Learning All Subfunctions of a Function

Sanjay Jain [a,1] Efim Kinber [b] Rolf Wiehagen [c]

[a] *School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543. Email: sanjay@comp.nus.edu.sg*

[b] *Department of Computer Science, Sacred Heart University, Fairfield, CT 06432-1000, U.S.A. Email: kinbere@sacredheart.edu*

[c] *Department of Computer Science, University of Kaiserslautern, D-67653 Kaiserslautern, Germany. Email: wiehagen@informatik.uni-kl.de*

Abstract

Sublearning, a model for learning of subconcepts of a concept, is presented. Sublearning a class of total recursive functions informally means to learn all functions from that class together with all of their *subfunctions*. While in *language* learning it is known to be impossible to learn any infinite language together with all of its sublanguages, the situation changes for sublearning of *functions*.

Several types of sublearning are defined and compared to each other as well as to other learning types. For example, in some cases, sublearning coincides with *robust* learning. Furthermore, whereas in usual function learning there are classes that cannot be learned *consistently*, all sublearnable classes of some natural types *can* be learned consistently.

Moreover, the power of sublearning is characterized in several terms, thereby establishing a close connection to *measurable* classes and variants of this notion. As a consequence, there are rich classes which do not need any self-referential coding for sublearning them.

1 Introduction

In Gold's model of learning in the limit, see [15], the machine learner gets all examples of a total recursive function f, without loss of generality in natural order $(0, f(0)), (1, f(1)), (2, f(2)), \ldots$. Based on this information, the learner creates a sequence of hypotheses which eventually converges to a hypothesis

[1] Supported in part by NUS grant number R252–000–127–112.

exactly describing this function f. One might argue that getting *all* examples may be somewhat unrealistic, at least in some situations. On the other hand, what one can learn depends, intuitively, on the information one gets. Thus, also intuitively, the less information the learner gets the less it can learn. If it receives only information describing some *subconcept* of a certain "master" concept, then it seems reasonable that it can learn only this subconcept. From another, positive, point of view, the less data the learner is provided with, the wider is the spectrum of hypotheses which are consistent with these data and hence can serve as correct descriptions of the corresponding (sub-)concept to be learned. Situations like these of learning subconcepts of concepts we want to model and to study in the present paper. Possible scenarios of such "hierarchies" of concepts and corresponding subconcepts might include:

- learning a "theory of the universe", or learning only "subconcepts of nature" such as gravitation, quantum theory, or relativity,

- diagnosing the complete health status of a patient, or detecting only some of his/her deficiencies, or only one illness,

- forecasting the weather for a whole country or for some smaller region, or for a town only.

We do not intend, of course, to solve these problems within a model from abstract computation theory. What we want is to present a, in our opinion, technically easy model for learning of concepts and subconcepts and to study the corresponding learning capabilities.

In our model, we represent concepts by total recursive functions, i.e. computable functions mapping the natural numbers into the natural numbers and being everywhere defined (total). Subconcepts are then, consequently, represented by subfunctions of total recursive functions. Informally, we will call a class $\mathcal{C}$ of total recursive functions *sublearnable* iff all the functions from that class $\mathcal{C}$ *together with all of their subfunctions*, finite and infinite ones, are learnable. This goal might seem too ambitious, since, for example, in learning of *languages* from positive data it is known to be already impossible to learn any infinite language together with all of its *finite* sublanguages, see [15]. However, in learning of *functions*, the situation changes provided we consider a hypothesis as correct if this hypothesis is consistent with all the data presented to the learner. In other words, we allow the learner to converge to a hypothesis describing a *superfunction* of the (finite, infinite or total) function to be learned. This approach was introduced in the paper of the Blums [6]. Within this approach, if the learner is provided with all examples of any *total* function, then it is supposed to learn that function exactly. But if the learner is provided with exactly all examples of any finite or infinite *subfunction* of some total function, then it suffices to create a final hypothesis which, on the

one hand, is consistent with this subfunction, but which, on the other hand, describes a function that, on arguments never shown, can be arbitrarily defined or even undefined. Thus, indeed, when learning a *proper* subfunction of a total function by being presented only all the examples of that subfunction, the learner has "more freedom" to generate a correct final hypothesis.

We will also modify this approach, namely by strengthening and by weakening it, respectively. *Strengthening* means that we always require the final hypothesis to be *total* even when the learner was presented only a *partial* function. However, we do not require that this total final hypothesis has to describe a (total) function *from the learnable class* $\mathcal{C}$. The reason for not considering this additional strengthening is that then already simple classes (namely subclasses of recursively enumerable classes) would be no longer sublearnable. Nevertheless, it may be worth to study this additional strengthening as well in more detail in future work. *Weakening* the approach above means to require to learn only all the *infinite* subfunctions of the functions from $\mathcal{C}$, that is missing the finite subfunctions. As it turns out, this weakening indeed increases the learning possibilities. Finally, we will also combine this strengthening and this weakening, that is learning only infinite subfunctions but requiring total hypotheses as the final result of the learning process.

As for some historical background, note that in the seminal paper [15], Gold showed that in his model every *recursively enumerable* class of total recursive functions is learnable by the so-called identification-by-enumeration principle. Informally, this kind of learning strategy always outputs the minimal hypothesis (with respect to a given total recursive enumeration of the class to be learned) which is consistent with all the data seen so far. It is then easy to see that this strategy converges to the minimal correct hypothesis within the given enumeration. The naturalness of this strategy led Gold to conjecture that *every* learnable class can be learned using identification-by-enumeration. In other words, Gold's conjecture was that every learnable class is contained in a recursively enumerable class. However, as Bārzdiņš [2] proved, this conjecture is false. He exhibited the following "self-describing" class $\mathcal{SD}$ of total recursive functions, $\mathcal{SD} = \{f \mid f(0) \text{ is a program for } f\}$. Each function f in $\mathcal{SD}$ can be trivially learned by just outputting the program $f(0)$. On the other hand, no recursively enumerable class contains $\mathcal{SD}$.

It seems worth to be noted that the class $\mathcal{SD}$ above can also be learned *without* making explicit use of its self-coding, namely by some "generalized" identification-by-enumeration. The same is true for other classes learnable in Gold's model. This in turn led to the thesis that for *each* type of Gold-style learning, there is an *adequate* enumeration technique, i.e. an enumeration technique which can be used to learn exactly the concept classes of that type. This thesis is stated and technically motivated in [19]. In the present paper, we verify this thesis for several types of sublearning, see Theorems 48 and 50.

Also in the 1970's, Bārzdiņš suggested a more sophisticated version of Gold's conjecture above designed to transcend such self-referential counterexamples as the class $\mathcal{SD}$. He reasoned that if a class is learnable by way of such a self-referential property, then there would be an "effective transformation" that would transform the class into another one that is no longer learnable. The idea is that if a learner is able to find the embedded self-referential information in the functions of the class, so can an effective transformation, which then can weed out this information. A reasonable way to make the notion of an effective transformation precise consists in using the concept of general recursive operators, i.e. effective and total mappings from total functions to total functions, see Definitions 51 and 52. In order to illustrate Bārzdiņš' intuition in the context of the class $\mathcal{SD}$ above, consider the operator Θ weeding out the self-referential information $f(0)$ as follows: $\Theta(f) = g$, where $g(x) = f(x+1)$ for all arguments x. Then one can show that $\Theta(\mathcal{SD}) = \{\Theta(f) \mid f \in \mathcal{SD}\} = \mathcal{R}$, the class of *all* the total recursive functions. Since $\mathcal{R}$ is not learnable, see [15], $\Theta(\mathcal{SD})$ is not learnable as well. Informally, Bārzdiņš' conjecture can then be stated as follows: If all the projections of a class of total recursive functions under all general recursive operators *are* learnable (or, in other words, if the class is learnable *robustly*), then the class is contained in a recursively enumerable class of total recursive functions, and, consequently, it is learnable by use of identification-by-enumeration. This was how the notion of robust learning appeared historically. This notion was then studied in several papers, see [27,18,14,21,8,16,9].

Clearly, the notion of sublearning in the present paper can intuitively be viewed as some special case of learning robustly. Indeed, while general robust learning requires that *all* projections of a given class of total recursive functions under all general recursive operators be learnable, in sublearning only a *special* kind of projections is required so, namely the given class of total recursive functions together with all of their subfunctions (or all of their infinite subfunctions, respectively). Thus, the question of comparing the capabilities of these two learning paradigms, sublearning and robust learning, naturally arises. As we will show, in general, these capabilities turn out to be set-theoretically incomparable, see Theorems 59 and 63. Consequently, each of these notions has its "right of existence", since no one of them majorizes the other one by its learning power. On the other hand, in some natural cases, sublearning and robust learning *coincide*! This is true if the function classes to be learned are closed under finite variations, i.e. if some total function f belongs to such a class then any total function, which differs from f at most on finitely many arguments, also belongs to that class. Thus, intuitively, changing a function a "little bit" will keep the resulting function still within the class. In this case, we can show that sublearning and robust learning are of the same power, and, moreover, any such class is even contained in a recursively enumerable class, see Theorem 64.

Further note that Gold's classical identification-by-enumeration was later shown to be successfully applicable to learning of more than merely the recursively enumerable classes of functions. Actually, this technique can directly be applied also to learning of so-called *measurable* classes, see Definition 43. Informally, a function class is measurable iff it can be embedded into a computable numbering η such that the predicate $\eta_i(x) = y$ is decidable uniformly in i, x and y. For example, the running times of the total recursive functions form a measurable class. Somewhat more generally, any complexity measure in the sense of [7] also constitutes a measurable class. Clearly, measurability here just ensures the *computability* of the identification-by-enumeration strategy, i.e. the *effectiveness* of finding the corresponding minimal hypothesis which is consistent with the data received so far. As to our concept of sublearning, we will see that some of the corresponding types of sublearning contain *all* the measurable classes, as it follows from Theorem 44. This result has yet another interesting consequence, namely that there are sublearnable classes beyond the world of recursive enumerability which turn out to be not at all self-referential!

There are further results showing that the connection between sublearnable classes and measurable classes is really close. Actually, if we confine ourselves again to classes being closed under finite variations, then sublearnability and measurability *coincide*, see Theorem 45. Moreover, if we drop the property of closedness under finite variations, then sublearnability coincides with *weak* measurability, see Definition 47 and Theorem 48. Furthermore, the close connection between sublearnability and measurability can be considered as the substantial reason for another unexpected phenomenon. It is known that in Gold's model there are learnable classes which *cannot* be learned consistently; i.e. every learner of such a class must be allowed to produce intermediate hypotheses that are not consistent with the data seen so far, see [3,24,26]. Thus, paradoxically, the learners of such classes are forced to output intermediate hypotheses which *contradict known* data. Conversely, as it will be shown in Theorem 28, sublearnable classes *can* always be learned consistently!

The paper is organized as follows. In Section 2, the needed definitions and results from existing function learning theory are presented. In Section 3, the types of sublearning are formally introduced and some basic facts will be derived. In Section 4, we compare these types with respect to their corresponding learning power. In Section 5, we prove some characterizations for several sublearning types. In Section 6, we compare sublearning with robust learning. Finally, in Section 7, we present further comparisons of sublearning types with known types of function learning.

2 Notation and Preliminaries

Recursion-theoretic concepts not explained below are treated in [22]. N denotes the set of natural numbers. $*$ denotes a non-member of N and is assumed to satisfy $(\forall n)[n < * < \infty]$. $a \in A$ denotes a is a member of set A. $\subseteq, \subset, \supseteq, \supset, \triangle$, respectively, denote the subset, proper subset, superset, proper superset, and incomparability relations for sets. The empty set is denoted by $\emptyset$. We let $\mathrm{card}(S)$ denote the cardinality of the set S. So "$\mathrm{card}(S) \leq *$" means that $\mathrm{card}(S)$ is finite. The minimum and maximum of a set S are denoted by $\min(S)$ and $\max(S)$, respectively. We take $\max(\emptyset)$ to be 0 and $\min(\emptyset)$ to be ∞. χ_A denotes the characteristic function of A, that is, $\chi_A(x) = 1$, if $x \in A$, and 0 otherwise.

$\langle \cdot, \cdot \rangle$ denotes a 1-1 computable mapping from pairs of natural numbers onto natural numbers. π_1, π_2 are the corresponding projection functions. $\langle \cdot, \cdot \rangle$ is extended to n-tuples of natural numbers in a natural way. η, with or without subscripts, superscripts, primes and the like, ranges over partial functions. If η_1 and η_2 are both undefined on input x, then, we take $\eta_1(x) = \eta_2(x)$. We say that $\eta_1 \subseteq \eta_2$ iff for all x in domain of η_1, $\eta_1(x) = \eta_2(x)$. We let $\mathrm{domain}(\eta)$ and $\mathrm{range}(\eta)$ respectively denote the domain and range of the partial function η. $\eta(x)\downarrow$ denotes that $\eta(x)$ is defined. $\eta(x)\uparrow$ denotes that $\eta(x)$ is undefined. For a partial function η, $\eta^{-1}(y)$ denotes the set $\{x \mid \eta(x) = y\}$.

We say that a partial function η is *consistent* with η' (denoted $\eta \sim \eta'$) iff for all $x \in \mathrm{domain}(\eta) \cap \mathrm{domain}(\eta')$, $\eta(x) = \eta(x')$. η is *non-consistent* with η' (denoted $\eta \not\sim \eta'$) iff there exists an x such that $\eta(x)\downarrow \neq \eta'(x)\downarrow$.

For $r \in N$, the r-extension of η denotes the function f defined as follows:

$$f(x) = \begin{cases} \eta(x), & \text{if } x \in \mathrm{domain}(\eta); \\ r, & \text{otherwise.} \end{cases}$$

f, g and h, with or without subscripts, superscripts, primes and the like, range over total functions. $\mathcal{R}$ denotes the class of all total recursive functions, i.e., total computable functions with arguments and values from N. $\mathcal{T}$ denotes the class of all *total* functions. $\mathcal{R}_{0,1}$ ($\mathcal{T}_{0,1}$) denotes the class of all total recursive functions (total functions) with range contained in $\{0, 1\}$. $\mathcal{C}$ and $\mathcal{S}$, with or without subscripts, superscripts, primes and the like, range over subsets of $\mathcal{R}$. φ denotes a *fixed* acceptable programming system. φ_i denotes the partial recursive function computed by the φ-program i. Below we will interpret the hypotheses of our learning machines just as programs in this numbering φ. We let Φ be an arbitrary Blum complexity measure [7] associated with the acceptable programming system φ; many such measures exist for any acceptable programming system [7]. We assume without loss of generality that $\Phi_i(x) \geq x$,

for all i, x. $\varphi_{i,s}$ is defined as follows:

$$\varphi_{i,s}(x) = \begin{cases} \varphi_i(x), & \text{if } x < s \text{ and } \Phi_i(x) < s; \\ \uparrow, & \text{otherwise.} \end{cases}$$

We let $W_i = \text{domain}(\varphi_i)$, and $W_{i,s} = \text{domain}(\varphi_{i,s})$.

For a given partial computable function ψ, we define $\text{MinProg}(\psi) = \min(\{i \mid \varphi_i = \psi\})$.

For an r.e. set S of programs, we let $Union(S)$ denote a program for the partial recursive function defined as follows: $\varphi_{Union(S)}(x) = \varphi_p(x)$, for the first $p \in S$ found such that $\varphi_p(x)$ is defined, using some standard dovetailing mechanism for computing φ_p's. If $\varphi_p(x)$ is undefined for all $p \in S$, then $\varphi_{Union(S)}(x)$ is undefined. Note that one can get a program for $Union(S)$ effectively from an index for the r.e. set S. When programs $q_1, q_2, \ldots, q_n$ for partial recursive functions $\eta_1, \eta_2, \ldots, \eta_n$ are implicit, we sometimes abuse notation and use $Union(\{\eta_1, \eta_2, \ldots, \eta_n\})$, to denote $Union(\{q_1, q_2, \ldots, q_n\})$.

A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be *recursively enumerable* iff there exists an r.e. set X such that $\mathcal{C} = \{\varphi_i \mid i \in X\}$. For any non-empty recursively enumerable class $\mathcal{C}$, there exists a total recursive function f such that $\mathcal{C} = \{\varphi_{f(i)} \mid i \in N\}$.

A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be *closed under finite variations* iff for all $f, g \in \mathcal{R}$ such that $\text{card}(\{x \mid f(x) \neq g(x)\}) < \infty$, $f \in \mathcal{C}$ iff $g \in \mathcal{C}$.

We say that a function F *dominates* [23] a function f iff $F(x) \geq f(x)$ for all but finitely many x.

The following functions and classes are commonly considered below. Zero is the everywhere 0 function, i.e., $\text{Zero}(x) = 0$, for all $x \in N$. $\text{CONST} = \{f \mid (\forall x)[f(x) = f(0)]\}$ denotes the class of the constant functions. $\text{FINSUP} = \{f \mid (\forall^\infty x)[f(x) = 0]\}$ denotes the class of all total recursive functions of finite support.

2.1 Function Identification

We first describe inductive inference machines. In this paper we will be concerned about learning of functions, often being partial ones. For the purpose of learning the (partial) functions, the data given to the learner is the graph of the function presented in the form of infinite sequence of pairs from that graph (or a special pause symbol #).

A text is a mapping from N to $(N \times N) \cup \{\#\}$, such that if (x, y) and (x, z) are in the range of the text, then $y = z$. $\mathbf{T}$ denotes the set of all texts. A segment

is an initial sequence of a text. That is, a segment is a mapping from $\{x \in N \mid x < n\}$ to $(N \times N) \cup \{\#\}$, for some natural number n (where if (x, y) and (x, z) are in the range of the segment, then $y = z$). For a segment σ, content(σ) denotes the set of pairs in the range of σ: content(σ) = range(σ) $- \{\#\}$. Similarly, for a text T, content(T) = range(T) $- \{\#\}$. SEG denotes the set of all finite segments. $\text{SEG}_{0,1} = \{\sigma \in \text{SEG} \mid (x, y) \in \text{content}(\sigma) \Rightarrow y \in \{0, 1\}\}$. We let σ and τ, with or without subscripts, superscripts, primes and the like, range over SEG. Λ denotes the empty segment. For $f \in \mathcal{R}$ and $n \in N$, we let $f[n]$ denote the finite segment $(0, f(0)), (1, f(1)), \ldots, (n-1, f(n-1))$. Clearly, $f[0]$ denotes the empty segment. We let INITSEG $= \{f[n] \mid f \in \mathcal{R} \ \wedge \ n \in N\}$. Similarly, $\text{INITSEG}_{0,1} = \{f[n] \mid f \in \mathcal{R}_{0,1} \ \wedge \ n \in N\}$. For elements of INITSEG, we sometimes abuse notation and represent $f[n]$ by the string $f(0), f(1), \ldots, f(n-1)$. We assume some computable ordering of elements of SEG. $\sigma < \tau$, if σ appears before τ in this ordering. Similarly one can talk about the least element of a subset of SEG.

We let $\sigma \cdot \tau$ denote the concatenation of σ and τ. Sometimes we abuse notation slightly and use $\sigma \cdot (x, w)$ to denote the concatenation of σ with the segment of length one consisting of (x, w).

Let $|\sigma|$ denote the length of σ. $T[n]$ denotes the initial segment of T of length n. If $|\sigma| \geq n$, then we let $\sigma[n]$ denote the prefix of σ of length n. $\sigma \subseteq \tau$ denotes that σ is a prefix of τ.

A text T is for a (partial) function η, iff content(T) = η.

An *inductive inference machine* (IIM) $\mathbf{M}$ [15] is an algorithmic device that computes a (possibly partial) mapping from SEG into N. Since the set of all finite initial segments, SEG, can be coded onto N, we can view these machines as taking natural numbers as input and emitting natural numbers as output. For a text T and $i \in N$, we say that $\mathbf{M}(T) = i$ iff the sequence $\mathbf{M}(T[n])$ converges to i. We write $\mathbf{M}(T)\downarrow$ iff there is some $i \in N$ such that $\mathbf{M}(T) = i$. $\mathbf{M}(T)$ is undefined if no such i exists. $\mathbf{M}_0, \mathbf{M}_1, \ldots$ denotes a recursive enumeration of all the IIMs. The next definitions describe several criteria of function identification.

Definition 1 [15] Let $f \in \mathcal{R}$ and $\mathcal{C} \subseteq \mathcal{R}$.

(a) $\mathbf{M}$ **Ex**-*identifies* f (written: $f \in \mathbf{Ex}(\mathbf{M})$) just in case, for all texts T for f, there exists a φ-program i for f such that $\mathbf{M}(T) = i$.

(b) $\mathbf{M}$ **Ex**-*identifies* $\mathcal{C}$ iff $\mathbf{M}$ **Ex**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{Ex} \ = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Ex}(\mathbf{M})]\}$.

By the definition of convergence, only finitely many data points from a function

f have been observed by an IIM $\mathbf{M}$ at the (unknown) point of convergence. Hence, some form of learning must take place in order for $\mathbf{M}$ to identify f. For this reason, hereafter the terms *identify*, *learn* and *infer* are used interchangeably.

Note that in the literature, often canonical ordering of data for the input function is considered: the input consists of $(0, f(0)), (1, f(1)), \ldots$. For **Ex**-learning of total functions, the ordering is not important. However, for the criteria considered in this paper, ordering is often important. Thus, it is more suitable for us to use *arbitrary* ordering in the input.

Definition 2 [20] A machine $\mathbf{M}$ is said to be *set-driven* iff for all σ and τ such that content(σ) = content(τ), $\mathbf{M}(\sigma) = \mathbf{M}(\tau)$.

Definition 3 [13,6] A machine $\mathbf{M}$ is said to be *rearrangement-independent* iff for all σ and σ' such that content(σ) = content(σ'), and $|\sigma| = |\sigma'|$, $\mathbf{M}(\sigma) = \mathbf{M}(\sigma')$.

A machine $\mathbf{M}$ is said to be *order-independent* iff for all texts T and T' such that content(T) = content(T'), $\mathbf{M}(T) = \mathbf{M}(T')$.

Theorem 4 *[13,6] For every $\mathcal{C} \in \mathbf{Ex}$, there exists a rearrangement-independent and order-independent machine $\mathbf{M}$ such that $\mathbf{M}$ **Ex**-identifies $\mathcal{C}$.*

Theorem 4 holds for many criteria of learning besides **Ex**. In particular it can be shown for **AllTotSubEx**, **InfTotSubEx**, **AllPartSubEx** and **InfPartSubEx** defined below.

Definition 5 [13] σ is said to be an **Ex**-*stabilizing sequence* for $\mathbf{M}$ on η, iff (i) content$(\sigma) \subseteq \eta$, and (ii) for all σ' such that $\sigma \subseteq \sigma'$ and content$(\sigma') \subseteq \eta$, $\mathbf{M}(\sigma) = \mathbf{M}(\sigma')$.

Definition 6 [6,20] σ is said to be an **Ex**-*locking sequence* for $\mathbf{M}$ on η, iff (i) content$(\sigma) \subseteq \eta$, (ii) for all σ' such that $\sigma \subseteq \sigma'$ and content$(\sigma') \subseteq \eta$, $\mathbf{M}(\sigma) = \mathbf{M}(\sigma')$, and (iii) $\varphi_{\mathbf{M}(\sigma)} \supseteq \eta$.

Theorem 7 *[6,20] Suppose for all texts T for η, $\varphi_{\mathbf{M}(T)} \supseteq \eta$. Then, there exists an **Ex**-locking sequence for $\mathbf{M}$ on η.*

A similar theorem as above holds for many other criteria of inference, in particular, for **AllPartSubEx**, **InfPartSubEx**, **AllTotSubEx**, **InfTotSubEx**, defined below.

Definition 8 [4,10] Let $f \in \mathcal{R}$ and $\mathcal{C} \subseteq \mathcal{R}$.

(a) $\mathbf{M}$ **Bc**-*identifies* f (written: $f \in \mathbf{Bc}(\mathbf{M})$) iff, for all texts T for f, for all but finitely many $n \in N$, $\mathbf{M}(T[n])$ is a φ-program for f.

(b) **M Bc**-*identifies* $\mathcal{C} \subseteq \mathcal{R}$ iff **M Bc**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{Bc} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Bc}(\mathbf{M})]\}$.

Definition 9 (Based on [6,20]) σ is said to be a **Bc**-*locking sequence* for **M** on η, iff (i) content$(\sigma) \subseteq \eta$, (ii) for all σ' such that $\sigma \subseteq \sigma'$ and content$(\sigma') \subseteq \eta$, $[\eta \subseteq \varphi_{\mathbf{M}(\sigma')}]$.

Theorem 10 *(Based on [6,20]) Suppose for all texts T for η, for all but finitely many n, $\varphi_{\mathbf{M}(T[n])} \supseteq \eta$. Then, there exists a* **Bc**-*locking sequence for* **M** *on η.*

An analogous theorem holds for the sublearning types **AllPartSubBc**, **InfPartSubBc**, **AllTotSubBc** and **InfTotSubBc**, defined below.

Definition 11 [3] **M** is said to be *consistent* on f iff, for all texts T for f, for all n, $\mathbf{M}(T[n])\downarrow$ and content$(T[n]) \subseteq \varphi_{\mathbf{M}(T[n])}$.

The above consistency notion is referred to as $\mathbf{Cons}^{arb}$ in the literature (to denote that ordering of the input may be arbitrary rather than in canonical order), see [17]. As we will only be dealing with arbitrary input in this paper, we drop "arb" from the notation.

Definition 12 (a) [3] **M Cons**-*identifies* $f \in \mathcal{R}$ iff **M** is consistent on f, and **M Ex**-identifies f.

(b.1) [3] **M Cons**-*identifies* $\mathcal{C} \subseteq \mathcal{R}$ iff **M Cons**-identifies each $f \in \mathcal{C}$.

(b.2) $\mathbf{Cons} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Cons}\text{-identifies } \mathcal{C}]\}$.

(c.1) [17] **M** $\mathcal{R}$**Cons**-*identifies* $\mathcal{C} \subseteq \mathcal{R}$ iff **M** is total, and **M Cons**-identifies $\mathcal{C}$.

(c.2) $\mathcal{R}\mathbf{Cons} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathcal{R}\mathbf{Cons}\text{-identifies } \mathcal{C}]\}$.

(d.1) [25] **M** $\mathcal{T}$**Cons**-*identifies* $\mathcal{C} \subseteq \mathcal{R}$ iff **M** is consistent on each $f \in \mathcal{T}$, and **M Cons**-identifies $\mathcal{C}$.

(d.2) $\mathcal{T}\mathbf{Cons} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathcal{T}\mathbf{Cons}\text{-identifies } \mathcal{C}]\}$.

Note that for **M** to **Cons**-identify a function f, it must be defined on each initial segment of each text for f.

Definition 13 **M TEx**-*identifies* $f \in \mathcal{R}$, iff **M Ex**-identifies f, and for all texts T for f, for all n, $\mathbf{M}(T[n])$ is a program for a total function.

M TEx-identifies class $\mathcal{C} \subseteq \mathcal{R}$, iff **M TEx**-identifies each $f \in \mathcal{C}$.

$\mathbf{TEx} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{TEx}\text{-identifies}\ \mathcal{C}]\}$.

Definition 14 $\mathbf{NUM} = \{\mathcal{C} \mid (\exists \mathcal{C}' \mid \mathcal{C} \subseteq \mathcal{C}' \subseteq \mathcal{R})[\mathcal{C}' \text{ is recursively enumerable}]\}$.

For inductive inference within **NUM**, the set of all recursively enumerable classes and their subclasses, the reader is referred to [15,5,11].

The following theorems relate the criteria of inference discussed above.

Theorem 15 [25,26,3,4,6,24,10]

$\mathbf{NUM} \subset \mathcal{T}\mathbf{Cons} \subset \mathcal{R}\mathbf{Cons} \subset \mathbf{Cons} \subset \mathbf{Ex} \subset \mathbf{Bc}$.

Theorem 16 *[17]* $\mathbf{NUM} \subset \mathbf{TEx} \subset \mathbf{Cons}$.

$\mathcal{T}\mathbf{Cons} - \mathbf{TEx} \neq \emptyset$.

$\mathbf{TEx} - \mathcal{R}\mathbf{Cons} \neq \emptyset$.

3 Definitions for Sublearning

In this section, we formally define our types of sublearning. Notice that each of these types includes, by definition, only classes of *total* recursive functions – though a class be sublearnable means, as said above, to be learnable together with all (or all infinite, respectively) of the corresponding subfunctions as well. The formal reason for confining us to classes of *total* recursive functions in the definitions of the sublearning types below is the following. We then can compare these types to the established types of function learning (which also contain only classes of *total* recursive functions, see the definitions in Subsection 2.1) without any *formal* difficulty. On the other hand, obviously, once a class of total recursive functions has been fixed, then the class of all (or all infinite, respectively) corresponding subfunctions is uniquely determined and, hence, needs no additional specification. After giving these definitions we show that all the *recursively enumerable* classes are sublearnable with respect to *every* of our sublearning criteria, see Proposition 22. Consequently, in the following, we will mainly deal with those sublearnable classes which are *not* contained in any recursively enumerable class.

In our first definition, the learner is required to stabilize on a program for a *total* function extending the concept to be learned.

Definition 17 (a) We say that $\mathbf{M}$ **AllTotSubEx**-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{AllTotSubEx}(\mathbf{M})$), iff, for all subfunctions $\eta \subseteq f$, for all texts T for η, $\mathbf{M}(T)\downarrow$, $\varphi_{\mathbf{M}(T)} \supseteq \eta$, and $\varphi_{\mathbf{M}(T)} \in \mathcal{R}$.

(b) **M AllTotSubEx**-identifies $\mathcal{C} \subseteq \mathcal{R}$, iff **M AllTotSubEx**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{AllTotSubEx} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{AllTotSubEx}(\mathbf{M})]\}$.

In the next definition, the final conjecture is *not* required to be total.

Definition 18 (a) We say that **M AllPartSubEx**-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{AllPartSubEx}(\mathbf{M})$), iff, for all subfunctions $\eta \subseteq f$, for all texts T for η, $\mathbf{M}(T)\downarrow$, and $\varphi_{\mathbf{M}(T)} \supseteq \eta$.

(b) **M AllPartSubEx**-identifies $\mathcal{C} \subseteq \mathcal{R}$, iff **M AllPartSubEx**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{AllPartSubEx} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{AllPartSubEx}(\mathbf{M})]\}$.

In the next definition, the final conjecture must be total, but only all *infinite* subconcepts are required to be learned.

Definition 19 (a) We say that **M InfTotSubEx**-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{InfTotSubEx}(\mathbf{M})$), iff, for all subfunctions $\eta \subseteq f$ with infinite domain, for all texts T for η, $\mathbf{M}(T)\downarrow$, $\varphi_{\mathbf{M}(T)} \supseteq \eta$, and $\varphi_{\mathbf{M}(T)} \in \mathcal{R}$.

(b) **M InfTotSubEx**-identifies $\mathcal{C} \subseteq \mathcal{R}$, iff **M InfTotSubEx**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{InfTotSubEx} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{InfTotSubEx}(\mathbf{M})]\}$.

The next definition requires only infinite subconcepts to be learned, but does not require the final conjecture to be total.

Definition 20 (a) We say that **M InfPartSubEx**-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{InfPartSubEx}(\mathbf{M})$), iff, for all subfunctions $\eta \subseteq f$ with infinite domain, for all texts T for η, $\mathbf{M}(T)\downarrow$, and $\varphi_{\mathbf{M}(T)} \supseteq \eta$.

(b) **M InfPartSubEx**-identifies $\mathcal{C} \subseteq \mathcal{R}$, iff **M InfPartSubEx**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{InfPartSubEx} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{InfPartSubEx}(\mathbf{M})]\}$.

One can extend the above definitions to use other criteria of inference such as **Bc** or require consistency by the learning machine. Such criteria are named **AllTotSubBc** and **InfPartSubCons**, etc. We define **AllTotSubBc** as an example.

Definition 21 (a) We say that **M AllTotSubBc**-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{AllTotSubBc}(\mathbf{M})$), iff, for all subfunctions $\eta \subseteq f$, for all texts T for η,

for all but finitely many n, $\varphi_{\mathbf{M}(T[n])} \supseteq \eta$ and $\varphi_{\mathbf{M}(T[n])} \in \mathcal{R}$.

(b) **M AllTotSubBc**-identifies $\mathcal{C} \subseteq \mathcal{R}$, iff **M AllTotSubBc**-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{AllTotSubBc} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{AllTotSubBc}(\mathbf{M})]\}$.

Using identification-by-enumeration one can easily show that already the strongest among the sublearning types, **AllTotSubEx**, contains all the recursively enumerable classes. Notice that, by Proposition 55 and Theorem 63 below, the inclusion of Proposition 22 is even *proper*.

Proposition 22 $\mathbf{NUM} \subseteq \mathbf{AllTotSubEx}$.

4 Comparison of Sublearning Criteria

In this section, we first compare various criteria of sublearning to each other. Then we deal with consistent sublearning. In particular, we show that the classes from **AllPartSubEx** and from **AllTotSubEx** can even be learned *consistently*. Finally, we consider behaviourally correct sublearning.

A summary of the results of this section can be seen in Figure 1. If there is no sequence of directed arrows connecting two types then these types are incomparable.

4.1 Comparing the Basic Types of Sublearning to Each Other

As it turns out, the trivial inclusions immediately implied by the definitions are all *proper*, while **AllPartSubEx** and **InfTotSubEx** are incomparable, see Corollaries 26 and 27.

Theorem 23 $\mathbf{AllPartSubEx} - \mathbf{InfTotSubEx} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid [\mathrm{card}(\mathrm{range}(f)) < \infty]$ and $(\forall e \in \mathrm{range}(f))[W_e = f^{-1}(e)]\ \}$.

It is easy to verify that $\mathcal{C} \in \mathbf{AllPartSubEx}$. The learner on input σ, first computes $D = \{e \mid (\exists x)[(x, e) \in \mathrm{content}(\sigma)]\}$. Then, the learner outputs a program for the following function g: $g(x) = e$, for the first $e \in D$ found (in some standard search) such that $x \in W_e$; if no such e exists, then $g(x) = \uparrow$. (Here for **Ex**-identification we assume that the program output by the learner, on input σ, depends only on D as computed above).

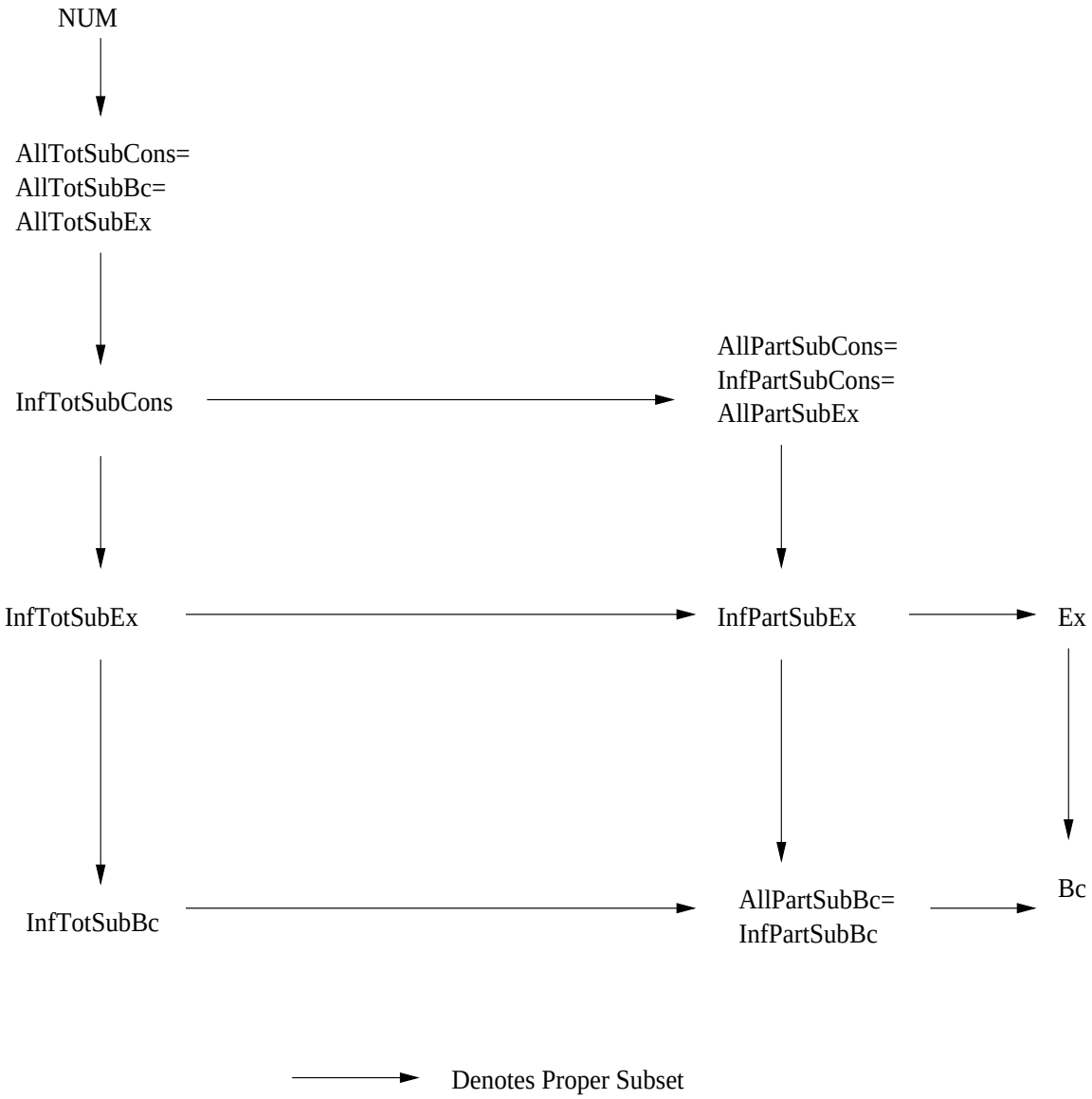

Fig. 1. Comparison of Sublearning Criteria

Instead of proving $\mathcal{C} \notin$ **InfTotSubEx**, we will prove a *stronger* result (which is needed in the proof of both Corollary 24 and Theorem 35 below), namely $\mathcal{C} \notin$ **InfTotSubBc**. Thus, suppose by way of contradiction **M InfTotSubBc**-identifies $\mathcal{C}$. Then, by Smullyan's double recursion theorem [22], there exist distinct a, b such that W_a, W_b may be described as follows. We will simultaneously define a function f, subfunctions of which will be used for the diagonalization.

Before stage 0, let f be the empty function. Let x_s denote the least x such that $f(x_s)$ is not defined before stage s. Let $\sigma_0 = \Lambda$. It will be the case that content(σ_s) = graph of $f[x_s]$. Initially, $W_a = W_b = \emptyset$. At the beginning of any stage s, W_a would contain $\{x < x_s \mid f(x) = a\}$ and W_b would contain $\{x < x_s \mid f(x) = b\}$.

Stage s

1. Dovetail steps 1.1, 1.2, until step 1.1 succeeds. If and when 1.1. succeeds, go to step 2.
   - 1.1 Search for τ extending σ_s such that
     - a) $\text{content}(\tau) - \text{content}(\sigma_s) \subseteq \{(x, a) \mid x > x_s\}$, and
     - b) $\varphi_{\mathbf{M}(\tau)}(x_s)$ converges.
   - 1.2 Enumerate in W_a, one by one, elements $x > x_s$.
2. Let $xm = \max((W_a \text{ enumerated up to now}) \cup \{x \mid (x, a) \in \text{content}(\tau)\})$.
   Let $f(x) = a$, for $x_s < x \leq xm$.
   Enumerate $x_s + 1, \ldots, xm$ in W_a.
   Let $f(x_s) = a$ if $\varphi_{\mathbf{M}(\tau)}(x_s) \neq a$, otherwise let $f(x_s) = b$. Correspondingly enumerate x_s in W_a or W_b, respectively.
   Let σ_{s+1} be an extension of τ such that $\text{content}(\sigma_{s+1})$ is the same as the graph of f defined up to now.
   Go to stage $s + 1$.

End stage s

Clearly, if infinitely many stages exist then f defined above is total and in $\mathcal{C}$. Let T denote the text $\bigcup_s \sigma_{s+1}$ for f. Now $\mathbf{M}(\tau)$ makes convergent errors on infinitely many initial segments τ of T (for the τ found at each stage).

On the other hand, if stage s does not end, then extend f as follows. Let c be such that $W_c = \{x_s\}$. Let $f(x_s) = c$. Let $f(x) = a$, for $x > x_s$. (f for $x < x_s$ is already defined before stage s). Clearly, $f \in \mathcal{C}$. Now $\mathbf{M}$ on any input τ, such that $\sigma_s \subseteq \tau$ and $\text{content}(\tau) \subseteq f - \{(x_s, c)\}$, does not output a program for a total function (as step 1.1 did not succeed).

Thus $\mathbf{M}$ does not **InfTotSubBc**-identify $\mathcal{C}$. ∎

Corollary 24 $\mathbf{AllPartSubEx} - \mathbf{InfTotSubBc} \neq \emptyset$.

Proof. Immediately from the proof of Theorem 23. ∎

Theorem 25 $\mathbf{InfTotSubEx} - \mathbf{AllPartSubEx} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall^\infty x)[\pi_1(f(x)) = e]\}$.

Clearly, $\mathcal{C} \in$ **InfTotSubEx**.

Now suppose by way of contradiction that $\mathbf{M}$ witnesses that $\mathcal{C} \in$ **AllPartSubEx**.

We will first construct a function φ_e. If φ_e is total, then φ_e will be in $\mathcal{C}$ and φ_e will be a diagonalizing function.

If φ_e is not total, then we will construct another diagonalizing function $\varphi_{e'}$ based on φ_e.

By Kleene recursion theorem [22], there exists an e such that φ_e may be described as follows. Let x_s denote the least x such that $\varphi_e(x)$ has not been defined before stage s. Initially, $x_0 = 0$. Let $\sigma_0 = \Lambda$. Go to stage 0.

Stage s

Dovetail steps 1 and 2, until one of them succeeds. If step 1 succeeds before step 2, then go to step 3. If step 2 succeeds before step 1, then go to step 4.

1. Search for a τ extending σ_s such that:
   (a) $\text{content}(\tau) \subseteq \{(x, \langle e, z\rangle) \mid x, z \in N\}$,
   (b) $\mathbf{M}(\tau) \neq \mathbf{M}(\sigma_s)$.
2. Search for a w such that
   (a) for all y, $(w, y) \notin \text{content}(\sigma_s)$,
   (b) $\varphi_{\mathbf{M}(\sigma_s)}(w)\downarrow$.
3. Let $\varphi_e(x) = \langle e, z\rangle$, for all $(x, \langle e, z\rangle)$ in $\text{content}(\tau)$.
   Let x' be the maximum x such that, for some z, $(x, \langle e, z\rangle) \in \text{content}(\tau)$.
   Let $\varphi_e(x) = \langle e, 0\rangle$ for $x \leq x'$ such that $\varphi_e(x)$ has not been defined up to now.
   Let σ_{s+1} be an extension of τ such that $\text{content}(\sigma_{s+1})$ is the graph of φ_e defined up to now.
   Go to stage $s + 1$.
4. Let $\varphi_e(w) = \langle e, 0\rangle$, if $\varphi_{\mathbf{M}(\sigma_s)}(w)\downarrow = \langle e, 1\rangle$; $\varphi_e(w) = \langle e, 1\rangle$, otherwise.
   Let $\varphi_e(x) = \langle e, 0\rangle$ for $x_s \leq x < w$.
   Let σ_{s+1} be an extension of σ_s such that $\text{content}(\sigma_{s+1})$ is the graph of φ_e defined up to now.
   Go to stage $s + 1$.

End stage s

If all stages in the above construction complete, then clearly, φ_e is total, is a member of $\mathcal{C}$ and $\mathbf{M}$ either makes infinitely many mind changes on $\bigcup_s \sigma_s$ (due to success of step 1 infinitely often), or the final program output by $\mathbf{M}$ on $\bigcup_s \sigma_s$ makes infinitely many convergent errors on φ_e (due to success of step 2 infinitely often, and diagonalization in step 4). Thus, $\mathbf{M}$ cannot **AllPartSubEx**-identify $\mathcal{C}$.

We now consider the case that some stage s does not complete. This means that step 1 in stage s does not succeed. In particular, it means that for some finite function η extending σ_s, $\mathbf{M}$ does not partially extend η on some input

text for η. Fix one such η. Now again using Kleene recursion theorem [22] there exists an e' such that

$$\varphi_{e'} = \begin{cases} \eta(x), & \text{if } x \in \text{domain}(\eta); \\ \langle e', 0 \rangle, & \text{otherwise.} \end{cases}$$

Clearly, $\varphi_{e'}$ is in $\mathcal{C}$. However, $\mathbf{M}$ does not partially extend the subfunction η of $\varphi_{e'}$, on some text for η. Thus, $\mathbf{M}$ does not **AllPartSubEx**-identify $\mathcal{C}$. ∎

An alternative proof of above theorem suggested by one of the anonymous referees can be obtained as follows: Let $\Omega(f)(x) = \langle x, f(0), f(1), \ldots, f(x) \rangle$. Let $\Omega(\mathcal{C}) = \{\Omega(f) \mid f \in \mathcal{C}\}$. Then, it is easy to see that:

(a) $\mathcal{C} \in \mathbf{Ex}$ iff $\Omega(C) \in \mathbf{Ex}$ iff $\Omega(C) \in \mathbf{InfTotSubEx}$.

(b) $\mathcal{C} \in \mathbf{Cons}^{can}$ iff $\Omega(\mathcal{C}) \in \mathbf{Cons}$ iff $\Omega(\mathcal{C}) \in \mathbf{AllPartSubCons} = \mathbf{AllPartSubEx}$. (The last equality is due to Theorem 28 below).

Here we say that $\mathbf{M}$ $\mathbf{Cons}^{can}$-identifies f iff, the sequence $\mathbf{M}(f[n])$ converges to a φ-program for f, and for all n, for all $x < n$, $\varphi_{\mathbf{M}(f[n])}(x) = f(x)$. Thus $\mathbf{M}$ is required to be consistent only when f is given in *canonical* rather than in arbitrary order. One can now define the type $\mathbf{Cons}^{can}$ in a way similar to Definition 12(b.1–b.2).

Now take a class $\mathcal{C} \in \mathbf{Ex}-\mathbf{Cons}^{can}$ [3]. Then, $\Omega(\mathcal{C})$ belongs to $\mathbf{InfTotSubEx}-\mathbf{AllPartSubEx}$ using (a) and (b) above.

Corollaries 26 and 27 immediately follow from Theorems 23 and 25.

Corollary 26 $\mathbf{AllPartSubEx} \triangle \mathbf{InfTotSubEx}$.

Corollary 27 $\mathbf{AllTotSubEx} \subset \mathbf{AllPartSubEx}$.

$\mathbf{InfTotSubEx} \subset \mathbf{InfPartSubEx}$.

$\mathbf{AllPartSubEx} \subset \mathbf{InfPartSubEx}$.

$\mathbf{AllTotSubEx} \subset \mathbf{InfTotSubEx}$.

4.2 Consistent Sublearning

While in Gold's model there are **Ex**-learnable classes which *cannot* be learned consistently, see Theorem 15, all the classes from **AllPartSubEx** as well as from **AllTotSubEx** *can* be learned consistently. This surprising fact will be

proved now in Theorem 28 using a technique from [9]. Note that this result will be useful at several subsequent places.

Theorem 28 **AllPartSubEx** = **AllPartSubCons**.

AllTotSubEx = **AllTotSubCons**.

PROOF. Suppose **M** **AllPartSubEx**-identifies (**AllTotSubEx**-identifies) $\mathcal{C}$. Without loss of generality, we can assume **M** to be total on SEG. We define a (monotonic) mapping F from SEG to SEG $\cup$ **T**, such that either (a) or (b) holds.

(a) $F(\sigma)$ is infinite, content$(F(\sigma)) \subseteq$ content(σ), and either $\mathbf{M}(F(\sigma))$ is not defined or $\varphi_{\mathbf{M}(F(\sigma))}$ is not an extension of content$(F(\sigma))$. (Thus content(σ) is not extended by any function in $\mathcal{C}$).

(b) $F(\sigma)$ is of finite length, content$(F(\sigma)) =$ content(σ), and $\varphi_{\mathbf{M}(F(\sigma))}$ extends content(σ).

This can be done by defining $F(\Lambda) = \Lambda$,

$$F(\sigma\cdot(x,w)) = \begin{cases} F(\sigma), & \text{if } F(\sigma) \text{ is of infinite length;} \\ F(\sigma)\cdot(x,w)\cdot\#^{\infty}, & \text{if } F(\sigma) \text{ is of finite length, and} \\ & \text{there does not exist a } j \text{ such that} \\ & \varphi_{\mathbf{M}(F(\sigma)\cdot(x,w)\cdot\#^{j}),j} \supseteq \text{content}(\sigma\cdot(x,w)); \\ F(\sigma)\cdot(x,w)\cdot\#^{j}, & \text{if } F(\sigma) \text{ is of finite length, and} \\ & j \text{ is the least number such that} \\ & \varphi_{\mathbf{M}(F(\sigma)\cdot(x,w)\cdot\#^{j}),j} \supseteq \text{content}(\sigma\cdot(x,w)). \end{cases}$$

F is clearly computable and satisfies the properties (a) and (b) above. Furthermore, for all η with an extension in $\mathcal{C}$, for all texts T for η, it is easy to verify that $\bigcup_n F(T[n])$ is also a text for η.

Define $\mathbf{M}'$ as follows. $\mathbf{M}'(\sigma) = \mathbf{M}(F(\sigma))$, if $F(\sigma)$ is finite in length. $\mathbf{M}'(\sigma)$ is undefined otherwise.

Now, as **M** **AllPartSubEx**-identifies (**AllTotSubEx**-identifies) $\mathcal{C}$, it follows using property (b) of F above, that $\mathbf{M}'$ **AllPartSubCons**-identifies (**AllTotSubCons**-identifies) $\mathcal{C}$. ∎

Proposition 29 **InfTotSubCons** $\subset$ **InfTotSubEx**.

PROOF. Let $\mathcal{C} = \{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall^{\infty} x)[\pi_1(f(x)) = e]\}$. $\mathcal{C}$ is clearly in **InfTotSubEx**. $\mathcal{C} \notin$ **Cons**, and hence $\mathcal{C} \notin$ **InfTotSubCons**, can be shown as follows. Suppose by way of contradiction otherwise. Suppose **M** is a machine which **Cons**-identifies $\mathcal{C}$.

If $\mathbf{M}$ is inconsistent on some inputs, then let σ be one such input (i.e. content$(\sigma) \not\subseteq \varphi_{\mathbf{M}(\sigma)}$). By Kleene recursion theorem [22], there exists an e such that

$$\varphi_e(x) = \begin{cases} y, & \text{if for some } y,\ (x,y) \in \text{content}(\sigma); \\ \langle e, 0\rangle, & \text{otherwise.} \end{cases}$$

Now $\varphi_e \in \mathcal{C}$, but $\mathbf{M}$ is not consistent on φ_e.

On the other hand, if $\mathbf{M}$ is consistent on all inputs and $y \neq z$, then $\mathbf{M}(\sigma \cdot (x,y)) \neq \mathbf{M}(\sigma \cdot (x,z))$, for all σ such that x is not in domain of content(σ). Thus one may define φ_e using Kleene recursion theorem [22] as follows: $\varphi_e(x) = \langle e, w\rangle$, for a $w \in \{0,1\}$, which causes a mind change $\mathbf{M}(\varphi_e[x]) \neq \mathbf{M}(\varphi_e[x] \cdot (x, \langle e, w\rangle))$. This φ_e is in $\mathcal{C}$, but $\mathbf{M}$ on φ_e makes infinitely many mind changes. ∎

By requirement of consistency, we have that any machine $\mathbf{M}$ **InfPartSubCons**-identifying f is consistent with all σ such that content$(\sigma) \subseteq f$. For any σ, let trunc(σ) be obtained by deleting any repetition in σ. Now let $\mathbf{M}'(\sigma) = \mathbf{M}(\text{trunc}(\sigma))$. It is easy to see that $\mathbf{M}'$ **AllPartSubCons**-identifies any f which is **InfPartSubCons**-identified by $\mathbf{M}$. Thus,

Proposition 30 **InfPartSubCons** = **AllPartSubCons**.

Corollary 31 **InfPartSubCons** ⊂ **InfPartSubEx**.

Now consider the class $\mathcal{C} = \{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall^\infty x)[\pi_1(f(x)) = e]$ and $(\forall x > 0)[f[x] \subseteq \varphi_{\pi_1(f(x-1))}\}$. $\mathcal{C}$ clearly belongs to **InfTotSubCons**. A modification of the proof of Theorem 25 can be used to show that $\mathcal{C} \notin$ **AllTotSubEx** = **AllTotSubCons**. We leave the details to the reader. Thus, we get the following proposition.

Proposition 32 **AllTotSubCons** ⊂ **InfTotSubCons**.

4.3 Behaviourally Correct Sublearning

We now derive some, partly surprising, effects for behaviourally correct sublearning. We start with the following observation. While **AllPartSubEx** is a *proper* subset of **InfPartSubEx**, see Corollary 27, this is no longer true for **Bc**-sublearning.

Theorem 33 **AllPartSubBc** = **InfPartSubBc**.

Proof. Suppose $\mathcal{C} \in$ **InfPartSubBc** as witnessed by machine $\mathbf{M}$. Now defined $\mathbf{M}'$ as follows.

$\mathbf{M}'(\sigma) = p$ such that

$$\varphi_p(x) = \begin{cases} y, & \text{if } (x,y) \in \text{content}(\sigma); \\ \varphi_{\mathbf{M}(\sigma)}(x), & \text{if for all } z,\ (x,z) \notin \text{content}(\sigma). \end{cases}$$

Note that for any finite function η, for any text T for η, $\varphi_{\mathbf{M}(T[n])} \supseteq \eta$, for all but finite many n.

Furthermore, if T is a text for infinite partial function η, and $\varphi_{\mathbf{M}(T[n])} \supseteq \eta$, then $\varphi_{\mathbf{M}'(T[n])} \supseteq \eta$ too. Theorem follows. ∎

Note that this proof does not work for "**Tot**" instead of "**Part**", as the initial partial functions from the **InfTotSubBc**-machine cannot be made total by the above patching.

Another difference comes with the **AllTot**-type of sublearning. While in traditional learning **Ex** $\subset$ **Bc** holds, see [4,10], this is not valid for **AllTot**-sublearning.

Theorem 34 **AllTotSubBc** = **AllTotSubEx**.

Proof. Suppose $\mathbf{M}$ **AllTotSubBc**-identifies a class $\mathcal{C}$. Note that, without loss of generality, we may assume that $\mathbf{M}$ is consistent on all inputs, i.e., for all $\sigma \in$ SEG, content$(\sigma) \subseteq \varphi_{\mathbf{M}(\sigma)}$.

For each segment σ, define $F(\sigma)$ as follows: Let $Cand_\sigma = \{\mathbf{M}(\sigma') \mid \sigma \subseteq \sigma' \wedge \text{content}(\sigma) = \text{content}(\sigma')\}$. Then, $F(\sigma) = Union(Cand_\sigma)$.

F satisfies the following properties.

(a) For all $\sigma \in$ SEG such that content(σ) has an extension in $\mathcal{C}$, $F(\sigma)$ is a program for a total function extending content(σ) (by definition of **AllTotSubBc** and consistency assumption on $\mathbf{M}$).

(b) For all partial functions η with an extension in $\mathcal{C}$, there exists a $\sigma \in$ SEG such that content$(\sigma) \subseteq \eta \subseteq \varphi_{F(\sigma)}$ (since there exists a locking sequence for $\mathbf{M}$ on η, for **AllTotSubBc**-identification, see remark after Theorem 10).

Now define $\mathbf{M}'$ as follows. $\mathbf{M}'$ on input σ outputs $F(\tau)$, for the least segment τ (in some ordering of elements of SEG) such that content$(\tau) \subseteq$ content(σ) and $\varphi_{F(\tau)}$ extends content(σ).

Now consider any subfunction η of $f \in \mathcal{C}$ and any text T for η. It follows using property (b) that $\mathbf{M}'(T)$ converges to $F(\tau)$ such that τ is the least element of SEG satisfying content$(\tau) \subseteq \eta \subseteq \varphi_{F(\tau)}$ (such τ exists due to property

(b)). Furthermore, $\varphi_{F(\tau)}$ in the previous statement is total (by property (a)). Theorem follows. ∎

We now exhibit some tradeoff between weakening the sublearning criterion, on the one hand, and strengthening the mode of convergence of the sequence of hypotheses, on the other hand.

Theorem 35 **InfTotSubBc** $\triangle$ **InfPartSubEx**.

Proof. By Corollary 24, **AllPartSubEx**$-$**InfTotSubBc** $\neq \emptyset$. Consequently, **InfPartSubEx** $-$ **InfTotSubBc** $\neq \emptyset$ as well. Conversely, the class $\{f \mid (\forall^\infty x)[\varphi_{f(x)} = f]\}$ obviously belongs to **InfTotSubBc**. However, this class is not in **Ex**, see [10], and hence it does not belong to **InfPartSubEx**. ∎

Theorem 35 together with Theorem 33 yield the following corollary.

Corollary 36 **InfTotSubBc** $\subset$ **AllPartSubBc**.

Finally, in order to complete Figure 1, we need the following separations. In particular, these results imply that, in contrast to **AllTotSubBc**, all the other types of **Bc**-sublearning go *beyond* the borders of usual **Ex**-learning.

Proposition 37 *(a)* **Ex** $-$ **InfPartSubBc** $\neq \emptyset$.

(b) **InfTotSubBc** $-$ **Ex** $\neq \emptyset$.

Proof. (a) The class $\mathcal{C} = \{f \in \mathcal{R} \mid \varphi_{f(0)} = f\}$ witnesses the separation. $\mathcal{C}$ is clearly in **Ex**. However, it is not in **InfPartSubBc**, as a machine missing the input $(0, f(0))$, cannot identify $\mathcal{C}$. To see this, suppose by way of contradiction that **M InfPartSubBc**-identifies $\mathcal{C}$. Then we show how to **Bc**-identify $\mathcal{R}$, contradicting a result of Case and Smith [10]. Note that for every function $f \in \mathcal{R}$, there exists an e such that $\varphi_e(x) = e$, if $x = 0$; $\varphi_e(x) = f(x)$, otherwise. Thus, for every program $f \in \mathcal{R}$, there exists a function $g \in \mathcal{C}$, which differs from f only on input 0. Thus, **M** extends every partial recursive function with domain $N - \{0\}$. We will use this property to get a contradiction.

For a segment σ, let σ' denote the segment obtained from σ by replacing all occurences of $(0, w)$ in σ by $\#$, for any $w \in N$. For a program p, and a number z, let $E(p, z)$ be defined as follows:

$$\varphi_{E(p,z)}(x) = \begin{cases} z, & \text{if } x = 0; \\ \varphi_p(x), & \text{otherwise.} \end{cases}$$

Now define machine $\mathbf{M}'$ as follows.

$$\mathbf{M}'(\sigma) = \begin{cases} E(\mathbf{M}(\sigma'), z), & \text{if } (0, z) \in \text{content}(\sigma); \\ 0, & \text{otherwise.} \end{cases}$$

As $\mathbf{M}$ extends every partial recursive function which is not defined on input 0, it is easy to verify that $\mathbf{M}'$ **Bc**-identifies $\mathcal{R}$. However, this is not possible [10]. Thus, $\mathcal{C} \notin$ **InfPartSubBc**.

(b) In proof of Theorem 35 we showed that **InfTotSubBc** $-$ **Ex** $\neq \emptyset$. ∎

As **Ex** $\subset$ **Bc** (see [10]), Proposition 37 yields the following corollary.

Corollary 38 **InfPartSubBc** $\subset$ **Bc**.

5 Characterizations for Sublearning

In this section, we derive some characterizations for several types of sublearning. The first of these characterizations, for **AllTotSubEx**, turns out to be useful for proving other results.

Theorem 39 $\mathcal{C} \in$ **AllTotSubEx** *iff there exists a total recursive function F mapping SEG to programs such that:*

(a) For all $\sigma \in SEG$, such that content(σ) *has an extension in $\mathcal{C}$, $F(\sigma)$ is a program for a total function extending* content(σ).

(b) For all partial functions η with an extension in $\mathcal{C}$, there exists a $\sigma \in SEG$ such that content$(\sigma) \subseteq \eta \subseteq \varphi_{F(\sigma)}$.

Proof. $\Rightarrow$: By Theorem 28, **AllTotSubEx** $=$ **AllTotSubCons**. Suppose $\mathcal{C} \in$ **AllTotSubCons** as witnessed by $\mathbf{M}$. Then for each segment σ, define $F(\sigma)$ as follows:

Let $Cand_\sigma = \{\mathbf{M}(\sigma') \mid \sigma \subseteq \sigma' \ \wedge \ \text{content}(\sigma) = \text{content}(\sigma')\}$. Then, $F(\sigma) = Union(Cand_\sigma)$.

It is easy to see that F satisfies the requirement (a) of theorem, by consistency requirement on $\mathbf{M}$. To see (b), note that for each subfunction η of $f \in \mathcal{C}$, there exists a locking sequence for $\mathbf{M}$ on η (see remark after Theorem 7). Let this locking sequence be σ. This σ shows part (b).

$\Leftarrow$: Suppose F as in theorem is given. Then $\mathbf{M}$ on input σ, outputs $F(\tau)$, for the least segment τ (in some ordering of elements of SEG) such that

content(τ) $\subseteq$ content(σ), and $\varphi_{F(\tau)}$ extends content(σ).

For any subfunction η of $f \in \mathcal{C}$, and any text T for η, it follows using clause (b) that **M** would find a σ as in (b) (or a lesser one according to the fixed ordering of segments), such that F maps σ to a total extension of η. ∎

The following corollary "liberalizes" the characterization of **AllTotSubEx** from Theorem 39 in a sense, by making the function F mapping now from arbitrary finite functions rather than from the set SEG of segments.

Corollary 40 $\mathcal{C} \in$ **AllTotSubEx** *iff there exists a total recursive function F mapping finite functions to programs such that:*

(a) For all finite functions α with an extension in $\mathcal{C}$, $F(\alpha)$ is a program for a total function extending α.

(b) For all infinite partial functions η with an extension in $\mathcal{C}$, there exists a finite subfunction α of η such that $F(\alpha)$ is a program for an extension of η.

Proof. Note that **M** constructed in the $\Leftarrow$ direction of the proof of Theorem 39 is set-driven. Thus, we may assume without loss of generality that $\mathcal{C} \in$ **AllTotSubEx** is witnessed by a set-driven machine. Corollary now follows by noting that $\Rightarrow$ direction of the proof of Theorem 39 gives F to be set-driven, if **M** is set-driven. ∎

Corollary 42 below shows that a class which is closed under finite variations belongs to **AllTotSubEx** iff this class is a subclass of a recursively enumerable class. In order to prove this result we need Corollary 41 which, in turn, is a consequence from the characterization in Theorem 39.

Corollary 41 *Suppose $\mathcal{C} \in$ **AllTotSubEx**. Suppose further that $\mathcal{C}$ contains an extension of every finite partial function. Then $\mathcal{C} \in$ **NUM**.*

Proof. Let $\mathcal{C}$ be as in the hypothesis. Thus, the characterization Theorem 39, implies that range of F (as defined in Theorem 39) contains programs for only total functions. As range of F contains programs for all functions in $\mathcal{C}$, corollary follows. ∎

Recall that a class $\mathcal{C} \subseteq \mathcal{R}$ is closed under finite variations iff for all $f, g \in \mathcal{R}$ such that card($\{x \mid f(x) \neq g(x)\}$) $< \infty$, $f \in \mathcal{C}$ iff $g \in \mathcal{C}$.

Corollary 42 *Suppose $\mathcal{C}$ is closed under finite variations. Then $\mathcal{C} \in$ **AllTotSubEx** iff $\mathcal{C} \in$ **NUM**.*

Proof. Immediately from Corollary 41 and Proposition 22. ∎

Note that Corollary 42 does not hold for **InfTotSubEx** as the class: $\{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall^\infty x)[\pi_1(f(x)) = e]\}$ shows. This class and its closure under finite variations are in **InfTotSubEx**. However, the class is not contained in **NUM**.

Our next results show that there is a close connection between **AllPartSubEx**-learnability and measurability.

Definition 43 [7] A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be *measurable* iff there exists a numbering η such that (a) $\mathcal{C} \subseteq \{\eta_i \mid i \in N\}$, and (b) there exists a total recursive function F such that, for all i, x, y,

$$F(i,x,y) = \begin{cases} 1, & \text{if } \eta_i(x) = y; \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 44 *If $\mathcal{C}$ is measurable, then $\mathcal{C} \in$* **AllPartSubEx**.

Proof. Suppose $\mathcal{C}$ is measurable, as witnessed by numbering η. Let h be a total recursive function reducing η-programs to equivalent φ-programs. Then one can define $\mathbf{M}$ as follows:

$$\mathbf{M}(\sigma) = h(\min(\{i \mid \text{content}(\sigma) \subseteq \eta_i\}))$$

By measurability, it immediately follows that $\mathbf{M}$ **AllPartSubEx**-identifies $\mathcal{C}$ (moreover, $\mathbf{M}$ is also consistent on any input from the class). ∎

The converse of Theorem 44 is also valid provided the corresponding classes are closed under finite variations.

Theorem 45 *Suppose $\mathcal{C}$ is closed under finite variations. Then $\mathcal{C} \in$* **AllPartSubEx** *iff $\mathcal{C}$ is measurable.*

Proof. The sufficiency follows from Theorem 44. For the necessity, note that **AllPartSubEx** $\subseteq$ **AllPartSubCons**, by Theorem 28. Thus, if $\mathcal{C}$ is closed under finite variations, then $\mathcal{C}$ must be in $\mathcal{T}$**Cons**. The theorem now follows using Theorem 46 below. ∎

Theorem 46 *If $\mathcal{C} \in \mathcal{T}$**Cons**, then $\mathcal{C}$ is measurable.*

Proof. Suppose $\mathbf{M}$ $\mathcal{T}$**Cons** identifies $\mathcal{C}$. For $\sigma \in$ SEG, define a (possibly partial) function η_σ as follows.

$$\eta_\sigma(x) = \begin{cases} y, & \text{if } (x,y) \in \text{content}(\sigma); \\ y, & \text{if } (x,z) \notin \text{content}(\sigma) \text{ for all } z, \text{ and } \varphi_{\mathbf{M}(\sigma)}(x) = y \\ & \text{and } \mathbf{M}(\sigma) = \mathbf{M}(\sigma \cdot (x,y)). \\ \uparrow, & \text{otherwise.} \end{cases}$$

Note that one can test whether $\eta_\sigma(x) = y$ as follows. If content(σ) contains (x,z), for some z, then clearly, $\eta_\sigma(x) = y$ iff $(x,y) \in$ content(σ). Otherwise, $\eta_\sigma(x) = y$ iff $\mathbf{M}(\sigma \cdot (x,y)) = \mathbf{M}(\sigma)$. To see this, suppose $\mathbf{M}(\sigma \cdot (x,y)) = \mathbf{M}(\sigma)$. Then, by consistency of $\mathbf{M}$ on all inputs, we have $\varphi_{\mathbf{M}(\sigma)}(x) = y$, and thus $\eta_\sigma(x) = y$. On the other hand, if $\mathbf{M}(\sigma \cdot (x,y)) \neq \mathbf{M}(\sigma)$, then, by definition of η_σ, we have that $\eta_\sigma(x)$ cannot be y.

Thus, in all cases, we can determine if $\eta_\sigma(x) = y$.

Moreover, for every function $f \in \mathcal{C}$, there is $\sigma \in$ SEG with $\eta_\sigma = f$ due to the locking sequence property (see remark after Theorem 7) for $\mathbf{M}$ on functions from $\mathcal{C}$. Finally, define a numbering ψ by $\psi_i = \eta_{\sigma_i}$, where $\sigma_0, \sigma_1, \ldots$ is an effective enumeration of SEG. Then, obviously, $\mathcal{C}$ is measurable as witnessed by the numbering ψ. ∎

In general, a class is **AllPartSubEx**-learnable iff it is *weakly measurable*, as we will show now. Intuitively, for a weakly measurable class $\mathcal{C}$, the measurability property is required only for those functions within the corresponding numbering which have a "good chance" to belong to $\mathcal{C}$.

Definition 47 A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be *weakly measurable* iff there exist a computable numbering η and a recursive sequence $\alpha_0, \alpha_1, \ldots$ of finite functions (here recursive sequence $\alpha_0, \alpha_1, \ldots$ means that there exists a program which, on input i, enumerates all of α_i and then stops) such that

(1) for each i, $\alpha_i \subseteq \eta_i$,

(2) for each partial function ψ which has an extension in $\mathcal{C}$, there exists an i such that $\alpha_i \subseteq \psi \subseteq \eta_i$,

(3) there exists a partial recursive function F such that, for all i, x, y such that $\alpha_i \cup \{(x,y)\}$ has an extension in $\mathcal{C}$,

$$F(i,x,y) = \begin{cases} 1, & \text{if } \eta_i(x) = y; \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 48 $\mathcal{C} \in$ **AllPartSubEx** *iff* $\mathcal{C}$ *is weakly measurable.*

Proof. $\Leftarrow$: Suppose η and α_i are given as in the definition of weakly measurable. Then $\mathbf{M}(\sigma)$ is defined as follows. Notice that $\mathbf{M}$ may be undefined on some inputs (even for some inputs which are initial segments of texts for

functions in the class). However, for all texts T for subfunctions of functions in $\mathcal{C}$, $\mathbf{M}$ converges on almost all initial segments of T.

If there exists an i such that: (a) $\alpha_i \subseteq \text{content}(\sigma)$, and (b) for each $(x, y) \in \text{content}(\sigma)$, $F(i, x, y)$ converges to 1 within $|\sigma|$ steps, or $F(i, x, y)$ does not halt within $|\sigma|$ steps, *then* $\mathbf{M}(\sigma)$ outputs a standard φ program for η_i, for least such i (note that least such i, if it exists, can be found effectively). Otherwise, $\mathbf{M}(\sigma)$ is undefined.

Now suppose ψ is an input function, which is extended by some function f in $\mathcal{C}$. Let T be a text for ψ. Let m be least program such that $\alpha_m \subseteq \psi \subseteq \eta_m$. Let n be large enough so that: for all $j < m$, (c) and (d) below are satisfied.

(c) If $\alpha_j \subseteq \psi$, then for minimum x such that $\psi(x)\downarrow$ and $\eta_j(x) \neq \psi(x)$ (note that there exists such an x due to assumption on m), $F(j, x, \psi(x))$ converges within n steps, and $(x, \psi(x)) \in \text{content}(T[n])$

(d) $\alpha_m \subseteq \text{content}(T[n])$.

Note that there exists such an n, due to condition (3) in definition of weakly measurable, and the fact that T is a text for ψ, and $\alpha_m \subseteq \psi \subseteq \eta_m$. Thus for all $n' \geq n$, $\mathbf{M}(T[n'])$ is m.

$\Rightarrow$: By Theorem 28, we know that **AllPartSubEx** $\subseteq$ **AllPartSubCons**. Suppose $\mathbf{M}$ witnesses that $\mathcal{C} \in$ **AllPartSubCons**.

Define (possibly partial) function g_σ as follows.

$$g_\sigma(x) = \begin{cases} y, & \text{if } (x, y) \in \text{content}(\sigma); \\ y, & \text{if } (x, z) \notin \text{content}(\sigma) \text{ for all } z, \text{ and } \varphi_{\mathbf{M}(\sigma)}(x) = y \\ & \text{and } \mathbf{M}(\sigma) = \mathbf{M}(\sigma \cdot (x, y)); \\ \uparrow, & \text{otherwise.} \end{cases}$$

Assume some recursive ordering $\sigma_0, \sigma_1, \ldots$, of all the members of SEG. Now let $\eta_i = g_{\sigma_i}$, and $\alpha_i = \text{content}(\sigma_i)$.

Now define $F(i, x, y)$ as follows.

$$F(i, x, y) = \begin{cases} 1, & \text{if } (x, y) \in \text{content}(\sigma_i); \\ 0, & \text{if } (x, z) \in \text{content}(\sigma_i) \text{ for some } z \neq y; \\ 1, & \text{if } (x, z) \notin \text{content}(\sigma_i) \text{ for all } z \\ & \text{and } \mathbf{M}(\sigma_i \cdot (x, y))\downarrow = \mathbf{M}(\sigma_i)\downarrow; \\ 0, & \text{if } (x, z) \notin \text{content}(\sigma_i) \text{ for all } z \\ & \text{and } \mathbf{M}(\sigma_i \cdot (x, y))\downarrow \neq \mathbf{M}(\sigma_i)\downarrow; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Now, if $\sigma_i \cdot (x,y)$ is extended by a function in $\mathcal{C}$, then by consistency we have that $\mathbf{M}(\sigma_i)\downarrow$ and $\mathbf{M}(\sigma_i \cdot (x,y))\downarrow$, and $\varphi_{\mathbf{M}(\sigma_i\cdot(x,y))}(x) = y$. Thus, $g_{\sigma_i}(x) = y$, iff $\mathbf{M}(\sigma_i) = \mathbf{M}(\sigma_i \cdot (x,y))$. It follows that F satisfies (3) in the definition of weakly measurable.

(1) in the definition of weakly measurable follows by construction.

(2) in the definition of weakly measurable holds by locking sequence argument: for all ψ which have extension in the class, there exists a σ such that $\text{content}(\sigma) \subseteq \psi \subseteq \varphi_{\mathbf{M}(\sigma)}$, and for all x such that $\psi(x)\downarrow$, $\mathbf{M}(\sigma) = \mathbf{M}(\sigma \cdot (x,\psi(x)))$. Thus, for i such that $\sigma_i = \sigma$, we have $\alpha_i = \text{content}(\sigma_i) \subseteq \psi \subseteq \eta_i$. Thus (2) is satisfied. ∎

Finally, we characterize the classes from **AllTotSubEx** to be exactly the *weakly enumerable* classes. In a certain analogy to the notion of weak measurability, intuitively, a class $\mathcal{C}$ is weakly enumerable if any function within the corresponding numbering is *total* in case this function has a "good chance" to belong to $\mathcal{C}$.

Definition 49 A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be *weakly enumerable* iff there exist a computable numbering η and a recursive sequence $\alpha_0, \alpha_1, \ldots$ of finite functions such that

(1) for each i, $\alpha_i \subseteq \eta_i$,

(2) for each partial function ψ which has an extension in $\mathcal{C}$, there exists an i such that $\alpha_i \subseteq \psi \subseteq \eta_i$,

(3) for all i, such that α_i has an extension in $\mathcal{C}$, η_i is total.

Theorem 50 *$\mathcal{C} \in$ **AllTotSubEx** iff $\mathcal{C}$ is weakly enumerable.*

Proof. $\Leftarrow$: Suppose $\mathcal{C}$ is weakly enumerable as witnessed by η. Let $\eta_{i,s}$ denote the time-bounded computation of η:

$$\eta_{i,s} = \begin{cases} \eta_i(x), & \text{if } x < s, \text{ and } \eta_i(x) \text{ converges within } s \text{ steps;} \\ \uparrow, & \text{otherwise.} \end{cases}$$

Let h be such that, for all i, $\varphi_{h(i)} = \eta_i$. Define $\mathbf{M}$ as follows. Note that $\mathbf{M}$ may be undefined on some initial segments of texts T for partial functions with extensions in $\mathcal{C}$. However, $\mathbf{M}$ would be defined on almost all initial segments of T. $\mathbf{M}(\sigma) = h(i)$, for the least i such that $\alpha_i \subseteq \text{content}(\sigma)$, and $\text{content}(\sigma) \sim \eta_{i,|\sigma|}$. If no such i exists, then $\mathbf{M}(\sigma)$ diverges. Now fix any ψ with an extension in $\mathcal{C}$, and a text T for ψ. By property (2) of weak enumerability, $\mathbf{M}$ is defined on almost all initial segments of T, and by property (3) outputs only programs

for total functions on T. By property (2), and using consistency check done by $\mathbf{M}$, $\mathbf{M}(T)\downarrow = h(i)$ for the least i, such that $\alpha_i \subseteq \psi \subseteq \eta_i$.

$\Rightarrow$: Assume some recursive ordering $\sigma_0, \sigma_1, \ldots$, of all the members of SEG. Suppose $\mathcal{C} \in \mathbf{AllTotSubEx}$. Suppose F is as given by Theorem 39. Let $\alpha_i = \text{content}(\sigma_i)$, and

$$\eta_i = \begin{cases} \alpha_i(x), & \text{if } x \in \text{domain}(\alpha_i); \\ \varphi_{F(\sigma_i)}(x), & \text{otherwise.} \end{cases}$$

It is easy to verify that η satisfies the requirements (1), (2), (3) of the definition of weakly enumerable. ∎

Notice that all the characterizations above rely on certain *finite* subfunctions of the functions to be sublearned. These finite subfunctions may remind to the so-called telltale sets which were used in [1] for characterizing *language* learning from positive data. On the one hand, such a relation is not surprising, since any function can also be interpreted as a (special) language. Moreover, for **All**-sublearning which has been characterized in this section, we need *every* finite subfunction of any function from the class to be sublearned in order to form a stabilizing sequence for that finite function, that is, for learning *itself*. This, too, is an analogue to learning of *finite languages*, where, as a rule, also the whole (finite) languages constitute the corresponding telltale sets. On the other hand, our characterizations are, in a sense, more general than those from [1]. Indeed, while those were established for *enumerable* language classes, many sublearnable function classes are *not* contained in any enumerable class, see Figure 1.

6 Sublearning Versus Robust Learning

We start with defining robust learning formally.

Definition 51 [22] A *recursive operator* is an effective total mapping, Θ, from (possibly partial) functions to (possibly partial) functions, which satisfies the following properties:

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

Definition 52 [22] A recursive operator Θ is called *general recursive* iff Θ maps all total functions to total functions.

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

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

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

Proposition 53 [16] *There exists an effective enumeration, $\Theta_0, \Theta_1, \ldots$, of recursive operators satisfying condition* (d) *above such that, for all recursive operators Θ, there exists an $i \in N$ satisfying:*

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

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

Now, we are ready to define *robust* learning.

Definition 54 [14,16]

RobustEx $= \{\mathcal{C} \subseteq \mathcal{R} \mid (\forall$ general recursive operators $\Theta)[\Theta(\mathcal{C}) \in \mathbf{Ex}]\}$.

RobustCons $= \{\mathcal{C} \subseteq \mathcal{R} \mid (\forall$ general recursive operators $\Theta)[\Theta(\mathcal{C}) \in \mathbf{Cons}]\}$.

Proposition 55 [27,16] **NUM** $\subseteq$ **RobustEx**.

In this section, we compare the capabilities of sublearning and robust learning. The question of comparing these capabilities arises naturally insofar, as sublearning can intuitively be viewed as some special case of learning robustly. Actually, while robust learning requires that *all* projections of a given class of total recursive functions under all general recursive operators be learnable, see Definition 54, in sublearning only a special kind of projection is required so, namely, the given class of total recursive functions together with all of their subfunctions (or all of their infinite subfunctions, respectively). Nevertheless, as it follows from Theorems 59 and 63 below, the capabilities of robust learning and sublearning turn out to be incomparable. For proving this, we show that, on the one hand, **RobustCons**, and hence **RobustEx** contains classes which do not belong to the largest type of **Ex**-sublearning, **InfPartSubEx**, see Theorem 59. Notice that the proof of Theorem 59 is based on the proof of separation of robust and uniform robust learning in [9]. On the other hand,

we derive that already the smallest sublearning type, **AllTotSubEx**, contains classes which are out of **RobustEx**, see Theorem 63. Propositions 56 and 57 will be needed in order to prove Theorem 59. Finally, we exhibit that, under certain circumstances, the power of sublearning and robust learning *coincides*, see Theorem 64.

Proposition 56 *[9] There exists a K-recursive sequence of initial segments, $\sigma_0, \sigma_1, \ldots \in INITSEG_{0,1}$, such that for all $e \in N$, the following are satisfied.*

(a) $0^e1 \subseteq \text{content}(\sigma_e)$.

(b) *For all $e' \leq e$, if $\Theta_{e'}$ is general recursive, then either $\Theta_{e'}(\sigma_e) \not\sim \Theta_{e'}(0^{|\sigma_e|})$ or for all $f \in \mathcal{T}_{0,1}$ extending* $\text{content}(\sigma_e)$, $\Theta_{e'}(f) = \Theta_{e'}(\text{Zero})$.

Proof. We define σ_e (using oracle for K) as follows. Initially, let $\sigma_e^0 = 0^e1$. For $e' \leq e$, define $\sigma_e^{e'+1}$ as follows: if there exists an extension $\tau \in \text{INITSEG}_{0,1}$ of $\sigma_e^{e'}$, such that $\Theta_{e'}(\tau) \not\sim \Theta_{e'}(0^{|\tau|})$, then let $\sigma_e^{e'+1} = \tau$; otherwise, let $\sigma_e^{e'+1} = \sigma_e^{e'}$.

Now let $\sigma_e = \sigma_e^{e+1}$ as defined above. It is easy to verify that the proposition is satisfied. ∎

Proposition 57 *[9] There exists an infinite increasing sequence $a_0, a_1, \ldots$ of natural numbers such that for $A = \{a_i \mid i \in N\}$, the following properties are satisfied for all $k \in N$.*

(a) *The complement of A is recursively enumerable relative to K.*

(b) *φ_{a_k} is total.*

(c) *For all $e \leq a_k$ such that φ_e is total, $\varphi_e(x) \leq \varphi_{a_{k+1}}(x)$ for all $x \in N$.*

(d) *For σ_e as defined in Proposition 56, $|\sigma_{a_k}| \leq a_{k+1}$.*

Proof. The construction of a_i's is done using movable markers (using oracle for K). Let a_i^s denote the value of a_i at the beginning of stage s in the construction. It will be the case that, for all s and i, either $a_i^s = a_i^{s+1}$, or $a_i^{s+1} > s$. This allows us to ensure property (a). The construction itself directly implements properties (b) to (d). Let *pad* be a 1–1 padding function [22] such that for all i, j, $\varphi_{pad(i,j)} = \varphi_i$, and $pad(i,j) \geq i + j$.

We assume without loss of generality that φ_0 is total. Initially, let $a_0^0 = 0$, and $a_{i+1}^0 = pad(0, |\sigma_{a_i^0}|)$ (this ensures $a_{i+1}^0 \geq |\sigma_{a_i^0}| > a_i^0$). Go to stage 0.

Stage s
1. If there exist a k, $0 < k \leq s$, and $x \leq s$ such that:

(i) $\varphi_{a_k^s}(x)\uparrow$ or
(ii) for some $e \leq a_{k-1}^s$, $[(\forall y \leq s)[\varphi_e(y)\downarrow]$ and $\varphi_e(x) > \varphi_{a_k^s}(x)]$

Then pick least such k and go to step 2. If there is no such k, then for all i, let $a_i^{s+1} = a_i^s$, and go to stage $s+1$.

2. For $i < k$, let $a_i^{s+1} = a_i^s$.
3. Let j be the least number such that
(i) $(\forall y \leq s)[\varphi_j(y)\downarrow]$ and
(ii) for all $e \leq a_{k-1}^s$, if for all $y \leq s$, $\varphi_e(y)\downarrow$, then for all $y \leq s$, $\varphi_j(y) \geq \varphi_e(y)$.

Let $a_k^{s+1} = pad(j, |\sigma_{a_{k-1}^s}| + s + 1)$.
4. For $i > k$, let $a_i^{s+1} = pad(0, |\sigma_{a_{i-1}^{s+1}}| + s + 1)$.
5. Go to stage $s+1$.

End stage s

We claim (by induction on k) that $\lim_{s\to\infty} a_k^s\downarrow$ for each k. To see this, note that once all the a_i, $i < k$, have stabilized, step 3 would eventually pick a j such that φ_j is total, and for all $e \leq a_{k-1}$, if φ_e is total then $\varphi_e \leq \varphi_j$. Thereafter a_k would not be changed.

We now show the various properties claimed in the proposition. One can enumerate $\overline{A}$ (using oracle for K) using the following property: $x \in \overline{A}$ iff there exists a stage $s > x$ such that, for all $i \leq x$, $a_i^s \neq x$. Thus (a) holds. (b) and (c) hold due to the check in step 1. (d) trivially holds due to padding used for definition of a_i^s for all s. ∎

Definition 58 Suppose $h \in \mathcal{R}$. Let $\mathcal{B}_h = \{\varphi_e \mid \varphi_e \in \mathcal{R}_{0,1} \ \wedge \ (\forall^\infty x)[\Phi_e(x) \leq h(x)]\}$.

Intuitively, $\mathcal{B}_h$ denotes the class of total recursive predicates whose complexity is almost everywhere bounded by h. We assume without loss of generality that φ_0 is large enough to ensure FINSUP $\subseteq \mathcal{B}_{\varphi_0}$. Thus for a_i as in Proposition 57, FINSUP $\subseteq \mathcal{B}_{\varphi_{a_i}}$, for all i.

Theorem 59 RobustCons – InfPartSubEx $\neq \emptyset$.

Proof. Fix $\sigma_0, \sigma_1, \ldots$ as in Proposition 56, and $a_0, a_1, \ldots$ as in Proposition 57.

Let $G_k = \mathcal{B}_{\varphi_{a_k}} \cap \{f \in \mathcal{R}_{0,1} \mid \sigma_{a_k} \subseteq f\}$.

The main idea of the construction is to build a diagonalizing class by taking at most finitely many functions from each G_k.

Claim 60 *For each i, $\mathbf{M}_i$ does not* **InfPartSubEx***-identify $\bigcup_{k\geq i} G_k$.*

PROOF. For each $i \in N$, $\sigma \in \text{INITSEG}_{0,1}$, we define $g_{\langle i,\sigma\rangle}$ in stages as follows. Initially, let $g_{\langle i,\sigma\rangle}(x) = \sigma(x)$, for x in domain(σ). Let n_s denote the least number x such that $g_{\langle i,\sigma\rangle}(x)$ is not defined before stage s. For $\sigma \subseteq \tau$, let $X(\tau,\sigma)$ denote the segment formed by replacing all elements in τ, which belong to content(σ), by #.

Intuitively, the construction below would try to find a total extension $g_{i,\sigma}$ of content(σ) such that $\mathbf{M}_i$ makes infinitely many mind changes on some text for $g_{i,\sigma} -$ content(σ).

Stage s
  Search for an extension $\tau \in \text{INITSEG}_{0,1}$ of $g_{\langle i,\sigma\rangle}[n_s]$ such that $\mathbf{M}_i(X(\tau,\sigma)) \neq \mathbf{M}_i(X(g_{\langle i,\sigma\rangle}[n_s],\sigma))$.
  If and when such a τ is found, extend $g_{\langle i,\sigma\rangle}$ to content(τ) and go to stage $s+1$.
End stage s

Note that G_i contains every function in FINSUP which extends content(σ_{a_i}). Thus, if $\mathbf{M}_i$ **InfPartSubEx**-identifies G_i, then for all $\sigma \in \text{INITSEG}_{0,1}$ such that $|\sigma| \geq |\sigma_{a_i}|$, $g_{\langle i,\sigma\rangle}$ is total (as $\mathbf{M}_i$ must converge to an extension on texts of all partial functions with finite support, whose domain is a subset of the complement of the domain of σ_{a_i}; thus search for mind change in the construction above is always successful). Thus, the complexity of all functions in $\{g_{\langle i,\sigma\rangle} \mid \sigma \in \text{INITSEG}_{0,1} \wedge |\sigma| \geq |\sigma_{a_i}|\}$ is dominated by a total recursive function, say h. It follows that for all but finitely many k, $g_{\langle i,\sigma_{a_k}\rangle} \in \mathcal{B}_{\varphi_{a_k}}$. However, $\mathbf{M}_i$ does not **InfPartSubEx**-identify $g_{\langle i,\sigma_{a_k}\rangle}$, for all $k \geq i$. Claim follows. □

We continue with the proof of the theorem. For each $e \in N$, let f_i denote a function in $\bigcup_{k\geq i} G_k$, such that $\mathbf{M}_i$ does not **InfPartSubEx**-identify f_i.

Let $\mathcal{S} = \{f_i \mid i \in N\}$. Let $H_k = \mathcal{S} \cap G_k$. It is easy to verify that H_k is finite (since $f_i \notin \bigcup_{k<i} G_k$).

Claim 61 $\mathcal{S} \notin$ **InfPartSubEx**.

PROOF. Follows by the selection of f_i diagonalizing against $\mathbf{M}_i$. □

Claim 62 $\mathcal{S} \in$ **RobustCons**.

PROOF. Suppose $\Theta = \Theta_k$ is general recursive. We need to show that $\Theta_k(\mathcal{S}) \in$ **Cons**. Let $A = \{a_i \mid i \in N\}$. Since $\overline{A}$ is r.e. in K, there exists a recursive sequence $c_0, c_1, \ldots$, such that each $a \in A$, $a > a_k$, appears infinitely often in the sequence, and each $a \notin A$ or $a \leq a_k$, appears only finitely often in

the sequence. Let $\sigma_{e,t} \in \text{INITSEG}_{0,1}$ be such that $\sigma_{e,t} \supseteq 0^e1$, and $\sigma_{e,t}$ can be obtained effectively from e, t, and $\lim_{t\to\infty} \sigma_{e,t} = \sigma_e$. Note that there exist such $\sigma_{e,t}$ due to K-recursiveness of the sequence $\sigma_0, \sigma_1, \ldots$.

Note that there exists a total recursive h such that, if φ_e is total recursive then, $\mathbf{M}_{h(e)}$ **Cons**-identifies $\Theta(\mathcal{B}_{\varphi_e})$. Fix such a total recursive h.

Let $H = \{\text{Zero}\} \cup H_0 \cup H_1 \cup \ldots \cup H_k$. H and $\Theta(H)$ are finite sets of total recursive functions.

Define $\mathbf{M}$ as follows.

$\mathbf{M}(T[n])$
1. If for some $g \in \Theta(H)$, content$(T[n]) \subseteq g$, then output a canonical program for one such g.
2. Else, let $t \leq n$ be the largest number such that $\Theta(\sigma_{c_t,n}) \sim$ content$(T[n])$, and $\Theta(\sigma_{c_t,n}) \not\sim \Theta(\text{Zero})$. (Note: if no such t exists, then take $t = 0$.)
   Dovetail the following steps until one of them succeeds. If steps 2.1 or 2.2 succeed, then go to step 3. If step 2.3 succeeds, then go to step 4.
   2.1 There exists an $s > n$, such that $c_s \neq c_t$, and $\Theta(\sigma_{c_s,s}) \sim$ content$(T[n])$, and $\Theta(\sigma_{c_s,s}) \not\sim \Theta(\text{Zero})$.
   2.2 There exists an $s > n$, such that $\sigma_{c_t,s} \neq \sigma_{c_t,n}$.
   2.3 $\mathbf{M}_{h(c_t)}(T[n])\downarrow$, and content$(T[n]) \subseteq \varphi_{\mathbf{M}_{h(c_t)}(T[n])}$.
3. Output a program for 0-extension of content$(T[n])$.
4. Output $\mathbf{M}_{h(c_t)}(T[n])$.

End

It is easy to verify that whenever $\mathbf{M}(T[n])$ is defined, content$(T[n]) \subseteq \varphi_{\mathbf{M}(T[n])}$. Also, if $f \in \Theta(H)$, then $\mathbf{M}$ **Cons**-identifies f.

Now, consider any $f \in \Theta(\mathcal{S}) - \Theta(H)$, and any text T for f. Note that there exists a unique $i > k$ such that $f \sim \Theta(\sigma_{a_i})$ and $\Theta(\sigma_{a_i}) \not\sim \Theta(\text{Zero})$ (due to definition of σ_{a_j}'s). Fix such i. Also, since $f \neq \Theta(\text{Zero})$, there exist only finitely many e such that $f \sim \Theta(0^e1)$.

We first claim that $\mathbf{M}(T[n])$ is defined for all n. To see this, note that if $c_t \neq a_i$ or $\sigma_{c_t,n} \neq \sigma_{a_i}$, then step 2.1 or step 2.2 would eventually succeed. Otherwise, since $f \in \Theta(H_i) \subseteq \Theta(\mathcal{B}_{\varphi_{a_i}})$, step 2.3 would eventually succeed (since $\mathbf{M}_{h(a_i)}$ **Cons**-identifies $\Theta(\mathcal{B}_{\varphi_{a_i}})$).

Thus, it suffices to show that $\mathbf{M}$ **Ex**-identifies f. Let r be such that $f \not\sim \Theta(0^r)$. Let m and $n > m$ be large enough such that (i) to (iv) hold.

(i) content$(T[n]) \not\sim \Theta(0^r)$.

(ii) $c_m = a_i$, and for all $s \geq m$, $\sigma_{a_i,s} = \sigma_{a_i,m}$.

(iii) For all $e < r$ and $t > m$, if $e \notin A$ or $e \leq a_k$, then $c_t \neq e$.

(iv) For all $e < r$ and $t > m$, if $e \in A - \{a_i\}$ and $e > a_k$, then $\Theta(\sigma_{e,t}) \not\sim$ content$(T[n])$ or $\Theta(\sigma_{e,t}) \sim \Theta(\text{Zero})$.

Note that there exist such m, n. Thus, for all $n' \geq n$, in computation of $\mathbf{M}(T[n'])$, c_t would be a_i, and step 2.1 and step 2.2 would not succeed. Thus step 2.3 would succeed, and $\mathbf{M}$ would output $\mathbf{M}_{h(a_i)}(T[n'])$. Thus $\mathbf{M}$ **Ex**-identifies f, since $\mathbf{M}_{h(a_i)}$ **Ex**-identifies f. □

Theorem follows from the above claims. ∎

We now show that sublearning is "rich" in comparison to robust learning.

Theorem 63 $\mathbf{AllTotSubEx} - \mathbf{RobustEx} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid (\forall x)[\varphi_{\pi_1(f(x))} = f]\}$. $\mathcal{C}$ is clearly in **AllTotSubEx**, as any data point gives away a program for f.

On the other hand, $\mathcal{C} \notin \mathbf{RobustEx}$. To see this, consider $\Theta(f)(x) = \pi_2(f(x))$. Now $\Theta(\mathcal{C})$ contains every total recursive function, as for any total recursive function g, there exists an e such that $\varphi_e(x) = \langle e, g(x)\rangle$. As $\mathcal{R} \notin \mathbf{Ex}$ (see [15]), we immediately have that $\mathcal{C} \notin \mathbf{RobustEx}$. ∎

While the class from the proof of Theorem 63 is in a sense "*maximally* self-describing", this property turns out to be far from necessary for the classes from $\mathbf{AllTotSubEx} - \mathbf{RobustEx}$. Actually, as an alternative proof of Theorem 63, consider the following class:

$$\mathcal{C} = \{f \mid (\exists e \mid \varphi_e = f)[(\forall x < e)[f(x) \in \{0,1\}] \ \wedge \ (\forall x \geq e)[f(x) \in \{2,3\}]]\}.$$

This class, in turn, could be called "*minimally* self-describing", since, for any function $f \in \mathcal{C}$, there is only *one* point, namely the least x such that $f(x) \in \{2,3\}$, which yields a program for f in a self-describing manner. Nevertheless, $\mathcal{C}$ belongs to $\mathbf{AllTotSubEx} - \mathbf{RobustEx}$ as well. Indeed, $\mathcal{C}$ is in **AllTotSubEx** despite the fact that this "self-describing" point may *not* belong to the corresponding subfunction to be learned. But this possibly missing information can be compensated as follows. On input σ, the learner outputs a program for the 0-extension of the input, if the input function has range only in $\{0,1\}$. Otherwise, the least x, such that $(x,2)$ or $(x,3)$ is in content(σ), gives away a *bound* on the program for f. This bound allows us to learn an extension of the input, by using the technique from [12]: we first cancel out all programs less than the bound which are inconsistent with the input. Then

we use *Union* of the remaining programs.

On the other hand,

$$\Theta(f)(x) = \begin{cases} f(x), & \text{if } f(x) \leq 1; \\ f(x) - 2, & \text{otherwise.} \end{cases}$$

is general recursive, and $\Theta(\mathcal{C}) = \mathcal{R}_{0,1}$. To see the latter note that for every $\{0,1\}$-valued total recursive function g, there exists an e such that

$$\varphi_e(x) = \begin{cases} f(x), & \text{if } x < e; \\ f(x) + 2, & \text{otherwise.} \end{cases}$$

Since $\mathcal{R}_{0,1} \notin \mathbf{Ex}$, see [15], it follows that $\mathcal{C} \notin \mathbf{RobustEx}$.

Finally, we show that sublearning and robust learning are of the *same* power if we confine ourselves to classes that are closed under finite variations.

Theorem 64 *Suppose $\mathcal{C}$ is closed under finite variations. Then $\mathcal{C} \in$ **AllTotSubEx** iff $\mathcal{C} \in$ **RobustEx** iff $\mathcal{C} \in$ **NUM**.*

Proof. Let $\mathcal{C} \subseteq \mathcal{R}$ be closed under finite variations. Then, by Corollary 42, $\mathcal{C} \in$ **AllTotSubEx** iff $\mathcal{C} \in$ **NUM**. On the other hand, $\mathcal{C} \in$ **RobustEx** iff $\mathcal{C} \in$ **NUM** was shown in [21]. ∎

7 Sublearning Versus Other Learning Criteria

7.1 Consistent Learning

We have already seen in Theorem 28 that there is a close connection between general sublearning and consistent sublearning. Consequently, we find it interesting enough to clarify the relations between consistent sublearning and consistent learning as well. This will be done now by Theorems 65, 66, and 67. These results tell us, informally, that each type of consistent sublearning contains classes which cannot be learned by the "next stricter" (in the sense of Theorem 15) type of consistent learning.

Theorem 65 **AllTotSubCons** $-$ $\mathcal{R}$**Cons** $\neq \emptyset$

Proof. Let F be an increasing limiting recursive function which dominates all total recursive functions, for example, $F(x) = x + \sum_{i \leq x, y \leq x, \varphi_i(y)\downarrow} \varphi_i(y)$.

Let $\mathcal{C} = \{f \in \mathcal{R}_{0,1} \mid f \neq \text{Zero} \ \wedge \ F(\min(\{x \mid f(x) \neq 0\})) \geq \text{MinProg}(f)\}$. $\mathcal{C} \notin \mathcal{R}\textbf{Cons}$ was shown in [9].

We will now show that $\mathcal{C} \in \textbf{AllTotSubBc}$. $\mathcal{C} \in \textbf{AllTotSubCons}$ will then follow from Theorem 28 and Theorem 34.

Suppose F is computed in the limit by $g(\cdot, \cdot)$.

Consider the following machine $\mathbf{M}$. If $\text{content}(\sigma) \subseteq \text{Zero}$, then output a standard program for Zero. Otherwise, let $m_\sigma = \min(\{x \mid (x, 1) \in \text{content}(\sigma)\})$. Let $Cand_\sigma = \{i \mid (\exists s)[i \leq g(m_\sigma, s)] \ \wedge \ \text{content}(\sigma) \subseteq \varphi_i\}$. Output $\mathbf{M}(\sigma) = Union(Cand_\sigma)$. For $f \in \mathcal{C} - \{\text{Zero}\}$, and any $\eta \subseteq f$, for any text T for η, it is easy to verify that, for all n, with $\text{content}(T[n]) \not\subseteq \text{Zero}$, (i) $Cand_{T[n]}$ contains a program for f, (as, for $m = \min(\{x \mid f(x) = 1\})$, $\text{MinProg}(f) \leq F(m)$ and thus there exists an s such that $g(m, s) \geq \text{MinProg}(f)$); (ii) $\lim_{n\to\infty} Cand_{T[n]}$, is finite and consists only of programs extending η. Thus, for all but finitely many n, $\mathbf{M}(T[n])$ would be a program for an extension of η. ∎

Theorem 66 $\textbf{AllTotSub}\mathcal{R}\textbf{Cons} - \mathcal{T}\textbf{Cons} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall x)[\pi_1(f(x)) = e]\}$.

Clearly $\mathcal{C} \in \textbf{AllTotSub}\mathcal{R}\textbf{Cons}$. However $\mathcal{C} \notin \mathcal{T}\textbf{Cons}$, using diagonalization as follows.

Suppose $\mathbf{M}$ $\mathcal{T}\textbf{Cons}$-identifies above class. Note that $\mathcal{T}\textbf{Cons}$ machine is always consistent with the input (even from outside the class). Thus, if $y \neq z$, then $\mathbf{M}(\sigma \cdot (x, y)) \neq \mathbf{M}(\sigma \cdot (x, z))$, for all σ such that x is not in domain of $\text{content}(\sigma)$. Thus one may define φ_e using Kleene recursion theorem [22] as follows: $\varphi_e(x) = \langle e, w\rangle$, for a $w \in \{0, 1\}$, which causes a mind change $\mathbf{M}(\varphi_e[x]) \neq \mathbf{M}(\varphi_e[x] \cdot (x, \langle e, w\rangle))$. This φ_e is in $\mathcal{C}$, but $\mathbf{M}$ on φ_e makes infinitely many mind changes. ∎

Theorem 67 $\textbf{InfTotSubEx} - \textbf{Cons} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall^\infty x)[\pi_1(f(x)) = e]\}$. $\mathcal{C}$ is clearly in $\textbf{InfTotSubEx}$. $\mathcal{C} \notin \textbf{Cons}$ was shown in the proof of Proposition 29. ∎

An alternative proof of above theorem can also be obtained using the alternative proof given for Theorem 25.

Total learning requires that not only the final hypothesis of the learning process must describe a total recursive function, namely the function to be learned, but also the *intermediate* hypotheses have to be *total* ones as well, see Definition 13. As Theorem 68 shows, this additional requirement can also be fulfilled for all sublearnable classes from **AllTotSubEx**. Recall that in **Tot**-sublearning, by definition, only the *final* hypothesis is required to describe a *total* recursive function, see Definition 17. On the other hand, all the other sublearning types turn out to be *incomparable* to total learning, see Corollary 72.

Theorem 68 $\mathbf{AllTotSubEx} \subseteq \mathbf{TEx}$.

Proof. It suffices to note that the machine constructed in the proof of $\Leftarrow$ direction of Theorem 39 witnesses the class $\mathcal{C}$ to be in **TEx**. ∎

Theorem 69 $\mathbf{TEx} - \mathbf{InfPartSubEx} \neq \emptyset$.

Proof. The class $\mathcal{C} = \{f \in \mathcal{R} \mid \varphi_{f(0)} = f\}$ witnesses the separation. $\mathcal{C}$ is clearly in **TEx**. It was shown in Proposition 37 that $\mathcal{C} \notin \mathbf{InfPartSubBc}$, and hence not in **InfPartSubEx**. ∎

Theorem 70 $\mathbf{InfTotSubEx} - \mathbf{TEx} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid (\exists e \mid \varphi_e = f)(\forall^\infty x)[\pi_1(f(x)) = e]\}$. Clearly, $\mathcal{C} \in \mathbf{InfTotSubEx}$. $\mathcal{C} \notin \mathbf{TEx}$ can be shown as follows.

Suppose by way of contradiction that $\mathbf{M}$ **TEx**-identifies $\mathcal{C}$. Clearly, $\mathcal{C} \notin \mathbf{NUM}$. Thus there must exist an input σ such that $\mathbf{M}(\sigma)$ is not a program for a total function. Now, by Kleene recursion theorem [22], there exists an e such that

$$\varphi_e(x) = \begin{cases} y, & \text{if } (x,y) \in \text{content}(\sigma) \text{ for some } y; \\ \langle e, 0\rangle, & \text{otherwise.} \end{cases}$$

Now $\varphi_e \in \mathcal{C}$, but $\mathbf{M}$ does not **TEx**-identify φ_e. ∎

Theorem 71 $\mathbf{AllPartSubEx} - \mathbf{TEx} \neq \emptyset$.

Proof. Let $\mathcal{C} = \{f \in \mathcal{R} \mid [\text{card}(\text{range}(f)) < \infty]$ and $(\forall e \in \text{range}(f))[W_e = f^{-1}(e)]\ \}$.

Clearly, $\mathcal{C} \in \mathbf{AllPartSubEx}$. The proof of Theorem 23 showing that $\mathcal{C}$ is not in **InfTotSubEx** can also be used to show that $\mathcal{C} \notin \mathbf{TEx}$, as step 1.1 b) would

always succeed for diagonalizing against **TEx** machines. ∎

Corollary 72 **AllTotSubEx** $\subset$ **TEx**.

AllPartSubEx $\triangle$ **TEx**.

InfPartSubEx $\triangle$ **TEx**.

InfTotSubEx $\triangle$ **TEx**.

Proof. Immediately from Theorems 68 to 71. ∎

8 Acknowledgements

Sanjay Jain was supported in part by NUS grant number R252-000-127-112. We would like to thank the anonymous referees for valuable comments and suggestions.

References

[1] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[2] J. Bārzdiņš. Prognostication of automata and functions. *Information Processing*, 1:81–84, 1971.

[3] J. Bārzdiņš. Inductive inference of automata, functions and programs. In *Int. Math. Congress, Vancouver*, pages 771–776, 1974.

[4] J. Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.

[5] J. Bārzdiņš and R. Freivalds. Prediction and limiting synthesis of recursively enumerable classes of functions. *Latvijas Valsts Univ. Zimatm. Raksti*, 210:101–111, 1974.

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

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

[8] J. Case, S. Jain, M. Ott, A. Sharma, and F. Stephan. Robust learning aided by context. *Journal of Computer and System Sciences (Special Issue for COLT'98)*, 60:234–257, 2000.

[9] J. Case, S. Jain, F. Stephan, and R. Wiehagen. Robust learning – rich and poor. *Journal of Computer and System Sciences*, 2004. To Appear.

[10] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[11] R. Freivalds, J. Bārzdiņš, and K. Podnieks. Inductive inference of recursive functions: Complexity bounds. In J. Bārzdiņš and D. Bjørner, editors, *Baltic Computer Science*, volume 502 of *Lecture Notes in Computer Science*, pages 111–155. Springer-Verlag, 1991.

[12] R. Freivalds and R. Wiehagen. Inductive inference with additional information. *Journal of Information Processing and Cybernetics (EIK)*, 15:179–195, 1979.

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

[14] M. Fulk. Robust separations in inductive inference. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 405–410. IEEE Computer Society Press, 1990.

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

[16] S. Jain, C. Smith, and R. Wiehagen. Robust learning is rich. *Journal of Computer and System Sciences*, 62(1):178–212, 2001.

[17] K. P. Jantke and H.-R. Beick. Combining postulates of naturalness in inductive inference. *Journal of Information Processing and Cybernetics (EIK)*, 17:465–484, 1981.

[18] S. Kurtz and C. Smith. On the role of search for learning. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 303–311. Morgan Kaufmann, 1989.

[19] S. Kurtz, C. Smith, and R. Wiehagen. On the role of search for learning from examples. *Journal of Experimental and Theoretical Artificial Intelligence*, 13:24–43, 2001.

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

[21] M. Ott and F. Stephan. Avoiding coding tricks by hyperrobust learning. In P. Vitányi, editor, *Fourth European Conference on Computational Learning Theory*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 183–197. Springer-Verlag, 1999.

[22] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.

[23] R. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, 1987.

[24] R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)*, 12:93–99, 1976.

[25] R. Wiehagen and W. Liepe. Charakteristische Eigenschaften von erkennbaren Klassen rekursiver Funktionen. *Journal of Information Processing and Cybernetics (EIK)*, 12:421–438, 1976.

[26] R. Wiehagen and T. Zeugmann. Learning and consistency. In K. P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 1–24. Springer-Verlag, 1995.

[27] T. Zeugmann. On Bārzdiņš' conjecture. In K. P. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the International Workshop*, volume 265 of *Lecture Notes in Computer Science*, pages 220–227. Springer-Verlag, 1986.