conditions can be satisfied via delaying mind changes [43, Note 2.14]. For any function f let $M^{\alpha}(f)$ abbreviate the result of computing $M(f(0)f(1)\ldots f(|\alpha|))$ using any oracle extending α where, according to the delayment above, it does not have any effect what the values of the oracle outside the domain of α are. There are two cases:

(I) There is a function $f \in S$ with $(\forall \alpha)\,(\exists \beta \succeq \alpha)\,[M^{\beta}(f) \neq M^{\alpha}(f)]$. Now it is possible to compute inductively binary strings $\alpha_0, \alpha_1, \ldots$ such that, for each n and $a_0, a_1, \ldots, a_n \in \{0,1\}$, there are α, β with $a_0\alpha_0a_1\alpha_1\ldots a_n \preceq \alpha \preceq \beta \preceq a_0\alpha_0a_1\alpha_1\ldots a_n\alpha_n$ and $M^{\alpha}(f) \neq M^{\beta}(f)$. So, each α_n has to cope with all 2^{n+1} possible values of $a_0, a_1, \ldots, a_n$. Therefore, α_n is produced by concatenating strings γ_k for $k = 0, 1, \ldots, 2^{n+1} - 1$ where the γ_k are defined inductively as follows: suppose that $a_0a_1\ldots a_n$ equals the binary representation of k and let $\alpha = a_0\alpha_0a_1\alpha_1\ldots a_n\gamma_0\gamma_1\ldots\gamma_{k-1}$ for this binary representation. Then γ_k is the first string found such that $M^{\alpha}(f) \neq M^{\beta}(f)$ for $\beta = \alpha\gamma_k$. After doing this for all k, all possible values for $a_0, a_1, \ldots, a_n$ are covered.

It follows that M does not converge for any oracle of the form $a_0\alpha_0a_1\alpha_1\ldots$, in particular not for $B = A(0)\alpha_0A(1)\alpha_1\ldots$ which is m-equivalent to A. So the case (I) does not hold.

(II) For each function $f \in S$ there is an α with $M^{\beta}(f) = M^{\alpha}(f)$ for all $\beta \succeq \alpha$. Now the Ex-learner N for S works as follows: On input $f(0)f(1)\ldots f(n)$, N searches for the first string α such that $M^{\beta}(f(0)f(1)\ldots f(|\beta|)) = M^{\alpha}(f(0)f(1)\ldots f(|\alpha|))$ for all strings $\beta \succeq \alpha$ of length up to n and outputs $M^{\alpha}(f(0)f(1)\ldots f(|\alpha|))$.

Some first string α satisfies the condition at (II) for the given function f and thus N converges to the value $M^{\alpha}(f)$. Since there is some oracle B in the m-degree of A with $B \succeq \alpha$, M converges using this oracle B also to $M^{\alpha}(f)$ and $M^{\alpha}(f)$ is the correct value. So N infers S without the help of any oracle.

(b) Proof for NV-Learning Here $M^{\alpha}(f(0)f(1)\ldots f(n))\!\downarrow = y$ means that the oracle Turing machine queries only within $dom(\alpha)$ and converges to the output y. Again there is a case-distinction.

(I) $(\exists f \in S)\,(\forall \alpha)\,(\forall n)\,(\exists m > n)\,(\exists \beta \succ \alpha)\,[M^{\beta}(f(0)f(1)\ldots f(m))\!\downarrow \neq f(m+1)]$. As in the Ex-case it is possible to construct a computable sequence $\alpha_0, \alpha_1, \ldots$ such that M makes infinitely many mistakes during the attempt to NV-learn f for any oracle of the form $a_0\alpha_0a_1\alpha_1\ldots$: The strings $\alpha_n = \gamma_0\gamma_1\ldots\gamma_{2^n-1}$ are defined inductively for $k = 0, 1, \ldots, 2^{n+1} - 1$ such that one searches in parallel for a γ_k and $m > n$ with $M^{\beta}(f(0)f(1)\ldots f(m))\!\downarrow \neq f(m+1)$ where $\beta = a_0\alpha_0a_1\alpha_1\ldots a_n\gamma_0\gamma_1\ldots\gamma_k$ and $a_0a_1\ldots a_n$ is the dual representation for k. The construction gives that M does not NV-learn f with oracle $B = A(0)\alpha_0A(1)\alpha_1\ldots$ although the strings $\alpha_0, \alpha_1, \ldots$ are determined recursively and independent of A and thus

B has the same m-degree as A.

(II) $(\forall f \in S)\,(\exists\alpha)\,(\exists n)\,(\forall m > n)\,[(\exists\beta \succ \alpha)\,[M^\beta(f(0)f(1)\dots f(m))\downarrow = f(m+1)] \wedge (\forall\beta \succ \alpha)\,(\forall y)\,[M^\beta(f(0)f(1)\dots f(m))\downarrow = y \Rightarrow y = f(m+1)]]$. Here the learning-algorithm is a bit different to that of the Ex-case but has the same basic idea. Note that for every input $f(0)f(1)\dots f(n)$ and for every α there is some $\beta \succeq \alpha$ such that $M^\beta(f(0)f(1)\dots f(n))\downarrow$ since some oracle in the m-degree of A extends β. The new inference machine N tries always to extrapolate B from a finite amount of information α in the just indicated way and crosses out every α which once produced an error via moving it into a book-keeping set C.

> Let $\alpha \notin C$ be the first binary string within a given enumeration which is not already crossed out. Now let
>
> $$N(f(0)f(1)\dots f(n)) = M^\beta(f(0)f(1)\dots f(n))$$
>
> for the first $\beta \succeq \alpha$ where this computation terminates within $|\beta|$ computation steps. If it turns out later (when the input $f(0)f(1)\dots f(n)f(n+1)$ is processed) that this prediction was wrong then α is crossed out and C is replaced by $C \cup \{\alpha\}$.

Note that at every stage of the algorithm only one string α is crossed out. Furthermore, whenever α and n satisfy the condition (II) for a function $f \in S$, then so does also either $\alpha 0$ or $\alpha 1$ with the same n. In particular infinitely many extensions of α satisfy (II) with the same n and at most n of them can be crossed out. Since the algorithm uses always the first α not yet in the set C of crossed out candidates, it uses the same α in almost all steps. Thus, almost all predictions are correct since otherwise this α would also be crossed out. The algorithm learns every f in S. ∎

For the criteria Fin, PEx and PFin the same result holds also with 1-degrees in place of m-degrees.

Theorem 6.1.2 *Assume that a single machine M PEx$[B]$-learns S via access to oracle B for any B in the 1-degree of A. Then S can be PEx-learned without any oracle. The same result holds also for the criteria Fin and PFin.*

Proof First it is necessary to note that each finite binary string can be extended to a set in the 1-degree of A since A is not recursive and therefore infinite and coinfinite — otherwise one could fix A and replace queries to the oracle by computations. Now let S be PEx-learnable via uniform access to some oracle in the 1-degree of A via a machine M. The set

$$E = \{M^B(\sigma) : B \equiv_1 A \text{ and } \sigma \in \mathbb{N}^*\} = \{e : (\exists\beta \in \{0,1\}^*)\,(\exists\sigma \in \mathbb{N}^*)\,[M^\beta(\sigma) = e]\}$$

is an enumerable set of indices: since M uses for any output only a finite prefix of B, the search can go over all binary strings instead over all oracles 1-equivalent to A. Since any such string can be extended to an oracle 1-equivalent to A, each index in E is an index of a total recursive function. On the other hand, E contains all guesses $M^A(f(0)f(1)\dots f(n))$ for each $f \in S$. Since M learns S from oracle A, E contains for each $f \in S$ an index. Thus E is an enumerable set containing only indices of total recursive functions and for each function in S there is an index in E. It follows that S is PEx-learnable.

The proofs for the criteria Fin and PFin are based on the same idea. For each σ define

— similarly to above — the sets

$$E(\sigma) = \{M^\beta(\sigma) : \beta \in \{0,1\}^*\}$$
$$G(\sigma) = \cup_{\tau \prec \sigma} E_{|\sigma|}(\tau)$$

where the $E_s(\tau)$ are a recursive enumeration of the $E(\tau)$ uniform in τ. The algorithm outputs the symbol "?" until it reaches some $\sigma \preceq f$ such that $G(\sigma)$ is not empty. Then the algorithm outputs some $e \in G(\sigma)$ and abstains from any mind change. This first guess is computed relative some finite binary string β and since some $B \equiv_1 A$ extends β, the output must be a correct index for f provided that $f \in S$. Furthermore, in the case PFin e has to be a total index, also if σ does not belong to any $f \in S$. So again it follows that uniform access to 1-degrees does not support learning for the criteria Fin and PFin. ∎

Some 1-degrees are also trivial for Ex-learning and NV-learning. For example the 1-degree of a cylinder A, where A is called a *cylinder* if $A(\langle x, y\rangle) = A(\langle x, 0\rangle)$ for all pairs $\langle x, y\rangle$. But if A is sufficiently thin then the class REC of all recursive functions can be Ex-learned and NV-learned uniformly relative to every $B \equiv_1 A$ by a single machine M.

Theorem 6.1.3 *If the principal function p_A of A dominates every recursive function $f \in$ REC then there is a machine M which Ex-learns REC relative to any oracle B in the 1-degree of A. The same holds for NV.*

Proof Recall that the principal function p_A of A assigns to each x the x-th element of A. It can be shown that for each $B \equiv_1 A$, the principal function p_B of B also dominates every recursive function: There is a computable one-one function f such that $A = \{f(x) : x \in B\}$. If p_B would not dominate the recursive function g then p_A would also not dominate the recursive function $n \to \max\{f(m) : m \leq g(n)\}$.

Now the following algorithm M^B Ex-learns all recursive functions g: On input $\sigma \preceq g$ of length n, M^B first computes $x = p_B(n)$. Then M^B searches for the least e such that $\varphi_e(y)\downarrow = \sigma(y)$ within x computation steps for all $y \in dom(\sigma)$. If M^B finds such an e below n then M^B outputs this program e. Otherwise M^B outputs the symbol "?" to indicate that M^B could not make up its mind because of either too few data or too few computation time.

M^B converges to the minimal index e of g: Since the principal function p_B dominates the computation time of φ_e, the learner M^B outputs almost always either e or an index below e. The second case only occurs finitely often because there are only finitely many indices $i < e$ and, for each $i < e$, there is x_i such that $\varphi_i(x_i)$ either diverges or computes a value different form $f(x_i)$. So whenever $x_0, x_1, \ldots, x_{e-1} \in dom(\sigma)$, $M^B(\sigma)$ does not output a guess strictly below e and thus M^B converges to e.

The modification from Ex-learning to NV-learning is that M^B in place of outputting e simulates $\varphi_e(n+1)$ for x computation steps and outputs the result if it is found within x steps. Otherwise it outputs 0. Since p_B dominates the computation-time of φ_e whenever φ_e is total (and in particular equals f) the procedure predicts almost always every recursive function. Note that M must be total only for oracles $B \equiv_1 A$ and may diverge on others, in particular on oracles represented by finite sets. ∎

This proof gives the nice (and already well-known) fact that whenever a dominating function can be computed from the oracle then REC can be learned under the criterion Ex using this dominating function [1]. This function needs not to be the same for all permitted oracles but each permitted oracle must give a dominating function via the same algorithm. The construction will be used in several proofs below. Martin [97] and Tennenbaum [142] showed that the complement of every maximal set has a dominating principal function. Therefore it follows immidiately that the 1-degrees of maximal sets are omniscient for robust learning.

Corollary 6.1.4 *If A is maximal then* REC *is* NV*-learnable and also* Ex*-learnable with robust access to the oracles within the 1-degree of A.*

The following result allows to characterize the robust learning power of enumerable Turing degrees. As the last results indicates, it is possible to learn the whole class REC relative to a suitable 1-degree inside any given high degree. All other enumerable Turing degrees do somehow not contain any 1-degree which is helpful for robust learning, so enumerable Turing degrees are either omniscient or trivial.

Theorem 6.1.5 *If A has enumerable but not high Turing degree then every S robustly* Ex*-learnable inside the 1-degree of A is also* Ex*-learnable without any oracle.*

Proof The proof follows mainly the lines of Theorem 6.1.1. Assume that M is a robust learner. Now it is sufficient to consider case (I) since the translation of the algorithm in case (II) is literally the same. Case (I) assumed that there is a function $f \in S$ with $(\forall \alpha)\, (\exists \beta \succeq \alpha)\, [M^\beta(f) \neq M^\alpha(f)]$. Without loss of generality, one can assume that M is increasing so that whenever there is a mind change the new value is properly larger than the old one.

Again there is a routine which given an n produces a string β_n such that, for every $\alpha \in \{0,1\}^n$, some mind change occurs if the oracle extends $\alpha\beta_n$: $M^{\alpha\beta_n}(f) \neq M^\alpha(f)$. Furthermore, let each string β_n contain at least one 0 and one 1. Now one constructs a set $B \equiv_1 A$ such that there are infintely many n with $B(x+n) = \beta_n(x)$ for all $x \in dom(\beta_n)$. Let $A_0, A_1, \ldots$ be an approximation of A with A-recursive modulus. There is a function g computable relative to A which assigns to each n the value $m + n + s$ such that $m = p_A(n+|\beta_n|) + p_{\overline{A}}(n+|\beta_n|)$ and s is the first stage where $A_s(x)$ has converged to $A(x)$ for all $x \leq m$. Recall that $p_A(k)$ is the position of the k-th element of A and $p_{\overline{A}}(k)$ is the position of the k-th element of $\overline{A}$, that is, of the k-th nonelement of A. Note that g is increasing. Since A does not have high Turing degree, there is a computable and increasing function h such that, for infinitely many k, $h(k) \geq g(g(k))$. Now a permutation π is constructed in stages. If at stage s the first n values of π are defined then one extends π on the next $|\beta_n|$ values by the following rule:

$$\pi(n+x) = \min\{y \notin \{\pi(0), \pi(1), \ldots, \pi(n+x-1)\} : A_{h(n)}(y) = \beta_n(x)\}$$

The resulting function π is computable. Let $B = \pi(A)$. There are infinitely many k with $h(k) \geq g(g(k))$. For each such k consider the first s for which $\pi(k)$ is defined. The corresponding n is above k but below $g(k)$. Therefore $h(n) \geq h(k) \geq g(g(k)) \geq g(n)$. So it follows that at this stage s for $y \leq g(n)$ the values $A_{h(n)}(y)$ and $A(y)$ coincide and therefore also for any $x \in dom(\beta_n)$ the equation $B(n+x) = \beta_n(x)$ holds. So for infinitely many n and the corresponding $\alpha = B(0)B(1)\ldots B(n-1)$ it holds that $B \succeq \alpha\beta_n$ and that $M^{\alpha\beta_n}(f) \neq M^\alpha(f)$. Thus M makes for each such n a mind change and does not converge on f. Since each β_n contains an 1 and 0, each element and each nonelement of A is at some time associated to some x and π is a computable permutation. So B is in the same 1-degree as A but M does not learn f relative to B.

So case (I) does not hold and therefore (II) from Theorem 6.1.1 holds. Now it follows by exactly the same argument as in Theorem 6.1.1 that $S \in$ Ex. ∎

The last theorem showed that enumerable nonhigh Turing degrees do not contain any 1-degree relative to which it is possible to learn anything nontrivial robustly. The next theorem shows that some nonhigh Turing degrees D differ from this in the sense that they contain a 1-degree which allows to learn every $S \in \mathrm{Ex}[D]$ robustly. So for any oracle in this 1-degree the learning power given by robust and normal access to the oracle is the same.

Theorem 6.1.6 *Let $D \leq_T K$ have the Turing degree of a complete extension of Peano Arithmetic. Then there is $A \equiv_T D$ such that exactly the $S \in \mathrm{Ex}[D]$ can be learned robustly relative to the 1-degree of A. Furthermore, D can be chosen such that D is not trivial and also not omniscient, that is, some nontrivial class S is learnable relative to D while* REC *cannot be learned using oracle D.*

Proof A number x is Kolmogorov random if there is no index $e < \frac{x}{3}$ such that $x = \varphi_e(0)$. Let C be the set of all Kolmogorov random numbers. The complement of C is an enumerable set. Furthermore, C has in each interval $I_n = \{2^n, 2^n + 1, \ldots, 2^{n+1} - 1\}$ at least one element. Relative to the oracle D one can select from each interval I_n an element x_n which is not enumerated into the complement of C and thus C has a subset $\{x_0, x_1, \ldots\} \leq_T D$.

Let $c_D(x)$ be the first stage $s \geq x$ such that $D_s(y) = D(y)$ for $y = 0, 1, \ldots, x$ where D_s is an approximation of D. If the sequence $b_0, b_1, \ldots$ dominates c_D then D is computable relative to $B = \{b_0, b_1, \ldots\}$ [135, Exercise IX.2.18 (a)]. There is even a single machine N such that $N^B(x) = D(x)$ whenever $b_z \geq c_D(z)$ for all $z \geq x$. N can be modified such that $N^B(x, y) = D(x)$ whenever $b_z \geq c_D(z)$ for all $z \geq y$. Furthermore, N is total for every infinite oracle.

Now let $a_n = x_{c_D(n)}$ for all n. The set $A = \{a_0, a_1, \ldots\}$ has the same Turing degree as D. Let B be in the same 1-degree as A. There is a recursive one-one function f such that $B = f(A)$. Now for each a_n it holds that $f(a_n) \geq q \cdot a_n - c$ for some rational constant $q > 0$ and some c since otherwise the sequence $a_0, a_1, \ldots$ would not consist of Kolmogorov random numbers. It follows that $f(a_n) \geq \log(a_n)$ for almost all n and thus $f(a_n) \geq c_D(n)$ for almost all n. Since B is the range of A, $b_n \geq c_D(n)$ for almost all n.

Given $S \in \mathrm{Ex}[D]$, there is a D-recursive Ex-learner M for S such that $M^D(\sigma)$ queries D only below $|\sigma|$ for every $\sigma \in \mathbb{N}^*$. Since $D \leq_T K$ there is an approximation M_s for M. Let $\sigma_0, \sigma_1, \ldots$ be an enumeration of all strings and define as in Theorem 2.4.4 the function

$$c_M(x) = \min\{s \geq x : M_s(\sigma_y) = M(\sigma_y) \text{ for all } y \leq x\}.$$

Since $c_M(x) \leq c_D(x)$ for all x, the sequence $b_0, b_1, \ldots$ dominates also c_M. Now it is shown that the machine N given by

$$N(\sigma_y) = M_x(\sigma_y) \text{ for the first } x \geq y \text{ with } M_x(\sigma_y) = M_s(\sigma_y) \text{ for } s = x, x{+}1, \ldots, b_x$$

robustly Ex-learns every $f \in S$: Since $M_s(\sigma_y)$ converges to $M(\sigma_y)$ at some t, any $x \geq y+t$ satisfies the condition $M_x(\sigma_y) = M_s(\sigma_y)$ for $s = x, x{+}1, \ldots, b_x$ and N^B is total. There is a number z such that $c_M(x) \leq b_x$ for all $x \geq z$. Almost all $\sigma_y \preceq f$ satisfy $y \geq z$, so for each of them the algorithm produces an $x \geq z$ such that $M_s(\sigma_y) = M_x(\sigma_y)$ for $s = x, x+1, \ldots, b_x$. By the choice of c_M, one of these $M_s(\sigma_y)$ coincides with $M(\sigma_y)$. So the output of M and N coincide for almost all σ_y. So if M Ex-learns f, then $N^B(f(0)f(1)\ldots f(y)) = M(f(0)f(1)\ldots f(y))$ for almost all y and so also N Ex-learns f. Therefore it is possible to learn every $S \in \mathrm{Ex}[D]$ also using robustly any oracle B within the 1-degree of A.

On the other hand, if some class is learnable with robust use of the oracle A, then it is also learnable with standard use of the oracle A and since $A \leq_T D$ also with standard use of the oracle D. So robust and standard use of the oracle A coincide.

Now an intermediate degree D is constructed: By the Low Basis Theorem [135, VI.5.13] there is a low set $D \leq_T K$ whose Turing degree is the one of a complete extension of the Peano Arithmetic. Since such a set is not below a 1-generic set, it is not trivial for Ex [132]. On the other hand, D is not high and therefore REC is not Ex-learnable relative to D [1]. It follows that the 1-degree of the corresponding set A is neither trivial nor omniscient with respect to robust Ex-learning. ▮

6.2 Lists

Gold [56] showed that every uniformly recursive class of functions can be learned by enumeration. Also in more restricted notions of learning, a uniformly recursive representation of the class — also often called an indexed family — is very helpful. So Angluin [2] initiated a study of the learnability of classes of sets with an indexed family. Jantke [68] introduced within this model the intensively studied notions of monotonic inference. Lange and Zeugmann [150, 151] give an overview on these studies.

In the present work such a uniformly recursive computation procedure is replaced by an oracle which consists of a list of all functions in S. It is investigated how well such an oracle supports learning.

For a given array F let F_x denote the function $F_x(y) = F(x, y)$. Such an array F is a list for a class S iff $S = \{F_x : x \in \mathbb{N}\}$, so a list for S contains just all functions in S (but no nonmembers of S).

A folklore result is that every uniformly recursive class can be learned with respect to its enumeration as hypothesis space. Learning by enumeration allows also to predict the next value, thus the following NV-learner P succeeds on all functions in S using any list F for S.

$$P(a_0 a_1 \dots a_y) = \begin{cases} F_x(y+1) & \text{for the first } x \leq y \text{ with} \\ & F_x(0) = a_0, F_x(1) = a_1, \dots, F_x(y) = a_y; \\ 0 & \text{if there is no such } x \leq y. \end{cases}$$

So, one obtains the following theorem.

Theorem 6.2.1 $S \in \mathrm{NV[List]}$ *for all* $S \subseteq \mathrm{REC}$, *that is, lists are omniscient for* NV.

Many natural examples can be Ex-learned using a list.

Example 6.2.2 The following classes are in Ex[List].
(a) The classes REC of all recursive functions.
(b) The class $\mathrm{REC}_{0,1}$ of all $\{0, 1\}$-valued recursive functions.
(c) The class $S_0 = \{f : (\exists e) [\varphi_e = f \wedge 0^e 1 \preceq f]\}$ of all self-describing functions.
(d) The class $S_1 = \{f : (\forall^\infty x) [f(x) = 0]\}$ of all functions with “finite support”.
(e) The union $S_0 \cup S_1$.

The classes S_0 and S_1 are already in Ex, so one does not need to proof Example 6.2.2 (c) and (d). The other three cases are covered by Theorem 6.2.3 below.

Alternatively to Theorem 6.2.3, one could prove (a) and (e) also by adapting the proof of Theorem 6.1.3. The basic idea is the following: A list F contains, in both cases, for every recursive function f, a function F_x which dominates f. Then, the function $h = F_0(x) + F_1(x) + \ldots + F_x(x)$ also dominates every recursive function. Although the specific form of the function h depends on the oracle F, the dominating property holds for every list F and can be exploited uniformly for the learning algorithm.

Theorem 6.2.3 *If* $\mathrm{REC}_{0,1} \cap S_0 \subseteq S$ *then* $S \in \mathrm{Ex[List]}$.

Proof This proof follows an idea of Jockusch [71]. There is a partial-recursive $\{0,1\}$-valued function ψ which has no total recursive extension. Based on this ψ one defines via dovetailing

$$\varphi_{g(i,j)}(x) = \begin{cases} 1 & \text{if } x = j; \\ 0 & \text{if } x < j \text{ or } \varphi_i(y) \text{ converges for all } y \leq x; \\ \psi(x) & \text{if } x > j \text{ and } \psi(x) \text{ converges before} \\ & \text{the condition above is satisfied;} \\ \uparrow & \text{otherwise.} \end{cases}$$

By the Fixed-Point Theorem, there is a function e such that $\varphi_{g(i,e(i))} = \varphi_{e(i)}$ for all i. The resulting function $\varphi_{e(i)}$ is either partial or in $S_0 \cap \mathrm{REC}_{0,1}$. Furthermore, $\varphi_{e(i)}$ has a recursive $\{0,1\}$-valued extension iff φ_i is total. In particular, $\varphi_{e(i)}$ has an extension F_j iff φ_i is total. So the learner M uses the list F to check whether φ_i is total and searches for the least pair $\langle i, k\rangle$ such that φ_i differs not from f and $\varphi_{e(i)}$ is extended by F_k, that is, the totalness of φ_i is "witnessed" by F_k:

> $M(f(0)f(1)\ldots f(x))$ outputs the i from the least pair $\langle i,k\rangle$ such that, for all $y \leq x$, $f(y) = \varphi_{i,x}(y)$ whenever $\varphi_{i,x}(y)$ is defined and $\varphi_{e(i),x}(y) = F_j(y)$ whenever $\varphi_{e(i),x}(y)$ is defined.

For the verification note, that such a pair $\langle i,k\rangle$ exists, since each $\{0,1\}$-valued total recursive function has an index i and then the function $\varphi_{e(i)}$ is also total and recursive and equals some F_k. Furthermore, all false pairs $\langle i,k\rangle$ are thrown out eventually since either $\varphi_i(y)\downarrow \neq f(y)$ for some y or $\varphi_{e(i)}$ has no recursive extension and thus differs from F_k somewhere on $dom(\varphi_{e(i)})$. So M converges to the i of the least pair $\langle i,k\rangle$ such that $\varphi_i = f$ and $\varphi_{e(i)} = F_k$. ∎

Most learning-criteria are closed with respect to taking subclasses, that is, if $S \subseteq S'$ and S' is learnable, so is S. But the previous theorem gives some incidence that this might fail for the criteria considered here: Learning $\mathrm{REC}_{0,1}$ required a different method than learning REC. Indeed since information on the class S' is given to the learner for S', some of this information might be lost when replacing S' by the subclass S and the lack of this information might destroy learnability. The next result establishes this conjecture and shows in addition, that the learning criterion Ex[List] is not closed under union.

Theorem 6.2.4 *There are two classes* $S_2, S_3 \in \mathrm{PFin[List]}$ *such that their union* $S_2 \cup S_3$ *and their difference* $S_2 - S_3$ *are not in* Ex[List]. *That is, none of the criteria* PFin, Fin, PEx *and* Ex *is closed under union or difference for learning from lists.*

Proof Let S_3 contain all constant functions. The class S_2 is defined using a construction from [89, Theorem 7.1]. This theorem shows that there is a family $\varphi_{g(0)}, \varphi_{g(1)}, \ldots$ and a list $f_0, f_1, \ldots$ of functions in $\mathrm{REC}_{0,1}$ such that

- $range(\varphi_{g(i)}) = \{0,1\}$ and $0^i1 \preceq \varphi_{g(i)}$;
- For all i there is at most one x with $\varphi_{g(i)}(x)\uparrow$;
- f_i extends $\varphi_{g(i)}$ and is recursive;
- The set $A = \{(x,i) : f_i(x) = 1\}$ has low Turing degree;
- The class $S_4 = \{f_i : i \in \mathbb{N}\}$ is not Ex-learnable relative to A.

Now let S_2 contain all functions in S_4 plus all constant functions of the form $f(x) = \langle i,j\rangle + 2$ where $(\forall j \geq i)\,[\varphi_{g(i)}(j)\downarrow]$. The following four observations give a proof of the theorem.

(I) $S_2 \in$ PFin[List]: Learning a function $f \in S_2$, the learner M checks whether $f(0) > 1$. If so, the function is a constant function and M outputs a total index for it. If not, M outputs "?" until an i is known with $0^i 1 \preceq f$ and a j is found such that the function h with $h(0) = \langle i,j\rangle + 2$ is in the list. Then the function $\varphi_{g(i)}$ is total beyond j and M outputs the index $g'(i,j)$ of

$$\varphi_{g'(i,j)}(x) = \begin{cases} f(x) & \text{if } x \leq j; \\ \varphi_{g(i)}(x) & \text{otherwise;} \end{cases}$$

where f is the function on the input. Only its first j values are necessary, but the j can depend on the concrete form of the list. By the choice of the constant functions in S_2 and the fact that the list contains exactly those functions which belong to S_2, the algorithm always outputs exactly one total program and this one is correct if the data belongs to some $f \in S_2$.

(II) $S_3 \in$ PFin[List]: This follows directly from the fact, that a constant function f is known after seeing the value $f(0)$.

(III) $S_2 - S_3 \notin$ Ex[List]: $S_4 = S_2 - S_3$ and A is a list for S_4. By the choice of A and S_4, the class S_4 cannot be learned with A-oracle, in particular not under the criterion Ex[List] since the list presented can be exactly A.

(IV) $S_2 \cup S_3 \notin$ Ex[List]: There is also an A-recursive list for $S_4 \cup S_3 = S_2 \cup S_3$. Since $S_4 \notin \text{Ex}[A]$, the same holds for the superclass $S_2 \cup S_3$ and so this class can also not be learned with the help of a list. Indeed the point is that by the union the particular information, from where on a function $\varphi_{g(i)}$ is total, is overwritten. ∎

A direct corollary is, that the class S_2 can be learned under the criteria PFin[List], PEx[List], Fin[List] and Ex[List], but not under the criteria PFin, PEx, Fin or Ex. So lists are really a help for several learning criteria.

6.3 Predictors

Bārzdins [11] and Blum and Blum [19] introduced the learning criterion NV where the learner has to interpolate the next value from the previous ones. In this section it is investigated to which extent such a predicting device can be uniformly translated into a learner for one of the other four criteria. Formally, a total device P is called a predictor for S iff

$$(\forall f \in S)\,(\exists x)\,(\forall y > x)\,[P(f(0)f(1)\dots f(y)) = f(y+1)],$$

that is, iff it predicts each function $f \in S$ at almost all places $y+1$ from the data $f(0), f(1), \dots, f(y)$. Using the algorithm above Theorem 6.2.1, one can turn any list F to a predictor P. But this translation is not reversible: a predictor may also predict functions outside the class S to be learned and so hide the information which functions belong to S and which not.

While lists help under all inference-criteria, predictors are no longer helpful for PFin, Fin and PEx. This is due to the fact, that every finite modification of a predictor is again a predictor and so the inference machine has to fulfill the requirements for these three learning criteria also under all finite modifications of the predictors. Then it follows by an easy adaption of the proof of Theorem 6.1.2 that the criteria are not supported by predictors as additional information.

Theorem 6.3.1 PEx[Predictor] = PEx, Fin[Predictor] = Fin *and* PFin[Predictor] = PFin.

Proof Let $S \in$ PEx[Predictor] via a machine M. The sets

$$E(\sigma) = \{e : (\exists \alpha \in \mathbb{N}^*)\,[M^\alpha(\sigma){\downarrow} = e]\}$$

are uniformly enumerable and contains all possible outputs of M on σ, in particular a program for every $f \in S$. If P is a predictor for S, so is the predictor Q defined as

$$Q(\sigma) = \begin{cases} \alpha(\sigma) & \text{if } \sigma \in dom(\alpha); \\ P(\sigma) & \text{otherwise;} \end{cases}$$

since the change from P to α on the domain of α can in the worst case add only finitely many errors to those of P on any function f. So every α is a prefix of a predictor of S and all indices in $E(\sigma)$ are total in the case of the learning criteria PEx and PFin. If $f \succeq \sigma$ is in S, then every index in $E(\sigma)$ is a program for f in the case of PFin and Fin.

In the case of PEx, the union of all $E(\sigma)$ is again recursively enumerable and contains only total programs, among them for each $f \in S$ at least one. It follows that S is the subset of an enumerable family and has already a PEx-learner which does not use any oracle.

In the case of Fin and PFin, the learner outputs "?" until an input $f(0)f(1)\dots f(n)$ is processed which is so long that some index e in the set $E(f(0)f(1)\dots f(m))$ is found in time n^2 for some $m \leq n$. Then this e is by assumption a program for f and the learner succeeds on f. It again follows that the modified learner succeeds on S but does not use any oracle. ∎

While predictors are omniscient for NV-learning (by definition) and trivial for Fin, PFin and PEx, they are intermediate for Ex-learning. In particular the natural class REC is learnable by predictor while the class $\mathrm{REC}_{0,1}$ is not.

Theorem 6.3.2 REC $\in$ Ex[Predictor] *and* $\mathrm{REC}_{0,1} \notin$ Ex[Predictor].

Proof The first result is due to the fact that a dominating function can be computed using a predictor. For each σ the predictor P defines inductively a total function f_σ via extending the string σ by P:

$$f_\sigma(n) = \begin{cases} \sigma(n) & \text{for } n \in dom(\sigma); \\ P(f_\sigma(0)f_\sigma(1)\dots f_\sigma(n-1)) & \text{for } n \notin dom(\sigma). \end{cases}$$

So if n is the first number outside the domain of σ then $f_\sigma(n) = P(\sigma)$, $f_\sigma(n+1) = P(\sigma f_\sigma(n))$ and so on. Given an enumeration $\sigma_0, \sigma_1, \dots$ of all strings, the function

$$h(x) = f_{\sigma_0}(x) + f_{\sigma_1}(x) + \dots + f_{\sigma_x}(x)$$

is uniformly recursive in the given predictor P and dominates every recursive function. As in Theorem 6.1.3 it follows that REC can be learned in the limit using this h obtained from P.

The construction fails in the case of $\mathrm{REC}_{0,1}$. Indeed there is a low oracle predicting all $\{0,1\}$-valued functions. This oracle gives a predictor, but the predictor is not sufficiently powerful to learn $\mathrm{REC}_{0,1}$ in the limit since this requires a high oracle [1, 43]. ∎

6.4 Classifiers

A one-sided classifier C [138] assigns to every string σ a binary value. C classifies S iff

$$(\forall f)\,[f \in S \Leftrightarrow (\forall^\infty \sigma \preceq f)\,[C(\sigma) = 1]\,].$$

For suitable classes S, the one-sided classifier cannot be recursive. Furthermore, one cannot use two-sided instead of one-sided classifiers, since several classes like the class S_1 of all almost everywhere 0 functions do not even have a nonrecursive two-sided classifier [124, Chapter 15.1]. Chapter 7 provides more information on classifiers. One-sided classifiers still do not help the criteria PEx, Fin and PFin via the same argument as in the case of 1-degrees and predictors. The following theorem is stated without proof, since the proofs for Theorem 6.1.2 and 6.3.1 could be adapted with minor changes.

Theorem 6.4.1 PEx[Classifier] = PEx, Fin[Classifier] = Fin *and* PFin[Classifier] = PFin.

Reliable inference [19, 25, 81, 106] means, that a machine converges on a function f iff it learns this function. In the context of learning total functions, there are two definitions: the first postulates only divergence on the not learned functions in REC, the second postulates also divergence on the total functions outside REC. The next theorem shows, that every class S learnable in the limit using any classifier can even be learned reliably in the second, more restrictive sense (here called REx), again using any classifier.

Theorem 6.4.2 Ex[Classifier] = REx[Classifier].

Proof The criterion Ex is more general than REx, thus it is sufficient to show only the direction Ex[Classifier] $\rightarrow$ REx[Classifier]. Let $S \in$ Ex[Classifier] via a classifier C and an inference-machine M. Furthermore, let *pad* be an injective padding-function such that $\varphi_{pad(i,j)} = \varphi_i$ for all i and j. The new REx-learner N uses *pad* to enforce a mind change whenever C takes the value 0.

$$N(\sigma)\!=\!pad(M(\sigma), |\tau|) \text{ for the longest } \tau\!\preceq\!\sigma \text{ with } C(\tau)\!=\!0 \vee M(\tau)\!\neq\! M(\sigma).$$

Without loss of generality, $C(\lambda) = 0$ and thus N is total. Now one shows that N is a reliable inference algorithm for S: If $f \in S$, then C converges on f to 1 and M converges to some index e with $\varphi_e = f$. There is a longest $\tau \preceq f$ such that $C(\tau) = 0 \vee M(\tau) \neq e$. Thus the learner N converges to $pad(e, |\tau|)$. If f is not in S then there are infinitely many $\tau \preceq f$ with $C(\tau) = 0$. For all these τ, N takes the value $pad(M(\tau), |\tau|)$ and all these values are different, that is, N does not converge. It follows that S is learned via the reliable machine N. ∎

The next Theorem uses — as the corresponding Theorem 6.2.3 for lists — Jockusch's construction [71] in order to show that every class containing all $\{0, 1\}$-valued self-describing functions is learnable using a classifier.

Theorem 6.4.3 *If* $\mathrm{REC}_{0,1} \cap S_0 \subseteq S$ *then* $S \in$ Ex[Classifier].

Proof Let ψ be a $\{0, 1\}$-valued partial recursive function without any total recursive extension. By Jockusch's construction [71] there is a recursive function g such that the functions $\varphi_{g(n,m)}$ satisfy the following requirements:

- $0^m1 \preceq \varphi_{g(n,m)}$ and $range(\varphi_{g(n,m)}) = \{0, 1\}$;
- If φ_n is total so are all functions $\varphi_{g(n,m)}$;

- If φ_n is partial then $\varphi_{g(n,m)}(x)\downarrow = \psi(x)$ for almost all $x \in dom(\psi)$.

By the recursion theorem with parameters [135, II.3.5] there is a recursive function h such that $\varphi_{h(n)} = \varphi_{g(n,h(n))}$ for all n. Note that every function $\varphi_{h(n)}$ is self-describing and that $\varphi_{h(n)}$ is total iff φ_n is. Furthermore, $\varphi_{h(n)}$ is either total or has no total recursive extension at all.

It can be computed effectively in the limit from any classifier C for S whether the function $\varphi_{h(n)}$ is total or not: To see this let σ_s be the longest prefix of the function $\varphi_{h(n)}(0)\varphi_{h(n)}(1)\ldots$ such that all its values are calculated within s stages. An extension $\tau \succeq \sigma_s$ is said to be consistent with $\varphi_{h(n)}$ ($\varphi_{h(n),s}$) iff for every $x \in dom(\tau) \cap dom(\varphi_{h(n)})$ ($x \in dom(\tau) \cap dom(\varphi_{h(n),s})$), the values $\tau(x)$ and $\varphi_{h(n)}(x)$ coincide. Consider the following sequence, which is uniformly recursive in the parameters s and n.

$$a_s = \begin{cases} 1 & \text{if there is an extension } \tau \in \{0,1\}^{s+1} \text{ of } \sigma_s \\ & \text{which is consistent with } \varphi_{h(n),s} \text{ and which satisfies} \\ & C(\eta) = 1 \text{ for all } \eta \text{ with } \sigma_s \preceq \eta \preceq \tau; \\ 0 & \text{otherwise.} \end{cases}$$

If φ_n is total, then $\varphi_{h(n)}$ is total and C converges on $\varphi_{h(n)}$ to 1. If s is sufficiently large, then σ_s is sufficiently long and the string $\tau = \varphi_{h(n)}(0)\varphi_{h(n)}(1)\ldots\varphi_{h(n)}(s)$ satisfies the requirements. τ extends σ_s. τ is obviously consistent with $\varphi_{h(n),s}$. $C(\eta) = 1$ for all η between σ_s and τ. So the a_s converge to 1.

If φ_n is partial, then $\varphi_{h(n)}$ has no recursive extension and C does not converge to 1 on any f extending $\varphi_{h(n)}$. Let σ be the longest prefix of $\varphi_{h(n)}$ such that all its values are defined. Now consider the following binary tree T_σ:

> A binary string τ is in T_σ either if $\tau \preceq \sigma$ or if τ extends σ, τ is consistent with $\varphi_{h(n)}$ and $C(\eta) = 1$ for all η between σ and τ.

Since C does not converge to 1 on any f extending $\varphi_{h(n)}$, the binary tree T_σ does not have any infinite branch f. So the tree T_σ is finite and there is some x bounding the length of every string in T_σ. Let $s > x$ be a stage such that for all $y \leq x$ the value $\varphi_{h(n)}(y)$ is calculated within s steps whenever it is defined. Now $\sigma_s = \sigma$. Furthermore, whenever $\tau \notin T_\sigma$, there is either some η between σ_s and τ with $C(\eta) = 0$ or there is some y with $\tau(y)\downarrow \neq \varphi_{h(n)}(y)\downarrow$. If the first case does not hold, then it follows by the construction of T_σ that the second case holds for some $y \leq x$. So τ is also inconsistent with $\varphi_{h(n),s}$. It follows that $a_s = 0$ since a_s is not 1 via any $\tau \in \{0,1\}^{s+1}$. The a_s converge to 0 in this second case.

So it can be computed in the limit using C which functions φ_n are total and this computation does not depend on the particular form of C. The learner M uses this information for the following construction: At every stage, M outputs the first e which is at stage $|\sigma|$ assumed to be total and for which $\varphi_{e,|\sigma|}$ is consistent with the data σ seen so far, that is, which satisfies $\varphi_{e,|\sigma|}(x) = \sigma(x)$ for all $x \in dom(\varphi_{e,|\sigma|}) \cap dom(\sigma)$. ∎

This result also holds for NV-learning (after modifying the last part of the proof above).

Any list F can be transferred into a one-sided classifier C: Let e_σ be the smallest index such that either F_{e_σ} extends σ or $e_\sigma = |\sigma|$. Now let $C(\lambda) = 0$ and, for any non-empty string σa,

$$C(\sigma a) = \begin{cases} 1 & \text{if } e_\sigma = e_{\sigma a}; \\ 0 & \text{otherwise.} \end{cases}$$

This C is the classification-version of the well-known algorithm to learn by enumeration; C converges to 1 exactly on the functions in the list F. So everything which can be learned from a classifier can also be learned from a list.

Theorem 6.4.4 Ex[Classifier] $\subseteq$ Ex[List].

So both concepts Ex[Classifier] and Ex[Predictor] are weaker than Ex[List]. The next theorem shows that they are incomparable and thus both concepts are strictly weaker than Ex-learning from a list.

Theorem 6.4.5 Ex[Classifier] *and* Ex[Predictor] *are incomparable.*

Proof Since $REC_{0,1} \in$ Ex[Classifier] $-$ Ex[Predictor], only the other noninclusion remains to be shown: Ex[Predictor] $\not\subseteq$ Ex[Classifier]. The class to witness this noninclusion is the union of the following two classes:

- The class S_4 from Theorem 6.2.4.
- The class $S_5 = \{\Phi_e : e \geq 0\} \cap REC$ of all total step-counting functions, where $\Phi_e(x)$ is defined as the time to compute $\varphi_e(x)$ if $\varphi_e(x)\downarrow$ and $\Phi_e(x)$ is undefined otherwise.

Blum [18] introduced abstract measures where the step-counting functions are the best known example of such an abstract measure — S_5 can be defined using any such abstract measure instead of the step-counting functions. Note that the uniform graph $G = \{(x, y, e) : \Phi_e(x)\downarrow = y\}$ of all step-counting functions is decidable and thus S_5 has even a recursive one-sided classifier C given by

- $C(f(0)f(1)\ldots f(n)) = 0$ if $n = 0$ or $a_n > a_{n-1}$ where $a_n = \max\{i \leq n : (\forall j \leq i)\, (\exists x < n)\, [(x, f(x), j) \notin G]\}$;
- $C(f(0)f(1)\ldots f(n)) = 1$ otherwise.

The class S_4 has a list relative to some low oracle A and therefore it also has a classifier relative to A. So, the union of $S_4 \cup S_5$ has a classifier of degree A, but as already mentioned in Theorem 6.2.4, S_4 and every superclass can only be learned from oracles of high degree. Therefore $S_4 \cup S_5 \notin$ Ex[Classifier].

On the other hand, if M is a predictor for S_5 then M must predict the computation-time for each function φ_e almost everywhere. So uniformly in M some function dominating all computation-times can be calculated and using this function it is possible to infer every recursive function — in particular every function in $S_4 \cup S_5$. ∎

A direct corollary is, that whenever M is a predictor for S_5, then a dominating and therefore nonrecursive function can be computed relative to M. In particular S_5 has no predictor which uses only the computable above constructed classifier as oracle and thus $S_5 \notin$ NV[Classifier].

Corollary 6.4.6 *The class S_5 of all total step-counting functions is not in* NV[Classifier].

6.5 Identifiers

An identifier is a generalization of an Ex-learner: it converges on every function in S to a number which is unique for this function within S. Such a number could be viewed as an abstract code for this program. The criterion LimEx is a generalization of Ex in the sense that the learner converges to a program which computes f in the limit — or equivalently with help of the oracle K.

Identifiers are the next step of this direction: Given an identifier I, there exists a (generally not recursive) two-variable function F such that, for all $f \in S$, there is an index i where the identifier I converges on the data $f(0)f(1)\ldots$ to i and $f(x) = F(i,x)$ for all x. Note that this definition is consistent since there is, for every i, at most one $f \in S$ on which I converges to i. One may complete the definition of F by taking $F(i,x) = 0$ for those i where I converges on no $f \in S$ to i. Formally the definition of an identifier is as follows.

An abstract device I is an *identifier for a class S of functions* iff I converges on every $f \in S$ and I takes on every two distinct functions $f, g \in S$ different values in the limit. There is no requirement how I behaves on functions outside S, on these I may either diverge or converge to any index e without caring whether e is an index of some function in S or not. The only requirement is that, also in these cases, I as a function is total, that is, $I(\sigma)$ is defined for every $\sigma \in \mathbb{N}^*$.

There are classes which are learnable under the criterion LimEx but not under the criterion Ex [26] and LimEx is not omniscient. Furthermore, some classes $S \notin$ LimEx have an recursive identifier but REC does not have one. So an identifier can still provide some nontrivial information. But the surprising result is, that this information is of no help for Ex-learning at all: Roughly spoken, it is as hard to decode and translate the indices produced by an arbitrary I as to learn programs for functions in a given S without any help. On the other hand certain classes are NV-learnable with the help of an identifier which cannot be NV-learned without any help. So while the translation of indices produced by an identifier is always impossible it is sometimes possible to "evaluate" them in nontrivial situations.

Theorem 6.5.1 *If S is robustly learnable under one of the criteria* Ex, PEx, Fin, PFin *with the help of an identifier then S is learnable under the same criterion without any help.*

Proof The proofs for PEx, Fin and PFin use only the fact that identifiers are closed under finite variations, so the results follow immidiately the lines of Theorems 6.1.2 and 6.3.1. The case Ex needs a new proof which is based on three ideas: (I) there is a computable binary tree T such that relative to every branch it is possible to compute uniformly an identifier for REC, (II) if $S \in$ Ex[Identifier] then S is learnable via a machine N which succeeds with every oracle represented by an infinite branch of T and (III) if S is Ex-learnable robustly relative to every infinite branch of T then S is already Ex-learnable without the help of any oracle.

(I): It is convenient to identify the levels of the tree with all triples (i,j,k) of natural numbers. The nodes of T are now those binary strings $\alpha \in \{0,1\}^*$ which satisfy the following two conditions:

- For each i, j there is at most one k with $\alpha(i,j,k)\downarrow = 1$.
- If $\varphi_i(j) = k$ within $|\alpha|$ steps and $(i,j,k) \in dom(\alpha)$ then $\alpha(i,j,k) = 1$.

The following program computes for every infinite branch A of T an identifier for REC:

$$I^A(\sigma) = \begin{cases} i & \text{for the least } i \leq |\sigma| \text{ such that} \\ & (i,j,\sigma(j)) \in A \text{ for all } j \in dom(\sigma); \\ |\sigma| & \text{if there is no such } i. \end{cases}$$

The verification is straightforward and based on the following two observations: First, for every $f \in$ REC, there is an i such that $(i,j,k) \in A \Leftrightarrow f(j) = k$. An example for such an i is the index of some program for f. Therefore the identifier converges on every computable function to some index. Second, if f and g are different computable functions, then there is a pair (j, k) such that $f(j) = k$ and $g(j) \neq k$. It follows that I^A converges on f to some index i with $(i, j, k) \in A$ and for g to some index i' with $(i', j, k) \notin A$. Thus I^A converges on every two different computable functions to different indices. So I^A is an identifier for REC. Note that an identifier for REC is also an identifier for every $S \subseteq$ REC.

(II): Let M witness that $S \in$ Ex[Identifier]. Then M works also with every I^A as an oracle. Now the learner N simulates M and just computes for every σ the value $I^A(\sigma)$ according to the algorithm given above, so N learns S from every oracle A which is an infinite branch of the tree T given above.

(III): This part needs only the recursiveness of T, not its special form. Furthermore, one can assume that whenever N makes a mind change, that is, whenever $N(\sigma a) \neq N(\sigma)$, then N takes an index which is larger then the input seen so far: $N(\sigma a) > |\sigma| + N(\sigma)$. Such an index can be found effectively by padding [111, Proposition II.1.6]. Furthermore, the convergence can be slowed down such that N queries only values below $|\sigma|$ in order to compute $N^A(\sigma)$. The basic idea of such a slow down is just to postpone a computation if it takes too long or queries too large data-items [43, Note 2.14]. All in all one obtains a monotonic increasing machine which queries for each σ only values below $|\sigma|$ and which converges for every $f \in S$ on all infinite branches to an index for f. Now the following algorithm H Ex-learns S even without oracle-queries — note that T is computable.

$$H(\sigma) = \min\{N^\alpha(\sigma) : \alpha \in \{0,1\}^{|\sigma|} \cap T\}.$$

Given some $f \in S$, let e be the minimal value such that there is an infinite branch A of T for which N^A with input f converges to e. This index e is a program for f and it remains to be shown that H also converges to e.

Since N^A is nondecreasing on f, the relation $H(\sigma) \leq e$ holds for all $\sigma \preceq f$. On the other hand, there is no infinite branch of T on which N converges to some value smaller than e and so the subtree

$$T' = \{\beta \in T : N^\beta(f(0)f(1)\ldots f(|\beta|)) < e\}$$

of T has no infinite branch. T' is finite by König's Lemma [111, Theorem V.5.23]. There is some string $\sigma \preceq f$ such that no string β of length $|\sigma|$ is in T'. Thus $N^\beta(\sigma) \geq e$ for all strings $\beta \in T$ of the same length as σ and $H(\sigma) \geq e$. Since H is increasing and does not take on f values greater than e, the learner H converges on f to e. So H is an Ex-learner for S, that is, $S \in$ Ex. ∎

So, the only interesting concept is NV[Identifier]. The next theorems show that many natural classes are also not learnable under this concept but there are some classes which can be NV-learned. NV[Identifier] and NV[Classifier] turn out to be incomparable but in some cases the combined concept is quite powerful.

Theorem 6.5.2 *If $S \in$ NV[Identifier] then some computable function g dominates every $f \in S$.*

Proof In the proof of Theorem 6.5.1 a computable binary tree T and an algorithm I is given such that I^A is an identifier for REC relative to every infinite branch A of T. By the Hyperimmune-Free Basis Theorem [111, Proposition V.5.34] T has an infinite branch A of hyperimmune-free degree. If $S \in$ NV[Identifier] then also $S \in$ NV[A] via some total A-recursive machine M. Along the lines of the proof of Theorem 6.3.2 it follows that some A-recursive function h dominates every function in S. By the definiton of hyperimmune-free degrees [111, Definition V.5.2] there is a computable function g which dominates this function h and with it also all functions in S. ∎

This result shows immidiately that NV[Classifier] is not contained in NV[Identifier]: By Theorem 6.4.3 every class containing S_0 is in NV[Classifier], in particular the whole class REC itself. Since REC is not dominated by a computable function, REC is not in NV[Identifier]. The reverse inclusion also does not hold. The next example shows this and also provides a nontrivial class in NV[Identifier].

Theorem 6.5.3 *The criteria* NV[Classifier] *and* NV[Identifier] *are incomparable.*

Proof As already mentioned, one noninclusion is already known. For the other one, let g be the function from Theorem 6.2.4 satisfying the following requirements:

- $range(\varphi_{g(i)}) = \{0,1\}$ and $0^i 1 \preceq \varphi_{g(i)}$;
- For all i there is at most one x with $\varphi_{g(i)}(x)\uparrow$;
- The class $S_6 = \{f \in \mathrm{REC}_{0,1} : f \text{ extends some } \varphi_{g(i)}\}$ is not Ex-learnable.

Here S_6 contains the class S_4 from Theorem 6.2.4, the main difference is that S_4 contains exactly one function extending each $\varphi_{g(i)}$ while S_6 contains every $\{0,1\}$-valued function extending $\varphi_{g(i)}$.

$S_6 \notin$ NV[Classifier]: This is done via showing that S_6 has already a computable classifier C. Then this C cannot provide any help for learning and the statement follows from $S_6 \notin$ NV — note that NV without oracle is less powerful than Ex. The computable classifier for S_6 is given as follows

$$C(\sigma) = \begin{cases} 1 & \text{if } \sigma \text{ and } \varphi_{g(i),|\sigma|} \text{ are consistent} \\ & \text{for the } i \text{ with } 0^i 1 \preceq \sigma; \\ 0 & \text{otherwise, that is, either no } 0^i 1 \preceq \sigma \text{ or} \\ & \sigma \text{ and } \varphi_{g(i),|\sigma|} \text{ are not consistent} \\ & \text{for the } i \text{ with } 0^i 1 \preceq \sigma; \end{cases}$$

where $\varphi_{g(i),|\sigma|}$ is the part of the function $\varphi_{g(i)}$ obtained within $|\sigma|$ computational steps and consistency means that, for all $x \in dom(\sigma)$, if $\varphi_{g(i)}(x)$ outputs within $|\sigma|$ computational steps a value y then $y = \sigma(x)$.

$S_6 \in$ NV[Identifier]: This result uses the computable classifier from above and the fact that every $f \in S_6$ is $\{0,1\}$-valued. Let I be an identifier for S_6. Now for any σ, consider the following trees

$$T_a = \{\tau \in \{0,1\}^* : (\forall \eta \preceq \tau)\,[\, I(\sigma a \eta) = I(\sigma) \wedge C(\sigma a \eta) = 1\,]\,\}.$$

If for example $\sigma 0 T_0$ has an infinite branch f then $f \in S$ since C outputs on f only finitely often a 0. Furthermore the tree $\sigma 1 T_1$ cannot have also an infinite branch g since then g would belong to S and I on both functions, f and g, converge to $I(\sigma)$. Thus one of the trees T_a is finite and the NV-learner M outputs that $1-a$ for that a where M finds out first that T_a is finite. If $f \in S$ then C outputs 0 only on finitely many $\sigma \preceq f$ and similarly I makes only finitely many mind changes, thus for almost all $\sigma a \preceq f$, the corresponding tree T_a is infinite and the prediction is the correct value a. ∎

So as a corollary of the last part of the proof one gets the following theorem that states that all classes of $\{0,1\}$-valued functions can be learned if both, an identifier and a classifier, are supplied.

Corollary 6.5.4 $S \in$ NV[Classifier,Identifier] *for all* $S \subseteq \mathrm{REC}_{0,1}$.

6.6 Martingales

A martingale calculates the gambling-account of someone who always tries to predict the next value of a function. In each round the gambler places an amount q on some number a, that is, for each string σ there is a rational number q, $0 \leq q < m(\sigma)$, such that $m(\sigma a) = m(\sigma) + q$ for some a and $m(\sigma b) = m(\sigma) - q$ for all $b \neq a$. The gambler wins on a function f iff the martingale takes on prefixes of f arbitrary large amounts of money. m is a martingale for S iff m wins on every function $f \in S$. The interested reader can find more on martingales in Schnorr's book [128].

Theorem 6.6.1 *If $S \in$ Ex[Martingale] then $S \in$ Ex. The same holds for all other inference criteria. In short: martingales do not help.*

Proof There is a martingale $m \leq_T A$ for some 1-generic set $A \leq_T K$ which wins on every recursive function — indeed every set A of hyperimmune degree is suitable. Let $g \leq_T A$ be a monotone function which is not dominated by any recursive function. Now the strategy of m is the following:

> Let σ be the input, $x = |\sigma|$ and $a = f(x)$ be the value to be predicted. Now look for the least $e \leq x$ such that $\varphi_e(y)$ converges to $\sigma(y)$ for $y = 0, 1, \ldots, x-1$ and $\varphi_e(x)$ also converges to some value a within $g(x)$ steps. If there are such an e and a then bet $q = \frac{m(\sigma)}{2}$ on a and otherwise do not bet ($q = 0$).

This martingale succeeds: Let e be the least index of f. g is not dominated by h where $h(x)$ is the time to compute all values $\varphi_e(0), \ldots, \varphi_e(x)$. There are even infinitely many x with $g(x) > h(3x + 3e)$. For these x, the martingale m bets for $y = x, x+1, \ldots, 3x+3e$ either on $\varphi_e(y)$ or on $\varphi_j(y)$ for some $j < e$. It happens for each $j < e$ at most once that m bets on $\varphi_j(y)$ and $\varphi_j(y) \neq \varphi_e(y)$, so this phenomenon produces in total at most e wrong bets. On the other hand, $\varphi_e(y)$ is computed within $g(x) \leq g(y)$ steps and so whenever m takes no value $\varphi_j(y)$ with $j < e$ then it predicts the value $\varphi_e(y)$. So at least $2x + 2e$ of the predictions between x and $3x + 3e$ are correct and $m(f(0)f(1)\ldots f(3x+3e)) \geq (\frac{9}{8})^{x+e}$. Since this holds for infinitely many x, m succeeds on g and so m succeeds on every recursive function.

If now $S \in$ Ex[Martingale] then S can also be learned via any oracle relative to which such a martingale exists. In particular S can be inferred relative to a low 1-generic oracle and thus S can be learned in the limit without any oracle [132]. So martingales do not help for learning in the limit. The same holds for learning under the criterion NV.

As in the case of predictors and classifiers, each finite part of any martingale can be extended to a martingale for S. The set of all such finite parts is enumerable and therefore the arguments from Theorem 6.1.2 and 6.3.1 can be used to show that martingales also do not help to learn under the criteria Fin, PFin and PEx. ∎

So martingales are on the bottom of the inclusion-structure of these five types of additional information as it is summarized in the following theorem. While unrelativized NV is much more restricted than unrelativized Ex, the opposite holds for many types of oracles: Most types of oracles can be exploited much better by an NV-learner than by an Ex-learner.

Theorem 6.6.2 *The inclusion-structure of the five types of additional information with respect to the learning criteria* Ex *and* NV *are given by the diagrams in Figure 1 on page 98. For the criteria* Fin, PFin *and* PEx *only lists provide some help while the other four types of additional information are trivial, that is, they do not increase the learning-power.*

Ex-learning

$$\begin{tikzcd}[column sep=small]
 & \text{List} & \\
\text{Predictor} \arrow[ur] & & \text{Classifier} \arrow[ul] \\
 & \text{Identifier} = \text{Martingale} = \text{Trivial} \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

NV-learning

$$\begin{tikzcd}[column sep=small]
 & \text{List} = \text{Predictor} = \text{Omniscient} & \\
\text{Identifier} \arrow[ur] & & \text{Classifier} \arrow[ul] \\
 & \text{Martingale} = \text{Trivial} \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

Figure 1: The Inclusion-Structure for Ex and NV.

7 Classification

Classification means to identify whether an object is contained in a class or not, these objects are in the present work subsets of the natural numbers. Classification is a concept common to two fields: (a) In recursion theory, classes can be defined via formulas describing whether a set belongs to a class or not. (b) In inductive inference, classification is just a natural generalization of a learning-process. The learner does not any longer infer a program for a set but decides only whether a set belongs to a class or not.

Here, two basic forms of classification are investigated. In the more restrictive setting of two-sided classification, the classifier converges to 1 on the data from members of the class and to 0 on the data from nonmembers of the class. In the more general setting of one-sided classification, the classifier converges on data from members of the class and diverges on data from nonmembers. These notions coincide with the basic recursion-theoretic concepts of Δ_2^0 and Σ_2^0 classes, respectively. These definitions reflect a similarity to the world of computable (Δ_0^0) and enumerable (Σ_0^1) sets. So there are many parallel definitions and phenomena, which are also investigated in the present work. This similarity has also its limitations, for example, there exists an infinite one-sided class not having an infinite two-sided subclass. This is due to the fact that besides the computational aspects known from the theory of sets, the theory of classes has also to deal with a aspects of a second type: many results are influenced by topological properties of the classes. Another example for this influence is that it is impossible to turn every one-sided class into a two-sided one with help of an oracle: For example, the class of all finite sets is in Σ_2^0 but not in Π_2^0 since it is not the intersection of topologically open classes [124, Theorem 15.IX].

For recursion theorists, picking Σ_2^0 and Δ_2^0 classes seems to be arbitrary and random, looking at Σ_3^0 versus Δ_3^0 classes could also be interesting. But from the viewpoint of learning theory, the Σ_2^0 and Δ_2^0 classes fit well into the until now studied concepts; choosing them has therefore some kind of naturalness.

Wiehagen and Smith [147] introduced a model, in which a finite number of classes is given and the classifier has to detect at least one class where the language to be classified belongs to. As in many other definitions in the field of inductive inference [19, 56], the process has only to converge in the limit and so the machine has the right to withdraw hypotheses and to replace them by new ones. Wiehagen and Smith [147] expected the process only to converge on the domain, so they avoided the topological problems which arize when the classifier has also to signal that a language does not belong to any class of the given collection.

Ben-David [17] and Kelly [78] started to investigate these topological aspects of classification: they showed that a (not necessarily computable) device can classify all sets with respect to one class in the limit iff the given class is the union of countably many closed classes and the intersection of countably many open classes at the same time; for this definition they use the Baire topology generated by the subbasis $\mathcal{A}_{x,y} = \{A : A(x) = y\}$ ($x \in \mathbb{N}, y \in \{0,1\}$). So their work is a direct motivation for the notion of two-sided classification since their model is just equivalent to relativized two-sided classification. Gasarch, Pleszkoch and Velauthapillai [53, 54] extended the result by establishing a close relation between the topological Borel hierarchy and the quantifier-hierarchy of query-languages during classification.

Minicozzi [106] introduced the notion of reliable inference: In this notion, given data of an arbitrary function f, the learner either converges to a correct program for f or diverges. This concept motivates one-sided classification since the learner converges exactly on the functions belonging to the class learned.

A further and early approach to classification was to design finite automata which decide in the limit whether an infinite string (representing the characteristic function of a language) belongs to a given ω-language or not [21, 100, 108, 144]. The restrictive computational ability of these finite automata led Büchi [21] and his successors to consider nondeterministic automata. The present approach takes the alternative way of choosing Turing machines as classifiers. In fact, already Büchi and Landweber [22, 93] did some first research into this direction.

7.1 The Basic Model of Classification

As already mentioned, Ben-David and Kelly already showed that a class is classifiable in the limit iff it is a (relativized) Δ_2^0 class; Rogers [124] called the Δ_2^0 classes also $\Delta_2^{(s)}$ classes. Reliable inference [106] can be generalized to the concept of one-sided learning classes since the learner signals membership by converging for the sets in the class and nonmembership by divergence. This is equivalent to the notion of Σ_2^0 classes or $\Sigma_2^{(s)}$ classes as Rogers [124] called them. That is, a machine H is a classifier for a class $\mathcal{A}$ iff H accepts every set in $\mathcal{A}$ by converging to 1 and rejects every set outside $\mathcal{A}$ by either diverging (one-sided classification) or converging to 0 (two-sided classification); the formal definition follows.

Definition 7.1.1 H is a one-sided classifier for $\mathcal{A}$ iff

$$\begin{aligned} & (\forall A \in \mathcal{A})\,(\forall^\infty \sigma \preceq A)\,[H(\sigma) = 1] \\ \text{and}\quad & (\forall A \notin \mathcal{A})\,(\exists^\infty \sigma \preceq A)\,[H(\sigma) = 0]; \end{aligned}$$

M is a two-sided classifier for $\mathcal{A}$ iff

$$\begin{aligned} & (\forall A \in \mathcal{A})\,(\forall^\infty \sigma \preceq A)\,[M(\sigma) = 1] \\ \text{and}\quad & (\forall A \notin \mathcal{A})\,(\forall^\infty \sigma \preceq A)\,[M(\sigma) = 0]. \end{aligned}$$

The relations between these two natural concepts of classification is the central topic of the present work. For each n, the concepts of Σ_n^0 sets relate to that of the Δ_n^0 almost in the same way as that of the enumerable ($= \Sigma_1^0$) sets to that of the computable ($= \Delta_1^0$) sets. This work shows that on one hand for one-sided versus two-sided classes this analogy basically also holds but that on the other hand the similarities are much more restricted than the parallel definitions suggest at the first glance.

Example 7.1.2 [124] The notions of effective topology follow the definitions as stated by Ko [82, p. 72, p. 165] and Rogers [124, §15.1]: A class $\mathcal{A}$ is recursively open iff there is a recursive sequence $\sigma_0, \sigma_1, \ldots$ of strings such that $A \in \mathcal{A} \Leftrightarrow (\exists n)\,[\sigma_n \preceq A]$. A class $\mathcal{A}$

is a recursively G_δ class iff there is a recursive array $\sigma_{m,n}$ of strings such that $A \in \mathcal{A} \Leftrightarrow (\exists m)\,(\forall n)\,[\sigma_{m,n} \not\preceq A]$.

Any recursively open class is two-sided and any recursively G_δ class is one-sided. There are recursively G_δ classes which are one-sided but not two-sided. An example for such a class is the class of all finite sets.

One important tool in recursion theory is that there is an acceptable numbering of all enumerable sets. Similarly there is an acceptable numbering of all one-sided classes given by total one-sided classifiers.

Fact 7.1.3 *There is an effective list $H_0, H_1, \ldots$ of one-sided classifiers such that every one-sided class is generated by such a classifier and every machine H_e is total. The so defined numbering of the one-sided classes is acceptable: every further effective numbering $\mathcal{G}_0, \mathcal{G}_1, \ldots$ can be represented via a computable function f as $\mathcal{H}_{f(e)} = \mathcal{G}_e$.*

This effective list of classifiers H_e is generated from an acceptable numbering φ_e of all partial computable functions which of course contains all classifiers. The formal definition is

$$H_e(\sigma) = \begin{cases} \varphi_e(\tau) & \text{for the longest } \tau \preceq \sigma \text{ such that} \\ & \varphi_e(\tau) \text{ outputs 0 or 1 within } |\sigma| \text{ steps;} \\ 0 & \text{if there is no such } \tau. \end{cases}$$

It is easy to verify that whenever φ_e is a one-sided classifier for $\mathcal{A}$, then so is H_e; and whenever φ_e is a two-sided classifier for $\mathcal{A}$, then so is H_e.

At many places this list H_e of one-sided classifiers will be quite useful; in particular it is much more handy to use the H_e in diagonalizations than the φ_e since the H_e are always total and $\{0,1\}$-valued. $\mathcal{H}_e$ denotes the one-sided class generated by H_e.

In the case of sets, the relations between Σ_n and Δ_n sets are — for $n > 1$ — practically the same as those between enumerable and recursive sets. In the case of classes, many but not all analogies from the world of enumerable versus recursive sets carry over. The result that Σ^0_n and Π^0_n classes are closed under union and intersection, that Δ^0_n classes are closed under all Boolean operations and that a class is Δ^0_n iff the class and its complement is Σ^0_n all carry over [124, Chapter 15]. But the statement that "every infinite enumerable set contains an infinite recursive subset" causes trouble in the world of classes. It is still true for the lowest level: every infinite Σ^0_1 class contains an infinite Δ^0_1 class. On the next level, it does not longer hold. The next theorem shows that there is an infinite one-sided ($= \Sigma^0_2$) classes which does not contain an infinite two-sided ($= \Delta^0_2$) class. Nevertheless one can still save the result, if "infinite" is replaced by the stronger requirement "uncountable".

Theorem 7.1.4 *Every uncountable one-sided class has a two-sided subclass of same cardinality. There is a one-sided infinite class which has no two-sided infinite subclass.*

Proof For the first statement, consider any uncountable one-sided class $\mathcal{A}$ and let H be its one-sided classifier. $\mathcal{A}$ is the ascending union of the Π^0_1 classes

$$\mathcal{A}_k = \{A \in \mathcal{A} : (\forall n \geq k)\,[H(A(0)A(1)\ldots A(n)) = 1]\,\}$$

and some $\mathcal{A}_k$ is not countable since $\mathcal{A}$ is not countable. This class $\mathcal{A}_k$ has then the cardinality of the continuum $\{0,1\}^\infty$. The same holds for the superclass $\mathcal{A}$. Since every Π^0_1 class is in Δ^0_2, $\mathcal{A}$ has the two-sided subclass $\mathcal{A}_k$ of the same cardinality.

The second statement is proven by constructing an infinite sequence $a_0, a_1, \ldots$ which is computable relative to K such that the class $\mathcal{A}$ of all A_n with $A_n = \{a_0, a_1, \ldots, a_n\}$

is one-sided but does not have an infinite two-sided subclass. The sequence starts with $a_0 = 0$ and one defines

$$a_{n+1} = 1 + \max\{b : (\exists e < n)\,[\varphi_e(a_0, a_1, \ldots, a_n)\!\downarrow = b]\}.$$

The sequence is obviously computable relative to K, so there is a uniformly recursive approximation $a_{n,0}, a_{n,1}, \ldots$ to the a_n. Now let

$$H(\sigma a) = \begin{cases} 1 & \text{if } a = 0 \text{ and there exists an } n \text{ with} \\ & \sigma(x)\!\downarrow = 1 \Leftrightarrow x \in \{a_{0,|\sigma|}, a_{1,|\sigma|}, \ldots, a_{n,|\sigma|}\}; \\ 0 & \text{otherwise.} \end{cases}$$

H outputs infinitely many often a 0 on every infinite set, so H accepts only finite sets and it is easy to verify that any of these finite sets has to be the set $A_n = \{a_0, a_1, \ldots, a_n\}$ for some n. So H is a one-sided classifier for $\mathcal{A}$.

Assume now by contradiction that M is a two-sided classifier for some infinite subclass of $\mathcal{A}$. Now there is a partial recursive function φ_e such that for each finite set D given as $\{b_0, b_1, \ldots, b_n\}$ the value $\varphi_e(b_0, b_1, \ldots, b_n)$ is the first $x > \max(D)$ such that $M(D(0)D(1)\ldots D(x)) = 1$ provided that this x exists. If M converges on A_n to 1 and $n > e$ then $\varphi_e(a_0, a_1, \ldots, a_n)$ exists and is properly below a_{n+1}. So M outputs an 1 between a_n and a_{n+1} on the characteristic function of A_n which up to that place also equals to the characteristic function of the infinite set $A = \{a_0, a_1, \ldots\}$. It follows that M outputs on A infinitely many 1s if M converges on infinitely many A_n to 1. So either M outputs on A infinitely many 1s or M converges only on finitely many A_n to 1. Therefore a two-sided classifier either accepts the set $A \notin \mathcal{A}$ or accepts only a finite subclass of $\mathcal{A}$. ▮

7.2 Classification and Turing Complexity

Post [120] asked whether there are enumerable sets which have neither the Turing degree $\mathbf{0}$ nor $\mathbf{0}'$. The work to solve this and similar problems initiated a large study of the Turing degrees of enumerable sets. Soare [135] gives a comprehensive overview on this work.

The analogous question for one-sided classes is to determine the amount of complexity which is necessary to compute a two-sided classifier for them. The straight forward implementation of this idea would be to identify each class with the easiest two-sided classifier for it — but such two-sided classifier sometimes do not exist and if they exist, there may be no one of least complexity.

Therefore classes are (in general) not related to a single Turing degree but to a collection of Turing degrees. This collection is called the Turing complexity of a class and consists of all oracles which allow to compute a two-sided classifier for the given class.

The one-sided classes are ordered in terms of their Turing complexity: So $\mathcal{A}$ has Turing complexity below that of $\mathcal{B}$ iff a two-sided classifier for $\mathcal{A}$ can be computed relative to every two-sided classifier for $\mathcal{B}$ considered as an oracle. With respect to this ordering, there are one-sided classes of least and greatest Turing complexity. Furthermore, among those of intermediate Turing complexity, there are some classes whose Turing complexity can be identified with a single Turing degree: Such a class has Turing degree $\mathbf{a}$ iff the Turing-degrees of the two-sided classifiers just form a cone above $\mathbf{a}$:

$$\{deg_T(M) : M \text{ is a two-sided classifier for } \mathcal{A}\} = \{\mathbf{b} : \mathbf{a} \leq \mathbf{b}\}.$$

So there are four types of one-sided classes with respect to their Turing complexity.

- A two-sided class has the least possible Turing complexity since it is two-sided relative to every oracle. $\emptyset$ and $\{0,1\}^\infty$ are examples of two-sided classes. Every two-sided class has a Turing degree, namely the degree $\mathbf{0}$ of the computable sets.

- There are one-sided classes which are not two-sided relative to any oracle. So they have the greatest possible Turing complexity: Example 7.2.6 gives three such one-sided classes.

- There is a one-sided class which has a nonrecursive Turing degree.

- There is a one-sided class which is two-sided relative to some oracles but which does not have a Turing degree.

The next results deal with the classes of intermediate Turing complexity according to the third and fourth case. Sacks [126, Section II.4] called a set A a Π^0_2 singleton iff A is the only set B which satisfies $(\forall x)(\exists y)[R(x,y,B)]$ for some recursive predicate R. Such Π^0_2 singletons allow to construct a one-sided classes $\mathcal{A}$ which has a nonrecursive Turing degree.

Example 7.2.1 *The cosingle class $\mathcal{A} = \{B : B \neq A\}$ has a Turing degree which is exactly that of A. Furthermore, $\mathcal{A}$ is one-sided iff A is a Π^0_2 singleton. So every hyperarithmetic set is below the Turing degree of some one-sided class.*

Proof Assume that M classifies two-sided $\mathcal{A}$. Then there is a finite string $\sigma \preceq A$ such that $M(\tau) = 0$ for all τ with $\sigma \preceq \tau \preceq A$. The binary tree

$$T = \{\tau : (\forall \eta \preceq \tau)\,[\eta \preceq \sigma \vee M(\eta) = 0]\}$$

is recursive in M and A is its only recursive branch. Therefore $M \geq_T A$. On the other hand, an A-oracle is obviously sufficient for two-sided classification since this means only to compare with A.

The second statement follows directly from the definition: The class $\mathcal{A}$ is one-sided iff it is Σ^0_2 iff its complement is Π^0_2; since this complement contains exactly the set A is it a Π^0_2 singleton iff it is Π^0_2.

The third statment follows from the fact that the hyperarithmetic sets are just the Turing closure downward of the Π^0_2 singletons [126, Section II.4]. ∎

The single one-sided classes $\{A\}$ are less complex than the cosingle ones: they are already two-sided without oracle and so have least Turing complexity. Some of the cocountable classes, which are a natural generalization of the cosingle classes, do not have a Turing degree, but they all are two-sided relative to some hyperarithmetic oracle. Example 7.2.2 gives an example of a class which does not have a Turing degree and whose Turing complexity is just the collection of all high Turing degrees. There are also cocountable examples with the same property. But for having a better readable proof, the easiest example is chosen.

Example 7.2.2 *There is a one-sided class $\mathcal{A}$ whose Turing complexity is the collection of all high Turing degrees. $\mathcal{A}$ does not have a Turing degree.*

Proof Let $\mathcal{A} = \{\{e\} : e \in K'\}$. The Limit-Lemma states that K' has a $\{0,1\}$-valued approximation $a(e,s)$ such that $a(e,s)$ is almost everywhere 1 iff $e \in K'$. Now $\mathcal{A}$ is one-sided via a classifier H which outputs $a(e,s)$ for the input 0^e10^s and 0 otherwise.

It is easy to see that for two-sided classifiers only the behaviour on sets with exactly one element e is interesting. A two-sided classifier converges on these sets $\{e\}$ to 1 iff $e \in K'$

and to 0 iff $e \notin K'$. So such a classifier exists in a Turing degree $\mathbf{a}$ iff K' is computable relative to $\mathbf{a}'$, that is, iff $\mathbf{a}$ is high.

The second statement of the theorem follows from the fact that the high Turing degrees do not form a cone. ∎

The next three theorems establish further results for classes of intermediate Turing complexity.

Theorem 7.2.3 *If a one-sided class is two-sided relative to some oracle, then it is two-sided relative to an oracle in Δ^1_2.*

Proof Let H be a computable one-sided classifier for $\mathcal{C}$. Assume furthermore that $\mathcal{C}$ is two-sided relative to some oracle. Now any, not necessarily recursive, two-sided classifier M satisfies the following Π^1_1 equation:

$$\begin{aligned}(\forall A)\ [\,(\,(\exists^\infty \sigma \preceq A)\,[H(\sigma)=0] \ \Rightarrow\ (\forall^\infty \sigma \preceq A)\,[M(\sigma)=0]\,)\ \wedge \\ (\,(\forall^\infty \sigma \preceq A)\,[H(\sigma)=1] \ \Rightarrow\ (\forall^\infty \sigma \preceq A)\,[M(\sigma)=1]\,)\,]\end{aligned}$$

It states that for all A, whenever H accepts or rejects A one-sidedly, so does M two-sidedly. It follows that M is a solution to the predicate iff M is a two-sided classifier for $\mathcal{C}$. So M is specified via a Π^1_1 predicate. Provided that this predicate has at least one solution, Addison and Kondo [126, Corollary 9.4] showed that there is a further Π^1_1 predicate $(\forall A)[P(M,A)]$ which has exactly one solution and whose solution N is also a solution to the original predicate. The sets $\{\sigma : N(\sigma)=c\}$ are in Σ^1_2 for $c=0,1$:

$$N(\sigma)=c \ \Leftrightarrow\ (\exists M)\,(\forall A)\,[M(\sigma)=c \wedge P(M,A)].$$

Since one set is the complement of the other, it follows that the machine N is in Δ^1_2. ∎

Theorem 7.2.4 *Let H be a one-sided classifier for $\mathcal{A}$ and for any $A \notin \mathcal{A}$ let $f_A(m)$ denote the first $k > m$ such that $H(A(0)A(1)\ldots A(k)) = 0$. Then $\mathcal{A}$ has a two-sided classifier of degree $\mathbf{a}$ iff there is a function g of degree $\mathbf{a}$ which dominates the functions f_A for all $A \notin \mathcal{A}$.*

Proof Assume that g dominates the functions f_A for all $A \notin \mathcal{A}$. The following $M \leq_T g$ is a two-sided classifier for $\mathcal{A}$:

$$M(A(0)A(1)\ldots A(n)) = \begin{cases} 1 & \text{if } g(m) < n \text{ for all } m \leq n \\ & \text{with } H(A(0)A(1)\ldots A(m)) = 0; \\ 0 & \text{otherwise.}\end{cases}$$

If $A \in \mathcal{A}$ then there are only finitely many m with $H(A(0)A(1)\ldots A(m)) = 0$. Almost all n are greater than $g(m)$ for each such m and thus $M(A(0)A(1)\ldots A(n)) = 1$ for almost all n. Otherwise $A \notin \mathcal{A}$ and g dominates f_A. So for almost all m there is a k with $m < k \leq g(m)$ and $H(A(0)A(1)\ldots A(k)) = 0$. So for almost all n, the greatest $m \leq n$ with $H(A(0)A(1)\ldots A(m)) = 0$ satisfies the condition $m \leq n \leq g(m)$ and it follows that $M(A(0)A(1)\ldots A(n)) = 0$ for these n. So M is a two-sided classifier for $\mathcal{A}$.

For the converse direction, let $M \leq_T E$ be a two-sided classifier for $\mathcal{A}$. The following function g is computable relative to E:

$$\begin{aligned} g(n) &= 2 + \max\{|\sigma| : \sigma \in T_n\} \text{ where} \\ T_n &= \{\sigma : (\forall \tau \preceq \sigma)\,[\,|\tau| \leq n \vee (M(\tau)=0 \wedge H(\tau)=1)\,]\,\}.\end{aligned}$$

Assume that g would be undefined for some n. Then T_n is infinite and by König's Lemma the tree T_n has an infinite branch A. M converges on A to 0 while H outputs on input A only finitely many 0s, that is, M and H classify A differently in contradiction to the choice of M and H. So T_n is finite and g is total. Since T_n is computable relative to E, its maximal string can be found using the oracle E and $g \leq_T E$.

Let $A \notin \mathcal{A}$. There is an n such that $M(A(0)A(1)\ldots A(m)) = 0$ for all $m \geq n$. Assume now by way of contradiction that $f_A(m) > g(m)$ for some $m \geq n$. Then $M(\sigma) = 0$ and $H(\sigma) = 1$ for all $\sigma \preceq A$ with $m < |\sigma| \leq g(m)$ and the string $A(0)A(1)\ldots A(g(m))$ is in T_m in contradiction to $g(m)$ being greater than the length of all strings in T_m. Thus such an m does not exist and $f_A(m) \leq g(m)$ for almost all m. ∎

A consequence of this is that every one-sided class, which is two-sided relative to a hyperimmune-free oracle, is already two-sided via a classifier without any access to oracles.

Theorem 7.2.5 *If $\mathcal{A}$ has a Turing degree then $\mathcal{A}$ has a hyperarithmetic Turing degree.*

Proof Assume that $\mathcal{A}$ has Turing degree $\mathbf{a}$ and let M be a two-sided classifier for $\mathcal{A}$ of degree $\mathbf{a}$. Theorem 7.2.4 implies that $\mathbf{a} \leq \mathbf{b}$ whenever every $\mathbf{a}$-recursive function is dominated by a $\mathbf{b}$-recursive function. There is a function f_0 dominating every function computable relative to $\mathbf{a}$ such that whenever g majorizes f_0 then $M \leq_T g$. First it has to be shown that this can be done via a single index, that is,

$$(\exists e, f)\,(\forall g \text{ majorizing } f)\,[M = \varphi_e^g]$$

and in a second step it is deduced that M has hyperarithmetic Turing degree. The existence of such e and f is shown via an algorithm which either provides the information to find e and f or which constructs a g majorizing f_0 such that $M \not\leq_T g$. In this proof $\sigma_0, \sigma_1, \ldots$ denote strings of numbers and not of bits. σ_0 is the empty string.

> Given σ_n and f_n check whether there is a f_{n+1} majorizing f_n and an extension $\sigma_{n+1} \succeq \sigma_n$ such that
>
> - $\sigma_n \prec \sigma_{n+1} \preceq f_{n+1}$ and
> - there is some $\eta \in \{0,1\}^*$ such that either $\varphi_n^{\sigma_{n+1}}(\eta)\downarrow \neq M(\eta)$ or $\varphi_n^g(\eta)\uparrow$ for all g majorizing f_{n+1} with $g \succeq \sigma_{n+1}$.
>
> If there are such σ_{n+1}, f_{n+1} the algorithm proceeds with them in the next step otherwise it terminates.

If the algorithm goes through all steps, then $g = \lim_n \sigma_n$ exists and majorizes f_0. By construction, for all e there is some $\eta \in \{0,1\}^*$ such that either $\varphi_e^g(\eta)\downarrow \neq M(\eta)$ (since $\varphi_e^{\sigma_{e+1}}(\eta)\downarrow \neq M(\eta)$ and $\sigma_{e+1} \preceq g$) or $\varphi_e^g(\eta)\uparrow$ (since g majorizes f_{e+1}). So M is not computed relative to g via any e in contradiction to the choice of f_0.

Thus the algorithm terminates in some stage n. Now for each g the following set is not empty:

$$F(g,\eta) = \{\tau \succeq \sigma_n : \varphi_n^\tau(\eta)\downarrow \wedge (\forall m \in dom(\tau) - dom(\sigma_n))\,[\tau(m) \geq g(m)]\}.$$

Furthermore, whenever g majorizes f_n and $\tau \in F(g,\eta)$ then $\varphi_n^\tau(\eta)\downarrow = M(\eta)$. Using these two facts it is possible to construct e:

If within $|\eta|$ steps no triple (θ, τ_1, τ_2) has been enumerated witnessing inconsistency in the way that $\tau_1, \tau_2 \in F(g, \theta)$ and $\varphi_n^{\tau_1}(\theta) \neq \varphi_n^{\tau_2}(\theta)$
then $\varphi_e^g(\eta) = \varphi_n^\tau(\eta)$ for the first τ found in $F(g, \tau)$
else $\varphi_e^g(\eta)\uparrow$.

So $M = \varphi_e^g$ for all g majorizing f_n and φ_e^g is partial if $M \neq \varphi_e^g$. The second step is now easy. The sets $M_c = \{\eta : M(\eta) = c\}$ are Π_1^1 according to the following definition:

$$M(\eta) = c \;\Leftrightarrow\; (\forall g)\,[\varphi_e^g \text{ is total } \Rightarrow \varphi_e^g(\eta) = c].$$

Since M_0 is the complement of M_1, both sets are in Δ_1^1 and M is hyperarithmetic. ∎

Example 7.2.6 *The following classes are one-sided but not relatively two-sided:*
(a) $\mathcal{A} = \{A : A \textit{ is cofinite}\}$ [53, 124].
(b) $\mathcal{B} = \{B : B \textit{ is primitive recursive}\}$ [124].
(c) $\mathcal{C} = \{C \oplus D : D \neq C'\}$.

Proof (a): The one-sided classifier H outputs a on input σa. It takes almost always the value 1 iff the input is a cofinite set. On the other hand this class is not relatively two-sided since for any function g there is a set $A \notin \mathcal{A}$ such that g does not dominate f_A. Namely for given g, this set is $A = \mathbb{N} - \{x_0, x_1, \ldots\}$ where $x_0 = g(0)+1$ and $x_{n+1} = g(x_n) + x_n + 1$. From the definition of H it follows that $f_A(x_n) = x_{n+1} = g(x_n) + x_n + 1 > g(x_n)$ and g does not dominate f_A.

(b): Rogers [124, §15.1] showed that dense classes with a uniform enumeration are one-sided but not two-sided relative to any oracle, so this proof covers also part (a): Let $B_0, B_1, \ldots$ be a uniform enumeration, in this case of all primitive recursive sets. Now on input σa, the one-sided machine looks for the first k with $\sigma \preceq B_k$. If then also $\sigma a \preceq B_k$ it outputs 1 otherwise it outputs 0. It is easy to see that the machine outputs 1 on almost all inputs $\sigma \preceq B$ iff $B = B_k$ for some k. The other part of the proof uses that no countable and dense class like $\mathcal{B}$ is the intersection of countably many open sets and thus not a relativized Π_2^0 class [124, Theorem 15.IX(b)].

(c): For each set C there is a uniform approximation of C' via strings γ_n^C of C such that each γ_n^C queries C only at the places $0, 1, \ldots, n$, $|\gamma_n^C| \leq n$ and $\gamma_n^C \preceq C'$ infinitely often. Now the one-sided machine H outputs on strings of odd length an 1 and processes strings of even length as follows:

$$H(C(0)D(0)C(1)D(1)\ldots C(n)D(n)) = \begin{cases} 0 & \text{if } \gamma_n^C \preceq D(0)D(1)\ldots D(n); \\ 1 & \text{otherwise.} \end{cases}$$

As mentioned, γ_n^C can be computed using only the $C(m)$ with $m \leq n$, thus the whole procedure needs no oracle but retrieves the answers from the input. If $D \neq C'$ then $\gamma_n^C \not\preceq D$ for almost all n; therefore H accepts all sets in $\mathcal{C}$. If $D = C'$ then there are infinitely many n with $\gamma_n^C \preceq D(0)D(1)\ldots D(n)$. H takes 0 at these n and thus rejects all sets outside $\mathcal{C}$. So H is a one-sided classifier for $\mathcal{C}$.

Assume now by way of contradiction that $\mathcal{C}$ is two-sided via M. Now M can also be viewed as a set and so one can consider the class $\{C \oplus D : C \neq M \vee D \neq M'\}$. A machine to identify this set can be derived from M as follows:

$$N(\sigma) = \begin{cases} M(\sigma) & \text{if } \sigma(x) = M(\frac{x}{2}) \text{ for all even } x \in dom(\sigma); \\ 1 & \text{otherwise;} \end{cases}$$

N is obviously recursive in M. On the other hand, the new class is cosingle and contains all sets except $M \oplus M'$. Then $M \oplus M'$ has to be recursive in M, a contradiction. ∎

Singleton one-sided classes are always two-sided and therefore less complex than cosingleton ones which can have a Turing degree above every given hyperarithmetic Turing degree. The relation between countable and cocountable one-sided classes is the other way round: While some countable one-sided classes can have the highest possible Turing complexity this is not true for cocountable classes.

Theorem 7.2.7 *Any cocountable one-sided class* $\mathcal{A} = \{B : (\forall n)\,[B \neq A_n]\}$ *is two-sided relative to some hyperarithmetic set.*

Proof Let H be a one-sided classifier for $\mathcal{A}$. Sacks [126, Theorem III.6.2] showed that every Σ^1_1 class either contains a perfect subclass (and is just uncountable) or has only members below some hyperarithmetic set. Since each one-sided class is defined without quantification over sets or functions, its complement is a Σ^1_1 class (indeed it is even a Δ^1_1 class). So there is a hyperarithmetic set C such that $A_n \leq_T C$ for all n. Now for each n the function f_{A_n} as defined in Theorem 7.2.4 is recursive in A_n. Some function $g \leq_T C'$ dominates all functions computable relative to C, in particular g dominates each function f_{A_n}. By Theorem 7.2.4 the class $\mathcal{A}$ is two-sided relative to C' which has hyperarithmetic Turing degree since the hyperarithmetic Turing degrees are closed under the jump. ∎

7.3 Complete Classes

There are some other reducibilities between sets besides Turing reduction. Post [120] introduced the concept of 1-reduction: A set A is 1-reducible to B iff there is a one-one computable function f such that $x \in A \Leftrightarrow f(x) \in B$. The set K is complete within the enumerable sets, that is, every enumerable set can be 1-reduced to K.

It is possible to transfer the notion of 1-reduction to the world of classification. Here a 1-reduction from a class $\mathcal{A}$ to a class $\mathcal{B}$ is a one-one computable and continuous operator Γ translating every set A into a set $\Gamma(A)$ such that $A \in \mathcal{A} \Leftrightarrow \Gamma(A) \in \mathcal{B}$.

Definition 7.3.1 A computable operator Γ is called a 1-reduction from $\mathcal{A}$ to $\mathcal{B}$ if

- Γ is strictly monotone (with respect to $\preceq$), that is, $\Gamma(\sigma) \preceq \Gamma(\tau)$ iff $\sigma \preceq \tau$ for all $\sigma, \tau \in \{0,1\}^*$.
- $A \in \mathcal{A}$ iff $\Gamma(A) = \lim_{\sigma \preceq A} \Gamma(\sigma) \in \mathcal{B}$ for all sets A.

A class $\mathcal{A}$ is called 1-complete iff every one-sided class is 1-reducible to it and $\mathcal{A}$ itself is one-sided.

It is easy to see that if $\mathcal{A} \leq_1 \mathcal{B}$ and $\mathcal{B}$ is two-sided via a (nonrecursive) machine M then $\mathcal{A}$ is also two-sided via a classifier computable relative to M. Since there are classes which are not relatively two-sided, the following 1-complete class is not relatively two-sided and does not have a Turing degree.

Theorem 7.3.2 *The class* $\mathcal{K} = \{A : (\forall^\infty \textit{even } x)\,[x \in A]\}$ *is 1-complete.*

Proof The classifier

$$H(a_0 a_1 \dots a_n) = \begin{cases} 1 & \text{if } n \text{ is odd;} \\ a_n & \text{if } n \text{ is even;} \end{cases}$$

witnesses that $\mathcal{K}$ is one-sided. Assume now that L is a computable one-sided classifier for a further class $\mathcal{A}$. A 1-reduction Γ from $\mathcal{A}$ to $\mathcal{K}$ is defined as follows:

$$\begin{aligned}\Gamma(\lambda) &= L(\lambda);\\ \Gamma(\sigma a) &= \Gamma(\sigma) a L(\sigma a).\end{aligned}$$

From this equation it follows that $H(\Gamma(\sigma)) = L(\sigma)$ and that whenever $\eta \preceq \Gamma(A)$ and $H(\eta) = 0$ then $\eta = \Gamma(\sigma)$ for some $\sigma \preceq A$. The equivalence

$$(\exists^\infty \sigma \preceq A)\,[L(\sigma) = 0] \;\Leftrightarrow\; (\exists^\infty \eta \preceq \Gamma(A))\,[H(\eta) = 0]$$

gives that $A \in \mathcal{A}$ iff $\Gamma(A) \in \mathcal{K}$ and $\mathcal{A}$ is 1-reducible to $\mathcal{K}$. So $\mathcal{K}$ is 1-complete. ▮

Theorem 7.3.3 *The class $\mathcal{A}$ of all cofinite sets has greatest Turing complexity but is not 1-complete.*

Proof By Example 7.2.6 (a), the class $\mathcal{A}$ of all cofinite sets has greatest Turing complexity. But $\mathcal{A}$ is not 1-complete: Consider the full class $\{0,1\}^\infty$ of all sets. If $\{0,1\}^\infty$ is 1-reducible to $\mathcal{A}$ via Γ then every infinite branch of the tree $\Gamma(\{0,1\}^*)$ would be contained in $\mathcal{A}$. This contradicts the fact that $\mathcal{A}$ contains only countably many sets. ▮

Post [120] showed that simple sets are not complete under various constructions. Indeed it is possible to define something analogous to simple set and to show that it is not 1-complete: A one-sided class is called simple iff it intersects every other infinite one-sided class.

Theorem 7.3.4 *No 1-complete class is simple.*

Proof Let $\mathcal{A}$ be a 1-complete class. Then the class $\mathcal{C}$ of all cofinite sets is 1-reducible to $\mathcal{A}$ via some reduction Γ. Now it is shown that $\mathcal{A}$ is not simple via showing that the class $\{\Gamma(A_0), \Gamma(A_1), \ldots\}$ is a two-sided infinite class disjoint to $\mathcal{A}$ where $A_x = \{x\}$. No set $A_x = \{x\}$ is contained in $\mathcal{C}$. Thus also no set $\Gamma(A_x)$ is in $\mathcal{A}$ and $\mathcal{B} = \{\Gamma(A_0), \Gamma(A_1), \ldots\}$ is an infinite class disjoint to $\mathcal{A}$. It remains to be shown that $\mathcal{B}$ is one-sided; indeed it will be shown that the following machine M is a two-sided classifier for $\mathcal{B}$.

$$M(\sigma) = \begin{cases} 1 & \text{if there is an } x \text{ with } \Gamma(0^x 1) \preceq \sigma \preceq \Gamma(0^x 1 0^\infty);\\ 0 & \text{otherwise.}\end{cases}$$

The check whether such an x exists, is computable: Only the $x \leq |\sigma|$ have to be considered since the string $\Gamma(0^x 1)$ is longer than σ for $x > |\sigma|$. Furthermore, the sets $\Gamma(A_x)$ are uniformly recursive, so the whole check and thus M is a computable procedure.

During the classification of any set A, M makes only two mind changes: from the initial guess 0 to 1 if it turns out that $\Gamma(0^x 1) \preceq A$ for some x. A further mind change back to 0 if A turns out to be different from $\Gamma(A_x)$. Since no string $\Gamma(0^y 1)$ with $y \neq x$ extends $\Gamma(0^x 1)$, there is no danger of a third mind change from 0 to 1 because of such a $\Gamma(0^y 1)$ being a prefix of A.

It is now easy to verify is that M converges on the sets $\Gamma(A_x)$ to 1 and on all other sets to 0; so M is a classifier for $\mathcal{B}$.

In particular $\mathcal{B}$ is a one-sided infinite class which is disjoint to $\mathcal{A}$ and thus witnesses that $\mathcal{A}$ is not simple. ▮

7.4 Index Sets of One-Sided Classes

Let $\mathcal{G}$ be a collection of classes and $\mathcal{H}_e$ denote the class generated by the e-th one-sided classifier H_e. Then the set $E = \{e : \mathcal{H}_e \in \mathcal{G}\}$ is called the index set of $\mathcal{G}$ and every such set E belonging to a collection of classes is called an index set. So this section deals with the analogue of the index sets of classes of enumerable sets; while index-sets of sets are mostly situated in the arithmetical hierarchy, index sets of one-sided classes have often the complexity Π^1_1. The first example of such an index set is the equality problem $\{\langle e, e'\rangle : \mathcal{H}_e = \mathcal{H}_{e'}\}$.

Theorem 7.4.1 *The set* $\{\langle e, e'\rangle : \mathcal{H}_e = \mathcal{H}_{e'}\}$ *is* Π^1_1 *complete.*

Proof The formula

$$(\forall A)\,[\,(\exists^\infty \sigma \preceq A)\,[H_e(\sigma) = 0] \;\Leftrightarrow\; (\exists^\infty \sigma \preceq A)\,[H_{e'}(\sigma) = 0]\,]$$

witnesses that equality is in Π^1_1. Fixing e' to be an index of $\{0,1\}^\infty$ the next Theorem 7.4.2 witnesses that the set is also Π^1_1 hard. ∎

Theorem 7.4.2 *The sets* $I = \{e : \mathcal{H}_e$ *is two-sided*$\}$ *and* $J = \{e : \mathcal{H}_e = \{0,1\}^\infty\}$ *are* Π^1_1 *complete.*

Proof The set $J = \{e : (\forall A)\,(\forall^\infty \sigma \preceq A)\,[H_e(\sigma) = 1]\,\}$ is in Π^1_1. Furthermore, $e \in I$ iff there is an index e' such that $\mathcal{H}_e = \mathcal{H}_{e'}$ and $H_{e'}$ makes on any A only finitely many mind changes, that is,

$$\begin{aligned} e \in I \quad \Leftrightarrow \quad (\exists e') \quad & [\,(\forall A)\,[(\exists^\infty \sigma \preceq A)\,[H_e(\sigma) = 0] \;\Leftrightarrow\; (\exists^\infty \sigma \preceq A)\,[H_{e'}(\sigma) = 0]\,] \;\wedge \\ & (\forall A)\,(\exists c)\,(\forall^\infty \sigma \preceq A)\,[H_{e'}(\sigma) = c]\,] \end{aligned}$$

So it follows that also I is in Π^1_1.

Now it is shown that both sets are complete via the same m-reduction f. Let $T_0, T_1, \ldots \subseteq \mathbb{N}^*$ be a computable enumeration of all primitive recursive trees. The set

$$E = \{e : T_e \text{ is well-founded}\}$$

is Π^1_1 complete [126] where a tree is well-founded iff it does not contain an infinite branch. A string σ is said to code a finite branch $a_0 a_1 \ldots a_n$ iff there are $b_0, b_1, \ldots, b_n \in \mathbb{N}$ such that $\sigma = 1^{\langle a_0, b_0\rangle} 0 1^{\langle a_1, b_1\rangle} 0 \ldots 1^{\langle a_n, b_n\rangle} 0$. E is m-reducible to both index sets via the following reduction:

$$H_{f(e)}(\sigma) = \begin{cases} 0 & \text{if } \sigma \text{ codes a finite branch of } T_e \text{, that is,} \\ & \text{if } \sigma = 1^{\langle a_0, b_0\rangle} 0 1^{\langle a_1, b_1\rangle} 0 \ldots 1^{\langle a_n, b_n\rangle} 0 \text{ and } a_0 a_1 \ldots a_n \in T_e; \\ 1 & \text{otherwise.} \end{cases}$$

It follows that $H_{f(e)}$ outputs infinitely many 0 on some set A iff $A = 1^{\langle a_0, b_0\rangle} 0 1^{\langle a_1, b_1\rangle} 0 \ldots$ for an infinite branch $a_0 a_1 \ldots$ of T_e.

In the case that T_e is well-founded, $H_{f(e)}$ outputs on every set A only finitely many 0s and therefore $\mathcal{H}_{f(e)} = \{0,1\}^\infty$. This class is two-sided, so $f(e) \in I$ and $f(e) \in J$.

Otherwise T_e has an infinite branch $a_0 a_1 \ldots$ and for any sequence $b_0 b_1 \ldots$ the set A with the characteristic function $1^{\langle a_0, b_0\rangle} 0 1^{\langle a_1, b_1\rangle} 0 \ldots$ is not in $\mathcal{H}_{f(e)}$. It follows that $\mathcal{H}_{f(e)} \neq \{0,1\}^\infty$ and $f(e) \notin J$. Furthermore, for each function g, the sequence $b_0 b_1 \ldots$ can be chosen such that $\langle a_{n+1}, b_{n+1}\rangle \geq b_{n+1} > g(c_n)$ where $c_n = n + \langle a_0, b_0\rangle + \langle a_1, b_1\rangle + \ldots + \langle a_n, b_n\rangle$. It follows that $f_A(c_n) > g(c_n)$ for each n and that g does not dominate f_A. So there is no function

g dominating the functions f_A for all $A \notin \mathcal{H}_{f(e)}$. By Theorem 7.2.4, $\mathcal{H}_{f(e)}$ is not two-sided — not even relative to any oracle — and $f(e) \notin I$. ▮

Theorem 7.4.2 has an immediate application: it shows that there is nothing equivalent to a Friedberg numbering. If all one-sided classes would have a Friedberg numbering, then there would be also a numbering where one class, namely $\{0,1\}^\infty$, is omitted. But such a numbering does not exist.

Theorem 7.4.3 *No numbering contains all one-sided classes except* $\{0,1\}^\infty$.

Proof Assume by way of contradiction that there is a computable function f such that the numbering $\mathcal{H}_{f(0)}, \mathcal{H}_{f(1)}, \ldots$ covers all one-sided classes except $\{0,1\}^\infty$. Then the set $\{e : (\exists e')\,[\mathcal{H}_e = \mathcal{H}_{f(e')}]\}$ is in Π^1_1 since Π^1_1 is closed under quantification on numbers as e' and since the equality problem is in Π^1_1. For any given e such an e' exists iff $\mathcal{H}_e \neq \{0,1\}^\infty$. So the complement of this Π^1_1 set is the Π^1_1 complete index set of $\{0,1\}^\infty$ and such a recursive function f cannot exist. ▮

Theorem 7.4.4 *The sets* $\{e : \mathcal{H}_e \leq_1 \mathcal{A}\}$ *and* $\{e : \mathcal{H}_e \equiv_1 \mathcal{A}\}$ *are in* Π^1_1 *for every one-sided class* $\mathcal{A}$. *In particular* $\{e : \mathcal{H}_e$ *is 1-complete*$\}$ *is in* Π^1_1.

Proof The proofs are very similar. There is an enumeration of all operators Γ_i such that whenever Γ_i is total then it is strictly monotone. Furthermore, there is a one-sided classifier H for $\mathcal{A}$. Now

$$\begin{array}{rcl} \mathcal{H}_e \leq_1 \mathcal{A} & \Leftrightarrow & (\exists i)\,(\forall A)\,[\,\Gamma_i \text{ is total } \wedge \\ & & ((\forall^\infty n)\,[H_e(A(0)A(1)\ldots A(n)) = 1] \Leftrightarrow \\ & & \quad (\forall^\infty n)\,[H(\Gamma_i(A(0)A(1)\ldots A(n))) = 1]\,)\,] \\ \mathcal{H}_e \equiv_1 \mathcal{A} & \Leftrightarrow & (\exists i,j)\,(\forall A)\,[\,\Gamma_i \text{ and } \Gamma_j \text{ are total } \wedge \\ & & ((\forall^\infty n)\,[H_e(A(0)A(1)\ldots A(n)) = 1] \Leftrightarrow \\ & & \quad (\forall^\infty n)\,[H(\Gamma_i(A(0)A(1)\ldots A(n))) = 1]\,) \;\wedge \\ & & ((\forall^\infty n)\,[H(A(0)A(1)\ldots A(n)) = 1] \Leftrightarrow \\ & & \quad (\forall^\infty n)\,[H_e(\Gamma_j(A(0)A(1)\ldots A(n))) = 1]\,)\,] \end{array}$$

Since the existential quantifier ranges over numbers, these expressions are Π^1_1. They characterize the two index sets. ▮

These classes are not Π^1_1 complete for every $\mathcal{A}$. In particular if $\mathcal{A} = \emptyset$ then they are in Π^0_2: $\mathcal{H}_e \equiv_1 \emptyset$ iff $\mathcal{H}_e = \emptyset$ iff for each n there is an m such that every string $\sigma \in \{0,1\}^m$ has at least n prefixes $\tau \preceq \sigma$ with $H_e(\tau) = 0$. The difference in the complexity of the question whether $\mathcal{H}_e$ is empty or equals $\{0,1\}^\infty$ is the mirror image of the fact that the question whether $W_e = \emptyset$ is Π^0_1 complete while the question whether $W_e = \mathbb{N}$ is Π^0_2 complete.

Furthermore, it can be shown that the index set $\{e : \mathcal{H}_e = \emptyset\}$ has the least complexity of an index set of classes. Rice [121] showed for the world of enumerable sets that every nontrivial index set is Π^0_1 hard or Σ^0_1 hard. In the world of one-sided classes it can be shown that every nontrivial index set E is Π^0_2 hard or Σ^0_2 hard. In particular it is shown that the Σ^0_2 complete set F is m-reducible to E or $\overline{E}$.

Theorem 7.4.5 *Let E be a nontrivial index set of some collection $\mathcal{G}$ of classes. Then the set $F = \{e : W_e$ is finite$\}$ is m-reducible either to E or to $\overline{E}$.*

Proof First consider the case $\{0,1\}^\infty \in \mathcal{G}$. In this case it is shown that $F \leq_m E$ via a m-reduction f. This f then witnesses that E has at least complexity Π^0_2. Since E is not trivial there is some one-sided class $\mathcal{A} \notin \mathcal{G}$ with some computable one-sided classifier H. Now f is defined implicitly via giving an informal description for the classifier $H_{f(e)}$:

$H_{f(e)}$ outputs on A at least n 0s iff $|W_e| \geq n$ and H outputs on A at least n 0s.

If W_e is finite, $H_{f(e)}$ outputs on every A only finitely often a 0 and thus accepts every set; so $\mathcal{H}_{f(e)} = \{0,1\}^\infty$ and $f(e) \in E$ for every $e \in F$. If W_e is infinite then $H_{f(e)}$ accepts a set A iff H does; so $\mathcal{H}_{f(e)} = \mathcal{A}$ and $f(e) \notin E$ for every $e \notin F$. It follows that J is m-reducible to E via f.

The other case that $\mathcal{G}$ does not contain the class $\{0,1\}^\infty$ just gives an m-reduction from F to $\overline{E}$ using the above proof with $\overline{E}$ in place of E and $\{\mathcal{A} : \mathcal{A} \notin \mathcal{G}\}$ in place of $\mathcal{G}$. ∎

7.5 Classification and Measure

The measure ν given by $\nu(\emptyset) = 0$, $\nu(\{0\}) = \nu(\{1\}) = 0.5$, $\nu(\{0,1\}) = 1$ has an infinite product μ on the space $\{0,1\}^\infty$. This can be extended in such a way that every subclass of a class with measure 0 is again measurable and has measure 0. Lusin [126, Lemma II.6.2] showed that any Π^1_1 class and therefore also every one-sided class is measurable. So the main question is how effective the measure of the classes are. For two-sided classes, one can compute the measure in the limit.

Theorem 7.5.1 *The measure $\mu(\mathcal{A})$ of a two-sided class can be computed from any index e of a two-sided classifier H_e for $\mathcal{A}$.*

Proof For each set A there is a unique n such that H_e converges at $A(0)A(1)\ldots A(n)$ either to 1 or to 0. Now let $\mathcal{A}_e = \{A : H_e$ converges to 1 at $n\}$ and $\mathcal{B}_e = \{A : H_e$ converges to 0 at $n\}$. Since H_e converges on every set A either to 1 or to 0, these classes $\mathcal{A}_0, \mathcal{A}_1, \ldots$ and $\mathcal{B}_0, \mathcal{B}_1, \ldots$ form a partition of $\{0,1\}^\infty$. In particular $\mu(\mathcal{A}_0)+\mu(\mathcal{B}_0)+\mu(\mathcal{A}_1)+\mu(\mathcal{B}_1)+\ldots = 1$. Now consider the computable sequence

$$q_n = 2^{-1-n} \cdot \sum{}_{a_0,a_1,\ldots,a_n \in \{0,1\}} H_e(a_0a_1\ldots a_n)$$

of rational numbers. This sequence converges to $\mu(\mathcal{A})$ since $\mu(\mathcal{A}_0)+\mu(\mathcal{A}_1)+\ldots+\mu(\mathcal{A}_n) \leq q_n \leq 1 - (\mu(\mathcal{B}_0) - \mu(\mathcal{B}_1) - \ldots - \mu(\mathcal{B}_n))$ and therefore $|\mu(\mathcal{A}) - q_n| \leq \epsilon_n$ where $\epsilon_n = \mu(\mathcal{A}_{n+1}) + \mu(\mathcal{B}_{n+1}) + \mu(\mathcal{A}_{n+2}) + \mu(\mathcal{B}_{n+2}) + \ldots$; the ϵ_n converge to 0 since the sum $\mu(\mathcal{A}_0) + \mu(\mathcal{B}_0) + \mu(\mathcal{A}_1) + \mu(\mathcal{B}_1) + \ldots + \mu(\mathcal{A}_n) + \mu(\mathcal{B}_n)$ approaches monotonically to 1. Thus the sequence of the q_n converges to $\mu(\mathcal{A})$ and so the measure of $\mathcal{A}$ can be computed in the limit from any two-sided classifier for $\mathcal{A}$. ∎

This is not longer true for one-sided classes. The class of all sets A which are lexicographic before K' has the measure $2^{-1}K'(0) + 2^{-2}K'(1) + \ldots$ which can not be computed in the limit since otherwise K' would be computable in the limit. So one might ask whether the measure is at least in those cases computable where the measure of the class is a recursive real. But also this fails since one can construct a function f with $\mathcal{H}_{f(e)} = \{0,1\}^\infty$ if W_e is finite and $\mathcal{H}_{f(e)} = \emptyset$ if W_e is infinite.

For the further analysis, the notion of recursive measure 0 and 1 is introduced which can be defined using martingales [42, 95, 128]. A class has recursive measure 0 if a recursive martingale succeeds on all its members and effective measure 1 if a recursive martingale succeeds on all its nonmembers. A martingale is a function m which associates to every $\sigma \in \{0,1\}^*$ a rational number such that:

- $m(\sigma 0) + m(\sigma 1) = 2m(\sigma)$;
- $m(\sigma) > 0$ and $m(\lambda) = 1$.

A martingale witnesses that *a class $\mathcal{A}$ has measure* 0 iff for each $A \in \mathcal{A}$ and for each k there is an n such that $m(A(0)A(1)\ldots A(n)) \geq k$. It witnesses that a *class has measure* 1 iff it witnesses that the complement of this class has measure 0. A class has *recursive measure* 0 or 1 iff some recursive martingale witnesses that it has measure 0 or 1, respectively.

Easier than computing such a measure is to verify that a class has measure 0 or 1 via presenting a recursive martingale which either succeeds on the class or on its complement. But — as the next example shows — also this fails for certain one-sided classes with measure 1: there is just no such martingale.

Example 7.5.2 *There is a cosingle one-sided class $\mathcal{A}$ which does not have recursive measure* 1.

Proof Every set $A \leq_T K$ is a π_2^0 singleton. Therefore it follows that for every $A \leq_T K$ the class $\mathcal{A} = \{B : B \neq A\}$ is one-sided and it remains to be shown that there is some $A \not\leq_T K$ such that no recursive martingale succeeds on A. This is just the well-known fact that there is a random-set $A \leq_T K$. ▮

It is easier to determine that a one-sided class is small than that it is large. As already seen the index set of the empty class is much easier than that of the full class $\{0,1\}^\infty$. This result has a parallel with respect to measure. While some cosingle one-sided classes do not have measure 1, every one-sided class of measure 0 does also have recursive measure 0.

Theorem 7.5.3 *If a one-sided class $\mathcal{A}$ has measure* 0 *then $\mathcal{A}$ has recursive measure* 0.

Proof Let $\mathcal{A}$ be one-sided via a machine H and have measure 0. Now a recursive martingale m is constructed in order to witness that $\mathcal{A}$ has recursive measure 0. The inductive definition starts with $m(\lambda) = 0$.

Now in each step chose (one of) the shortest σ such that $m(\sigma)$ is defined but not $m(\sigma 0)$ and $m(\sigma 1)$. Let L_n be the set of all $\tau \in \{0,1\}^n$ such that $H(\sigma\eta) = 1$ for all nonempty $\eta \preceq \tau$. If for each n the cardinality of L_n would be larger than 2^{n-2} then H would output on a "quarter" of all sets $A \succeq \sigma$ never a 0 after processing σ which implies $\mu(\mathcal{A}) \geq 2^{-|\sigma|-2}$ in contradiction to the choice of $\mathcal{A}$. Thus there is an $n > 0$ such that L_n has at most 2^{n-2} elements, without loss of generality, n is the smallest such number.

Now let $m(\sigma\tau) = 1.5 \cdot m(\sigma)$ for every $\tau \in L_n$ and let $m(\sigma\tau) = \frac{2^n - 1.5 \cdot |L_n|}{2^n - |L_n|} \cdot m(\sigma)$ for the other strings τ of length n. All values $m(\sigma\tau)$ are above 0 and their sum is $2^n \cdot m(\sigma)$. Furthermore, define $m(\sigma\eta)$ for strings η of length $n-1, n-2, \ldots, 0$ according to the formula $m(\eta) = \frac{1}{2} \cdot (m(\eta 0) + m(\eta 1))$. This finishes the extension step.

Each such step finishes and since each step takes the shortest σ with $m(\sigma 0), m(\sigma 1)$ being undefined, m becomes a total function. Furthermore, all values of m are positive rational numbers and it can be verified that the equation $m(\sigma 0) + m(\sigma 1) = 2 \cdot m(\sigma)$ holds for all σ. Thus m is a recursive martingale.

Let now $A \in \mathcal{A}$. Let $\sigma_0 \preceq \sigma_1 \preceq \ldots \preceq A$ being that sequence of strings such that σ_{n+1} is always a string of the form $\sigma_n\tau$ when m is extended at σ_n. By the construction the following holds:

$$\begin{aligned} m(\sigma_{n+1}) < m(\sigma_n) &\Leftrightarrow H(\eta) = 0 \text{ for some } \eta \text{ with } \sigma_n \prec \eta \preceq \sigma_{n+1} \\ m(\sigma_{n+1}) = 1.5 \cdot m(\sigma_n) &\Leftrightarrow H(\eta) = 1 \text{ for all } \eta \text{ with } \sigma_n \prec \eta \preceq \sigma_{n+1} \end{aligned}$$

Since H outputs on A almost always 1s, the second case holds for almost all n and it follows that m takes on A arbitrary large values. So $\mathcal{A}$ has recursive measure 0 witnessed by the recursive martingale m. ▮

Example 7.5.2 showed already that one-sided classes of measure 1 do not need to have recursive measure 1. So it is natural to look for the help of oracles and the next result states, that a K-oracle is sufficient to do the job: If a class has measure 1 then it has already K-recursive measure 1, that is, a K-recursive martingale witnesses that the class has measure 1.

Theorem 7.5.4 *If a one-sided class $\mathcal{A}$ has measure 1 then $\mathcal{A}$ has K-recursive measure 1.*

Proof For given one-sided class $\mathcal{A}$ with measure 1 a K-recursive martingale m is constructed which succeeds on every $A \notin \mathcal{A}$ and so witnesses that $\mathcal{A}$ has K-recursive measure 1. Let H be a one-sided classifier for $\mathcal{A}$. Let μ denote the standard measure on $\{0,1\}^\infty$ and for any computable tree T let

$$\nu(T,\eta) = \mu(\{A \succeq \eta : A \text{ is infinite branch on } T\}).$$

Starting with $m(\lambda) = 1$, the inductive definition of m runs as follows:

- Choose the shortest σ such that $m(\sigma 0)\uparrow$ and $m(\sigma 1)\uparrow$.
  Indeed the domain of m will be a tree at each stage and by extending the domain on some shortest leaf, it is guaranteed that m is total at the end. This σ can be found by bookkeeping on the previously defined places.

- Let $T_n = \{\tau : |\{m \in dom(\tau) : H(\tau(0)\tau(1)\ldots\tau(m)) = 0\}| \leq n\}$. Find using the oracle K a suitable n such that $\nu(T_n,\sigma) > 2^{-|\sigma|-1}$.
  Such an n exists since the union of all T_n contains almost all branches through σ and so $\nu(\sigma,T_n)$ must approach to $2^{-|\sigma|}$ by the continuity of μ. Furthermore,

  $$\nu(\sigma,T_n) = 2^{-|\sigma|} - \sum_{\tau \text{ is leaf of } T_n} 2^{-|\tau|}$$

  is computable via K-oracle and thus a suitable n can be found.

- Define m on all nodes of T_n above σ such that $m(\tau) \geq 1.5 \cdot m(\sigma)$ for every leaf τ of T_n above σ.
  The definition $m'(\tau) = m(\sigma) \cdot (0.95 \cdot 2^{|\tau|-|\sigma|} \cdot \nu(\tau,T_n)/\nu(\sigma,T_n) + 0.05)$ satisfies all requirements but has the disadvantage of not giving rational numbers. But m' can be approximated by an extension of m onto T_n above σ such that m has on this extended domain the same computational complexity as m', takes only rational values and satisfies $0.9 \cdot m'(\eta) \leq m(\eta) \leq 1.1 \cdot m'(\eta)$ for all η. Since even $m'(\tau) \geq 1.9 \cdot m(\sigma)$ for all leaves of T_n, it follows that $m(\tau) \geq 0.9 \cdot 1.9 \cdot m(\sigma) \geq 1.5 \cdot m(\sigma)$ for these τ.

At each stage of the definition, a set A stays on the corresponding tree T_n only if H outputs a finite number of 0s on input A. Thus starting with $\sigma_0 = \lambda$, a given $A \notin \mathcal{A}$ leaves this tree through a leaf σ_1 and when m is extended on a tree above σ_1 then A leaves this tree through a leaf σ_2 and so on. So A goes through an infinite sequence $\sigma_0, \sigma_1, \ldots$ of nodes such that $m(\sigma_{k+1}) \geq 1.5 \cdot m(\sigma_k)$ for all these nodes. It follows that m takes on A arbitrary high values and so m witnesses that $\overline{\mathcal{A}}$ has K-recursive measure 0. ▮

Let I be an interval of real numbers. It follows from the definition of the Lebesgue measure, that every measurable set $E \subseteq I$ is approximable via a F_σ set F in the sense that the symmetric difference of E and F has measure 0. Lusin [118, Satz 8.2] showed a function $f : I \to I$ is measurable iff for each $\epsilon > 0$ there is a set D of measure less than ϵ such that the restriction of f to the domain $I - D$ is continuous. These results motivate to look at the question to which extent from the view-point of measure theory, a one-sided class can be approximated by a two-sided one.

Theorem 7.5.5 *For every one-sided class $\mathcal{A}$ and every $\epsilon > 0$ there is a two-sided class $\mathcal{B}$ such that the symmetric difference of both classes has a measure less than ϵ. But there is also a one-sided class $\mathcal{A}$ such that every two-sided class differs from $\mathcal{A}$ on a set of positive measure.*

Proof Recall that every one-sided class is measurable. For given one-sided class $\mathcal{A}$ and $\epsilon > 0$ consider the classes $\mathcal{A}_k$ containing all sets A on which H outputs at most k times a 0. These classes are two-sided and they approximate $\mathcal{A}$ from below. Since the measure is continuous, $\mu(\mathcal{A})$ is the upper limit of the $\mu(A_k)$ and so $\mu(\mathcal{A}) - \epsilon < \mu(\mathcal{A}_k) \leq \mu(\mathcal{A})$ for some k. Since $\mathcal{A}_k \subseteq \mathcal{A}$, the symmetric difference has the measure $\mu(\mathcal{A}) - \mu(\mathcal{A}_k)$ which is less than ϵ.

For the second result let G be 1-generic and below K. Consider the class $\mathcal{A} = \{A : A <_{lex} G\}$. This class is one-sided via outputting 1 if $\sigma <_{lex} G_{|\sigma|}$ and 0 otherwise where G_s is a recursive approximation to G. Now let $\mathcal{B}$ be any two-sided class with classifier M. M converges on G to some value a, with only finitely many changes one can obtain that $M(\sigma) = a$ for all $\sigma \preceq M$. Thus G avoids the computable set $\{\tau : M(\tau) \neq a\}$ and so there is some prefix σ such that $M(\tau) = a$ for all $\tau \succeq \sigma$. On the other hand there are $\eta_0, \eta_1 \succeq \sigma$ such that all sets extending η_0 belong to $\mathcal{A}$ and all sets extending η_1 belong to $\overline{\mathcal{A}}$, in particular all sets $A \succeq \eta_{1-a}$ are in the symmetric difference of $\mathcal{A}$ and $\mathcal{B}$ so that this symmetric difference has a measure larger than $2^{|\eta_{1-a}|} > 0$. ∎

There are two natural properties, one-sided classes can take: being simple and maximal. The first one was already introduced above: a simple class is one-sided and intersects every infinite one-sided class. The second one is the following: A one-sided class $\mathcal{A}$ is maximal if it is coinfinite and has the property that either $\mathcal{A} \cup \mathcal{B}$ or $\mathcal{A} \cup \overline{\mathcal{B}}$ is cofinite for every one-sided class $\mathcal{B}$. The easiest way to construct a maximal class is just to convert a maximal set into it: Let U be a set which is maximal relative to K, that is, U is enumerable relative to K, coinfinite and no further set which is enumerable relative to K can split the complement of U into two infinite parts. The class $\{A : |A| \neq 1 \vee (A = \{x\} \wedge x \in U)\}$ is maximal. So maximal classes exist. Simple and maximal classes are not only large in the sense that they intersect every infinite one-sided class. They are also large with respect to measure theory.

Theorem 7.5.6 *If $\mathcal{A}$ is simple then $\mu(\mathcal{A}) > 0$. If $\mathcal{A}$ is maximal then $\mu(\mathcal{A}) = 1$.*

Proof First it is shown that no one-sided class $\mathcal{A}$ of measure 0 is simple. So let $\mathcal{A}$ be a one-sided class of measure 0. $\mathcal{A}$ then also has recursive measure 0 and there is a computable martingale m witnessing this fact. The class $\mathcal{B}_c = \{A : (\exists n)\,[m(A(0)A(1)\ldots A(n)) > c]\}$ of all sets on which m obtains some value greater than c has measure at most $\frac{1}{c}$. So the class $\overline{\mathcal{B}_2}$ has at least measure $\frac{1}{2}$ and is therefore infinite. $\overline{\mathcal{B}_2}$ is disjoint to $\mathcal{A}$ since the martingale succeeds on every set in $\mathcal{A}$. Furthermore, $\overline{\mathcal{B}_2}$ is one-sided via guessing 1 on input A as long as $m(A(0)A(1)\ldots A(n)) \leq 2$ and then making a mind change to 0. So the infinite one-sided class $\overline{\mathcal{B}_2}$ is disjoint to $\mathcal{A}$ and $\mathcal{A}$ is not simple.

Second it is shown that $\mu(A) = 1$ for all maximal classes $\mathcal{A}$. Given a maximal class $\mathcal{A}$ some kind of "kernel" B of $\mathcal{A}$ is constructed as follows: for each n one of the classes

$\{A \notin \mathcal{A} : A(n) = 0\}$ and $\{A \notin \mathcal{A} : A(n) = 1\}$ is finite and the other one is infinite; so let $B(n)$ take that value b for which $\{A : A(n) = b\}$ is infinite. It follows that for every n the class $\{A \notin \mathcal{A} : A(n) \neq B(n)\}$ is finite and thus their union is countable. So $\overline{\mathcal{A}}$ has at most countably many members: those just mentioned plus perhaps B itself. Therefore the complement of any maximal class is countable and so every maximal class has measure 1. ∎

Theorem 7.5.6 has two limitations: first it is only claimed that maximal classes have measure 1 but not that they have recursive measure 1. Indeed this is not possible since by Example 7.5.2 there is a cosingle class $\mathcal{A}$ not having a recursive measure 1 and taking any maximal class $\mathcal{B}$, the new class $\mathcal{A} \cap \mathcal{B}$ does also not have recursive measure 1 but is still maximal.

The second restriction is that there are simple classes with measure below 1 as will be proven below. Indeed this tradeoff between the size of maximal and simple classes has an analogy in recursion theory given by the fact that a simple set can be arbitrary thin — there is for every given computable function f a simple set which has only n elements among the numbers $0, 1, \ldots, f(n)$ — while no similar result holds for maximal sets.

Example 7.5.7 *For each $\epsilon > 0$ there is a simple class $\mathcal{A}$ with $\mu(\mathcal{A}) < \epsilon$.*

Proof Each string σ generates an open class $\sigma \cdot \{0,1\}^\infty$ of sets. This open class is said to meet the one-sided class $\mathcal{H}_e$ generated by H_e effectively iff there is a set $A \in \sigma \cdot \{0,1\}^\infty$ such that $H_e(A(0)A(1)\ldots A(n)) = 1$ for all $n \geq |\sigma|$. This condition is coenumerable, that is, for each class $\mathcal{H}_e$ given by H_e the set

$$N_e = \{\sigma : \sigma \cdot \{0,1\}^\infty \text{ does not meet } \mathcal{H}_e \text{ effectively}\}$$

is enumerable. Furthermore, almost all sets $A(0)A(1)\ldots A(n) \cdot \{0,1\}^\infty$ meet $\mathcal{H}_e$ effectively whenever $A \in \mathcal{H}_e$. There is an algorithm which generates a three-dimensional array $\sigma_{e,k,s}$ of strings such that

- the length of each string $\sigma_{e,k,s}$ is at least k;
- if $\mathcal{H}_e \neq \emptyset$ then $\sigma_{e,k} = \lim_{s\to\infty} \sigma_{e,k,s}$ exists;
- if $\sigma_{e,k}$ exists then the open class $\sigma_{e,k} \cdot \{0,1\}^\infty$ meets $\mathcal{H}_e$ effectively.

This algorithm works after a simple schema: $\sigma_{e,k,s}$ is just the first string (with respect to some given ennumeration of all strings) whose length is at least k and which is not enumerated to N_e within s computation steps. So this algorithm converges to some $\sigma_{e,k}$ if $\mathcal{H}_e \neq \emptyset$ and diverges otherwise. Let E denote the index set of all nonempty classes $\mathcal{H}_e$, that is, the set of all e where the sequences $\sigma_{e,k,0}, \sigma_{e,k,1}, \ldots$ converge.

Given ϵ there is a number n such that $2^{1-n} < \epsilon$. Now $\mathcal{A}$ is taken to be the union of all classes $\sigma_{e,n+e} \cdot \{0,1\}^\infty$ with $e \in E$. The measure of $\mathcal{A}$ is bounded by $\mu(\mathcal{A}) \leq \Sigma_{e\in E}\, 2^{-e-n} \leq 2^{1-n} < \epsilon$ and so the requirement on the measure of $\mathcal{A}$ is satisfied. It remains to show that $\mathcal{A}$ is one-sided. A one-sided classifier for $\mathcal{A}$ is given as follows:

> H outputs on A at least n 0s iff there is $s \geq n$ such that
> $\sigma_{e,k,s} \neq A(0)A(1)\ldots A(m)$ for all $m \leq n$ and $e, k \leq n$.

If $A \in \mathcal{A}$ then some $\sigma_{e,k,s}$ converges to a prefix of A. So there are m, t such that $\sigma_{e,k,s} = A(0)A(1)\ldots A(m)$ for all $s \geq t$. It follows that H does not output more than $e + m + t$ 0s and H accepts A.

If $A \notin \mathcal{A}$ then for each n there is a stage s such that all strings $A(0)A(1)\ldots A(m)$ with

$m \leq n$ are enumerated to all N_e with $e \leq n$. It follows that all $\sigma_{e,k,s}$ with $e, k \leq n$ are different from all $A(0)A(1)\ldots A(m)$ with $m \leq n$ and H outputs on A eventually at least n 0s. Since this holds for each n, H rejects A.

So H is a one-sided classifier for $\mathcal{A}$. By construction, $\mathcal{A}$ meets every nonempty set $\mathcal{H}_e$, so $\mathcal{A}$ is a simple class. Furthermore, $\mu(\mathcal{A}) < \epsilon$ and so $\mathcal{A}$ satisfies all conditions of the theorem. ▮

A related notion to measure is that of the category. A class has first category or is meager iff it is the countable union of nowhere dense classes. Here a class $\mathcal{A}$ is dense if for every string σ there is a $A \in \mathcal{A}$ with $\sigma \preceq A$ and is nowhere dense iff for every σ there is an extension $\tau \succeq \sigma$ such that no $A \succeq \tau$ is an element of $\mathcal{A}$.

Mehlhorn [101] introduced an effective notion of meagerness. He called a class $\mathcal{A}$ *effectively meager* iff there is a uniformly recursive family $f_0, f_1, \ldots$ of function such that $f_n(\sigma) \succeq \sigma$ for every n and σ and such that for each $A \in \mathcal{A}$ there is a function f_n with $f_n(A(0)A(1)\ldots A(m)) \not\preceq A$ for all m.

Now one can show that for one-sided classes there are many natural relations between these notions.

Theorem 7.5.8 *Let $\mathcal{A}$ be a one-sided class. Then $\mathcal{A}$ is meager iff $\mathcal{A}$ is effectively meager iff the complement $\overline{\mathcal{A}}$ is dense. Furthermore, if $\mathcal{A}$ has measure* 0 *then $\mathcal{A}$ is meager but the converse does not hold.*

Proof If $\mathcal{A}$ is meager then no class $\sigma \cdot \{0,1\}^\infty$ is contained in $\mathcal{A}$ and thus the complement of $\mathcal{A}$ is dense. For the second transition let $\overline{\mathcal{A}}$ be dense and let M witness, that $\mathcal{A}$ is one-sided. Then for each n there is a function f_n which searches for given σ an extension $\tau \succeq \sigma$ such that $M(\eta) = 0$ for at least n prefixes $\eta \preceq \tau$. The f_n are total since $\overline{\mathcal{A}}$ is dense and so every σ has an infinite extension $A \succeq \sigma$ on which M outputs infinitely many 0s. Furthermore, the f_n are uniformly recursive, that is, the mapping $n, \sigma \to f_n(\sigma)$ is computable in both parameters. If $A \in \mathcal{A}$ then M outputs on A only finitely often, say n times, a 0 and so $f_{n+1}(A(0)A(1)\ldots A(m))$ is for no m a prefix of A. It follows that $\mathcal{A}$ is effectively meager. The third transition from effective meagerness to meagerness follows directly from the definition.

For the second statement of the theorem, the implication holds since no class of measure 0 contains a subclass $\sigma \cdot \{0,1\}^\infty$ and so every class of measure 0 has a dense complement. The properness of this relation can be obtained by considering the following two-sided class:

$$A \in \mathcal{A} \Leftrightarrow (\forall n)\,[A \not\succeq \sigma_n 0^{n+2}]$$

where $\sigma_0, \sigma_1, \ldots$ is an enumeration of all strings. The class $\mathcal{A}$ has a dense complement and so is meager, it is even nowhere dense. But its measure satisfies $\mu(\mathcal{A}) \geq 1 - \Sigma_n 2^{-n-2-|\sigma_n|} \geq \frac{1}{2}$ and so $\mathcal{A}$ does not have measure 0. ▮

7.6 Classifying Recursive Sets Only

Smith, Wiehagen and Zeugmann [134, 147] looked at classification tasks where only the behaviour on computable sets is considered. Case, Kinber, Sharma and Stephan [30] extended this work. They obtained results which are much more similar to the world of enumerable versus recursive sets — a reason for this is that topological constraints are much less important and that the whole world REC has only countably many objects which can be identified with their finite descriptions (programs) although one cannot find them without a high oracle. For classifying recursive sets only, Case, Kinber, Sharma and Stephan [30] obtained the following results:

- Every one-sided classifier H can be transformed into a two-sided one with the help of a high oracle — by inferring the program e of the function and then computing in the limit whether the set $\{x : H(\varphi_e(0)\varphi_e(1)\ldots\varphi_e(x)) = 0\}$ is finite or infinite.

- A class in Ex is one-sided iff it is in REx; the result holds also using LimEx instead of Ex where again a reliable LimEx learner diverges on every function not inferred. The corresponding result in the general framework would be that a class in Ex (which then is a subset of REC) is in R_{all}Ex iff it is one-sided.

- For every one-sided class $\mathcal{A}$, either $\mathcal{A}$ or its complement contain an infinite class $\mathcal{B}$ which is a uniformly recursive (or indexed) family. That is, $\mathcal{B}$ is of the form $\mathcal{B} = \{B_0, B_1, \ldots\}$ where the mapping $i, x \to B_i(x)$ is recursive in both parameters. But there is a two-sided infinite class $\mathcal{A}$ which does not have such a subclass.

Theorem 7.1.4 showed that some infinite one-sided class has no infinite two-sided subclass. It is natural to ask whether this also holds for the model of classifying recursive sets only, but this is until now unknown. Further research [30, Section 4] is dedicated to the study of classification from positive data. In this context it was again possible to construct an infinite one-sided class which does not have an infinite two-sided subclass, namely the class containing exactly the sets $\{0, 1, \ldots, a\}$ where $a \in \mathbb{N}$. A further interesting result is, that if $\mathcal{A}$ and $\mathcal{B}$ are two-sided classifiable from text, then $\mathcal{A} = \mathcal{B}$ iff $\mathcal{A}$ and $\mathcal{B}$ contain exactly the same finite sets.

This section now deals with the relation of the general model where all sets are classified versus the restricted model where only computable sets are classified. The next theorem shows that there is a class which is two-sided in the restricted model but does not have any two-sided classifier relative to any oracle in the general model.

Theorem 7.6.1 *There is a class $\mathcal{A}$ of computable sets such that some computable M classifies all computable sets with respect to $\mathcal{A}$ but there is not even a nonrecursive classifier which converges on every input-set and classifies all computable sets with respect to $\mathcal{A}$.*

Proof Let S be a simple set and $\mathcal{A} = \{\text{finite } A : A \cap S = \emptyset \wedge |A| \text{ is odd}\}$. First it is shown that some M classifies S one-sided where M converges on every computable set. This M is given by

$$M(\sigma) = \begin{cases} 0 & \text{if } \{x : \sigma(x){\downarrow} = 1\} \text{ meets } S_{|\sigma|} \text{ or has even cardinality;} \\ 1 & \text{otherwise.} \end{cases}$$

That means that M outputs 0 or 1 depending on the cardinality of the 1s in σ until M discovers that some x with $\sigma(x) \downarrow = 1$ is enumerated into S — then M switches to 0 forever.

Now it is shown that M converges on every computable set. If A is finite then M changes only finitely often its mind — either when M finds a new element or when some already found element is enumerated into S. If A is computable and infinite then M also makes only finitely often a mind change since M eventually discovers that there is an $x \in A \cap S$ and from this time on only outputs 0.

The second part is to show that every classifier N which is correct on all finite sets with respect to $\mathcal{A}$ diverges on some infinite set $A = \{a_0, a_1, \ldots\}$ where this set is defined inductively starting with $n = 0$ and some $a_0 \notin S$.

> Take some $\sigma_n \preceq \{a_0, a_1, \ldots, a_n\}$ such that $|\sigma_n| > a_n$ and $N(\sigma_n) = 1$ iff $n+1$ is odd. Then take some $a_{n+1} \notin S \cup dom(\sigma_n)$.

The σ_n is found since N classifies $\{a_0, a_1, \ldots, a_n\}$ with respect to $\mathcal{A}$. The a_{n+1} exists since S is coinfinite and $dom(\sigma_n)$ is finite. So the construction works for all n and the resulting set A is infinite and disjoint to S. Furthermore, $N(\sigma_n) = 0$ for all even n, $N(\sigma_n) = 1$ for all odd n and the σ_n are all different prefixes of A, so N does not converge on A. ∎

So sometimes two-sided classifiers for computable sets cannot be extended to two-sided classifiers for all sets. But as the next theorem shows, they can be extended to one-sided classifiers for all sets such that the corresponding class has measure 1.

Theorem 7.6.2 *Every one-sided class $\mathcal{A}$ has a one-sided "extension" $\mathcal{B}$ of measure 1 such that $A \in \mathcal{A} \Leftrightarrow A \in \mathcal{B}$ for every computable set A.*

Proof Let H be a one-sided classifier for $\mathcal{A}$. Now a new one-sided classifier N is constructed such that the class $\mathcal{B}$ defined via N has the desired properties. N just slows down the output of the 0s and meets the following definition:

> N outputs on A at least n 0s if H outputs on A at least n 0s and there is $m \geq n$ such that $\varphi_m(x)\downarrow = A(x)$ for all $x \leq m$.

So whenever A is recursive, A has infinitely many indices and in particular for each n there is an index $m \geq n$ of A. This m satisfies of course $\varphi_m(x)\downarrow = A(x)$ for all $x \leq m$. Therefore M outputs on A infinitely many 0s if H does and N classifies A to be in $\mathcal{B}$ iff H classifies A to be in $\mathcal{A}$. So $\mathcal{A}$ and $\mathcal{B}$ coincide on the computable sets.

Let $\mathcal{A}_m = \{A : (\forall x \leq m)\, [\varphi_m(x)\downarrow = A(x)]\}$. Each class $\mathcal{A}_m$ has measure 2^{-m-1} if φ_m is defined on the input $0, 1, \ldots, m$ and is empty otherwise. So whenever N outputs on a set A at least n 0s, then A belongs to some $\mathcal{A}_m$ with $m \geq n$. Since $\mu(\mathcal{A}_n \cup \mathcal{A}_{n+1} \cup \ldots) \leq \mu(\mathcal{A}_n) + \mu(\mathcal{A}_{n+1}) + \ldots \leq 2^{-n-1} + 2^{-n-2} + \ldots = 2^{-n}$, it follows that the measure of the class of all A on which N outputs at least n 0s has the upper bound 2^{-n}. Thus the measure of $\overline{\mathcal{B}}$ which is the class of all sets on which N outputs infinitely many 0s is 0 since it is bounded by each number 2^{-n}. It follows that $\mathcal{B}$ has measure 1. ∎

A similar Theorem does not hold with measure 0 in place of measure 1. Taking the one-sided class $\mathcal{A} = \{0,1\}^\infty$ of all sets, every one-sided class $\mathcal{B}$ which agrees with $\mathcal{A}$ on all computable sets just has to contain every computable set. The measure of $\mathcal{B}$ cannot be 0 since then $\mathcal{B}$ had recursive measure 0 and so there would be a recursive martingale succeeding on all computable sets - which does not exist.

8 Inside Truth-Table Degrees

A reducibility between two sets D and E is called strong, if it is a restriction of truth-table reducibility and is called weak otherwise. For example, at weak truth-table reducibility it is impossible to know for all illegal oracle-answers, what the result of the reduction would be while for the strong version, truth-table reducibility itself, it is possible to compute for each theoretically possible outcome of the oracle answers a value which coincides with the result of the reduction if the correct oracle answers are supplied. This chapter deals mainly with strong reducibilities, so first an overview on the most important of them is given.

The main goal is to compare Post's notion of truth-table reducibility [120] with two more restrictive variants: the bounded truth-table reducibility (btt) and the positive reducibility. The first variant uses only a fixed number of queries, the second variant a positive formula to evaluate the queries. A given tt-reduction is positive iff for all sets D_1, D_2, E_1, E_2 such that $D_1 \leq_{tt} E_1$ and $D_2 \leq_{tt} E_2$ via this same tt-reduction the implication $E_1 \subseteq E_2 \Rightarrow D_1 \subseteq D_2$ holds.

It follows directly from the definition that btt-degrees and positive degrees are subsets of tt-degrees. So the more interesting question is the reverse direction: how many btt-degrees and positive degrees are in every tt-degree? For the recursive tt-degree the answer depends somehow on the definition of positive degrees: the normal definition gives one but if a real dependence on the input data is requested then $\{\emptyset\}$ and $\{\mathbb{N}\}$ are two further positive degrees so that there are three positive degrees inside the recursive tt-degree. On nonrecursive sets the definition of positive reducibility is more robust and gives the same relationship for both ways to define it.

For btt-degrees, Kobzev [85] showed that every nonrecursive enumerable tt-degree consists of at least two btt-degrees. Dëgtev [36] extended the result by showing that there are at least two btt-degrees inside every nonrecursive tt-degree. These two btt-degrees are given by a semirecursive set and the tt-cylinder. Beigel, Gasarch, Gill and Owings [14] found an alternative short proof for this fact.

Within this chapter it is shown that every nonrecursive tt-degree contains infinitely many btt-degrees so that Dëgtev's lower bound is improved from 2 to ∞. This is done in two steps: first the result is shown for all hyperimmune tt-degrees (that is, for all tt-degrees inside hyperimmune Turing degrees) and second the same is shown for all degrees not enumerable relative to K which covers the case of hyperimmune-free nonrecursive tt-degrees.

Further investigations deal with the structure of the btt-degrees inside nonrecursive tt-degrees. It is shown that these btt-degrees form infinite chains and antichains. These antichains witness an affirmative answer to the following question of Jockusch [70]: does every nonrecursive tt-degree contain an antichain of m-degrees? The tt-cylinder represents the greatest btt-degree among all btt-degrees inside a given tt-degree. So it is natural to ask whether there is also a least one. Kobzev [84] has already constructed a minimal btt-degree inside some tt-degree. Within the present work, this result is extended by showing that on one hand some of these minimal degrees are not a least one inside their tt-degree and that on the other hand some tt-degrees have a least btt-degree. The construction of these tt-degrees having a least btt-degree is used to answer a question of Beigel, Gasarch and Owings [15]: there exists a Turing degree such that, for every set in this degree, arbitrary large portions of its characteristic function can be computed with only two nondeterministic queries to this set.

Jockusch [69] showed that every nonrecursive tt-degree consists of at least three positive degrees. It is shown that this result is optimal in the sense that some nonrecursive tt-degrees consist of exactly three positive degrees. These tt-degrees can even be chosen such that they fulfill additional requirements such as being hyperimmune-free or enumerable. The so constructed enumerable tt-degrees have some nice properties: one of their positive degrees consists of semirecursive enumerable sets, the second one of semirecursive coenumerable sets and the third degree of sets which are neither enumerable nor coenumerable nor semirecursive. So in these tt-degrees the notions "semirecursive" and "enumerable" coincide (modulo complementation). Furthermore, there is no tt-degree consisting of four or any other finite even number of positive degrees, so that the first next possible cardinality to come is five. The next known cardinality is nineteen, so it is open whether some tt-degrees consist of five, seven, nine, eleven, thirteen, fifteen or seventeen positive degrees.

The results can be extended in the way that there are infinitely many odd numbers which are the cardinality of the positive degrees inside some suitable tt-degree.

Also it is shown that many of the results transfer to the weak versions of the considered reducibilities. For example, every nonrecursive weak truth-table degree consists of infinitely many wbtt-degrees. Since the strong and weak versions coincide for hyperimmune-free sets, some wtt-degree consists of exactly three weak positive degrees. The results for enumerable wtt-degrees are different from those for enumerable tt-degrees: Every enumerable wtt-degree contains exactly one enumerable weak positive degree but infinitely many further weak positive degrees.

For certain other strong and weak reducibilities these questions are already solved. Jockusch [69] showed that some positive degrees consist of exactly one many-one degree (namely those of semirecursive sets) and others consist of infinitely many many-one degrees: Such an example is the nonsemirecursive positive degree in some tt-degrees which consist of three positive degrees. Every tt-degree contains infinitely many many-one degrees [70] and taking away the two semirecursive positive degrees consisting of one many-one degree the remaining one contains infinitely many of them. Furthermore, Dëgtev [37] showed for Bulitko's notion of linear reducibility [23] that some linear degrees are so large that each of them covers a whole tt-degree while other linear degrees are so small that each of them consists only of one many-one degree.

In contrast to many other areas where the recursion theoretic side is much better known than the complexity-theoretic one, the structure inside recursive polynomial time tt-degrees is well-known: There are nontrivial tt-degrees which consist only of one btt-degree. Ambos-Spies [6] constructed some natural examples; these are the tt-degrees of certain supersparse sets. Ladner, Lynch and Selman [91] showed that, whenever $X <_{btt} Y$ (in the context of polynomial time reducibilities) for two recursive sets X and Y, then there is a Z which is properly between X and Y: $X <_{btt} Z <_{btt} Y$. So having two different btt-degrees X, Y inside one tt-degree, one can take the upper bound $X \oplus Y$ of both which is still inside the same tt-degree. At least one of the btt-degrees, say X, is strictly below $X \oplus Y$ and one can construct an infinite ascending chain from X to $X \oplus Y$. So every recursive polynomial time tt-degree consists of either one or infinitely many btt-degrees.

Basically, the last result depends only on the dense embeddability of countable chains into every proper interval of the subrecursive btt-degrees. As a consequence, infinite chains can be replaced by arbitrary countable distributive lattices due to a corresponding result on lattice embeddings by Ambos-Spies [5]. In fact, these embeddability results do not depend on the use of polynomial time bounds. The results hold for abstractly defined subrecursive (or "bounded") reducibilities which comprise a wide class of time and space bounded reducibilities [102, 103]. Also they are independent of the exact nature of the reducibilities; therefore every time-bounded tt-degree contains either one or infinitely many positive degrees which contrasts from the result for recursion theoretic positive degrees versus tt-degrees. Nevertheless, the sets representing these degrees are still required to be recursive. Furthermore, also for the world of recursive polynomial time degrees, some questions are still open. For example, it is unknown whether there is a Turing degree consisting of exactly one many-one degree.

Now the formal definition of the reducibilities is given. A set D is Turing reducible to E iff there is an algorithm which computes for each x the value $D(x)$ using some queries to E which depend on x. All reducibilities considered within this chapter are obtained by restricting the way in which $D(x)$ is computed. In particular the number of the queries made is restricted and also the way the results are evaluated afterwards.

truth-table reducibility

$D \leq_{tt} E \Leftrightarrow (\exists \text{ total-recursive } f, g)\,(\forall x)\,[D(x) = g(x, E(0), E(1), \ldots, E(f(x)))]$.
The intuitive idea is that $D(x)$ is computed in two steps: First the algorithm queries E at $0, 1, \ldots, f(x)$ and then evaluates this information via a truth-table which may depend on x.

bounded truth-table reducibility

$$D \leq_{btt} E \Leftrightarrow (\exists k)\,(\exists \text{ total-recursive } f_1, f_2, \ldots, f_k, g)\,(\forall x)\,[D(x) = g(x, E(f_1(x)), E(f_2(x)), \ldots, E(f_k(x)))].$$

This is a more restrictive variant of the truth-table reducibility since the algorithm makes only k queries to E, these queries are selected by k total recursive function in parallel. The number k is called the norm of the btt-reducibility. The notion $D \leq_{btt(k)} E$ means that $D \leq_{btt} E$ with norm k.

positive reducibility

$D \leq_p E \Leftrightarrow (\exists \text{ total-recursive } f, g)\,(\forall x)\,[D(x) = g(x, E(0), E(1), \ldots, E(f(x)))]$ where in addition $g(x, a_0, a_1, \ldots, a_{f(x)}) \leq g(x, b_0, b_1, \ldots, b_{f(x)})$ whenever $a_i \leq b_i$ for $i = 0, 1, \ldots, f(x)$.
So positive reducibility is a restricted truth-table reducibility not in the way that the number of queries is bounded by a constant but in the way that the function evaluating the queries must be "positive" or "monotone". The easiest way to think of a "positive" truth-table is to think of a formula whose atoms have the form $E(i)$ or 0 or 1 and are connected by and-operations and or-operations. The formula does not contain any negation or negated form $\overline{E}(i)$ of an atom. The formula may of course depend on x.

many-one reducibility

$D \leq_m E \Leftrightarrow (\exists \text{ total-recursive } f)\,(\forall x)\,[D(x) = E(f(x))]$.
Many-one reducibility might be considered as a special case of btt-reducibility with norm 1. The evaluation does not depend on x and is given by the simple formula $x \in D \Leftrightarrow f(x) \in E$.

Closely related to the notion of reducibility is that of degree. Two sets are truth-table equivalent iff each of them is truth-table reducible to the other one and the truth-table degree is just a class which contains all sets truth-table equivalent to some given representative.

$$\begin{aligned} Y \equiv_{tt} X &\Leftrightarrow Y \leq_{tt} X \wedge X \leq_{tt} Y; \\ \text{tt-degree}\,(X) &= \{Y : Y \equiv_{tt} X\}. \end{aligned}$$

Similarly one can define the corresponding equivalence relations and degrees also for bounded truth-table, positive truth-table, many-one and other reducibilities.

8.1 Some Structural Properties

Post [120] looked at many structural properties of sets in order to decide whether they are complete for some strong reducibility or not. Post himself did not find a structural property which could be used to prove that there is some Turing incomplete enumerable set. The first constructions [47, 107] to show that such a set existed did not exploit such structural properties but introduced the priority method. Finally, Harrington and Soare [59] found such a structural property. Similarly the structure induced by the strong reducibilities is explored using representatives having certain structural properties. The sets A, B and C defined below have certain outstanding structural properties so that they are important at many places within this chapter. They are introduced at the next definition and the letters A, B and C are reserved for sets of these types. They are constructed from any arbitrary representative X of their tt-degree. Note that these sets are constructed as sets of strings or formulas instead of sets of numbers — in this case the constructions use implicitly some canonical translation between these concepts.

Definition 8.1.1 The set A contains the strings $X(0)X(1)\dots X(n)$ for all natural numbers n, that is, A is the set of all prefixes of the characteristic function of X. In short, $A = \{x : x \preceq X\}$. A is an example of a retraceable set as introduced by Dekker and Myhill [38].

The set B contains all strings which are lexicographically below those in A, that is, it contains every string $X(0)X(1)\dots X(n)$ and furthermore every string $Y(0)Y(1)\dots Y(n)$ such that there is an $m \leq n$ with $Y(k) = X(k)$ for $k < m$ and $Y(m) < X(m)$. In short, $B = \{x : x \leq_{lex} X\} = \{x : (\exists y \in A)\,[x \leq_{lex} y]\}$. B is just the standard semirecursive set derived from X as used by Martin [96], McLaughlin [99] and Jockusch [69, Theorem 3.6]. Jockusch [69, Corollary 4.3] showed that B is never in the greatest positive degree and Dëgtev [36] showed that B is never in the greatest btt-degree of its tt-degree.

The set C contains all tuples $\langle n, \phi, x_1, \dots, x_n\rangle$ where ϕ is a Boolean formula in n variables and $\phi(X(x_1), X(x_2), \dots, X(x_n))$ is true. Rogers [124, §8.4] called this set C the tt-cylindrification of X or simply a tt-cylinder. C represents the greatest m-degree within a tt-degree, that is, for all sets U the equivalence $U \leq_{tt} C \Leftrightarrow U \leq_m C$ holds. It is obvious that C also represents the greatest positive degree and btt-degree within its tt-degree.

The next definition deals with tt-reducibilities to A. Since A is of a special form, these tt-reducibilities can be represented in some standardized form.

Definition 8.1.2 Let $\tilde{A}$ be a set-variable which ranges over the sets of the same form as A, that is, $\tilde{A}$ ranges over all sets derived from some $\tilde{X}$ by $\tilde{A} = \{x : x \preceq \tilde{X}\}$.

If h is a tt-reduction from a set D to the set A, that is, if $D = h^A$, then one can define the following function:

$$h[x](a) = \begin{cases} 0 & \text{if } h^{\tilde{A}}(a) = 0 \text{ for all } \tilde{A} \text{ with } x \in \tilde{A}; \\ 1 & \text{if } h^{\tilde{A}}(a) = 1 \text{ for all } \tilde{A} \text{ with } x \in \tilde{A}; \\ ? & \text{otherwise.} \end{cases}$$

So the function h searches on all oracles $\tilde{A}$ of the same form as the given set A which contain x. If this search gives a unique opinion then $h[x](a)$ outputs this value, otherwise $h[x](a)$ outputs the symbol "?". The search terminates since h is a tt-reduction and therefore the oracles $\tilde{A}$ have to be inspected only on the use of the tt-reduction at input a. In particular, if $x \in A$ and x is sufficiently large, then $h[x](a) = D(a)$.

Now the relations between A, B and C with respect to some strong reducibilities are investigated. It is easy to see that A, B and C belong to the same tt-degree.

For positive reducibility, Jockusch [69] showed already that B and $\overline{B}$ define two separate positive degrees which are in addition different to that one of C. Since everything inside the tt-degree of C is many-one reducible to C, the positive degrees of B and $\overline{B}$ are below that of C. Furthermore, A and C are equivalent under positive reducibility since every tt-reduction to A can be turned into a positive one: Since $A(x) = A(x0) + A(x1)$, every query to some x can be replaced by two queries to $x0$ and $x1$. So one can adjust the queries of the tt-reduction such that A is evaluated just at the strings of one given length. Since A contains exactly one element of each length, exactly one of the queries is answered with 1 and every vector of answers containing no or several 1s does not come from A. So one can modify the truth-table on these vectors to 0 for the all-0-vector and to 1 on those vectors which contain at least two 1s. The truth-table reduction has been turned into a positive one and the sets A and C belong to the same positive degree.

So it remains to investigate which relations hold between A, B and C with respect to btt-reducibility. The first result is due to Dëgtev who proved his lower bound by showing that B and C represent two different btt-degrees inside every given nonrecursive tt-degree.

Proposition 8.1.3 [36, Theorem 1] *$B <_{btt} C$.*

Dëgtev showed this as follows. First C belongs to the greatest m-degree and therefore also to the greatest btt-degree inside the tt-degree of X, so $B \leq_{btt} C$. Second he showed that the sets

$$B_k = \{(x_1, x_2, \ldots, x_k) : |\{i : x_i \in B\}| \text{ is odd}\}$$

form an ascending chain of m-degrees, that is, $B_k <_m B_{k+1}$ for all k. Third he showed that under the assumption that $C \leq_{btt} B$ there is a k such that $C \leq_m B_k$. Since $B_{k+1} \leq_m C$ by the first result, Dëgtev obtained that $B_{k+1} \leq_m B_k$ in contradiction to his second result.

The next result shows that A forms a btt-degree strictly below B provided that B is neither enumerable nor coenumerable.

Proposition 8.1.4 *If B is neither enumerable nor coenumerable then $A <_{btt} B$.*

Proof Let x, y, z range over the strings $\{0,1\}^*$, let $x \preceq y$ mean that $y = xz$ for some $z \in \{0,1\}^*$ and let $x \leq_{lex} y$ mean that x is lexicographic before y, that is, either $x \preceq y$ or there is a string z such that $z0 \preceq x$ and $z1 \preceq y$. If B is not computable then A and B relate to each other as follows:

$$A = \{x : (\exists y, z \succeq x)\,[y \in B \wedge z \notin B]\}. \tag{29}$$

In other words, A is just the borderline between B and $\overline{B}$. For every x let x' denote the lexicographic successor of x of the same length (if it exists). For example, $00110' = 00111$, $101011' = 101100$ and $1111'$ does not exist. Since A is the borderline of B, x is in A iff x is in B and x' is not in B. More formally, $A \leq_{btt(2)} B$ via the following reduction.

$$x \in A \;\Leftrightarrow\; \begin{cases} x \in B \wedge x' \notin B & \text{if } x \in \{0,1\}^* - \{1\}^*\text{, that is, if } x' \text{ exists;} \\ x \in B & \text{otherwise, that is, } x \in \{1\}^* \text{ and } x' \text{ does not exist.} \end{cases}$$

For example, $0011 \in A \Leftrightarrow 0011 \in B \wedge 0100 \notin B$ and $11111 \in A \Leftrightarrow 11111 \in B$.

By way of contradiction, it is assumed that $B \leq_{btt} A$. First it is shown that each query to some y can be replaced by up to two queries to some $z \preceq x$.

So every query whether "$y \in A$?" where $y \not\preceq x$ has to be replaced by some queries to some prefixes of x. In the case that $y \succ x$ one knows that $y \in A \Rightarrow x \in A \Rightarrow x \in B$. So within the computation a query whether "$y \in A$?" can be replaced by a query whether "$x \in A$" such that the computation halts with "$B(x) = 1$" if the answer "$x \in A$" is received and continues using that "$y \notin A$" if the answer "$x \notin A$" is received. In the case that $y \not\preceq x$ and $x \not\preceq y$ a further case-distinction is made beginning with the subcase $x <_{lex} y$. There is a unique greatest lower bound z for x and y, that is, $z0 \preceq x$ and $z1 \preceq y$. So the query whether "$y \in A$?" is replaced by the query whether "$z \in A \wedge z0 \notin A$?". If so, the computation halts with "$x \in B$" since $z1 \in A$ and so some number lexicographic above

x is in B; otherwise one knows that $z1 \notin A$ and therefore $y \notin A$ and the computation continues in this case using that "$y \notin A$". In the other subcase where $x >_{lex} y$ a similar common lower bound z with $z0 \preceq y$ and $z1 \preceq x$ is computed. This is symmetric to the previous subcase with the two differences that the query is whether "$z \in A \wedge z1 \notin A$?" and that in the case that this holds the computation terminates with "$x \notin B$". Note that the number of queries is still bounded by a constant, but this constant may be twice the previous one.

So if $B \leq_{btt} A$ then the reduction $B(x) = g(x, A(f_1(x)), A(f_2(x)), \ldots, A(f_k(x)))$ can be taken such that $f_1(x) \preceq f_2(x) \preceq \ldots \preceq f_k(x) \preceq x$ for all x. Without loss of generality, let k be the minimal norm such that there is a reduction of this form. The next case-distinction shows that either the assumption on the minimality of the norm k or on the fact that B is neither enumerable nor coenumerable is false, so the contradiction is obtained.

(I): There is an $x \in A$ such that $f_1(y) \preceq x$ for all $y \succeq x$. Then it is possible to compute B with norm $k-1$ relative to A:

$$B(y) = \begin{cases} 0 & \text{if } y \not\succeq x \text{ and } y >_{lex} x; \\ 1 & \text{if } y \not\succeq x \text{ and } y \leq_{lex} x; \\ g(y, 1, A(f_2(y)), \ldots, A(f_k(y))) & \text{otherwise, that is, if } y \succeq x. \end{cases}$$

This contradicts to the choice of k as the optimal norm for the reduction and therefore case (I) does not occur.

(II): For every $x \in A$ there is a $y \succeq x$ with $f_1(y) \succeq x$. Now it is possible to find $d \in \{0,1\}$ such that in addition $g(y, 0, 0, \ldots, 0) = d$ can be required for infinitely many $x \in A$. One can now replace "for infinitely many x" by "for all x" since if such a y exists above some x then the same y exists also above every $x' \preceq x$ satisfying the same condition. Now a case distinction is made for the two possible values of d in order to show that for $d = 0$ the set $\overline{B}$ and for $d = 1$ the set B is enumerable.

(IIa): $d = 0$. Let $y \leq'_{lex} x$ denote that there is some $z \preceq y$ with $z0 \preceq y$ and $z1 \preceq x$. Note that $x \in \overline{B}$ iff some $y \leq'_{lex} x$ is in A.

Assume that for an arbitrary x there is some y such that $y \leq'_{lex} x$, $f_1(y) \leq'_{lex} x$ and $g(y, 0, 0, \ldots, 0) = 0$. If $f_1(y) \in A$ then $x \in \overline{B}$. Otherwise $f_1(y) \notin A$ and then none of the values $f_1(y), f_2(y), \ldots, f_k(y)$ is in A, $B(y) = 0$ and $y, x \in \overline{B}$. On the other hand for every $x \in \overline{B}$ there is an $x' \leq'_{lex} x$ in A and for this x' some $y \succeq x'$ satisfies $x' \preceq f_1(y)$ and $g(y, 0, 0, \ldots, 0) = 0$. This y then also satisfies $y \leq'_{lex} x$ and $f_1(y) \leq'_{lex} x$. So one obtains that

$$\overline{B} = \{x : (\exists y \leq'_{lex} x)\, [f_1(y) \leq'_{lex} x \wedge g(y, 0, 0, \ldots, 0) = 1]\,\}$$

and $\overline{B}$ is an enumerable set.

(IIb): $d = 1$. This case is symmetric to the previous one. Assume that for an arbitrary x there is some y with $x \leq_{lex} f_1(y)$, $x \leq_{lex} y$ and $g(y, 0, 0, \ldots, 0) = 1$. If $f_1(y) \in A$ then also $f_1(y) \in B$ and $x \in B$. Otherwise $f_1(y) \notin A$ and then $f_1(y), f_2(y), \ldots, f_k(y) \notin A$, $B(y) = 1$ and again $x \in B$. On the other hand for every $x \in B$ some $x' \in A$ and some y satisfy $x' >_{lex} x$, $g(y, 0, 0, \ldots, 0) = 1$, $y \succeq x'$ and $f_1(y) \succeq x'$. It follows that $y \geq_{lex} x$ and $f_1(y) \geq_{lex} x$. So one obtains that

$$B = \{z : (\exists y \geq_{lex} x)\, [f_1(y) \geq_{lex} x \wedge g(y, 0, 0, \ldots, 0) = 1]\,\}$$

is an enumerable set. ∎

Dëgtev [35, Theorem 3] constructed a tt-degree containing an enumerable set M such that every further nonrecursive enumerable set which is tt-reducible to M is already many-one equivalent to M. So let B belong to such a degree and be enumerable. Again let x'

denote the lexicographic successor of x (if it exists). The set B has a recursive enumeration $z_0, z_1, \ldots$ and let $D = \{y : z_y \notin A\}$. D is an enumerable set since $y \in D \Leftrightarrow (z_y)' \in B$. So D and B are many-one equivalent and there is a recursive function f such that $x \in B \Leftrightarrow f(x) \in D \Leftrightarrow z_{f(x)} \notin A$. This equivalence shows that $A \equiv_{btt} B$. Therefore it is essential to the previous proof that B is neither enumerable nor coenumerable. Since there is by Theorem 8.6.4 a tt-degree whose semirecursive sets are all enumerable or coenumerable, $A <_{btt} B$ cannot be achieved by a suitable choice of B within this tt-degree.

The next theorem shows that each class of sets btt-reducible to a given semirecursive set B is restricted. The same results hold also for A since $A \leq_{btt} B$.

Proposition 8.1.5 *Let $D \leq_{btt} B$ for some semirecursive set B and let D be nonrecursive. Then $D \equiv_T B$. Furthermore, if D is simple, then D is also hypersimple.*

Proof The proof works using a technique from the field of frequency computation. This technique uses the notion of strongly (m, n)-verbose sets [14, 16]: A set E is strongly (m, n)-verbose iff there is a total recursive function f which computes, for every $x_1, \ldots, x_n$, m binary vectors of length n such that $(E(x_1), \ldots, E(x_n))$ is among these m vectors.

Every semirecursive set is $(n+1, n)$-verbose for all n by the simple algorithm which generates all vectors compatible with the ordering $\sqsubset$ for which B is an initial segment [14]. Using this fact one proves the following claim: Every set btt-reducible to B with norm k is strongly $(nk+1, n)$-verbose for all n. Given arguments $x_1, x_2, \ldots, x_n$ one can calculate for each x_i exactly k numbers $y_{i,1}, y_{i,2}, \ldots, y_{i,k}$ such that the characteristic function at x_i can be computed from $B(y_{i,j})$ for $j = 1, 2, \ldots, k$. Since B is semirecursive, one can find $nk+1$ binary vectors among which one agrees to the characteristic function of B at all $y_{i,j}$. For each of these vectors, one can evaluate the btt-reduction and get $kn+1$ vectors, among them the characteristic function of the given set at $x_1, x_2, \ldots, x_n$. This completes the claim.

If D has a different nonrecursive Turing degree than B, then the set $B \oplus D$ is strongly $(m(n), n)$-verbose only for a function m which is at least quadratic: $m(n) \geq \frac{1}{4}(n+1)(n+3)$ [16, Theorem 3.3]. On the other hand, if $D \leq_{btt} B$ then $B \oplus D$ is also btt-reducible to B and would be strongly $(nk+1, n)$-verbose for some k — which is impossible for $n = 4k$. Then the upper bound of the verboseness is $4k^2 + 1 \leq (4k+1)k$ and the lower bound is $\frac{1}{4}(4k+1)(4k+3) > (4k+1)k$, a contradiction.

Kummer and Stephan [90, Theorem 4.7] showed that a simple set D which is not hypersimple are strongly $(m(n), n)$-verbose only if $m(n) \geq \frac{1}{2}n(n+1)+1$. Again one has that if $D \leq_{btt} B$ with some norm k, then D is on one hand strongly $(2k^2+1, 2k)$-verbose and on the other hand not strongly $(2k^2+k, 2k)$-verbose, a contradiction. So $D \not\leq_{btt} B$. ∎

Also any c-complete enumerable, hypersimple and maximal set D has a quadratic lower bound for the minimum complexity in terms of strong verboseness. As indicated in the proof of the previous theorem, no such set D is btt-reducible to a semirecursive set or a set retraced by a total recursive function. Jockusch [69, Corollary 4.6] showed the result for enumerable sets: No maximal set is btt-reducible to an enumerable semirecursive sets. The d-complete sets have even the exponential lower bound 2^n for their complexity in terms of strong verboseness and so are not btt-reducible to any set which is strongly $(2^n - 1, n)$-verbose for some n.

8.2 Inside Hyperimmune Truth-Table Degrees

A tt-degree is called hyperimmune iff it is contained inside a hyperimmune Turing degree. That is, a set has hyperimmune tt-degree iff there is a hyperimmune set H Turing equivalent to it.

Within this section it is shown that every hyperimmune tt-degree contains an infinite number of btt-degrees. This is the first step on the way to a proof that all nonrecursive tt-degrees contain infinitely many btt-degrees. In addition it is shown that the ordering of the enumerable sets under inclusion can be embedded into the ordering of the btt-degrees inside every given hyperimmune tt-degree (with respect to the ordering but not preserving meets and joins) and so there exist infinite ascending and descending chains as well as infinite antichains of btt-degrees within the given tt-degree.

The main result, Theorem 8.2.3, is based on some propositions. The first one of them, Proposition 8.2.1, shows that below every set A of hyperimmune tt-degree there is some hyperimmune set $H \leq_{tt} A$. This cannot be improved to $H \equiv_{tt} A$ since for example the tt-degree of K does not contain a hyperimmune set: Post [111, Theorem III.3.10] showed that K is not tt-reducible to a hypersimple set; actually the proof works for hyperimmune sets.

Proposition 8.2.1 *If the Turing degree of A is hyperimmune then there exists a function $f \leq_{tt} A$ such that, for every k, the set $F_k = \{x : f(x) = k\}$ and its complement are both hyperimmune.*

Proof First f is constructed. Given A, let $Y = \{y_0, y_1, \ldots\}$ be a hyperimmune set which is computable relative to A. For each n, let t_n be the time which is used to compute $Y(0), Y(1), \ldots, Y(y_n)$ relative to the oracle A. So t_n is just the time to search through Y until at least $n+1$ elements are found. Now let $x_0 = y_0 + t_0$ and $x_{n+1} = x_n + y_{n+1} + t_{n+1}$. The set $X = \{x_0, x_1, \ldots\}$ is also hyperimmune but it is also truth-table reducible to A while Y was only Turing reducible to A. Note that there is no computable function g such that $g(n) > x_n$ for all n [111, Proposition III.3.8]. This can even be strengthened in the way that for every computable function g there is an n with $g(x_n) < x_{n+1}$. To see this just consider the function given by $g'(0) = x_0$ and $g'(n+1) = \max\{g(y) : y \leq g'(n)\}$ and take the first n with $g'(n+1) < x_{n+1}$ which exists by [111, Proposition III.3.8].

The definition of f uses X and basically meets the overall goal by satisfying the requirements $\langle e, k\rangle$ inductively.

> $\langle e, k\rangle$: φ_e does not witness against $\overline{F_k}$ being hyperimmune, that is, either φ_e is partial or there is some s with $s, s+1, \ldots, \varphi_e(s) \in F_k$.

Let x_n be the largest member of X below s. One defines that

- Requirement $\langle e, k\rangle$ needs attention at stage s if $\varphi_{e,s}(x)\downarrow$ for all $x \leq x_n$.
- Requirement $\langle e, k\rangle$ is satisfied at stage s if there is $x < x_n$ with $\varphi_{e,s}(x)\downarrow \leq x_n$ and $f(y) = k$ whenever $x \leq y \leq \varphi_{e,s}(x)$.
- $f(s) = k$ for the least requirement $\langle e, k\rangle \leq s$ which needs attention at stage s and is not already satisfied at stage s. If every requirement below s either does not need attention or is already satisfied then $f(s) = 0$ as a default-value.

It is easy to see that $f \leq_{tt} X \leq_{tt} A$. Assume now by way of contradiction that for some total function φ_e some requirement $\langle e, k \rangle$ would never be satisfied, without loss of generality all requirements $\langle e', k' \rangle < \langle e, k \rangle$ are either satisfied or belong to a partial function $\varphi_{e'}$. The sequence $x_0, x_1, \ldots$ dominates G and is hyperimmune. There is an x such that all requirements $\langle e', k' \rangle < \langle e, k \rangle$ are either satisfied below x or have $\varphi_{e'}(y)\uparrow$ for some $y < x$. For input y, let $g(y)$ take the value $s + \varphi_e(s)$ for the first stage $s > y$ where $\varphi_{e,s}(z)$ is defined for all $z \leq y$. Since X is hyperimmune and since g is a computable function there are arbitrary large numbers n such that $g(x_n) < x_{n+1}$. Now taking n large enough and s as above the following conditions are satisfied.

- $x + e < x_n < s$;
- $\varphi_{e,s}(y)\downarrow$ for all $y \leq x_n$;
- $\varphi_e(s)\downarrow < x_{n+1}$.

Now the requirement $\langle e, k \rangle$ is the least one which needs attention at the stages $t = s, s+1, \ldots, x_{n+1} - 1$ and is not already satisfied. Therefore $f(t) = k$ for $t = s, s+1, \ldots, x_{n+1} - 1$, in particular for $t = s, s+1, \ldots, \varphi_e(s)$. Thus the requirement $\langle e, k \rangle$ is satisfied against the assumption; in particular φ_e is not a witness against $\overline{F_k}$ being hyperimmune.

It remains to be shown that for every k that f takes infinitely often the value k: For every m there is a function φ_e such that $\varphi_e(s) = s + m$. Since the requirement $\langle e, k \rangle$ is satisfied there is an s such that $f(t) = k$ for $t = s, s+1, \ldots, \varphi_e(s)$; so f takes at least $\varphi_e(s) + 1 - s = m + 1$ times the value k. Since this holds for every m, the value k is taken infinitely often. So every set F_k is infinite and since the sets F_k are disjoint, also every set $\overline{F_k}$ is infinite since its subset F_{k+1} is infinite.

So for all e and k, either φ_e is partial or φ_e is not a witness against $\overline{F_k}$ being hyperimmune. All sets $\overline{F_0}, \overline{F_1}, \ldots$ are hyperimmune since they are infinite and no total-recursive function witnesses the contrary. The sets F_k are also hyperimmune since they are infinite and every F_k is a subset of the hyperimmune set $\overline{F_{k+1}}$. ∎

Proposition 8.2.2 *Let A, f and $F_1, F_2, \ldots$ be defined as in Proposition 8.2.1. Furthermore, let $G_i = \{(x, f(x)) : f(x) \neq i\}$. Then $F_i \not\leq_{btt} G_i \oplus A$.*

Proof Assume by way of contradiction that $F_i \leq_{btt} G_i \oplus A$ on some infinite recursive set Y, that is, assume that there is an infinite recursive set Y and a reduction such that

$$(\forall x \in Y)\,[F_i(x) = g(x, G_i(d_1(x)), \ldots, G_i(d_m(x)), A(e_1(x)), \ldots, A(e_n(x)))\,]$$

where $g, d_1, \ldots, d_m, e_1, \ldots, e_n$ are total recursive functions. The sum $m + n$ is called the norm of the reduction. The recursive set Y is chosen such that the reduction takes the minimal possible norm. Furthermore, the following statements hold where Y may be replaced by a suitable computable infinite subset if necessary.

(I): There is an a such that $g(x, 0, \ldots, 0) = a$ for all $x \in Y$.

For $b = 0, 1$, consider the two subsets $Y_b = \{x \in Y : g(x, 0, \ldots, 0) = b\}$, that is, Y_b is the set of all $x \in Y$ such that the btt-reduction returns b whenever all queries to G_i and A are answered by 0. If Y_1 is infinite, let $a = 1$ and replace Y by Y_1. Otherwise Y_0 is infinite, let $a = 0$ and replace Y by Y_0.

(II): For no function d_k there is an infinite recursive subset $Y' \subseteq Y$ such that $G_i(d_k(x))$ is the same constant c for all $x \in Y'$. The same holds for $e_1, e_2, \ldots, e_n$. In particular, none of the functions $d_1, d_2, \ldots, d_m, e_1, e_2, \ldots, e_n$ takes the same value for infinitely many $x \in Y$.

Assume by way of contradiction, that this would fail for some d_k or e_k; say $G_i(d_1(x)) = c$ for all $x \in Y'$. Then let $g'(x, b_2, \ldots, b_{m+n}) = g(x, c, b_2, \ldots, b_{m+n})$ and it follows that

$$(\forall x \in Y')\, [F_i(x) = g'(x, G_i(d_2(x)), \ldots, G_i(d_m(x)), A(e_1(x)), \ldots, A(e_n(x)))\,]$$

in contradiction to the minimality of the norm $m + n$. The second statement is almost obvious since if d_1 takes the constant y on Y' then $G_i(d_1(x)) = G_i(y)$ for all $x \in Y'$.

(III): $n > 0$.

Assume by way of contradiction that $n = 0$. Now it is shown that under this assumption one of the sets $\{x : f(x) \neq j\}$ is not hyperimmune in contradiction to the choice of f. So for any y let

$$D(y) = \{z : (\exists z')\, [(z, z') \in \{d_1(y), d_2(y), \ldots, d_n(y)\}]\}.$$

Note that for every x the intersection $\{0, 1, \ldots, x\} \cap D(y)$ is empty for almost all $y \in Y$ since one could otherwise use (II) in order to construct a contradiction: some function d_k would infinitely often take some pair with the same coordinate z and one can then evaluate $G_i(d_1(y))$ whenever it takes such a pair just by comparing the second component of the pair with the constant $f(z)$. So given any x one can find effectively a $y(x)$ such that $y(x)$ and every value in $D(y(x))$ is greater than x. Note that $y(x)$ is simply the first $y > x$ with $y \in Y$ in the case that $m = 0$. Now let $I_x = \{x, x+1, \ldots, y(x) + \max(D(y(x)))\}$ for $m > 0$ and $I_x = \{x, x+1, \ldots, y(x)\}$ for $m = 0$. The set I_x is effectively computable from x. The further proof needs a case-distinction on the constant a from (I).

First the case $a = 0$. Now I_x intersects $\overline{F_i}$: If $D(y(x)) \subseteq F_i$ then $G_i(d_k(y(x))) = 0$ for all k and so $F_i(y(x)) = g(y(x), 0, \ldots, 0) = a$ and therefore $y(x) \in \overline{F_i}$. Since this holds for all x, it follows that $\overline{F_i}$ is not hyperimmune in contradiction to the choice of F_i.

Second the case $a = 1$. Since $\mathbb{N}$ is infinite and $\{1, 2, \ldots, m\}$ is finite, it is impossible that for every j there is a k such that $d_k(y)$ has the form (z, j) for almost all $y \in Y$. So there is an index $j \neq i$ such that for infinitely many $y \in Y$ no pair $d_k(y)$ has the form (z, j). So one can find a j and replace Y by a suitable subset Y' such that $d_k(y) \neq (z, j)$ for all $z \in \mathbb{N}$, all $k \in \{1, 2, \ldots, m\}$ and all $y \in Y'$. Now one knows that every such interval I_x intersects $\overline{F_j}$: either $G_i(d_k(y(x))) = 1$ and therefore the z with $d_k(y(x)) = (z, z')$ satisfies $f(z) = z' \neq j$ or $F_i(y(x)) = 1$ and $f(y(x)) = i \neq j$. So it follows that $\overline{F_j}$ is not hyperimmune, a contradiction.

(IV): For all $u \in A$ and almost all $x \in Y$: $u \preceq e_1(x) \wedge u \preceq e_2(x) \wedge \ldots \wedge u \preceq e_n(x)$.

Assume that this would fail for one $u \in A$. Since the finitely many values below u are taken only finitely often, one of the values $e_1(x), \ldots, e_n(x)$, say $e_1(x)$, is incomparable to u for infinitely many $x \in Y$. The recursive set $Y' = \{x \in Y : e_1(x) \not\preceq u \wedge u \not\preceq e_1(x)\}$ is infinite and $A(e_1(x)) = 0$ for all $x \in Y'$ in contradiction to (II).

(V): For all $y \in Y$: $e_1(y) \in A \vee e_2(y) \in A \vee \ldots \vee e_n(y) \in A$.

The proof of this fact is similar to the proof for (III). So assume by way of contradiction that there are infinitely many $y \in Y$ with $e_1(y) \notin A \wedge e_2(y) \notin A \wedge \ldots \wedge e_n(y) \notin A$. Again let $D(y) = \{z : (\exists z', k)\, [d_k(y) = (z, z')]\}$. Note that for each y with $e_1(y) \notin A \wedge e_2(y) \notin A \wedge \ldots \wedge e_n(y) \notin A$ there is a $y' \in Y$ and $u \in A$ such that $u' \not\preceq e_1(y), e_2(y), \ldots, e_n(y)$ and $u' \preceq e_1(y'), e_2(y'), \ldots, e_n(y')$. So it is possible to compute for every given x two elements $y(x), y'(x) \in Y$ and an interval $I_x = \{x, x+1, \ldots, x'\}$ such that $x' > x$, $y(x), y'(x) \in I_x$, $D(y(x)) \cup D(y'(x)) \subseteq I_x$ and either all values $e_k(y(x))$ or all values $e_k(y'(x))$ are not in A. Since $y(x)$ and $y'(x)$ can be interchanged without changing the construction, one might fix $e_1(y(x)), e_2(y(x)), \ldots, e_m(y(x)) \notin A$ for the verification that I_x witnesses that either $\overline{F_i}$ (in the case $a = 0$) or some $\overline{F_j}$ (in the case $a = 1$) is not hyperimmune.

The remaining part to show (v) can now be completed by copying word by word the part beginning with the case distinction whether $a = 0$ or $a = 1$ in the proof of (iii) where of course the definitions of I_x and $y(x)$ have to follow those given in this part (v). One obtains from the argumentation that in both cases there is a contradiction. So the converse of the assumption holds, that is, for almost all $y \in Y$, one of the values $e_k(y)$ is in A. By removing the finitely many exceptions in Y one can conclude that this holds for all $y \in Y$.

Now from the last two conditions (iv) and (v) it follows that A is recursive:

> On input w find the first $y \in Y$ and v with $|v| = |w|$ and $v \preceq e_1(y), \ldots, e_n(y)$.
> If $v = w$ then output "$w \in A$" else output "$w \notin A$".

The algorithm terminates, since for every w there is an $v \in A$ of length $|w|$ and this v satisfies for some $y \in Y$ the search-condition. Furthermore, $v \in A$ since by the last item one of the $e_i(y)$ is in A and A is closed under $\preceq$. But A was chosen of hyperimmune and therefore nonrecursive degree. By this contradiction, the proposition holds. ∎

Now the two propositions are put together to obtain the desired theorem. Recall that a tt-degree is defined to be hyperimmune iff it is inside a hyperimmune Turing degree.

Theorem 8.2.3 *Every hyperimmune tt-degree consists of infinitely many btt-degrees.*

Proof Let A be the prefix-set inside the given tt-degree and $f, F_0, F_1, \ldots$ as in Proposition 8.2.1. $F_y \not\leq_{btt} A \oplus F_z$ for $y \neq z$ by Proposition 8.2.2: note that every F_z is a row of G_y, that is, $F_z \leq_m G_y$, and the statement follows just from $F_y \not\leq_{btt} A \oplus G_y$. On the other hand $F_y \leq_{tt} A$ for all y. Thus the sets $A_0, A_1, \ldots$ given by

$$A_y \;=\; A \oplus F_y \;=\; A \oplus \{x : f(x) \neq y\}$$

represent an antichain of infinitely many incomparable btt-degrees inside a given hyperimmune tt-degree. ∎

A direct corollary is an intermediate result which increases Dëgtev's lower bound from 2 to 3: Either A, B and C represent three different btt-degrees inside the given tt-degree or B is enumerable or coenumerable, in particular B has hyperimmune tt-degree. In this latter case the previous result gives infinitely many and thus at least three btt-degrees.

A closely related question is to determine the quantity of enumerable btt-degrees in a enumerable tt-degree. Dëgtev [35] constructed a tt-degree, whose enumerable sets are all in one m-degree and therefore also in one btt-degree. On the other hand Kallibekov [74] and Kobzev [84] showed that the tt-degree of the halting problem K contains infinitely many enumerable btt-degrees. The next theorem shows that both, the structures of btt-degrees inside hyperimmune tt-degrees and the structures of enumerable btt-degrees inside the tt-degree of K, are rich.

Theorem 8.2.4 *The inclusion-structure of the enumerable sets can be embedded into the structures of*
(a) *the btt-degrees inside every hyperimmune tt-degree and*
(b) *the enumerable btt-degrees inside the tt-degree of the halting problem K.*

Proof (a): Let A be a prefix-set representing a given hyperimmune tt-degree and define the sets $F_1, F_2, \ldots \leq_{tt} A$ as in Proposition 8.2.2. Note that these sets and the sets H_e are uniformly recursive relative to A where the sets H_e are given as

$$H_e = \{(x, y, z) : x \in W_{e,y} \wedge z \in F_x\} \tag{30}$$

These sets have the following properties:

- If $W_e \subseteq W_{e'}$ then H_e is many-one-reducible to $H_{e'}$: Given (x, y, z) one first checks whether $x \in W_y$. If so, one enumerates $W_{e'}$ until the first stage y' is found with x appears there at stage y' and then has in this case that $H_e(x, y, z) = F_x(z) = H_e(x, y', z)$, so the many-one reduction maps (x, y, z) to (x, y, z'). If not, one knows that $H_e(x, y, z) = 0$ and the reduction maps (x, y, z) to some fixed nonelement of $H_{e'}$ which exists since the set $H_{e'}$ is not recursive.

- If $x \in W_e$ then F_x is many-one reducible to H_e.

- If $x \notin W_e$ then F_x is not bounded truth-table reducible to $A \oplus H_e$: This follows from the fact that F_x is not btt-reducible to $A \oplus G_x$ while H_e is many-one reducible to G_x by the fact that $H_e(x', y, z) = 0$ for $x' \notin W_{e,y}$ including the case $x' = x$ and $H_e(x', y, z) = G_x(z)$ for $x' \in W_{e,y}$.

- If $W_e \not\subseteq W_{e'}$ then H_e is not bounded truth-table reducible to $A \oplus H_{e'}$ since there is an $x \in W_e - W_{e'}$ so that the corresponding set F_x is many-one reducible to H_e but not bounded truth-table reducible to $A \oplus H_{e'}$.

It follows that the ordering among the btt-degrees of the form $A \oplus H_e$ is the same as among the enumerable sets W_e. The sets $A \oplus H_e$ have clearly the same tt-degree as A.

(b): Due to the infinite version of the Theorem of Friedberg and Muchnik [135, VII.2.2(a)] there are uniformly enumerable and semirecursive sets $B_0, B_1, \ldots$ such that no set B_i is computable relative to the join of the other sets B_j. Furthermore, one can chose the $B_0, B_1, \ldots$ such that K is not computable relative to the join of all. Now one uses a semirecursive set B with Turing degree K in place of A and B_x in place of F_x. The equation corresponding to (30) gives

$$H_e = \{(x, y, z) : x \in W_{e,y} \wedge z \in B_x\} \tag{31}$$

and from the four items the first, second and fourth hold immediately provided that the third item can be reproven:

- If $x \notin W_e$ then B_x is not bounded truth-table reducible to $B \oplus H_e$.

The Turing degree of $B_x \oplus H_e$ is properly between those of H_e and $B \oplus H_e$. By relativizing Proposition 8.1.5 to H_e-recursive computations, B_x is not btt-reducible to the semirecursive set B. Thus $B_x \oplus H_e$ and, in particular, B_x alone are not btt-reducible to $B \oplus H_e$ in the nonrelativized world.

As in case (a) one can now derive that the sets $B \oplus H_e$ represent enumerable btt-degrees inside the tt-degree of K such that the ordering of the H_e with respect to btt-reducibility is the same as that of the sets W_e under inclusion. ▮

8.3 Inside Truth-Table Degrees not Enumerable Relative to K

Miller and Martin [105] showed that every K-recursive set of hyperimmune-free Turing degree is recursive. It is straightforward to see that also every set A enumerable relative to K of hyperimmune-free Turing degree is also enumerable and thus by Miller and Martin's result as well recursive: Let A_s be a recursive approximation of A such that $A(x) = 1$ iff

$A_s(x) = 1$ for almost all s. The function f^A given as $f^A(x,s) = \min\{t \geq s : A_t(x) = A(x)\}$ is total and A-recursive. Since A has hyperimmune-free Turing degree, a computable function g majorizes f^A. Then the formula

$$x \in A \Leftrightarrow (\exists s)\,(\forall t, s \leq t \leq g(x,s))\,[A_t(x) = 1]$$

witnesses that A is enumerable. So, $A \leq_T K$ and A is recursive.

This section deals with the second step to show the existence of infinitely many btt-degrees inside nonrecursive tt-degrees: A structural construction gives infinitely many btt-degrees inside every given tt-degree not enumerable relative to K and by the previously mentioned fact, this construction covers the case of all hyperimmune-free nonrecursive Turing degrees. The gap left in the previous section is closed. Again sets of a special form are needed and these are based on a fixed truth-table reduction h from C to A.

Definition 8.3.1 Let h be a tt-reduction from C to A such that, for every finite tree $t \subseteq \{0,1\}^*$ and every labeling $d_x \in \{0,1\}$ of the leaves x of t, there is a number i such that the tree $t_i = \{x : h[x](i) = ?\}$ is equal to t and, for every leaf x of t, $h[xd_x](i) = 1 \wedge h[x(1-d_x)](i) = 0$. A recursive set D is compatible with C iff, for every finite tree t, either all or no i with $t_i = t$ are in D.

The next result is the keystone of the proof since it maps down the argumentation to mere topological arguments.

Theorem 8.3.2 *Let D and E be recursive sets compatible with C. If $C \cap D \leq_{btt} C \cap E$ then there is a constant k and a coenumerable tree T such that $A \subseteq T$ and, for every $i \in D$ where the leaves of t_i are branching nodes of T, there are $j_1, j_2, \ldots, j_k \in E$ with $t_i \subseteq t_{j_1} \cup t_{j_2} \cup \ldots \cup t_{j_k}$.*

Proof Let $g, f_1, f_2, \ldots, f_k$ define a btt-reduction from $C \cap D$ to $C \cap E$. Since every answer to queries outside E is 0, one can assume without loss of generality that the range of each of the functions $f_1, f_2, \ldots, f_k$ is a subset of E and that $g(i, a_1, a_2, \ldots, a_k) = 0$ whenever $i \notin D$.

Say that x is inconsistent iff there is an i such that the assumption $x \in A$ would imply that the given btt-reduction is incorrect. More formally, x is inconsistent iff there is a number $i \in D$ such that the values $h[x](i)$ and $h[x](f_1(i)), h[x](f_2(i)), \ldots, h[x](f_k(i))$ are all in $\{0,1\}$ but $h[x](i) \neq g(x, h[x](f_1(i)), h[x](f_2(i)), \ldots, h[x](f_k(i)))$. These inconsistent x cannot be members of A since the assumption $x \in A$ implies $h[x](i) = C(i) = (C \cap D)(i)$ and $h[x](f_l(i)) = (C \cap E)(f_l(i))$ for $l = 1, 2, \ldots, k$. Now the complement of the coenumerable tree T is given as the downward closure of all inconsistent x. In particular, every $x \notin T$ is either inconsistent or satisfies $x0, x1 \notin T$. Since every $x \in A$ has a successor $xd \in A$ for some $d \in \{0,1\}$, the tree T still contains every $x \in A$.

Let t be a finite tree such that $x0, x1 \in T$ for every leaf x of t and such that there is an $i \in D$ with $t_i = t$. Since all j with $t_j = t$ are also in D, one can chose i such that $h[xd](i) = d$ for every leaf x of t and every $d \in \{0,1\}$. Now the value $h[xd](i)$ depends on d, thus there must be some query $f_l(i)$ where the answer depends on d, that is, satisfies $h[y_0](f_l(i)) = 0 \wedge h[y_1](f_l(i)) = 1$ for two extensions $y_0, y_1 \succeq x$. It follows that $h[x](f_l(i)) = ?$. It follows that every leaf x of t_i is a node of some tree $t_{f_l(i)}$ and thus the tree t_i is covered by the union of the k trees $t_{f_1(i)}$, $t_{f_2(i)}$, …, $t_{f_k(i)}$. ∎

The next two results show that there are ascending and descending chains as well as antichains of btt-degrees inside every nonrecursive tt-degree.

Theorem 8.3.3 *There are infinite ascending and descending chains of btt-degrees inside every nonrecursive tt-degree.*

Proof Theorem 8.2.4 covers already the case of tt-degrees enumerable relative to K since these tt-degrees are all either hyperimmune or recursive. So it is sufficient to consider the case of tt-degrees not enumerable relative to K.

Let h, C, A and t_i be as in Definition 8.3.1. Now let, for every rational $q > 1$, the set D_q contain all indices of trees where, for every branching node x of t_i, there are not more than $q^{|x|}$ leaves above $x0$.

Clearly each set D_q is compatible with C and each set D_q is computable. Furthermore, whenever $q < r$, then $C \cap D_q$ is m-reducible to $C \cap D_r$ via the following function f: In the case that $i \in D_q$, then i is also in D_r and one takes $f(i) = i$; in the case that $i \notin D_q$ then there is some $j \notin C$ and $f(i) = j$ for this j. So the main task is to show the following:

$$C \cap D_q \not\leq_{btt} C \cap D_r \text{ for } q > r.$$

A consequence of this result is that the ordering of the classes $C \cap D_q$ is dense. In particular, there exist ascending and descending chains: they are given by the sequences $q = 2, 3, 4, \ldots$ and $q = 2, 1.1, 1.01, 1.001, 1.0001, \ldots$ for the parameter q of the classes $C \cap D_q$.

Assume by way of contradiction that $q > r$ but $C \cap D_q \leq_{btt} C \cap D_r$. Let T be the coenumerable tree and k be the constant from Theorem 8.3.2. Since A is not computable relative to K, there is above every $x \in A$ a full binary tree s_x of arbitrary finite depth embeddable into T [87, Lemma 1]. Say that the depth of s_x is so large that the number of its leafs m_x is between $2k \cdot r^{|x|} + 2$ and $4k \cdot r^{|x|} + 4$. The set of all those $x \in T$ where such a tree s_x exists is enumerable relative to K. If, for all the $x \in A$ with $x0 \notin A$ there would be only finitely many x such that s_{x0} exists then A would have a Turing degree enumerable relative to K: there would be a largest node $y \in A$ such that there is an $x \in A$ with $x0 \notin A$, s_{x0} exists and $y \preceq x$. It would follow that A is represented by the leftmost branch of the K-recursive tree

$$\{x \in T : x \preceq y \vee (x \succeq T \wedge s_x \text{ exists})\}$$

and thus the truth-table degree of A would be enumerable relative to K in contradiction to the choice of A.

Note that, for sufficiently large x, the number $q^{|x|}$ is larger than the number of leafs of s_{x0} whenever s_{x0} exists. There are k nodes $x_1, x_2, \ldots, x_k \in A$ such that for all these nodes x_l the node x_l0 is not in A but the tree s_{x_l0} exists and has less than $q^{|x_l|}$ leaves. Let u be a branching node of T which is in A and above all x_l1. The finite tree t containing all nodes $y \preceq u$ plus those which are below a branching node z of some s_{x_l} has the following property: whenever a node $x \in t$ is a branching node of t then the node $x0$ is in some tree s_{x_l} and there are at most $q^{|x|}$ leaves above $x0$. So it follows that there is an $i \in D_q$ with $t = t_i$.

By assumption, there are trees $t_{j_1}, t_{j_2}, \ldots, t_{j_k}$ whose union covers t_i and whose indices $j_1, j_2, \ldots, j_k$ are in D_r. For every $x_l \in \{x_1, x_2, \ldots, x_k\}$ those trees among the trees $t_{j_1}, t_{j_2}, \ldots, t_{j_k}$ having a branching node at x_l cover at most $r^{|x_l|}$ leaves of the more than $k \cdot r^{|x_l|}$ leaves of t_i above x_l0; so for each x_l some of the trees $t_{j_1}, t_{j_2}, \ldots, t_{j_k}$ must contain the node x_l0 without containing x_l1. As a consequence, the node u is not contained in any of the trees $t_{j_1}, t_{j_2}, \ldots, t_{j_k}$ and their union does not cover t_i. It follows that $C \cap D_q \not\leq_{btt} C \cap D_r$ ∎

Theorem 8.3.4 *Every nonrecursive tt-degree contains an infinite antichain of btt-degrees.*

Proof The case of hyperimmune tt-degrees enumerable relative to K is again covered by Theorem 8.2.4. The case of hyperimmune-free tt-degrees is contained in the case of

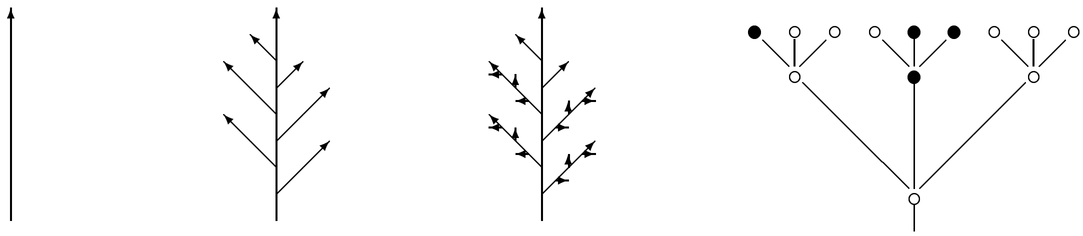

Figure 2: (a) Some trees of rank 0, rank 1 and rank 2. (b) Embedding a binary tree.

tt-degrees which are not enumerable relative to K. For this case consider the sets $E_m = \{i \in D_{(m+1)/m} : rank(t_i) \leq m+3\}$ where t_i has rank k or more iff the k-th full binary tree can be embedded topologically into t_i — Figure 2 (a) on page 132 gives some examples of trees of small rank.

Now one shows that $E_m \cap C$ and $E_n \cap C$ are btt-incomparable unless $m = n$. Let $m < n$. The direction that $E_m \cap C \not\leq_{btt} E_n \cap C$ needs one adaptation: the s_x should be built as trees of rank 2 so that the constructed tree t_i has at most rank 3. This can be obtained by first embedding a tree of depth 2^{m_x} into T above x and then taking for s_x all nodes z for which there is at most one node $y \prec z$ which is a branching node but not on the left-most branch. This subtree s_x has then m_x leaves. Still, every $x \in A$ has a tree s_x and the rest of the argumentation is absolutely parallel to Theorem 8.3.3. For the other direction, assume by way of contradiction that $E_n \cap C \leq_{btt(k)} E_m \cap C$.

For every finite tree t and every sufficiently large x, the expression $(\frac{n+1}{n})^{|x|}$ is greater than the number of t's leaves. So one can build a topologically equivalent copy of t below sufficiently large x and so obtain that this copy is in $D_{(n+1)/n}$. Thus one can reduce the proof to the pure topological argument that there is a tree t of rank $n+3$ which cannot be covered by a union of k trees of rank $n+2$; note that $n \geq m+1$ and thus $m+3 \leq n+2$. For this abstract topological argumentation, the tree t may be represented by some finite set of strings, omitting nonbranching nodes. The arity of t is increased by replacing each string 0^j1 simply by j.

Now let t contain all strings in $\{0, 1, \ldots, k\}^*$ whose length is at most $n+3$. A full binary tree of rank $n+4$ cannot be embedded into this tree since then one would need strings of length $n+4$, so the rank of t is at most $n+3$. Assume now that k trees $t_1, t_2, \ldots, t_k$ of rank $n+2$ would cover the tree t. Then each leaf y receives a colour j if $y \in t_j$. If there are several possibilities, the choice is arbitrary. Now whenever all successors $y0, y1, \ldots, yk$ of a node y have a colour, then at least two of them have the same colour. So one elects for y a colour such that two successors ya, yb of y have the same colour. This procedure is continued until the root has a colour j. Now it is immediate that each node of the tree t_j with colour j has at least two successors on the next level, so t_j contains the $n+3$-rd full binary tree and has rank $n+3$ in contradiction to the assumption. Figure 2 (b) on page 132 illustrates this process.

This argumentation gives that $E_n \cap C \not\leq_{btt} E_m \cap C$. So the sequence $E_1 \cap C, E_2 \cap C, \ldots$ is an antichain of bounded truth-table degrees within any given tt-degree not enumerable relative to K. ∎

Theorems 8.2.4 and 8.3.3 give either a m-reduction from one set to the other or show that there is no btt-reduction from one set to the other. So they extend Jockusch's result [70] who constructed an infinite ascending chain of m-degrees within every given nonrecursive tt-degree in the way that there is also an infinite descending chain. Theorem 8.3.4 also shows that there is an infinite antichain of m-degrees solving a problem left open by Jockusch [70].

Theorem 8.3.5 *Every nonrecursive tt-degree contains an infinite antichain of m-degrees.*

8.4 On Least Bounded Truth-Table Degrees

It was shown that tt-degrees contain infinite chains and antichains of btt-degrees. Further it is well-known that every tt-degree contains a greatest btt-degree given by the tt-cylinder. Now the attention turns to the other side: Does every tt-degree also contain a least btt-degree? The answer is: some do, but not all. Kobzev [84] already approximated this question by showing that some tt-degrees have a minimal btt-degree — minimal in the sense that there are no nontrivial btt-degrees below that but that there may still exist some incomparable further btt-degrees. The first result of this section shows that this can happen. The second result shows that on the other side there are tt-degrees which have even a least btt-degree: every set tt-reducible to this btt-degree is either recursive or above this btt-degree with respect to btt-reducibility. This second result will give a hyperimmune-free tt-degree, therefore this tt-degree is already a Turing degree. So there is a Turing degree with a least and a greatest btt-degree.

Example 8.4.1 *There is a tt-degree which does not have a least btt-degree.*

Proof Kobzev [84, Theorem 1] constructed a simple but not hypersimple set D which has minimal btt-degree in the following sense: Every set $E \leq_{btt} D$ is either recursive or btt-equivalent to D. Thus whenever the tt-degree has a least btt-degree, this btt-degree must contain D. The tt-degree of D has a semirecursive set B. By Proposition 8.1.5, D is not btt-reducible to B and therefore D does not have least btt-degree within its tt-degree. In particular the tt-degree of D does not have a least btt-degree. ∎

So one has tt-degrees without a least btt-degree. The next result shows that some other tt-degrees do have a least btt-degree.

Theorem 8.4.2 *There is a nonrecursive set A such that $A \leq_{btt(2)} D$ for all sets D within the same tt-degree.*

Proof The main idea of the proof is the construction of a tree T such that for a suitable branch X of T the set $A = \{X(0)X(1)\ldots X(n) : n \in \mathbb{N}\}$ represents the least btt-degree inside the tt-degree of X. It is equivalent to consider recursive or coenumerable trees — the coenumerable trees are somehow more practical since one can assume that they do not have dead ends: Each node $x \in T$ is on some infinite branch of T.

The construction of T uses movable markers similar to the construction of maximal sets [111, Theorem III.4.18]. While in the construction of the maximal sets the markers represent the intended nonelements of the enumerated set, the markers represent in this context the intended branching nodes of the tree: A tree can be viewed as a function which assigns to every $\sigma \in \{0,1\}^*$ a node $T(\sigma)$ such that the branching nodes are exactly the range of this function and that $T(\sigma a) \succeq T(\sigma)a$ for all $\sigma \in \{0,1\}^*$ and $a \in \{0,1\}$. This function T associated with the tree T is partial K-recursive; within this chapter the only such trees considered are where the associated function is also total.

Now $h_e[x](a)$ is defined as in Definition 8.1.2. The update-rule for the markers is that they move either if the move is induced by some markers below which also move or if they can increase their state where the l-th state in dependence of the current positions $m_{\sigma,s}$ of all markers with $\sigma \in \{0,1\}^l$ is the number of all different functions $\alpha_a : \sigma \in \{0,1\}^l \to h_e[m_{\sigma,s}](a)$ where $e \leq l$, $a \leq \max(\{m_{\sigma,s} : \sigma \in \{0,1\}^l\})$ and $h_e[m_{\sigma,s}](a) \in \{0,1\}$ for all $\sigma \in \{0,1\}^l$. Note that the l-th state is a natural number below $(l+1)2^{2^l}$ and that the markers with indices of length l have the l-th state in common which they maximize by moving simultaneously. The markers with longer indices might be forced to move because of moves below. Formally, these two types of moves are described by the following algorithm executed at stage s:

First: the original moves. At stage s it is checked for each length $l < s$ whether the l-th state can be increased if the markers with indices of length l move from the current position $m_{\sigma,s}$ to some position $m_{\tau,s}$ with $\tau \succeq \sigma$ and $|\tau| \leq s$. If such l does not exist, nothing is done and the algorithm proceeds with stage $s+1$. Otherwise one takes the shortest such l and every m_σ moves to the corresponding node $m_{\tau,s}$: $m_{\sigma,s+1} = m_{\tau,s}$. So if $l = 2$ and if for $\sigma = 00, 01, 10, 11$ the corresponding τ are $0000, 0101, 1001, 111$ then $m_{00,s+1} = m_{0000,s}$, $m_{01,s+1} = m_{0101,s}$, $m_{10,s+1} = m_{1001,s}$ and $m_{11,s+1} = m_{111,s}$.

Second: all markers with indices longer than l are adapted: If the marker m_σ moves from the position $m_{\sigma,s}$ to $m_{\tau,s}$ then $m_{\sigma\eta,s+1} = m_{\tau\eta,s}$ for all $\eta \in \{0,1\}^*$. So $m_{101,s+1} = m_{10011,s}$ for the example of the previous paragraph.

Third: the markers below level l do not move. With the movement of the markers, also the enumeration of the complement of T is given. The tree T_s is a superset of T which corresponds directly to the positions of the markers at stage s:

$$x \in T_s \Leftrightarrow (\exists \sigma \in \{0,1\}^{|x|})\,[x \preceq m_{\sigma,s}].$$

The sequence $T_0, T_1, \ldots$ is uniformly recursive, descending and converges pointwise to the tree T: As in the original construction of maximal sets it can be shown that every marker moves only finitely often and converges to a final position, this position is $T(\sigma)$ for the marker m_σ.

Given any path X through T, given the corresponding $A = \{x : x \preceq X\}$ and given some $D \leq_{tt} A$ via some reduction h_e, it can be shown that either D is recursive or there are, for all sufficiently large $x \in A$ and for the shortest σ with $x \preceq T(\sigma)$, two numbers a and b such that

- $h_e[T(\sigma)](a) = 0$ and $h_e[T(\sigma)](b) = 1$;
- $h_e[T(\tau)](a) = h_e[T(\tau)](b)$ for the other strings τ of the same length as σ.

Such a, b are said to *isolate* $T(\sigma)$ and x. Now it is shown that provided D is not recursive almost all $x \in A$ are isolated by some a and b. Let x be so large that the first σ with $T(\sigma) \succeq x$ has at least length e. Now one of the following two cases hold:

(I): there are c and $y, z \in T$ such that $y, z \succeq T(\sigma)$, $h_e[y](c) = 0$ and $h_e[z](c) = 1$. Then y, z must be incomparable with respect to $\preceq$. For each further τ of the same length as σ, there is some $x_\tau \in T$, $x_\tau \succeq T(\tau)$, such that $h_e[x_\tau](c) \in \{0,1\}$. Since the marker m_σ and all the markers m_τ remain on the positions $T(\sigma)$ and $T(\tau)$, respectively, they do not increase their state by moving to the values y, z in case of m_σ and x_τ in the case of m_τ. So there are a, b such that $h_e[T(\tau)](a) = h_e[T(\tau)](b) = h_e[x_\tau](c)$, $h_e[T(\sigma)](a) = h_e[y](c) = 0$ and $h_e[T(\sigma)](b) = h_e[z](c) = 1$; that is, a, b isolate $T(\sigma)$.

(II): for every c and for all $y, z \succeq T(\sigma)$ such that $y, z \in T$ either the values $h_e[y](c)$ and $h_e[z](c)$ are equal or one of them is "?". In this case D is recursive: There is some level s such that no query at the computation of $D(c)$ is beyond s. So $h_e[y](c) \in \{0,1\}$ for all the $y \succeq x$ of this level and then the complement of T is enumerated until a stage $t > s$ is found such that for all $y \succeq x$ with $|y| = s$ and $y \in T_t$, the function $h_e[y](c)$ takes a unique value which must be $D(c)$.

Now it is shown that it is possible to give a sequence $\eta_0, \eta_1, \ldots$ such that for the infinite branch $X = \lim_n T(\eta_n)$ and for set $A = \{x : x \preceq X\}$ it holds that every $D \leq_{tt} A$ either is recursive or satisfies $A \leq_{btt(2)} D$. Let $\eta_0 = 0$.

Now for each η_n there are two cases: Either every $x \in T$ with $x \succeq T(\eta_n)$ is isolated by some a and b. Then one just takes some arbitrary $\eta_{n+1} \succeq \eta_n$ whose length is at least $n+1$. Or there is some $x \in T$ with $x \succeq T(\eta_n)$ which is not isolated by any a, b. Then one takes $\eta_{n+1} \succeq \eta_n$ such that $x \preceq T(\eta_{n+1})$ and $|\eta_{n+1}| \geq n+1$.

In the second case, D is recursive by the argument given in (II). In the first case one knows that for every $x \in T$ with $x \succeq T_e(\eta_n)$ it is possible to find a and b which isolate x. Since the condition of being isolated is only enumerable and not computable, one works with the approximation that "a, b are isolated at stage s". This notion is formalized by requiring that

- $h_e[T_s(\sigma)](a) = 0$ and $h_e[T_s(\sigma)](b) = 1$;
- $h_e[T_s(\tau)](a) = h_e[T_s(\tau)](b)$ for the other strings τ of the same length as σ.

This notion is used in the following algorithm.

Search for the first s, a, b such that one of the following cases holds.

- $x \notin T_s$. Then output "$A(x) = 0$".
- x is incomparable to $T(\eta_n)$, that is, $x \not\preceq T(\eta_n) \wedge x \not\succeq T(\eta_n)$. Then output "$A(x) = 0$".
- $x \preceq T(\eta_n)$. Then output "$A(x) = 1$".
- $x \succ T(\eta_n)$ and $x \in T_s$ and a, b isolate x at stage s. Query $D(a), D(b)$ and output "$A(x) = 0$" if $D(a) = D(b)$ and output "$A(x) = 1$" if $D(a) \neq D(b)$.

The correctness of the algorithm in the first three cases is based on the observation that $T(\eta_n) \in A$, $A \subseteq T \subseteq T_s$ and A contains the nodes $x \preceq T(\eta_n)$ but no nodes incomparable to $T(\eta_n)$. Since every node in T above $T(\eta_n)$ is isolated by some a, b, the search terminates and a, b, s are found such that the algorithm goes into the fourth case, that is, a, b isolate x at stage s and still $x \in T_s$. The algorithm terminates always. The correctness of the algorithm in the fourth case is verified by considering the following two subcases.

- $x \in A$. Since a, b isolate x at stage s it holds that $h_e[x](a) = 0$ and $h_e[x](b) = 1$. It follows that $D(a) \neq D(b)$ and the algorithm returns the correct value for $A(x)$.
- $x \notin A$. There is an $y \in A$ such that $h_e[y](a) = D(a)$ and $h_e[y](b) = D(b)$. This y is in T_s and is incomparable to x. Thus $h_e[y](a) = h_e[y](b)$ and $D(a) = D(b)$. Again the algorithm returns the correct value for $A(x)$.

This completes the proof that $A \leq_{btt(2)} D$. ∎

Corollary 8.4.3 *There is a set A such that $A \leq_{btt(2)} D$ for all sets D within the same Turing degree.*

Proof This result is based on the fact that tt-degrees and Turing degrees are identical for sets of hyperimmune-free Turing degree. So it is sufficient to show that the construction in Theorem 8.4.2 can be adapted such that A has hyperimmune-free Turing degree.

First the notion of l-state is adapted. Now a more complicated l-state is defined by adding at stage s to the original l-state the number of all triples (e, x, σ) such that $e, x \leq |\sigma|$ and the computation $\varphi_e^X(x)$ terminates within $|m_{\sigma,s}|$ steps for some oracle X where $m_{\sigma,s} \preceq X$ and the computation does not ask queries to X which are longer than $|m_{\sigma,s}|$ — that is these queries to X can be answered by using the current position $m_{\sigma,s}$ so that X is introduced here only for formal reasons.

As a consequence, the marker m_σ also moves if this move makes some value $\varphi_e^X(u)$ defined for some $e, u \leq |\sigma|$. From this behaviour it follows that whenever $\varphi_e^X(u)$ is undefined for some u then there is some node $y \preceq X$ such that $\varphi_e^Y(u)\uparrow$ for all infinite branches $Y \succeq y$ of T.

Now the sequence $\eta_0, \eta_1, \ldots$ is replaced by a sequence $\eta_0, \eta_0', \eta_1, \eta_1', \ldots$ where η_e' is always an extension of η_e elected by the following condition: If there is a node $x \succeq T(\eta_e)$ in T such that $\varphi_e^X(u)\uparrow$ for all infinite branches X of T extending x then η_e' is taken to be the shortest string extending η_e such that $x \preceq T(\eta_e')$. Otherwise $\eta_e' = \eta_e$. After doing this step, η_{e+1} is derived from η_e' in the same way as η_{e+1} is derived from η_e in Theorem 8.4.2.

The condition that $A \leq_{btt(2)} D$ for every D in the tt-degree of A still holds. But now A and X do also have hyperimmune-free Turing degree: If $D \leq_T X$ via φ_e^X then one knows that φ_e^Y is also total for every infinite branch Y of T which extends $T(\eta_e')$. So one can make the computation total for all oracles by either taking $\varphi_e^Y(u)$ or outputting just 0 in the case that $T(\eta_e') \not\preceq Y$ or that some prefix $y \preceq Y$ is enumerated into $\overline{T}$ before the computation $\varphi_e^Y(u)$ terminates. The computation remains unchanged for the oracle X. So $D \leq_T X$ by a machine which terminates for every oracle and thus $D \leq_{tt} X$ [109, 143]. It follows that X has the greatest tt-degree among the tt-degrees inside X's Turing degree. So X has hyperimmune-free Turing degree [70, Theorem 8]. ∎

8.5 An Application of Least btt-Degrees

In the previous section a Turing degree was constructed such that it contains a retraceable set A which is btt(2)-reducible to any set inside it. This result is used to answer an open problem of Beigel, Gasarch and Owings [15], they asked whether every nonrecursive Turing degree contains an objective set. Before indicating how this is refuted, the definitions of k-subjective and objective sets from [15] are included.

Definition 8.5.1 A set D is k-subjective iff an algorithm ψ is computing arbitrary large parts of the characteristic function of D with k nondeterministic queries. The nondeterminism can be brought into the algorithm ψ via an additional input $z \in \mathbb{N}$:

> The input is a list of inputs $(x_1, x_2, \ldots, x_n)$ and a number z.
> During the computation, ψ may make up to k queries to D.
> For some numbers z, ψ outputs $(D(x_1), D(x_2), \ldots, D(x_n))$.
> For all other numbers z, ψ outputs the symbol "?" or does not terminate.

The opposite notion to k-subjective is that of an objective set: D is objective iff the n-fold characteristic function $(D(x_1), D(x_2), \ldots, D(x_n))$ cannot be computed with $n-1$ nondeterministic queries of the type above for any n. So every set is either k-subjective for some k or objective, but never both.

Beigel, Gasarch and Owings [15] asked whether every Turing degree contains an objective set. The next proposition is the key to decide this conjecture.

Proposition 8.5.2 *If the retraceable set $A = \{X(0)X(1)\dots X(n) : n \in \mathbb{N}\}$ represents the least btt-degree inside its Turing degree then the Turing degree of A does not contain objective sets.*

Proof Let D be an arbitrary set within the Turing degree of A. There is a k such that $A \leq_{btt(k)} D$, that is, $A(x) = g(x, D(f_1(x)), D(f_2(x)), \dots, D(f_k(x)))$ for all x. As already mentioned in Definition 8.1.1 the sets $\{y : y \preceq z\}$ and A coincide below z for all $z \in A$. Let $\phi(x, z)$ be the value of the Turing reduction from D to A if $\{y : y \preceq z\}$ instead of A is used as an oracle and no arguments $y > z$ are queried. Note that $\phi(x, z) = D(x)$ if $z \in A$ and ϕ is defined, that is, if no values $y > z$ are queried. Now the following algorithm ψ witnesses that D is indeed k-subjective.

$$\psi(x_1, x_2, \dots, x_n, z) = \begin{cases} (y_1, y_2, \dots, y_n) & \text{if } g(z, D(f_1(z)), D(f_2(z)), \dots, D(f_k(z))) = 1 \\ & \text{and } \phi(x_i, z) \text{ terminates within } z \text{ stages} \\ & \text{with output } y_i \text{ for all } i; \\ ? & \text{otherwise.} \end{cases}$$

The k queries to D check whether $z \in A$. If $z \notin A$ then ψ outputs the symbol "?" and so does not produce a wrong hypothesis in this case. If z is too small, that is, if some of the computations for $D(x_i)$ do not terminate within z steps or need some query to some value larger than z, then ψ also outputs the symbol "?" and avoids to produce a wrong hypothesis. In the remaining case that $z \in A$ and z is sufficiently large — such z exist since A is infinite — the algorithm can completely simulate the Turing reduction of D to A for all x_i and output the correct values $D(x_i)$. So D is k-subjective and not objective. ∎

The set A from Corollary 8.4.3 has hyperimmune-free Turing degree, is not recursive and is btt(2)-reducible to every set inside its Turing degree. Therefore one has according to the previous proof that every set in the Turing degree of A is even 2-subjective.

Theorem 8.5.3 *Some nonrecursive Turing degree consists only of 2-subjective sets. In particular, it does not contain any objective set.*

It is straightforward to show that every hyperimmune Turing degree contains an objective set. Beigel, Gasarch and Owings have done this for nonrecursive enumerable Turing degrees [15, Theorem 24]. The following Theorem shows that no nonrecursive Turing degree consists only of 1-subjective sets; thus the bound 2 in Theorem 8.5.3 is optimal.

Theorem 8.5.4 *Every semirecursive 1-subjective set is enumerable or coenumerable. Every nonrecursive Turing degree contains a set which is not* 1*-subjective.*

Proof Let B be a semirecursive 1-subjective set. B is the initial segment of some linear and recursive ordering $\sqsubseteq$. Now two enumerable sets $B_0 \subseteq \overline{B}$ and $B_1 \subseteq B$ are constructed with the intention that $B = B_1$ or $\overline{B} = B_0$. So either B_0 witnesses that B is coenumerable or B_1 witnesses that B is enumerable.

Let $f(x, y : z, c) = (a, b)$ denote that there is a computation which queries B at z and converges to (a, b) if the answer c is supplied to the query. Now one defines

$$\begin{aligned} B_0 &= \{y : (\exists x, z \sqsubseteq y)\,[f(x, y : z, 1) \in \{(0,0), (1,0)\}]\}; \\ B_1 &= \{x : (\exists y, z \sqsupseteq x)\,[f(x, y : z, 0) \in \{(1,0), (1,1)\}]\}. \end{aligned}$$

If $y \in B_0$ then there are $x, z \sqsubseteq y$ such that $f(x, y : z, 1) \in \{(0,0),(1,0)\}$. If $z \in B$ then $y \notin B$ by the correctness of f. If $z \notin B$ then $y \notin B$ by $z \sqsubseteq y$. So $B_0 \subseteq \overline{B}$ and by a symmetric argument $B_1 \subseteq B$.

Let $x \in B$ and $y \notin B$. Immediately it follows that $x \sqsubseteq y$. There is a z such that $f(x, y : z, B(z)) = (1, 0)$. If $B(z) = 1$ then $x, z \sqsubseteq y$ and $y \in B_0$. If $B(z) = 0$ then $x \sqsubseteq y, z$ and $x \in B_1$. It follows that either $B_0 = \overline{B}$ or $B_1 = B$. So B is enumerable or coenumerable.

For the second statement of the theorem an arbitrary nonrecursive Turing degree is considered. If it is an enumerable degree then it contains an objective set [15, Theorem 24] and otherwise it contains a semirecursive set which is neither enumerable nor coenumerable. This semirecursive set is also not 1-subjective. ∎

8.6 A Positive Result About Positive Degrees

The result that every tt-degree contains infinitely many btt-degrees seems to be natural since for many notions the interrelation between two types of degrees is that the weaker one consists of either one or infinitely many of the stronger degrees. A 1-degree is a degree in which every two sets are many-one reducible to each other via a 1-1 function. Young [149] showed that an m-degree consists of either one or of infinitely many 1-degrees. Jockusch [70] showed that every nonrecursive tt-degree consists of infinitely many m-degrees and that a Turing degree consists of either one (hyperimmune-free case) or infinitely many (hyperimmune case) tt-degrees.

Dëgtev [35] constructed an enumerable tt-degree which contains exactly one enumerable m-degree. So he shows that Jockusch's result on infinitely many m-degrees inside each tt-degree does not hold if one considers enumerable sets only. Cholak and Downey [32, 39, 40] extended this result by showing that for each n there are infinitely many enumerable tt-degrees containing exactly n enumerable m-degrees.

Within this section, a similar result is shown for positive versus truth-table reducibility within the general (nonenumerable) world. Jockusch [69, Corollary 4.3 (iv)] showed that every nonrecursive tt-degree contains at least three positive degrees which are represented by the sets A (or C), B and $\overline{B}$. The main result of this section is that this lower bound is optimal, that is, some nonrecursive tt-degree consists of exactly these three positive degrees.

Theorem 8.6.1 *Given $n \geq 1$, let m be the number of partial, transitive and irreflexive orderings on $\{0, 1, \ldots, n\}$. Then there is a nonrecursive tt-degree consisting of exactly m positive degrees. In particular, there are tt-degrees consisting of* 3*,* 19*,* 219*,* 4231*,* 130023 *and* 6129859 *positive degrees.*

Proof The construction follows Spector's basic way to construct minimal Turing degrees [111, Chapter V.5] via a descending sequence $T_0 \supseteq T_1 \supseteq T_2 \supseteq \ldots$ of trees such that the intersection is then a retraceable set $A = \{X(0)X(1)\ldots X(n) : n \in \mathbb{N}\}$ given by the unique common infinite branch X which is on all these trees. In each tree T_e every node has one or two successors and above each node there is some node with two successors. So T_e is a full binary tree and may also be viewed as a function that maps binary strings σ to nodes $T_e(\sigma)$ in the way that $T_e(\sigma a) \succeq T_e(\sigma)a$ for all strings σ and $a \in \{0, 1\}$. The node $T_e(\sigma a)$ is in fact the first node above $T_e(\sigma)a$ which has again two successors.

In the construction of minimal Turing degrees [111, Definition V.5.8] the notion of e-splitting pairs (x, y) is important. Here (x_0, x_1) are e-splitting on T iff there is some

Number	Relations	Representation
$-$	$0 \leftrightarrow 1 \leftrightarrow 2$	$\emptyset$ (recursive case)
1	$0 \to 1 \to 2$	B_{012}
2	$0 \to 2 \to 1$	B_{021}
3	$1 \to 0 \to 2$	B_{102}
4	$1 \to 2 \to 0$	B_{120}
5	$0 \to 2 \to 1$	B_{201}
6	$0 \to 2 \to 0$	B_{210}
7	$0 \to 1,\ 0 \to 2$	$B_{012} \oplus B_{021}$
8	$1 \to 0,\ 1 \to 2$	$B_{102} \oplus B_{120}$
9	$2 \to 0,\ 2 \to 1$	$B_{201} \oplus B_{210}$
10	$1 \to 0,\ 2 \to 0$	$B_{120} \oplus B_{210}$
11	$0 \to 1,\ 2 \to 1$	$B_{021} \oplus B_{201}$
12	$0 \to 2,\ 1 \to 2$	$B_{012} \oplus B_{102}$
13	$0 \to 1$	$B_{012} \oplus B_{021} \oplus B_{201}$
14	$0 \to 2$	$B_{102} \oplus B_{012} \oplus B_{021}$
15	$1 \to 0$	$B_{102} \oplus B_{120} \oplus B_{210}$
16	$1 \to 2$	$B_{012} \oplus B_{102} \oplus B_{120}$
17	$2 \to 0$	$B_{120} \oplus B_{210} \oplus B_{201}$
18	$2 \to 1$	$B_{021} \oplus B_{201} \oplus B_{210}$
19	none	joining all six sets

Figure 3: The nineteen positive degrees at $n = 2$

input u such that every $X \succeq x_0$ and every $Y \succeq x_1$ satisfy $\varphi_e^X(u) \neq \varphi_e^Y(u)$. The set V of all triples (x, y, z) such that (x, y) is e-splitting, $z0 \preceq x$ and $z1 \preceq y$ is enumerable. In general such a set V satisfies the following properties:

- V is enumerable.
- If $(x_0, x_1, y) \in V$ then $ya \preceq x_a$ for $a = 0, 1$.
- If $(x_0, x_1, y) \in V$ and $z_a \succeq x_a$ for $a = 0, 1$ then $(z_0, z_1, y) \in V$.

A tree T_e is homogeneous for such a set V if either no $x_0, x_1, y \in T_e$ satisfy $(x_0, x_1, y) \in V$ or $(T_e(\sigma 0), T_e(\sigma 1), T_e(\sigma)) \in V$ for all σ.

The main difference to the constructions of Spector [136] and Shoenfield [131] as presented in [111, Chapter V.5] is now that the trees are not only made homogeneous for those sets V given by the notion of e-splitting but also for all others and so covering the necessary steps for the requirements to obtain a tt-degree with only finitely many positive degrees. This construction then automatically satisfies that no branch is recursive; so it is not necessary to encode nonrecursiveness via an explicit separate requirement. Furthermore, the construction uses $n{+}1$-ary trees $T_e \subseteq \{0, 1, \ldots, n\}^*$ instead of binary trees. Also every node $x \in T_e$ either has exactly one successor in T_e or all $n + 1$ successors $x0, x1, \ldots, xn$ of x are in T_e — the latter type of node is called branching-node. Let $V_0, V_1, \ldots$ be an enumeration all enumerable sets V_e containing only legal $n{+}2$-tuples $(x_0, x_1, \ldots, x_n, y)$ and satisfying in addition $(z_0, z_1, \ldots, z_n, y) \in V_e$ whenever $(x_0, x_1, \ldots, x_n, y) \in V_e$ and $x_a \preceq z_a$ for $a = 0, 1, \ldots, n$, where a tuple is called *legal* if $x_a \succeq ya$ for $a = 0, 1, \ldots, n$.

> The formal construction of the trees T_e is inductive over $e = 0, 1, \ldots$ and starts with $T_0 = \{0, 1, \ldots, n\}^*$. The tree T_{e+1} is chosen as a recursive subtree of T_e which is homogeneous for V_e and which has above every node a branching node.

Such a tree T_{e+1} exists and can be constructed from T_e in the same way as Spector's e-splitting requirements [136] are satisfied. The method chosen here follows Odifreddi's way to construct T_{e+1} [111, Propositions V.5.5 and V.5.10] and is adapted to the case of trees over $\{0, 1, \ldots, n\}^*$.

Now one defines that a legal tuple $(x_0, x_1, \ldots, x_n, y)$ is *on a tree T_e above u* if all nodes $x_0, x_1, \ldots, x_n, y$ are in T_e and if $y \succeq u$. The construction of the tree T_{e+1} follows one of two cases.

First: For every $u \in T_e$ there is a tuple in V_e on T_e above u. In this case one can construct T_{e+1} inductively starting with $u = \lambda$ which is a node in T_e. A node is a momentary leaf if it is already fixed that $u \in T_{e+1}$ but one did not yet care on the nodes above u. At stage 0, λ is the momentary leaf. At stage $s + 1$ one takes the shortest momentary leaf u at stage s and looks for the first legal tuple $(x_0, x_1, \ldots, x_n, y) \in V_e$ which is on T_e and above u. Then one defines $z \in T_{e+1}$ for all z with $u \prec z \preceq x_a$ for some $a \in \{0, 1, \ldots, n\}$ and one defines $z \notin T_{e+1}$ for all nodes $z \succeq u$ which are incomparable to all strings $x_0, x_1, \ldots, x_n$ with respect to $\preceq$.

This construction satisfies the following observations: All added strings belong to T_e and so $T_{e+1} \subseteq T_e$. T_{e+1} has above every node z some node u which is a momentary leaf at some stage s and above this node u a branching node, which is y in this construction. Other branching nodes than these nodes y are not added to the tree T_{e+1}, therefore if $y = T_{e+1}(\sigma)$ then $T_{e+1}(\sigma a) \succeq x_a$ and so every tuple $(T_{e+1}(\sigma 0), T_{e+1}(\sigma 1), \ldots, T_{e+1}(\sigma n), T_{e+1}(\sigma))$ is in V_e. Thus T_{e+1} is homogeneous for the condition V_e.

Second: There is an $u \in T_e$ such that no tuple in V_e is on T_e above u. Now one just takes $T_{e+1} = \{x \in T_e : x \preceq u \vee u \preceq x\}$.

The tree T_{e+1} is also computable and is homogeneous for V_e since every legal n+2-tuple on T_{e+1} is above u and so by the choice of u not in V_e. In addition there is still above each node x a branching node y in T_e: If $x \succeq u$ then the property is just inherited from T_e since there is a branching node y above x in T_e and this node is also in T_{e+1} as well as its successors $y0$ and $y1$. If $x \preceq u$ then there is some branching node $y \succeq u$ in T_e and therefore in T_{e+1} and this node is also in T_{e+1}. Nodes incomparable to u are not in T_{e+1} by definition.

Note that for every k there is a set V_e containing all legal n+2-tuples $(x_0, x_1, \ldots, x_n, y)$ with $|y| > k$. Now above every node there is a tuple in V_e on T_e and thus the algorithm goes through the "First"-case. Then $u = T_{e+1}(\lambda)$ has at least length k. Now u is in all trees $T_{e'}$ with $e' > e$ the unique string of its length and thus $u \preceq X$. So the obtained sequence of trees has a unique common infinite branch X. From this X one defines $A = \{x : x \preceq X\}$ and $B = \{x : x \leq_{lex} X\}$.

In order to see that X is not recursive, consider any recursive infinite string $Y \in \{0, 1, \ldots, n\}^\infty$ and the set V_e containing all legal tuples $(x_0, x_1, \ldots, x_n, y)$ with $y \preceq Y$. The tree T_e contains a node $u \not\preceq Y$ since T_e has incomparable nodes. Now above u there is no tuple in V_e and thus the tree T_{e+1} is constructed according to the "Second"-case. It follows that Y is not an infinite branch of T_{e+1} and so $X \neq Y$. In particular X is not recursive.

Now it is shown that the tt-degree of X contains m positive degrees. Recall that m is the number of partial transitive and irreflexive orderings on $\{0, 1, \ldots, n\}$. This is shown by demonstrating that every set $D \leq_{tt} A$ can be associated with such an ordering and that two sets D, E from the tt-degree of X are in the same positive degree iff they have the same associate ordering.

For every $D \leq_{tt} A$, let h^D denote an associated truth-table reduction from D to A and define, for every x and a, the value $h^D[x](a)$ according Definition 8.1.2. Here the superscript "D" is added since also for some further set the corresponding function h^E is considered below. For any two digits b and c one enumerates a legal tuple $(x_0, x_1, \ldots, x_n, y)$ into a set $V_{b,c}$ iff there is an a such that $h^D[x_b](a) = 1$ and $h^D[x_c](a) = 0$. Now there is an e such that T_e is homogeneous for all sets $V_{b,c}$ where b, c are distinct digits from the set $\{0, 1, \ldots, n\}$. Let $b \to c$ denote that the implication

$$(\forall a)\,(\forall y)\,(\forall x_b \succeq yb)\,(\forall x_c \succeq yc)\,[h^D[x_b](a) = 1 \wedge x_b, x_c \in T_e \Rightarrow h^D[x_c](a) \in \{?, 1\}] \quad (32)$$

holds. The relation $\to$ defines a transitive ordering on $\{0, 1, \ldots, n\}$. Now the following is shown: (I) If $b \to c$ and $c \to b$ then D is recursive. (II) If the ordering induced by D is a subset of the ordering induced by E, that is, if whenever $b \to c$ holds for the ordering induced by D then it also holds for the ordering induced by E, then E is positive reducible to D. (III) If $b \to c$ holds for the ordering induced by D but not for the one induced by E then E is not positive reducible to D. (IV) For every transitive and irreflexive partial ordering there is a set D which is associated to this ordering.

(I): Assume that $b \to c$, $c \to b$ but $c \not\to d$ would hold for some pairwise different $b, c, d \in \{0, 1, \ldots, n\}$. Then there is an a such that $h^D[T_e(cc)](a) = 1$ and $h^D[T_e(cd)](a) = 0$. There is some $x \succeq T_e(b)$ such that $x \in T_e$ and $h^D[x](a) = 1$ since $T_e(cc) \succeq T_e(\lambda)c$, $x \succeq T_e(\lambda)b$, $c \to b$ and $h^D[x'](a) = ?$ for only finitely many x'. By $b \to c$ it follows now also that $h^D[T_e(cd)](a)$ is either the symbol "?" or 1 in contradiction to the choice of a. So this case cannot occur and one has for all a and for all $x \in T_e$ that $T_e[x](a)$ is either $D(a)$ or the symbol "?". Since the symbol "?" occurs only for finitely many x one can calculate $D(a)$ by just evaluating the computable function $h^D[x](a)$ for the $x \in T_e$ until some value in $\{0, 1\}$ is found. This completes (I).

A corollary of this result is that for nonrecursive D it never happens that there is a circle of the form $b \to c \to \ldots \to b$. So one can redefine that $b \not\to b$ for all $b \in \{0, 1, \ldots, n\}$ without loosing transitivity and so link to every nonrecursive set D an irreflexive and transitive ordering on $\{0, 1, \ldots, n\}$. The next steps (II) and (III) will show that every positive degree corresponds to exactly one irreflexive and transitive ordering.

(II): Assume that whenever $b \to c$ with respect to h^D, then the same implication would also hold with respect to h^E. Furthermore, let E be nonrecursive, since otherwise E is already clearly positive reducible to D. Now given any a, there is a level k such that $h^E[T_e(\sigma)](a) \in \{0, 1\}$ for all σ of length k. So let the set $One(a)$ contain all strings σ of length k with $h^E[T_e(\sigma)](a) = 0$ and the set $Zero(a)$ contain all strings τ of length k with $h^E[T_e(\tau)](a) = 0$. For each $\sigma \in One(a)$, for each $\tau \in Zero(a)$ and for their longest common prefix η, there are two digits b, c such that $T_e(\eta)b \preceq T_e(\sigma)$ and $T_e(\eta)c \preceq T_e(\tau)$. It follows that $b \not\to c$ for the ordering belonging to E. Now the same holds for the ordering belonging to D and so there is an $a_{\sigma,\tau}$ such that $h^D[T_e(\sigma)](a_{\sigma,\tau}) = 1$ and $h^D[T_e(\tau)](a_{\sigma,\tau}) = 0$. So one can define the following positive reduction from E to D at a:

$$E(a) = \bigvee_{\sigma \in One(a)} \ \bigwedge_{\tau \in Zero(a)} \ D(a_{\sigma,\tau}). \quad (33)$$

For the verification a case-distinction is made whether $E(a) = 0$ or $E(a) = 1$.

In the case $E(a) = 0$ one has that $T_e(\eta) \in A$ for some $\eta \in Zero(a)$. For any given σ the conjunction $\bigwedge_{\tau \in Zero(a)} E(a_{\sigma,\tau})$ is 0 since one of the τ equals η and $D(a_{\sigma,\eta}) =$

$h^D[T_e(\eta)](a_{\sigma,\eta}) = 0$. So all disjunctions in Equation (33) are evaluated to 0 and the equation is correct.

In the other case $E(a) = 1$ one has $T_e(\eta) \in A$ for some $\eta \in One(a)$. Now the corresponding conjunction $\bigwedge_{\tau \in Zero(a)} D(a_{\eta,\tau})$ ranges only over terms $D(a_{\eta,\tau})$ which are evaluated to 1 and thus the whole conjunction is evaluated to 1. The preceding disjunction preserves this 1 and so the equality above is correct also in this case.

The reduction given by Equation (33) is correct and positive for every a. The construction is recursive in the parameters a and k. The parameter e is a constant and the parameter k can be found effectively by inspecting one level after the other until all outputs are either 0 or 1 but do not take the undefined value "?". So one has a positive reduction from E to D. This completes the proof of (II).

(III): Let $b \to c$ for the ordering linked to the set D but not for the ordering linked to the set E. Assume by way of contradiction that E is positively reducible to D. For every σ there is an a such that $h^E[T_e(\sigma b)](a) = 1$ and $h^E[T_e(\sigma c)](a) = 0$. There is a level k such that for every a' queried during the positive reduction from $E(a)$ to D every expression $h^D[T_e(\tau)](a')$ is either 0 or 1 for all $\tau \in \{0, 1, \ldots, n\}^k$. So one can compute for every $\tau \in \{0, 1, \ldots, n\}^k$ the value u_τ which the reducibility would take under the assumption that $T_e(\tau) \in A$. If $u_\tau = 1$ for some $\tau \succeq \sigma b$ of length k then also $u_\eta = 1$ for all $\eta \succeq \sigma c$ since for every a' it holds that $h^D[T_e(\tau)](a') \leq h^D[T_e(\eta)](a')$ and since the reduction is positive. So one either has that for every τ extending σb the reducibility returns under the assumption that $T_e(\sigma b) \in A$ the wrong value 0 or one has that for every η extending σc the reducibility returns under the assumption that $T_e(\sigma c) \in A$ the wrong value 1. So it is possible to conclude that — depending on the actual value of the assumption — either $T_e(\sigma b)$ or $T_e(\sigma c)$ is not in A. So there is an enumerable set W which contains only elements x for which one knows that $x \notin A$. The set W contains in particular every string σ either the value $T_e(\sigma b)$ or the value $T_e(\sigma c)$.

Now one defines $V_{e'}$ such that $V_{e'}$ contains every legal tuple above every element of W. Since $V_{e'}$ has infinitely many indices, one can without loss of generality assume that $e' > e$. Now every branching node $T_{e'}(\tau)$ is equal to some $T_e(\sigma)$ and so either $T_e(\sigma b)$ or $T_e(\sigma c)$, say the first, belongs to W. It follows that $T_{e'}(\tau b) \succeq T_e(\sigma b)$ and therefore $T_{e'}(\tau b) \in W$. Therefore every legal tuple above $T_{e'}(\tau b)$ and so also some legal tuple above $T_{e'}(\tau)$ on $T_{e'}$ is in $V_{e'}$. So the algorithm to construct $T_{e'+1}$ takes the "First"-case in the construction above and every infinite branch on $T_{e'+1}$ goes through a node which witnesses that the given positive reduction from E to D fails. So the assumption that E is positive reducible to D is false and (III) holds.

Putting (II) and (III) together one obtains that every two sets D and E which are associated with the same partial and irreflexive ordering belong to the same positive degree but that those belonging to different orderings belong to different positive degrees. It remains to show that for every ordering there is also a set linked to it.

(IV): The ordering $\to$ on the digits can be extended to an ordering $\rightsquigarrow$ on all strings x, y by taking $x \rightsquigarrow y$ if $|x| = |y|$ and if either $x = y$ or there is a z with $zb \preceq x$, $zc \preceq y$ and $b \to c$. Now let $a \in D$ if $x \rightsquigarrow a$ for some $x \in A$ and let $a \notin D$ otherwise. It remains to show that the given ordering is just the one which is associated with D. According to Definition 8.1.2 it is possible to construct a tt-reduction from D to A and an associated function h^D such that the following holds: If $x \succeq y$ and $|y| = |a|$ then $h^D[x](a) = h^D[y](a)$, $h^D[y](a) = 1$ if $a \rightsquigarrow y$ and $h^D[y](a) = 0$ if $y \not\rightsquigarrow a$. If $|x| < |a|$ then $h^D[x](a) = h^D[y](a)$ in the case that this value is unique for all $y \succeq x$ of length $|a|$ and $h^D[x](a) = ?$ otherwise.

Now it is verified that for this particular function h^D and for any of the trees V_e the implication $b \to c$ holds iff Condition (32) is satisfied. So let b, c be two different digits in $\{0, 1, \ldots, n\}$.

If $b \to c$, then consider any a and any node y. If $|y| \geq |a|$ then one knows that $h^D[x](a) = h^D[y](a)$ for all $x \succeq y$. So the implication $h^D[x_b](a) = 1 \Rightarrow h^D[x_c](a) = 1$ holds for all $x_b \succeq yb$ and $x_c \succeq yc$. Now consider the case that $|y| < |a|$. If both strings $x_b \succeq yb$ and $x_c \succeq yc$ have the same length as a then $h^D[x_b](a) = 1$ if $a \rightsquigarrow x_b$ and $h^D[x_b](a) = 0$ if $a \not\rightsquigarrow x_b$. In the first case it also holds that $a \rightsquigarrow x_c$ by the transitivity of the relation $\rightsquigarrow$ and so $a \rightsquigarrow x_c$ follows. In particular if some $x_b \succeq yb$ satisfies $h^D[x_b](a) = 1$, so does some x_b of the same length as a and therefore every $x_c \succeq yc$ of length a and thus all $x_c \succeq yc$ map $h^D[x_c](a)$ to 1. So Condition (32) is satisfied.

If $b \not\to c$ then let $y = T_e(\lambda)$ and let $a = yb$. Since y is a branching node, the nodes yb and yc are both in T_e. Furthermore, $h^D[yb](a) = 1$ since $yb = a$. But $h^D[yc](a) = 0$ since $a \not\rightsquigarrow yc$ and $|a|$ and $|yc|$ have the same length. So Condition (32) is not satisfied.

From the case distinction it follows that $b \to c$ holds for different $b, c \in \{0, 1, \ldots, n\}$ if and only if Condition (32) is satisfied. So $\to$ is the relation associated with D. Furthermore, $x \in A$ iff the $a \in D$ of the same length as x are just those a with $x \rightsquigarrow a$. This definition specifies a unique x since $x \rightsquigarrow y \wedge y \rightsquigarrow x$ does not hold if $x \neq y$. So $A \leq_{tt} D$ and D has the same tt-degree as A. Thus it follows that for every transitive and irreflexive ordering $\to$ on the digits $\{0, 1, \ldots, n\}$ there is a set D which is associate to this ordering. This completes the proof of (IV).

Putting (I), (II), (III) and (IV) together one has that there are m positive degrees within the tt-degree of X where m is the number of irreflexive and transitive partial orderings on $\{0, 1, \ldots, n\}$. Sloane's encyclopedia [133] gives for $n = 1, 2, \ldots, 13$ the corresponding numbers m: 3, 19, 219, 4231, 130023, 6129859, 431723379, 44511042511, 6611065248783, 1396281677105899, 414864951055853499, 171850728381587059351 and 98484324257128207 032183. ∎

For $n = 1$ the 3 positive degrees are generated by A, B and $\overline{B}$. For $n = 2$ there are already 19 positive degrees; Figure 3 on page 139 displays the orderings linked to these degrees and also some representative sets. The six semirecursive sets denoted by B_{abc} are defined by the underlying ordering $a \to b \to c$ of the digits $0, 1, 2$ and the other positive degrees can be represented by joins of some of these six semirecursive sets.

The next result extends the knowledge of the possible cardinalities of the positive degrees inside tt-degrees. It in particular shows that not all numbers m are possible: there is no tt-degree consisting of an even and finite number of positive degrees.

Theorem 8.6.2 *Every tt-degree consists of an odd number or an infinite number of positive degrees. The recursive tt-degree consists of exactly one positive degree, some minimal tt-degrees consist of exactly three positive degrees and all other tt-degrees consist of at least five positive degrees. The tt-degree of K consists of infinitely many positive degrees.*

Proof Consider the mapping $D \to \overline{D}$ which assigns to every set its complement. First one should note that this mapping preserves degrees: If $D \equiv_p E$ then also $\overline{D} \equiv_p \overline{E}$. The main idea is that a positive formula to evaluate $D(x)$ from E at $x_1, x_2, \ldots, x_n$ can be turned into a positive formula to compute $\overline{D}$ from $\overline{E}$: The whole formula is negated at the output and at each input. Then one can move the negations over all "and" and "or" gates using the De Morgan Law and so obtain, that only double negations occur which can be left out.

The following example illustrates the procedure.

$$\begin{array}{rcll} D(x) & = & E(x_1) \wedge (E(x_2) \vee E(x_3)) & \Leftrightarrow \\ \overline{D}(x) & = & \neg(E(x_1) \wedge (E(x_2) \vee E(x_3))) & \Leftrightarrow \\ \overline{D}(x) & = & \neg(\neg\overline{E}(x_1) \wedge (\neg\overline{E}(x_2) \vee \neg\overline{E}(x_3))) & \Leftrightarrow \\ \overline{D}(x) & = & \neg\neg\overline{E}(x_1) \vee \neg(\neg\overline{E}(x_2) \vee \neg\overline{E}(x_3)) & \Leftrightarrow \\ \overline{D}(x) & = & \neg\neg\overline{E}(x_1) \vee (\neg\neg\overline{E}(x_2) \wedge \neg\neg\overline{E}(x_3)) & \Leftrightarrow \\ \overline{D}(x) & = & \overline{E}(x_1) \vee (\overline{E}(x_2) \wedge \overline{E}(x_3)) & \end{array}$$

So one obtains that negating both sets preserves positive reducibility between them and thus the negation maps positive degrees into positive degrees. Furthermore, the mapping is one-one since it is the reverse of itself. There are two cases:

(I): A set D is in the greatest positive degree within its tt-degree. Then any tt-reduction to D can be turned into a positive reduction to D and so $\overline{D}$ is positively reducible to D and vice versa. Thus the greatest positive degree within some tt-degree is mapped into itself.

(II): A set D does not belong to the greatest positive degree of its tt-degree. Since every set tt-reducible to $D \oplus \overline{D}$ is also positive reducible to $D \oplus \overline{D}$ it follows that $D \oplus \overline{D}$ belongs to a positive degree different than that of D. Then $\overline{D}$ belongs neither to the positive degree of D nor to that of $D \oplus \overline{D}$ but $\overline{D}$ is of course still in the same tt-degree as D.

Now let D_0 be in the greatest positive degree and let $D_1, D_2, \ldots, D_{2n}$ be positive degrees such that $D_{2m} = \overline{D_{2m-1}}$ for $m = 1, 2, \ldots, n$. If there is an other positive degree with D_{2n+1} representing it, then also the complement $D_{2n+2} = \overline{D_{2n+1}}$ is not in the positive degrees generated by $D_0, D_1, \ldots, D_{2n}$: If, for example, D_{2n+2} is in D_1 then D_{2n+1} is in the positive degree of the complement D_2 of D_1 which was previously excluded. So either $D_0, D_1, \ldots, D_{2n}$ represent all positive degrees or there are at least two more represented by D_{2n+1} and D_{2n+2}. Starting the induction with $n = 0$ one obtains that the number of positive degrees inside a truth-table degree is either a finite odd number or infinite.

The recursive degree consists by definition of just one positive degree. Every further tt-degree consists of at least three positive degrees [69, Corollary 4.3 (IV)] and by the preceding Theorem 8.6.1, this bound is already optimal: some minimal nonrecursive truth-table degree consists of three positive degrees. Now it is shown that nonminimal nonrecursive truth-table degrees have at least five positive degrees.

Let B be a semirecursive set representing some given nonminimal tt-degree. Then there is a nonrecursive set $C <_{tt} B$ which is a tt-cylinder. In the sequel it is shown that there is a chain of three positive degrees inside the tt-degree of B: the one of B, the one of $B \oplus C$ and the one of $B \oplus \overline{B} \oplus C$.

C is not positive reducible to B since any positive reducibility to the semirecursive set B can be turned into a many-one reducibility [69, Theorem 4.2 (II)], C would have to be semirecursive in contradiction to the choice of C as a nonrecursive tt-cylinder. So B is strictly below $B \oplus C$.

The proof that $\overline{B}$ is not positive reducible $B \oplus C$ is a bit more complicated. B is an initial segment of some recursive linear ordering $\sqsubset$. Assuming that $\overline{B}$ is positive reducible to $B \oplus C$ one replaces at the computation of $\overline{B}(x)$ every query to $B(y)$ be the constant 1 if $y \sqsubseteq x$ and by the constant 0 if $y \sqsupset x$. Call a the result of this procedure.

Now it is verified that $\overline{B}(x) = a$: If $\overline{B}(x) = 0$ then $B(x) = 1$ and thus whenever $B(y)$ was replaced by the answer 1 in the above algorithm, then this replacement was correct. But the replacement was perhaps incorrect at some places y where it assumed that $B(y) = 0$ — so the reduction replaced the queries to B by queries to some subset $\tilde{B}$ of B. Since the reducibility is positive, it follows that $a \leq \overline{B}(x)$ and since the reduction is $\{0,1\}$-valued, the output a takes the value $\overline{B}(x)$ which is 0. If $\overline{B}(x) = 1$ then $B(x) = 0$ and thus the algorithm computing a was correct when it assumed that $B(y) = 0$ but perhaps incorrect at some y where it assumed that $B(y) = 1$. So the reduction replaces the queries to B by queries to some superset $\tilde{B}$ of B. Therefore $a \geq \overline{B}(x)$ which is 1 and a is also correct in this case.

So a positive reduction from $\overline{B}$ to $B \oplus C$ can be turned into one to C alone. But then $\overline{B} \leq_{tt} C$ and $B \leq_{tt} C$ in contradiction to the choice of B and C. Thus one has that the given three positive degrees are properly above each other.

Two further positive degrees within the tt-degree of B are given by $\overline{B}$ and $\overline{B} \oplus C$. So if B does not belong to a minimal tt-degree then it contains at least five positive degrees.

The last part is to show that the tt-degree of K contains infinitely many positive degrees: Let C_i be the tt-cylindrifications of the uniformly enumerable sets G_i from Theorem 8.2.4 (b). Note that $C_i \oplus K \equiv_{tt} K$ since every enumerable set and thus also the tt-cylinder of every enumerable set is tt-reducible to K. So the sets $K \oplus C_i$ have all the same tt-degree as K.

Relative to the oracle C_j, the set $C_j \oplus K$ is enumerable but not the set $C_i \oplus K$ for $i \neq j$ since the tt-cylinder C_i is enumerable only relative to oracles above C_i. Since positive reducibility preserves enumerability [69, Proposition 3.5 (II)], it also preserves the property to be "enumerable relative to C_j" and so it follows that $C_i \oplus K$ is not positive reducible to $C_j \oplus K$. Thus the sets $C_0 \oplus K$, $C_1 \oplus K$, ... form an infinite antichain of positive degrees within the tt-degree of K. ▮

The tt-degree constructed in Theorem 8.6.1 is hyperimmune-free. So it is in fact a Turing degree.

Corollary 8.6.3 *There are nonrecursive Turing degrees consisting only of finitely many positive degrees.*

This does not hold for hyperimmune Turing degrees since these consist of infinitely many tt-degrees and thus also of infinitely many positive degrees. But it is natural to ask whether the result carries over to some hyperimmune tt-degrees. The next result shows that there are even enumerable tt-degrees consisting only of finitely many positive degrees.

Theorem 8.6.4 *There is an enumerable tt-degree consisting of exactly three positive degrees.*

Proof The construction consists of two major steps:

First a coenumerable tree T is constructed such that every node in it has either one or two successors. Furthermore, T is locally homogeneous for requirements V_e (as in the previous proof) in the sense that if X generates an infinite branch through T then $V_e(T(\sigma 0), T(\sigma 1), T(\sigma))$ is the same value for almost all $\sigma \preceq X$. Again this is obtained by using movable markers similar to the construction of maximal sets [111, Theorem III.4.18].

Second it is shown that some set $A \leq_m L$ where $L = \{x : (\exists k)\,[x \preceq T(0^k)]\}$ represents an enumerable tt-degree consisting of three positive degrees.

First, the construction of T is presented by describing how the markers move. The sets V_e are the same as in the proof of Theorem 8.6.1 (where the parameter n is fixed to 1) and $V_{e,s}$ is the set of all elements of V_e enumerated within s steps plus all tuples (x', y', z) for which there are $x \preceq x'$ and $y \preceq y'$ with $(x, y, z) \in V_e$. The markers m_σ move on T in order to maximize the following state:

$$state(e, x, y, z, s) = \textstyle\sum_{e'=0,1,\ldots,e} 2^{e-e'} V_{e',s}(x, y, z).$$

The tree T is initialized as $\{0,1\}^*$ with $m_{\sigma,0} = \sigma$ for all markers. At every stage s of coenumerating T, the following things are done: If there are a marker $m_{\sigma,s}$ and nodes $x, y, z \in T$ with $z0 \preceq x$, $z1 \preceq y$, $m_{\sigma,s} \preceq z$ and $state(|\sigma|, x, y, z, s) > state(|\sigma|, m_{\sigma,s}, m_{\sigma 0,s}, m_{\sigma 1,s}, s)$ then the marker with the first such σ, x, y, z moves in stage s where the strings are ordered $\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots$ in the standard way. The precise move of m_{sigma} is given by the equation $m_{\sigma,s+1} = z$. Furthermore, $m_{\sigma a \eta,s+1} = m_{\tau\eta,s}$ for the first string τ with $m_{\tau,s} \succeq x$ if $a = 0$ and with $m_{\tau,s} \succeq y$ if $a = 1$. All nodes $u \succeq m_{\sigma,s}$ which are incomparable to both, x and y, with respect to $\preceq$, are enumerated into the complement of T.

As in the construction of the maximal set it can be verified that every marker m_σ moves only finitely often and remains after these moves on the node equal to $T(\sigma)$. Furthermore, for each e the sequence $state(e, T(0^k0), T(0^k1), T(0^k), \infty)$ is descending for $k = e, e+1, \ldots$ where the symbol ∞ means that $V_e(x, y, z)$ is taken instead of $V_{e,s}(x, y, z)$ in the definition of the state above. Therefore the value $V_e(T(0^k0), T(0^k1), T(0^k))$ is the same constant c for almost all k.

Second, let X be the lexicographic first branch on T and let L be the corresponding retraceable subset of T: $L = \{x : x \preceq X\} = \{x : (\exists k)\,[x \preceq T(0^k)]\}$. Now the set of all nodes lexicographic before those in L including those in L is enumerable and there is a one-one enumeration f of these nodes which satisfies in addition that whenever x is enumerated then all $y \prec x$ have been enumerated before x.

Now the set $A = \{x : f(x) \in L\}$ is retraceable since it is one-one reduced to the retraceable set L preserving the ordering $\preceq$. The set $B = \{x : f(x) \notin L\}$ is also enumerable and is the complement of A. Furthermore, B is semirecursive since for every x, y it holds that whenever $x \in B$ and $f(y) \leq_{lex} f(x)$ then also $y \in B$. Every set D tt-reducible to A is also tt-reducible to L since $A(x) = L(f(x))$ for all x.

Now it is shown that the tt-degree of A consists of the three positive degrees given by A, $B = \overline{A}$ and $A \oplus B$. In particular it is shown that every $D \leq_{tt} A$ is either in one of these three positive degrees or is recursive.

So let h be a tt-reduction from D to L which is obtained by concatenating the original tt-reduction h' from D to A with the m-reduction f from A to L. For h there are two sets V and W given as

$$\begin{aligned} V &= \{(x,y,z) : z0 \preceq x \wedge z1 \preceq y \wedge (\exists a)\,[h[x](a) = 0 \wedge h[y](a) = 1\,]\,\}; \\ W &= \{(x,y,z) : z0 \preceq x \wedge z1 \preceq y \wedge (\exists a)\,[h[x](a) = 1 \wedge h[y](a) = 0\,]\,\}. \end{aligned}$$

There are now four cases which correspond to the positive degrees represented by the sets $A \oplus B$, A, B and $\emptyset$ (belonging to the recursive positive degree).

(I): $(T(0^k0), T(0^k1), T(0^k)) \in V, W$ for all $k \geq l$ for some fixed l. It is shown that D is in the positive degree of $A \oplus B$ which is the greatest positive degree since $B = \overline{A}$.

Now it is shown how to compute $A(x)$ for any given x. For all $k \geq l$ with $T(0^k) \preceq f(x)$ there is a number a such that $h[T(0^k0)](a) = 1$ and $h[T(0^k1)](a) = 0$. Therefore it is possible to give a positive reducibility from A to D:

$$A(x) = \begin{cases} 1 & \text{if } f(x) \preceq T(0^l); \\ 0 & \text{if } x \text{ is enumerated into } B \text{ before the next} \\ & \text{condition is satisfied;} \\ \min\{D(a_i)\} & \text{where the } i \text{ ranges over } \{l, l+1, \ldots, k\}, \\ & k \geq l,\ m_{0^l,s} = T(0^l),\ f(x) \preceq m_{0^k0,s}, \\ & h[m_{0^i1,s}](a_i) = 0 \text{ and } h[m_{0^i0,s}](a_i) = 1, \\ & \text{for the first stage } s \text{ where this is possible.} \end{cases} \tag{34}$$

The verification starts with showing that the computation terminates for all x. Obviously it terminates when $x \in B$ or when $f(x) \preceq T(0^l)$. The remaining x are in A and satisfy $f(x) \not\preceq T(0^l)$. For such an x there is a unique $k \geq l$ such that $f(x) \preceq T(0^{k+1})$ but $f(x) \not\preceq T(0^k)$. Now there are a_i for $i = l, l+1, \ldots, k$ such that $h[T(0^i1)](a_i) = 0$ and $h[T(0^i1)](a_i) = 1$ otherwise. Then all $D(a_i) = 1$ since $T(0^i0) \in L$ and so also $A(x) = 1$ is the correct output. So the search terminates and outputs the correct value.

Somehow one cannot be sure when this situation is found and so has to work with current marker positions as indicated in the Equation (34). Therefore it is necessary to verify that $A(x)$ is computed correctly whenever the value is computed by the third condition in the case distinction. If $A(x) = 1$ then $m_{0^k0,s} \in L$ and thus also $m_{0^i0,s} \in L$ for all $i \leq k$. Thus $D(a_i) = h[m_{0^i0,s}](a_i) = 1$ and the computation of $A(x)$ is correct. If $A(x) = 0$ then $m_{0^k0,s} \notin L$ and some marker $m_{0^i1,s}$ is currently in L. Then $D(a_i) = h[m_{0^i1,s}](a_i) = 0$ for this marker and $A(x)$ is also computed correctly. The case that the computation terminates with $A(x) = 0$ by enumerating x into B is also correct since $B = \overline{A}$ and the case that the algorithm outputs 1 because of $f(x) \preceq T(0^l)$ is also correct since $T(0^l) \in L$. So the correctness of the given reduction from A to D is verified.

Similarly one can give a positive reducibility from $B = \overline{A}$ to D. Note that every set E which is tt-reducible to A is also positive reducible to $A \oplus \overline{A}$. Since both parts of the join are positive reducible to D, E is also positive reducible to D itself. So D is in the greatest positive degree inside the truth-table degree of A.

(II): $(T(0^k0), T(0^k1), T(0^k)) \in V$ but $(T(0^k0), T(0^k1), T(0^k)) \notin W$ for all $k \geq l$ for some fixed l. The construction of the reduction from A to D in the previous case only uses the information that $(T(0^k0), T(0^k1), T(0^k)) \in V$ for $k \geq l$; this particular information is necessary to establish the third case in the Equation (34). So this reduction exists independently whether the triples $(T(0^k0), T(0^k1), T(0^k))$ are in W or not. So the same reduction exists also in this case (II) and it remains to show that D is positively reducible to A.

Every query within the tt-reduction h' of D to A which queries some x with $f(x) \preceq T(0^l)$ can be replaced by the constant 1 and which queries some x with $f(x)$ incomparable to $T(0^l)$ can be replaced by 0. Furthermore, one can enumerate B and replace any query to some $x \in B$ by 0 until a stage s is reached where there is a marker m_{0^k} such that all queries x to A satisfy $m_{0^l,s} \preceq f(x) \preceq m_{0^k,s}$ and $T(0^l) = m_{0^l,s}$. Now one knows that $D(a) = h[m_{0^i1}](a)$ for some $i \in \{l, l+1, \ldots, k\}$; formally also $D(a) = h[m_{0^k}](a)$ is possible but since this value is not "?", it follows that $h[m_{0^k1}](a) = h[m_{0^k}](a)$. From the fact that no triple (x, y, z) with $z = T(0^k)$ for some $k \geq l$ is in W, it follows that $h[T(0^i1)](a) \geq h[T(0^j1)](a)$ whenever $l \leq i \leq j$, so one can also continue the enumeration process if also $h[m_{0^i1,s}](a) \geq h[m_{0^j1,s}](a)$ for all i, j with $l \leq i \leq j$. Now either $h[m_{0^i1,s}](a)$ is a constant b for $i = l, l+1, \ldots, k$ and one knows that $D(a) = b$ or there is a maximal i with $h[m_{0^i1,s}](a) = 0$. There is a unique x with $f(x) = m_{0^i,s}0$. If $x \in A$ then $m_{0^i0,s}$ must be in L and $D(a) = 1$. If $x \notin A$ then $m_{0^j1,s} \in L$ for some $j \leq i$ and $D(a) = 0$. Putting all these things together, one can either compute $D(a)$ directly or find an x such that $D(a) = A(x)$. Therefore D is positively reducible to A.

(III): $(T(0^k0), T(0^k1), T(0^k)) \notin V$ but $(T(0^k0), T(0^k1), T(0^k)) \in W$ for all $k \geq l$ for some fixed l. If one interchanges the role of 0 and 1 in the given truth-table reduction then one obtains a reduction from $\overline{D}$ to A where also the roles of V and W are interchanged. This is then exactly the case (II) and one obtains that $\overline{D}$ and A are in the same positive degree. It follows immediately that D and $\overline{A}$ and therefore also B have the same positive degree.

(IV): $(T(0^k0), T(0^k1), T(0^k)) \notin V, W$ for all $k \geq l$ for some fixed l. In this case one can conclude that both, D and $\overline{D}$, are positively reducible to B. Since B is enumerable, so are D and $\overline{D}$ and therefore D is recursive.

This finishes the case distinction and thus one has exactly three positive degrees given by the retraceable set A, the enumerable set B and the tt-cylinder C which has the same positive degree as $A \oplus B$. ▮

Enumerable tt-degrees consisting of three positive degrees have some beautiful properties. The next theorem shows that such a tt-degree has a positive degree of enumerable semirecursive sets, a tt-degree of coenumerable semirecursive sets and a positive degree of sets which are neither enumerable nor coenumerable nor semirecursive.

Theorem 8.6.5 *Let B be an enumerable set whose tt-degree consists of three positive degrees and let $D \equiv_{tt} B$. For D the following holds:*
(a) *If D is semirecursive then D is either enumerable or coenumerable.*
(b) *If D is enumerable then D is semirecursive.*
(c) *If D is retraced by a total function then D is the difference of two enumerable sets.*

Proof (a): Since D is semirecursive, D is not in the greatest positive degree [69, Corollary 4.3.(IV)]. If D is in the positive degree of B then D is enumerable since positive reducibility preserves the property "enumerable" [69, Proposition 3.5.(II)]. If D is in the positive degree of $\overline{B}$ then D is coenumerable.

(b): If D is in the greatest positive degree then $\overline{D}$ is positive reducible to D and thus also enumerable in contradiction to the fact that D is not recursive. So D is either in the positive degree of B or in that of $\overline{B}$ and so semirecursive [69, Theorem 4.2.(III)].

(c): The retracing-function defines a tree-structure on $\mathbb{N}$. Here a node x is a direct successor of y if the retracing-function maps x to y. So D is an infinite branch of a tree T. The sets B_1 of all nodes lexicographic before some node on D and the set $B_2 = B_1 - D$ are both semirecursive. So B_1 and B_2 are enumerable or coenumerable. Since $D = B_1 \cap \overline{B_2}$ either is the intersection of an enumerable and coenumerable and thus the difference of two enumerable sets or is the intersection of two coenumerable sets and thus coenumerable or — hypothetically — is the intersection of two enumerable sets and thus enumerable, one has that D can always be represented as the difference of two enumerable sets. ▮

A careful analysis of the construction in Theorem 8.6.4 shows that the set B shares some properties of that one constructed by Dëgtev [35] in the sense that it is also semirecursive, nonrecursive and η-maximal for some suitable equivalence-relation η. Downey [40] showed that in many Turing degrees there are enumerable tt-degrees having exactly one enumerable m-degree, in particular such a tt-degree exists within the Turing degree of K. The next result shows that this is not true for enumerable tt-degrees consisting of three positive degrees. So the next result indicates why some additional work was required to construct them.

Theorem 8.6.6 *If every semirecursive set in some given tt-degree is either enumerable or coenumerable then this tt-degree belongs to a low$_2$ Turing degree.*

Proof Let X represent a tt-degree in which all semirecursive sets are either enumerable or coenumerable. Relativizing a result of Arslanov [7, Theorem 7], there is a partial X'-recursive $\{0,1\}$-valued function ψ which does not have a total extension which is computable relative to some X'-enumerable degree $Z <_T X''$. An easy example for such a function [135, Section V.5] is given by

$$\psi(x) = \begin{cases} \varphi_e^{X'}(e) & \text{if } \varphi_e^{X'}(e)\downarrow \leq 1; \\ \uparrow & \text{otherwise.} \end{cases}$$

Let $\tilde{\psi}$ be a total function of X'-enumerable degree Z which extends ψ. Every partial $\{0,1\}$-valued X'-recursive function is many-one reducible to ψ. There is a computable function u such that $X'(x) = \psi(u(x))\downarrow$ and one knows that $Z \geq_T X'$. Furthermore, by relativizing Arslanov's result [7, Theorem 1] one has that $Z \equiv_T X''$ for computations relative to X' and since $X'', Z \geq_T X'$, this equivalence also holds for nonrelativized computations.

The partial function ψ has a X-recursive approximation f such that $f(x,s)$ converges to a for $s \to \infty$ iff $\psi(x)\downarrow = a$. Now the set Y_x is given by

$$Y_x(2y+a) = \begin{cases} f(x,y) & \text{if } a = 0; \\ X(y) & \text{if } a = 1 \text{ and } f(x,y) = 0; \\ 1 - X(y) & \text{if } a = 1 \text{ and } f(x,y) = 1. \end{cases}$$

In all cases the set Y_x defines a semirecursive set B_x as the set of all finite strings which are lexicographic before Y_x. Let $\psi(0)\downarrow = 0$. Now B_0 is either enumerable or coenumerable, say the first. If $\psi(x)\downarrow = 0$ then $f(0,y) = f(x,y)$ for almost all y. It follows that Y_0 and Y_x differ only on finitely many elements and that thus $B_0 \equiv_m B_x$. If $\psi(x)\downarrow = 1$ then $Y_x(y) = 1 - Y_0(y)$ for almost all y and B_x is coenumerable. Now one defines:

$$g(x) = \begin{cases} 0 & \text{if } B_x \text{ is enumerable;} \\ 1 & \text{if } B_x \text{ is coenumerable.} \end{cases}$$

Each B_x is semirecursive and in the same tt-degree as X. So it follows that every B_x is either enumerable or coenumerable but not both. g is well-defined and total, furthermore g extends ψ. The following K'-recursive algorithm computes g: On input x, the algorithm searches the first e such that either $W_e = B_x$ or $\overline{W_e} = B_x$. Such an e exists since B_x is either enumerable or coenumerable. The test whether $W_e = B_x$ can be done using oracle K' since both arrays, the one of the B_x and the one of the W_e, are uniformly recursive in K. When the e is found, then $g(x) = 0$ for the case $B_x = W_e$ and $g(x) = 1$ for the case $B_x = \overline{W_e}$. So g is a K'-recursive function extending ψ. K' is enumerable relative to X' and so K' must have the same Turing degree as X''. Thus X has low$_2$ Turing degree. ∎

8.7 Weak Truth-Table Degrees

Friedberg and Rogers [49] introduced the weak forms of truth-table and bounded truth-table reducibility. While the functions f and $f_1, \ldots, f_k$ remain to be total-recursive, the function g is replaced by a partial recursive function γ, that is,

$$\begin{aligned} A \leq_{wtt} B \ &\Leftrightarrow\ (\exists f, \gamma)\,(\forall x)\,[A(x) = \gamma(x, B(0), B(1), \ldots, B(f(x)))\downarrow\ ]. \\ A \leq_{wbtt} B \ &\Leftrightarrow\ (\exists k)\,(\exists f_1, f_2, \ldots, f_k, \gamma) \\ &\qquad (\forall x)\,[A(x) = \gamma(x, B(f_1(x)), B(f_2(x)), \ldots, B(f_k(x)))\downarrow\ ]. \end{aligned}$$

Kobzev [86] showed, that every enumerable weak truth-table degree contains infinitely many enumerable weak bounded truth-table degrees (which he denoted as enumerable “bw-degrees”). So Dëgtev's result that some enumerable tt-degree contains only one enumerable btt-degree does not generalize to wbtt-degrees versus wtt-degrees. But the results on the nonenumerable btt-degrees transfer to results for wbtt-degrees:

Theorem 8.7.1 *Every nonrecursive wtt-degree contains infinitely many wbtt-degrees. The structure of the enumerable sets under inclusion can be embedded into the structure of the wbtt-degrees inside any given hyperimmune wtt-degree.*

Proof The proof of this theorem just uses the fact that whenever in Proposition 8.2.2, Theorem 8.2.4 and Theorem 8.3.3 there occur two sets X_1, X_2 with $X_1 \not\leq_{btt} X_2$ then in fact $X_1 \not\leq_{wbtt} X_2$ holds.

In Proposition 8.2.2 it is shown that for the two considered sets X_1, X_2 there is no infinite recursive set Y such that X_1 is btt-reducible to X_2 on Y. This result can be extended by showing:

> If X_1 is wbtt-reducible to X_2 then there is an infinite recursive set Y such that X_1 is btt-reducible to X_2 on Y.

Let $X_2(x) = \gamma(x, f_1(x), \ldots, f_k(x))$ for some k where γ is partial-recursive and $f_1, \ldots, f_k$ are total-recursive. For $l = 0, 1, \ldots, 2^k, 2^k + 1$ let

$$Y_l = \{x : \gamma(x, b_1, \ldots, b_k){\downarrow} \text{ for at least } l \text{ tuples } (b_1, \ldots, b_k) \in \{0,1\}^k\}.$$

All the Y_l are enumerable sets. Since $Y_0 = \mathbb{N}$ and $Y_{2^k+1} = \emptyset$ there is a maximal $l \leq 2^k$ such that Y_l is infinite. Since Y_{l+1} is finite, the set $Y_l - Y_{l+1}$ is enumerable and has an infinite recursive subset Y. On each $x \in Y$, the first l k-tuples $(b_1, \ldots, b_k) \in \{0,1\}^k$ such that $\gamma(x, b_1, \ldots, b_k)\downarrow$ are enumerated and on these tuples one defines $g(x, b_1, \ldots, b_k) = \gamma(x, b_1, \ldots, b_k)$. For the other $2^k - l$ tuples $(b_1, \ldots, b_k)$ one knows that $\gamma(x, b_1, \ldots, b_k)\uparrow$ and therefore g can be made total on Y by just letting $g(x, b_1, \ldots, b_k) = 0$. So $f_1, \ldots, f_k$ and g witness that $X_1 \leq_{btt} X_2$ on the infinite recursive set Y in contrary to the proof of Proposition 8.2.2. So $X_1 \not\leq_{wbtt} X_2$. It follows that even every hyperimmune tt-degree intersects with infinitely many wbtt-degrees.

For the hyperimmune-free case, standard and weak (bounded) truth-table reducibility coincide [111, Proof of Theorem IV.5.5] and so every hyperimmune-free tt-degree consists of infinitely many wbtt-degrees.

The result that the structure of enumerable sets under inclusion can be embedded into the structure of the wbtt-degrees inside a hyperimmune wtt-degree follows directly. ▮

It is also possible to define a positive version of weak truth-table reducibility. That is, a set D is weak positive reducible to E via f and γ iff

- f is total recursive and γ is partial recursive.
- $(\forall x)\,[D(x) = \gamma(x, E(0), E(1), \ldots, E(f(x))){\downarrow}]$.
- If $\gamma(x, a_0, a_1, \ldots, a_{f(x)}){\downarrow} = 0$ and $b_y \leq a_y$ for $y = 0, 1, \ldots, f(x)$ then $\gamma(x, b_0, b_1, \ldots, b_{f(x)}){\downarrow} = 0$.
- If $\gamma(x, a_0, a_1, \ldots, a_{f(x)}){\downarrow} = 1$ and $b_y \geq a_y$ for $y = 0, 1, \ldots, f(x)$ then $\gamma(x, b_0, b_1, \ldots, b_{f(x)}){\downarrow} = 1$.

Inside hyperimmune-free Turing degrees, the notions of weak positive degrees and positive degrees coincide by the same argument that weak truth-table and truth-table degrees coincide. So it follows directly from this argument that also some weak truth-table degrees consist only of finitely many positive degrees. The more interesting question is what happens on enumerable weak truth-table degrees. For them the next result shows that they have exactly one enumerable weak positive degree but that they consist of infinitely many weak positive degrees. Of these, all but one are not enumerable.

Theorem 8.7.2 *Let B be a nonrecursive enumerable set.*
(a) *The weak positive degree of B contains exactly the enumerable sets within the weak truth-table degree of B. In particular, wtt-reducibility and weak positive reducibility coincide on enumerable sets.*
(b) *The inclusion-structure of the enumerable sets can be embedded into the structure of the weak positive degrees inside the wtt-degree of B. In particular there are infinite ascending and descending chains as well as infinite antichains of weak positive degrees inside the wtt-degree of B.*

Proof (a): Let D be weak positive reducible to B where the reduction is given as $D(x) = \gamma(x, B(0), B(1), \ldots, B(f(x)))$. Now one knows that

$$x \in D \Leftrightarrow (\exists s)\,[\gamma_s(x, B_s(0), B_s(1), \ldots, B_s(f(x)))\!\downarrow = 1]$$

and thus D is also enumerable. So all sets in the weak positive degree of B are enumerable.

For the converse direction let D be an enumerable set which is weak truth-table reducible to B via some functions γ' and f'. Now a further partial function γ'' is constructed such that D is positively reducible to B via γ'' and f'. A stage s is called *consistent at x* if $D_s(x) = \gamma'_s(x, B_s(0), B_s(1), \ldots, B_s(f'(x)))$; all sets S_x of consistent stages are infinite and uniformly recursive. Now

$$\gamma''(x, b_0, b_1, \ldots, b_{f'(x)}) = \begin{cases} 0 & \text{if } (\exists s \in S_x)\,(\forall y \leq f'(x))\,[b_y \leq B_s(y) \wedge D_s(x) = 0]; \\ 1 & \text{if } (\exists s \in S_x)\,(\forall y \leq f'(x))\,[b_y \geq B_s(y) \wedge D_s(x) = 1]; \\ \uparrow & \text{otherwise.} \end{cases}$$

defines together with f' a weak positive reduction from D to B. Let $s \in S_x$. The reduction is positive and partial recursive by definition, nevertheless it has to be shown that it is correct and that it is not ambiguous. For the correctness note that there is an s such that $B_s(y) = B(y)$ for all $y \leq f'(x)$, $\gamma'_s(x, B(0), B(1), \ldots, B(f'(x)))$ is defined and such that $D(x) = D_s(x)$. It follows that then $s \in S_x$ and therefore also $\gamma''(x, B(0), B(1), \ldots, B(f'(x)))$ is defined according to the first or second case.

Now assume by way of contradiction that the definition of γ'' is ambiguous, that is, that there are stages $s, t \in S_x$ such that $\gamma''(x, b_0, b_1, \ldots, b_{f'(x)})$ is defined to be 0 at stage s and defined to be 1 at stage t. It follows directly that $D_s(x) = 0$ and $D_t(x) = 1$. Furthermore, $B_t(y) \geq B_s(y)$ for all y. Since now $b_y \leq B_s(y)$ and $b_y \geq B_t(y)$ holds for all $y \leq f'(x)$ and since by the fact that B is enumerable it holds that $B_s(y) \leq B_t(y)$ for $y = 0, 1, \ldots, f'(x)$, it follows that $b_y = B_s(y) = B_t(y)$ for $y = 0, 1, \ldots, f'(x)$. The reduction given by γ' and f' coincides with $D_s(x)$ at stage s and $D_t(x)$ at stage t by $s, t \in S_x$. Therefore $0 = D_s(x) = \gamma'(x, b_0, b_1, \ldots, b_{f'(x)})$ and $1 = D_t(x) = \gamma'(x, b_0, b_1, \ldots, b_{f'(x)})$ which shows that if γ'' is ambiguous so is γ'. From this contradiction it follows that the definition of γ'' is sound and D is weak positive reducible to B.

So one has on one hand that all sets D in the weak positive degree of B are enumerable and on the other hand that if D is an enumerable set in the wtt-degree of B then D is already weak positive reducible to B. Similarly B is weak positive reducible to D and so all enumerable sets from the wtt-degree of B are also in the weak positive degree of B.

(b): Sack's Splitting Theorem [135, Theorem VII.3.2] states that every nonrecursive enumerable set B is the disjoint union of two Turing incomparable enumerable sets, whose Turing degrees then are also strictly below B. Sacks [125, page 70] and Robinson [122, Theorem 3] present generalizations of the theorem by splitting B into a uniformly enumerable family of sets $B_0, B_1, \ldots$ such that no set B_i is computable relative to the join of all others. Let C_x be the tt-cylinder of B_x and let

$$H_e = \{(x, y, z) : x \in W_{e,y} \wedge z \in C_x\} \tag{35}$$

As in Theorem 8.2.4 one can prove that H_e is many-one reducible to $H_{e'}$ if $W_e \subseteq W_{e'}$. Furthermore, C_x is many-one reducible to H_e if $x \in W_e$. But if $x \notin W_e$ then C_x is by choice not computable relative to H_e and, since C_x is many-one equivalent to its complement, also not enumerable relative to H_e. It follows that $C_x \oplus B$ is not weakly positive reducible to $H_e \oplus B$ since $H_e \oplus B$ is enumerable relative to H_e and the sets enumerable relative to H_e are closed under weak positive reducibility. If $W_e \not\subseteq W_{e'}$ then $H_e \oplus B$ is not weakly positive reducible to $H_{e'} \oplus B$ since there is some $x \in W_e - W_{e'}$ and C_x is weakly positive reducible to W_e but not to $W_{e'}$.

So one has that $H_e \oplus B$ is weakly positive reducible to $H_{e'} \oplus B$ iff $W_e \subseteq W_{e'}$. Since all sets B_x, C_x and H_e are wtt-reducible to B, all sets $H_e \oplus B$ are in the wtt-degree of B and the structure of the enumerable sets under inclusion can be embedded to the structure of the weak positive degrees inside the wtt-degree of B. The existence of infinite chains and antichains follows from the corresponding result on the lattice of enumerable sets. ∎

9 Conclusion

The present work focuses on oracles, in particular on computation and learning with the help of an oracle. Oracles allow the analysis of the difficulty of learning and computing problems. For example, the difficulty to check whether a set W_e given by its index e is infinite needs an oracle of degree $\mathbf{0}''$ or higher. In learning theory, Adleman and Blum [1] showed that exactly the high oracles allow to Ex-learn the class REC of all total recursive functions. Also the difference between learning models has been analyzed in terms of the oracles necessary to make a problem S learnable with respect to some more pretentious Model 2 provided that the S is already learnable without the help of any oracle with respect to a less pretentious Model 1. The usefulness of the oracles with respect to a computation or learning task induces canonically a degree-structure on the oracles. Such degree-structures are research topics in their own right and there are numerous results on the degree-structures induced by computing (mostly on Turing, enumeration and many-one degrees). The present work (in particular in Chapter 2) gives an overview on the results about the degree-structures induced by learning.

For Turing reducibility, the question whether B is at least as powerful than A is equivalent to the question whether A can be computed relative to B, whence in this setting, degrees and their closures downward are always countable. In learning, an oracle B might be more powerful than A without giving any possibility to compute A relative to B, even in the limit. For example, Ex-learning has one uncountably infinite degree, namely the one of those oracles which allow to Ex-learn all classes of recursive functions. The structure of Ex-degrees is coarser than that of the Turing degrees. Only very restricted learning models like finite learning (Fin) generate the same degree-structure as the Turing reducibility.

Chapters 3 combines the notions of queries to oracles and of queries to teachers. Such a teacher answers questions — posed in a specific query language — on the function f to be learned. So the learner has more data on f as in the standard case. This extra-knowledge can sometimes be non-trivially combined with an oracle: There is an oracle A of trivial Ex-degree, which together with a teacher can learn a class S which can neither be learned from the oracle A alone nor from the teacher alone (Theorem 3.3.5).

An important research topic is the question to which extent errors and false information in the data-stream disturb learning. Many models of noisy data have the disadvantage that the disturbances do not only make learning difficult but do also permit identical data-streams for different concepts. Chapter 4 proposes a popular concept of noise [24, 27, 28, 58] which solves this problem: the basic idea is that each correct data-item occurs infinitely often while each incorrect one occurs only finitely often. Although each single item can be false, the data-stream as a whole determines which items are correct and which incorrect so that the function to be learned is uniquely determined by the data-stream. The central result is that Ex-learning functions from such data has a nice characterization in terms of learning with oracles: S is Ex-learnable from noisy data iff S is finitely learnable from noise-free data with the help of the oracle K.

The topic of Chapters 5 and 6 is the learning of sets and functions with infinite additional information. In both cases, the additional information is required to describe the whole class of languages to be learned and it must not be specific for a single set or function in S. Chapter 5 deals with various types of index-sets for classes of languages: if a set B contains for every language in S some index but no indices for languages outside S then a learner of Turing degree $\mathbf{0}''$ can learn S with access to this oracle B. Such a learner is even class-independent since it works for every principally learnable S when the corresponding B is supplied. If B is an index-set in the classical sense, that is, if $e \in B \Leftrightarrow W_e \in S$, then the learner can even be chosen to be recursive. In Chapter 5 it was required that the algorithm is universal in the sense that it worked for every S which is principally learnable, that is, learnable relative to some oracle. In contrast to this, the setting investigated in Chapter 6 permits that the algorithms are specific for the class S to be learned. However, the algorithms still have to be robust in the sense that they succeed with every oracle meeting the specification. Degree-theoretic descriptions of the oracle do not help much: if the learning-algorithm must be able to learn S with every oracle from a given m-degree then one can learn S even without an oracle. In contrast to this, syntactic descriptions of the class S turn out to be more helpful. The most powerful tools are lists of the functions in S. The considered syntactic descriptions are related to learning criteria, whence Chapter 6 is also some kind of study to which extent it is possible to translate learners of one type (represented by the oracle) uniformly into learners of an other type (represented by the learning algorithm using the oracle).

Classification is related to learning in the sense that, like a learner, a classifyer reads the characteristic function of the set as a learner but converges only to 1 or 0 standing for "set is in the class" and "set is not in the class". This notion of two-sided classification can also be weakened to one-sided classification where the classifyer on sets outside the class only outputs infinitely often a 0 without being required to converge to 0. Chapter 7 analyzes the relations between these two notions and the question, which oracles enable to transform a given one-sided classifyer into a two-sided one.

While the preceding chapters focus on degree-structures more general than that of the Turing reducibility, Chapter 8 investigates the structures induced by stronger reducibility notions, mainly by those of truth-table reducibility and its even more restrictive variants. The central question is whether there are always infinitely many positive and bounded truth-table degrees inside a truth-table degree. Dëgtev's result [35] that the number of bounded truth-table degrees inside a truth-table degree is at least two is improved by showing that this number in fact is always infinite. Moreover, there are infinite chains and antichains of bounded truth-table degrees inside every truth-table degree. The latter implies an affirmative answer to a question of Jockusch [70] whether every truth-table degree contains an infinite antichain of many-one degrees. Some but not all truth-table degrees have a least bounded truth-table degree. The technique to construct such a degree is used to solve an open problem of Beigel, Gasarch and Owings [15]: There are Turing degrees (constructed as hyperimmune-free truth-table degrees) which consist only of 2-subjective sets and do therefore not contain any objective set. Furthermore, a truth-table degree consisting of three positive degrees is constructed where one positive degree consists of enumerable semirecursive sets, one of coenumerable semirecursive sets and one of sets, which are neither enumerable nor coenumerable nor semirecursive. So Jockusch's result [69] that there are at least three positive degrees inside a truth-table degree is optimal. The number of positive degrees inside a truth-table degree can also be some other odd integers as for example nineteen, but it is never an even finite number.

The following are the main questions which are still open within the field covered in the present work: No characterization of the omniscient degree of BC-learning is known (Section 2.5). Furthermore, it would be interesting to know whether the omniscient BC degree coincides with the degree of all oracles permitting to learn the class of all functions which are either self-describing or almost everywhere equal to 0. Concerning the combination of teachers and oracles, it is still open whether there is a non-recursive oracle A which does not increase the learning power of the basic concepts QEx[Succ], QEx[<] and QEx[+] (Chapter 3). In the field of classification, a highly interesting problem is the following stated by Case, Kinber, Sharma and Stephan [30] for the model where only computable languages are considered (Section 7.6): Does every infinite one-sided class $\mathcal{A}$ have an infinite two-sided subclass $\mathcal{B}$? In Chapter 8 it is shown that there are infinitely many numbers a such that some non-recursive tt-degree consists of a positive degrees. It is shown that $a \leq 3$ and a is odd. But it is still open whether, for all odd numbers $a \geq 3$, there is a tt-degree consisting of exactly a positive degrees.

References

[1] Lenny Adleman and Manuel Blum. Inductive inference and unsolvability. *Journal of Symbolic Logic*, 56:891–900, 1991.

[2] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[3] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[4] Andris Ambainis, Sanjay Jain and Arun Sharma: Ordinal mind change complexity of language identification. *Proceedings of the Third European Conference on Computational Learning Theory, Jerusalem* (1997), *Springer Lecture Notes in Artificial Intelligence* 1208:301–316, 1997.

[5] Klaus Ambos-Spies. Sublattices of the polynomial time degrees. *Information and Control*, 65(1):63–84, 1985.

[6] Klaus Ambos-Spies. Inhomogeneities in the polynomial time degrees: the degrees of supersparse sets. *Information Processing Letters*, 22:113–117, 1986.

[7] Marat Arslanov. On some generalizations of a fixed-point theorem. *Soviet Mathematics (Iz. VUZ), Russian*, 25(5):9–16, 1981, *English translation*, 25(5):1–10, 1981.

[8] Ganesh Baliga and John Case. Learning with higher order additional information. *Joint Proceedings of the Fourth International Workshop on Analogical and Inductive Inference* (AII) *and of the Fifth Workshop on Algorithmic Learning Theory* (ALT) *Springer Lecture Notes in Artificial Intelligence* 872:64–75, 1994.

[9] Ganesh Baliga, John Case and Sanjay Jain. Synthesizing enumeration techniques for language learning. *Proceedings of the Ninth Conference on Computational Learning Theory* (COLT), 169–180, 1996.

[10] Ganesh Baliga, Sanjay Jain and Arun Sharma. Learning from multiple sources of inaccurate data. *Proceedings of the Third International Workshop on Analogical and Indcutiver Inference* (AII), *Springer Lecture Notes in Artificial Intelligence* 642:108–128, 1992.

[11] Janis Bārzdins. Prognostication of automata and functions. *Information Processing* '71, (1) 81–84. Edited by C. P. Freiman, North-Holland, Amsterdam, 1971.

[12] Janis Bārzdins and Rūsiņš Freivalds. On the prediction of general recursive functions. *Soviet Mathematical Doklady*, 13:1224–1228, 1972.

[13] Janis Bārzdins and Karlis Podnieks. The theory of inductive inference. *Proceedings of the Conference on Mathematical Foundations of Computer Science* (MFCS), 9–15, 1973.

[14] Richard Beigel, William Gasarch, John Gill and James Owings, Jr. Terse, superterse and verbose sets. *Information and Computation*, 103:68–85, 1993.

[15] Richard Beigel, William Gasarch and James Owings, Jr. Nondeterministic bounded query reducibilities. *Annals of Pure and Applied Logic*, 41:107–118, 1989.

[16] Richard Beigel, Martin Kummer and Frank Stephan. Quantifying the amount of verboseness. *Information and Computation*, 118:73–90, 1995.

[17] Shai Ben-David. Can finite samples detect singularities of real-valued functions? *Proceedings of the 24th Annual ACM Symposium on the Theory of Computer Science*, Victoria, B.C., 390–399, 1992.

[18] Manuel Blum. A machine independent theory of the complexity of recursive functions. *Journal of the Association of Computing Machinery*, 14:322–336, 1967.

[19] Lenore Blum and Manuel Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[20] Ulrike Brandt. The position of index sets of identifiable sets in the arithmetical hierarchy. *Information and Control*, 68:185–195, 1986.

[21] J. Richard Büchi. On a decision method in restricted second order arithmetic. *Proceedings of the International Congress on Logic, Methodology and Philosophy of Science*, Standford University Press, Standford, California, 1960.

[22] J. Richard Büchi and Lawrence H. Landweber: Definability in the monadic second order theory of successor. *Journal of Symbolic Logic*, 34:166–170, 1966.

[23] Valery K. Bulitko. Reducibility by linear Zhegalkin tables. *Siberian Mathematical Journal*, *Russian*, 21:23–31, 1980, *English translation*, 21:332–339, 1980.

[24] John Case and Sanjay Jain. Synthesizing learners tolerating computable noisy data. *Proceedings of the Ninth Annual Conference on Algorithmic Learning Theory* (ALT), *Springer Lecture Notes of Artificial Intelligence* 1501:205–219, 1998.

[25] John Case, Sanjay Jain and Suzanne Ngo Manguelle. Refinements of inductive inference by Popperian and reliable machines. *Journal Kybernetika*, 30:23–52, 1994.

[26] John Case, Sanjay Jain and Arun Sharma. On learning limiting programs. *International Journal of Foundations of Computer Science*, 1:93–115, 1992.

[27] John Case, Sanjay Jain and Arun Sharma. Synthesizing noise-tolerant language learners. *Proceedings of the Eighth Annual Workshop on Algorithmic Learning Theory* (ALT), *Springer Lecture Notes in Artificial Intelligence* 1316:228–243, 1997.

[28] John Case, Sanjay Jain and Frank Stephan. Vacillatory and BC learning on noisy data. *Proceedings of the Seventh Annual Workshop on Algorithmic Learning Theory* (ALT), *Springer Lecture Notes in Artificial Intelligence* 1160:285–298, 1996. See also: Electronic Archive for Computational Learning Theory eC-TR-96-002, Dortmund, 1996.

[29] John Case, Susanne Kaufmann, Efim Kinber and Martin Kummer. Learning recursive functions from approximations. *Proceedings of the Second European Conference on Computational Learning Theory* (EuroCOLT), 140-153, 1995.

[30] John Case, Efim Kinber, Arun Sharma and Frank Stephan. On the classification of computable languages. *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science* (STACS), 225–236, 1997.

[31] John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[32] Peter Cholak and Rod Downey. Recursive enumerable m- and tt-degrees. Part III: Realizing all finite distributive lattices. *The Journal of the London Mathematical Society, Second Series*, 50:440–453, 1994.

[33] Alonzo Church. An unsolvable problem of elementary number theory. *The American Journal of Mathematics*, 58:345–363, 1936.

[34] Martin Davis (editor). *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions.* Raven Press, 1965.

[35] Alexander Dëgtev. tt- and m-degrees. *Algebra and Logic*, *Russian*, 12:143–161, 1973, *English translation*, 12:78–89, 1973.

[36] Alexander Dëgtev. Three theorems on tt-degrees. *Algebra and Logic*, *Russian*, 17(3):270–281, 1978, *English translation*, 17:187–194, 1978.

[37] Alexander Dëgtev. Comparison of linear reducibility with other reducibilities of tabular type. *Algebra and Logic*, *Russian*, 21:511–529, 1982, *English translation*, 21:339–353, 1982.

[38] James Dekker and John Myhill. Retraceable sets. *Canadian Journal of Mathematics*, 10:357–373, 1958.

[39] Rod Downey. Recursively enumerable m- and tt-degrees. Part I: The quantity of m-degrees. *The Journal of Symbolic Logic*, 54:553–567. 1989.

[40] Rod Downey. Recursive enumerable m- and tt-degrees. Part II: The distribution of singular degrees. *Archive for Mathematical Logic*, 27:135–148, 1988.

[41] Anna-Maria Emde and Britta Schinzel. Aggregating inductive expertise on partial recursive functions. *Information and Computation*, 96:139–167, 1992.

[42] Lance Fortnow, Rūsiņš Freivalds, William Gasarch, Martin Kummer, Steven Kurtz, Carl Smith and Frank Stephan. On the relative size of learnable sets. *Theoretical Computer Science*, 197:139-156, 1998.

[43] Lance Fortnow, William Gasarch, Sanjay Jain, Efim Kinber, Martin Kummer, Steven Kurtz, Mark Pleszkoch, Theodore Slaman, Robert Solovay and Frank Stephan. Extremes in the degrees of inferability. *Annals of Pure and Applied Logic*, 66:231–276, 1994.

[44] Rūsiņš Freivalds, Efim Kinber and Rolf Wiehagen. On the power of inductive inference from good examples. *Theoretical Computer Science*, 110:131–144, 1993.

[45] Rūsiņš Freivalds and Carl H. Smith: On the Role of procrastination for machine learning, *Information and Computation*, 107:237–271, 1993.

[46] Rūsiņš Freivalds and Rolf Wiehagen. Inductive inference with additional information. *Elektronische Informationsverarbeitung und Kybernetik* 15:179–185, 1979.

[47] Richard Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability. *Proceedings of the National Academy of Sciences*, 43:236–238, 1957.

[48] Richard Friedberg. A criterion for completeness of degrees of unsolvability. *Journal of Symbolic Logic*, 22:159–160, 1957.

[49] Richard Friedberg and Hartley Rogers. Reducibilities and completeness for sets of integers. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 5:117–125, 1959.

[50] Mark Fulk and Sanjay Jain. Learning in the presence of inaccurate information. *Theoretical Computer Science*, 161:235–261, 1996.

[51] William Gasarch and Mark Pleszkoch. Learning via queries to an oracle. *Proceedings of the Second Annual Conference on Computational Learning Theory* (COLT), 214–229, 1989.

[52] William Gasarch, Mark Pleszkoch and Robert Solovay. Learning via queries in $[+, <]$. *Journal of Symbolic Logic*, 57:53–81, 1992.

[53] William Gasarch, Mark Pleszkoch, Frank Stephan and Mahendran Velauthapillai. Classification using information. *Annals of Mathematics and Artificial Intelligence, Selected papers from ALT 1994 and AII 1994*, 23:147–168, 1998.

[54] William Gasarch, Mark Pleszkoch and Mahendran Velauthapillai. Classification using information — conference version. *Joint Proceedings of the Fourth International Workshop on Analogical and Inductive Inference* (AII) *and of the Fifth Workshop on Algorithmic Learning Theory* (ALT) *Springer Lecture Notes in Artificial Intelligence* 872:290–300, 1994.

[55] William Gasarch and Carl Smith. Learning via queries. *Journal of the Association of Computing Machinery* 39(3):649–676, 1992.

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

[57] Sally Goldman and David Mathias. Teaching a smarter learner. *Journal of Computer and System Sciences*, 52:255–267, 1996.

[58] Gunter Grieser and Steffen Lange. Iterative learning from noisy data. *Tenth Conference on Algorithmic Learning Theory* (ALT), to appear, 1999.

[59] Leo Harrington and Robert Soare. Post's program and incomplete recursive enumerable sets. *Proceedings of the National Academy of Science, U.S.A.*, 88:10242–10246, 1991.

[60] Christine Ann Haught. The degrees below a 1-generic degree $< \mathbf{0}'$. *Journal of Symbolic Logic*, 51:770–777, 1986.

[61] Sanjay Jain. Program synthesis in the presence of infinite number of inaccuracies. *Journal of Computer and System Sciences*, 53(3):583–591, 1996.

[62] Sanjay Jain and Arun Sharma. Learning in the presence of partial explanations. *Information and Computation*, 95:162–191, 1991.

[63] Sanjay Jain and Arun Sharma. Learning with the knowledge of an upper bound on program size. *Information and Computation*, 102:118–166, 1993.

[64] Sanjay Jain and Arun Sharma. On the non-existence of maximal inference degrees for language identification. *Information Processing Letters*, 47:81–88, 1993.

[65] Sanjay Jain and Arun Sharma. On monotonic strategies for learning r.e. languages. *Joint Proceedings of the Fourth International Workshop on Analogical and Inductive Inference* (AII) *and of the Fifth Workshop on Algorithmic Learning Theory* (ALT) *Springer Lecture Notes in Artificial Intelligence* 872:349–364, 1994.

[66] Klaus-Peter Jantke. Automatic synthesis of programs and inductive inference of functions. *Proceedings of the International Conference on Fundamentals of Computation Theory in Berlin/Wendisch-Rietz* (FCT) 219–225, 1979.

[67] Klaus-Peter Jantke. Natural properties of strategies identifying recursive functions. *Elektronische Informationsverarbeitung und Kybernetik* 15:487-496, 1979.

[68] Klaus-Peter Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.

[69] Carl Jockusch. Semirecursive sets and positive reducibility. *Transactions of the American Mathematical Society*, 131:420–436, 1968.

[70] Carl Jockusch. Relationships between reducibilities. *Transactions of the American Mathematical Society*, 142:229–237, 1969.

[71] Carl Jockusch. Degrees in which recursive sets are uniformly recursive. *Canadian Journal of Mathematics*, 24:1092–1099, 1972.

[72] Carl Jockusch. Degrees of generic sets. *London Mathematical Society Lecture Notes*, 45:110–139, 1981.

[73] Carl Jockusch and Robert Soare. Π^0_1 classes and degrees of theories. *Transactions of the American Mathematical Society*, 173:33–56, 1972.

[74] Sejtnijas Kallibekov. On degrees of recursively enumerable sets. *Siberian Mathematical Journal*, 14(2):421–426, 1973, *English translation*, 14:200–203, 1973.

[75] Shyam Kapur. Monotonic language learning. *Proceedings of the Third Workshop on Algorithmic Learning Theory* (ALT) 147–158, 1992.

[76] Shyam Kapur and Gianfranco Bilardi. On uniform learnability of language families. *Information Processing Letters*, 44:35–38, 1992.

[77] Susanne Kaufmann and Frank Stephan (1997): Robust learning with infinite additional information. *Proceedings of the Third European Conference on Computational Learning Theory* (EuroCOLT), 316–330, 1997. *Forschungsberichte Mathematische Logik* 23 / 1996, Mathematisches Institut, Universität Heidelberg, 1996.

[78] Kevin Kelly. *The Logic of Reliable Inquiry.* Oxford University Press, Oxford, 1995.

[79] Efim Kinber. Some problems of learning with an oracle. *Proceedings of the Third Conference on Computational Learning Theory* (COLT), 178–186, 1990.

[80] Efim Kinber and Frank Stephan. Language learning from texts: Mind changes, limited memory and monotonicity. *Information and Computation*, 123:224–241, 1995.

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

[82] Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser, Boston, 1991.

[83] Georgi Kobzev. On btt-Reducibilities I. *Algebra and Logic*, *Russian*, 12(2):190–204, *English translation*, 12:107–115, 1973.

[84] Georgi Kobzev. On btt-Reducibilities II. *Algebra and Logic*, *Russian*, 12(4):433–444, *English translation*, 12:242–248, 1973.

[85] Georgi Kobzev. On Γ-separable sets. *Studies in Mathematical Logic and the Theory of Algorithms*, *Russian*, 19–30. Tblisi 1975.

[86] Georgi Kobzev. Recursive enumerable bw-degrees. *Matematicheskie Zametki*, *Russian*, 21(6):839–846, *English translation*, 21:473–477, 1977.

[87] Martin Kummer. A proof of Beigel's cardinality conjecture. *The Journal of Symbolic Logic*, 57:677–681, 1992.

[88] Martin Kummer and Matthias Ott. Learning Branches and Learning to Win Closed Games. *Proceedings of Ninth Annual Conference on Computational Learning Theory* (COLT), 280–291, 1996.

[89] Martin Kummer and Frank Stephan. On the structure of degrees of inferability. Journal of Computer and System Sciences, Special Issue COLT 1993, 52:214–238, 1996.

[90] Martin Kummer and Frank Stephan. Recursion theoretic properties of frequency computation and bounded queries. *Information and Computation*, 120:59–77, 1995.

[91] Richard Ladner, Nancy Lynch and Alan Selman. Comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[92] Steffen Lange, Jochen Nessel and Rolf Wiehagen. Language learning from good examples. *Joint Proceedings of the Fourth International Workshop on Analogical and Inductive Inference* (AII) *and of the Fifth Workshop on Algorithmic Learning Theory* (ALT) *Springer Lecture Notes in Artificial Intelligence* 872:423–437, 1994.

[93] Lawrence H. Landweber. Decision problems for ω-automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[94] Steffen Lange, Thomas Zeugmann and Shyam Kapur. Monotonic and dual monotonic language learning. *Theoretical Computer Science*, 155:365–410, 1996.

[95] Jack Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and Systems Science*, 44:226–258, 1992.

[96] Donald Martin. Completeness, the recursion theorem and effectively simple sets. *Proceedings or the American Mathematical Society*, 17:838–842, 1966.

[97] Donald Martin. Classes of recursively enumerable sets and degrees of unsolvability. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 12:295–310, 1966.

[98] Yasuhito Mukouchi. Characterization of finite identification. *Proceedings of the Third International Workshop on Analogical and Inductive Inference* (AII), *Springer Lecture Notes in Artificial Intelligence* 642:260–267, 1992.

[99] Thomas McLaughlin. On a class of complete simple sets. *Canadian Mathematical Bulleting*, 8:33–37, 1965.

[100] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:434–448, 1966.

[101] Kurt Mehlhorn. On the size of sets of computable functions. *Proceedings of the Fourteenth Annual Symposium on Switching and Automata Theory*, 190–196, IEEE Computer Society, 1973.

[102] Kurt Mehlhorn. Polynomial and abstract subrecursive classes. *Journal of Computer and System Sciences*,12:147–178, 1978.

[103] Wolfgang Merkle. *A Generalized Account of Resource Bounded Reducibilities.* Doctoral Dissertation, Universität Heidelberg, 1997.

[104] Wolfgang Merkle and Frank Stephan. Trees and learning. *Proceedings of the Ninth Conference on Computational Learning Theory* (COLT), 270–279, 1996.

[105] Webb Miller and Donald Martin. The degrees of hyperimmune sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 14:159–166, 1968.

[106] Eliana Minicozzi. Some natural properties of strong-identification in inductive inference. *Theoretical Computer Science*, 2:345–360, 1976.

[107] Al'bert Abramovich Muchnik. Negative answer to the problem of reducibility in the theory of algorithms. *Doklady Akademii Nauk S. S. S. R.*, 108:194–197, 1956.

[108] Maurice Nivat and Dominique Perrin (editors): *Automata on Infinite Words. Springer Lecture Notes in Computer Science* 192, 1984.

[109] Anil Nerode. General topology and partial recursive functionals. *Talks Cornell Summ. Inst. Symb. Log.*, Cornell 247–257, 1957.

[110] David B. Posner, Robert W. Robinson. Degrees joining to $\mathbf{0}'$. *Journal of Symbolic Logic*, 46(4):714–722, 1981.

[111] Piergiorgio Odifreddi. *Classical Recursion Theory.* North-Holland, Amsterdam, 1989.

[112] Daniel Osherson, Dick de Jongh, Eric Martin and Scott Weinstein. Formal learning theory. *Handbook of Logic and Language*, edited by J. van Benthem and A. ter Meulen. Elsevier, 737–775, 1997.

[113] Daniel Osherson, Michael Stob and Scott Weinstein. *Systems That Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists.* Bradford — The MIT Press, Cambridge, Massachusetts, 1986.

[114] Daniel Osherson, Michael Stob and Scott Weinstein. Synthesizing inductive expertise. *Information and Computation*, 77(2):138–161, 1988.

[115] Daniel Osherson, Michael Stob and Scott Weinstein. A universal inductive inference machine. *Journal of Symbolic Logic*, 56:661–672, 1991.

[116] Matthias Ott. *Learning Strategies for Infinite Games.* Doctoral Dissertation, Universität Karlsruhe, 1998.

[117] Matthias Ott and Frank Stephan. Structural measures for games and process control in the branch learning model. *Proceedings of the Third European Conference on Computational Learning Theory* (EuroCOLT), 94-108, 1997. *Interner Bericht* 39 / 96, Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, 1996.

[118] J. C. Oxtoby. *Maß und Kategorie.* Springer-Verlag, Heidelberg, 1971.

[119] Emil Post. Finite combinatory processes. Formulation I. *Journal of Symbolic Logic*, 1:103–105, 1936.

[120] Emil Post. Recursively enumerable sets of positive integers and their decision problems, *Bulletin of the American Mathematical Society*, 50:284–316, 1944.

[121] H. Gordon Rice. Classes of enumerable sets and their decision problems. *Transactions of the American Mathematical Society* 74:358–366, 1953.

[122] Robert W. Robinson. Interpolation and embedding in the recursive enumerable degrees. *Annals of Mathematics, Second Series*, 93:285–314, 1971.

[123] Robert W. Robinson. Jump restricted interpolation in the recursive enumerable degrees. *Annals of Mathematics, Second Series*, 93:586–596, 1971.

[124] Hartley Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[125] Gerald E. Sacks. *Degrees of Unsolvability.* Annals of Mathematics Studies 55, Princeton University Press, Princeton, New Jersey, 1966.

[126] Gerald E. Sacks. *Higher Recursion Theory.* Perspectives in Mathematical Logic, Springer-Verlag, Heidelberg, 1990.

[127] Gisela Schäfer. Some results in the theory of effective program synthesis — learning by defective information. *Springer Lecture Notes in Computer Science* 225:219–225, 1986.

[128] Claus Peter Schnorr. *Zufälligkeit und Wahrscheinlichkeit. Springer Lecture Notes in Mathematics*, 1971.

[129] Arun Sharma. A note on batch and incremental learnability. *Journal of Computer and System Sciences*, 56:272–276, 1998

[130] Joseph Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, 69:644–653, 1959.

[131] Joseph Shoenfield. A theorem on minimal degrees. *The Journal of Symbolic Logic*, 31:539–544, 1966.

[132] Theodore Slaman and Robert Solovay. When oracles do not help. *Proceedings of the Fourth Conference on Computational Learning Theory* (COLT), 379–383, 1991.

[133] Neil J. A. Sloane. *The On-Line Encyclopedia of Integer Sequences.* Homepage http://www.research.att.com/~njas/sequences/index.html.

[134] Carl H. Smith, Rolf Wiehagen and Thomas Zeugmann. Classifying predicates and languages. *International Journal of Foundations of Computer Science*, 8(1):15–42, 1997.

[135] Robert Soare. *Recursively Enumerable Sets and Degrees. A Study of Computable Functions and Computably Generated Sets.* Springer-Verlag, Heidelberg, 1987.

[136] Clifford Spector. On degrees of unsolvability. *Annals of Mathematics*, 64:581–592, 1956.

[137] Frank Stephan. Learning via queries and oracles. *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory* (COLT), 162-169, ACM-Press, New York, 1995.

[138] Frank Stephan. On one-sided versus two-sided classification. *Forschungsberichte Mathematische Logik* 25 / 1996, Mathematisches Institut, Universität Heidelberg, 1996.

[139] Frank Stephan. Noisy inference and oracles. *Theoretical Computer Science* 185:129–157, 1997.

[140] Frank Stephan. On the structures inside truth-table degrees. *Forschungsberichte Mathematische Logik 29 / 1997, Mathematisches Institut, Universität Heidelberg*, Heidelberg, 1997.

[141] Frank Stephan and Sebastiaan Terwijn. The complexity of universal text-learners. *Proceedings of the eleventh International Symposium on the Foundations of Computation Theory* (FCT), *Springer Lecture Notes in Computer Science* 1279:441–451, 1997.

[142] S. Tennenbaum. Degrees of unsolvability and the rate of growth of functions. *Proceedings of the Symposium on the Mathematical Theory of Automata, Microwave Research Institute Symposium Series* 12, Polytechnic Press, Brooklyn, New York 71–73, 1962.

[143] Boris A. Trakhtenbrot. Tabular representation of recursive operators. *Doklady Akademii Nauk S. S. S. R.*, 101:417–420, 1955.

[144] Boris A. Trakhtenbrot. Finite automata and the logic of one place predicates. *Siberian Mathematical Journal*, 3:103–131, 1962 [in Russian].

[145] Alan M. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936 and correction, 43:544–546, 1937.

[146] Rolf Wiehagen. A thesis in inductive inference. *Proceedings First International Workshop on Nonmonotonic and Inductive Logic*, *Springer Lecture Notes in Artificial Intelligence* 534:184–207, 1990.

[147] Rolf Wiehagen and Carl Smith. Generalization versus classification. *Proceedings Fifth Annual Workshop on Computational Learning Theory* (COLT), ACM-Press, New York, 224–230, 1992.

[148] Paul Young. On reducibility by recursive functions. *Proceedings of the American Mathematical Society*, 15:889–892, 1964.

[149] Paul Young. A theorem on recursively enumerable classes and splinters. *Proceedings of the American Mathematical Society*, 17:1050–1056, 1966.

[150] Thomas Zeugmann. *Algorithmisches Lernen von Funktionen und Sprachen.* Habilitationsschrift, Technische Hochschule Darmstadt, 1993.

[151] Thomas Zeugmann and Steffen Lange. A guided tour across the boundaries of learning recursive languages. *Algorithmic Learning for Knowledge-Based Systems*, final report on research project Gosler, edited by Klaus P. Jantke and Steffen Lange, *Springer Lecture Notes in Artificial Intelligence* 961:193–262, 1995.

[152] Thomas Zeugmann, Steffen Lange and Shyam Kapur. Characterizations of monotonic and dual monotonic language learning, *Information and Computation*, 120:155–173, 1995.