Measuring Learning Complexity with Criteria Epitomizers

John Case[1] **and Timo Kötzing**[2]

1 **Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586, USA**
`case@cis.udel.edu`

2 **Max-Planck Institute for Informatics, 66123 Saarbrücken, Germany**
`koetzing@mpi-inf.mpg.de`

Abstract

In prior papers, beginning with the seminal work by Freivalds et al. 1995, the notion of *intrinsic complexity* is used to analyze the learning complexity of sets of functions in a Gold-style learning setting. Herein are pointed out some weaknesses of this notion. Offered is an alternative based on *epitomizing sets* of functions – sets, which are learnable under a given learning criterion, but not under other criteria which are not at least as powerful.

To capture the idea of epitomizing sets, new reducibility notions are given based on *robust learning* (closure of learning under certain classes of operators). Various degrees of epitomizing sets are *characterized* as the sets complete with respect to corresponding reducibility notions! These characterizations also provide an easy method for showing sets to be epitomizers, and they are, then, employed to prove several sets to be epitomizing.

Furthermore, a scheme is provided to generate easily *very strong* epitomizers for a multitude of learning criteria. These strong epitomizers are so-called *self-learning* sets, previously applied by Case & Kötzing, 2010. These strong epitomizers can be generated and employed in a myriad of settings to witness the strict separation in learning power between the criteria so epitomized and other not as powerful criteria!

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases Algorithmic Learning Theory, Learning Complexity, Robustness in Learning

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.320

1 Introduction

We analyze the problem of algorithmically learning a description for an infinite sequence (a function from the natural numbers into the natural numbers) when presented larger and larger initial segments of that sequence. For example, a learner h might be presented more and more of the sequence $g = 1, 4, 9, 16, \ldots$. After each new datum of g, h may output a description of a function as its conjecture. For example, h might output a program for the constantly 1 function after seeing the first element of this sequence g and a program for the squaring function on all the other data from g. Many criteria for saying whether h is *successful* on g have been proposed in the literature. Gold, in his seminal paper [16], gave a first, simple learning criterion, later called in [11] *Ex-learning*[1], where a learner is

[1] *Ex* stands for *explanatory*.

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 320–331

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


successful iff it eventually stops changing its conjectures, and its final conjecture is a correct description for the input sequence.

Trivially, each single, describable sequence g has a suitable constant function as an Ex-learner (this learner constantly outputs a description for g). Thus, we are interested for which *sets of functions* $\mathcal{S}$ is there a *single learner* h learning *each* member of $\mathcal{S}$. We are interested herein in learning sets of total computable functions, and we will use (codes of) *programs* from a fixed *programming system* as possible conjectured descriptions for the functions.[2] This framework is known as *function learning in the limit* and has been studied extensively, using a wide range of learning criteria similar to Ex-learning (see, for example, the text book [19]).

Freivalds et al. [12] considered how to define *learning complexity* of a set of learnable functions. They introduced the seminal notion of *intrinsic complexity* and defined, for learning criteria I, a corresponding reducibility relation $\leq^I$. *Intrinsic* here is intrinsic to a learning task or problem $\mathcal{S}$, not to particular learning algorithms for $\mathcal{S}$. The idea is that, if $\mathcal{S} \leq^I \mathcal{S}'$, then $\mathcal{S}'$ is at least as hard to I-learn as is $\mathcal{S}$. In particular, [12] shows that, if $\mathcal{S} \leq^{\mathrm{Ex}} \mathcal{S}'$ and $\mathcal{S}'$ is Ex-learnable, then $\mathcal{S}$ is Ex-learnable. This intrinsic complexity has been further studied in some detail, see, for *example*, [12, 18, 17].

From [12], for a given learning criterion I, an I-learnable set of functions $\mathcal{S}_0$ is said to be $\leq^I$*-complete* iff, for all I-learnable sets of functions $\mathcal{S}$, $\mathcal{S} \leq^I \mathcal{S}_0$. As far as $\leq^I$ describes the relative difficulty of learnability, $\leq^I$-complete sets are the most difficult to I-learn. [12] shows that the set $\mathcal{S}_{\mathrm{FinSup}}$ of all computable functions of finite support[3] is $\leq^{\mathrm{Ex}}$-complete. These notions from [12] are structural analogs, for example, to the various notions from complexity theory of polynomial time reducibility and completeness.

There are at least two problems connected with the notion of intrinsic complexity from [12].

(i) For some learning criteria I, the relation $\leq^I$ is not very fine-grained. In particular, there are $\leq^I$-complete sets of functions which are also learnable with respect to much more restricted learning criteria (see Theorem 4.2 below).

(ii) There are learning criteria I and sets of functions $\mathcal{S}, \mathcal{S}'$ such that $\mathcal{S} \leq^I \mathcal{S}'$ and $\mathcal{S}'$ is I-learnable, but $\mathcal{S}$ is *not* I-learnable (see Theorem 4.3 below).

In this paper we quantify the difficulty of learning a given class of functions in a *new* way. First, we consider the following concept, essentially from [12]. A set of functions $\mathcal{S}$ *epitomizes* a learning criterion I *with respect to* some class of learning criteria $\mathcal{I}$, iff, $\mathcal{S}$ is I-learnable, and, for each $I' \in \mathcal{I}$, if some I-learnable task is too hard to be I'-learned, then $\mathcal{S}$ is already such an example task too hard to be I'-learned.[4]

We believe that epitomization nicely captures the learning complexity of a set of functions. Hence, the work herein aims at *finding* such epitomizers. Naturally, the interest is in epitomizers with respect to as large as possible classes of learning criteria $\mathcal{I}$. We give epitomizers with respect to classes of all learning criteria which are *robust* with respect to certain classes of operators (operating on functions). Essentially, a learning criterion I is *robust* with respect to a given class of operators $\mathfrak{O}$ iff, for each I-learnable task $\mathcal{S}$ and each

[2] One could, for example, think of the programming system as one of Java, C, Turing machines, ...
[3] A (total) function has *finite support* iff only finitely many arguments have a function value other than 0.
[4] Note that [12] called epitomizing sets *characteristic*. To our best knowledge, neither this term nor the concept caught on in the later literature — until now.

operator $\Theta \in \mathfrak{O}$, $\Theta(\mathcal{S})$ is also I-learnable, i.e., the class of I-learnable sets of functions is closed under operators from $\mathfrak{O}$.[5]

Furthermore, for any set of operators $\mathfrak{O}$, we define a reducibility $\leq^{\mathfrak{O}}$ and a corresponding completeness notion. As an important first theorem we have that a set $\mathcal{S}$ epitomizes a learning criterion I with respect to all $\mathfrak{O}$-robust learning criteria iff $\mathcal{S}$ is $\leq^{\mathfrak{O}}$-complete for all I-learnable sets (Theorem 3.6 below)! The benefits of this theorem are twofold.

First, since, as noted above, we believe that epitomization captures the complexity of learning, this Theorem 3.6 entails that our reducibility notions also capture this complexity.

Secondly, we now need only to prove completeness to get epitomization!

Other than structural insight, we get the following two benefits from epitomizers.

First, we can use epitomizers to show the identity of learning power of two learning criteria. For example, Theorem 5.2 establishes $\mathcal{S}_{\text{FinSup}}$ to be epitomizing with respect to various learning criteria and sets of operators. In Corollary 5.3 below we use this to show a learning criterion I to be as powerful as one of the epitomized learning criteria by showing that $\mathcal{S}_{\text{FinSup}}$ can be I-learned. With classic methods, the proof of this result might have required tedious work with attention to detail, while we can conclude it as a corollary to *structural properties* uncovered by our theorems.

The second way in which epitomization helps us is by providing canonical candidates to witness the separation of two learning criteria. To this end, *self-learning* classes (see Theorem 5.7) are particularly useful epitomizers and are used in the literature to prove particularly difficult separations (see, for example, [9], which solves two previously open problems using this technique, and see also [10]).

[12] noted that their $\leq^{\text{Ex}}$-completeness does not give epitomization with respect even to the set of learning criteria considered in [12].

Thus, we believe our approach to complexity of learning is both more *comprehensive* and more *useful* than the notion of intrinsic complexity from [12].

We present mathematical preliminaries in Section 2. The notions discussed above and some first theorems about them are given in Section 3, including the mentioned important characterization of epitomizers as complete sets (Theorem 3.6 below).

Section 4 gives definitions and results regarding the notion of intrinsic complexity introduced in [12]. We already mentioned our Theorems 4.2 and 4.3 below which witness drawbacks of this older notion; furthermore, in Theorem 4.5 below, we characterize $\leq^{\text{Ex}}$ in terms of one of our reducibility notions, and conclude in Corollary 4.6 that all sets complete with respect to a central one of our reducibility notions are $\leq^{\text{Ex}}$-complete.

Finally, in Section 5, we present a series of tasks and state which learning criteria they epitomize *at what strength*. N.B. Epitomizers with respect to larger classes of learning criteria are stronger. As indicated above, we give each epitomization result, for some set of operators $\mathfrak{O}$, and with respect to the corresponding set of $\mathfrak{O}$-robust learning criteria. N.B. It will be seen that the smaller the set of operators $\mathfrak{O}$, the larger the set of $\mathfrak{O}$-robust learning criteria. Theorem 5.2 entails that $\mathcal{S}_{\text{FinSup}}$, the set of functions of finite support introduced above, is a very weak epitomizer. Some so called self-*describing* classes are also surprisingly weak epitomizers (Theorem 5.4 below); some are of considerably greater strength (Theorem 5.5

[5] In the previous literature, a set $\mathcal{S}$ was called *robustly I-learnable* iff, for *all recursive operators* [22] Θ, the class of total functions in $\Theta(\mathcal{S})$ is I-learnable (see, for example, [15]). The motivation for such past notions of robustness was to eliminate self-referential examples. The motivation herein is quite different. Herein, as will be seen, it is very interesting that which operators are to be considered can be restricted, and ask for *all* I-learnable tasks to be robust with respect to some possibly restricted class of operators.

below); but, interestingly, the strongest are self-*learning* sets (Theorem 5.7 below).

Some of our proofs involve subtle infinitary program self-reference arguments employing (variants of) the Operator Recursion Theorem (ORT) from [5, 6, 19]. Note that, due to space constraints, all proofs are omitted from the paper.

We are working on extending the present paper to employ self-learning sets which work for learning criteria, unlike those herein, which require the learners to be total on *all* inputs.

2 Mathematical Preliminaries

Unintroduced computability-theoretic notions follow [22].

$\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$.

The symbols $\subseteq, \subset, \supseteq, \supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets. The symbol $\setminus$ denotes set-difference.

The quantifier $\forall^\infty x$ means "for all but finitely many $x \in \mathbb{N}$"; the quantifier $\exists^\infty x$ means "for infinitely many $x \in \mathbb{N}$". For any set A, $\mathrm{card}(A)$ denotes its cardinality, $\mathrm{Pow}(A)$ denotes the set of all subsets of A.

With $\mathfrak{P}$ and $\mathfrak{R}$ we denote, respectively, the set of all partial and of all total functions $\mathbb{N} \to \mathbb{N}$. With dom and range we denote, respectively, domain and range of a given function. Set-theoretically, (partial) functions are identified with their graphs, i.e., they are treated as sets of ordered pairs, and we sometimes compare them by $\subseteq$.

We sometimes denote a partial function f of $n > 0$ arguments $x_1, \ldots, x_n$ in lambda notation (as in Lisp) as $\lambda x_1, \ldots, x_n . f(x_1, \ldots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x . c$ is the constantly c function of one argument.

If $f \in \mathfrak{P}$ is not defined for some argument x, then we denote this fact by $f(x)\uparrow$, and we say that f on x *diverges*; the opposite is denoted by $f(x)\downarrow$, and we say that f on x *converges*. If f on x converges to p, then we denote this fact by $f(x)\downarrow = p$.

We say that $f \in \mathfrak{P}$ *converges to* p iff $\forall^\infty x : f(x)\downarrow = p$; we write $f \to p$ to denote this.[6]

For any (possibly partial) predicate P, we let $\mu x . P(x)$ denote the least x such that $P(x)$ and, for all $y < x$, $P(x)\downarrow$ (if no such x exists, $\mu x . P(x)$ is undefined).

We fix any computable 1-1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[7] With π_1 and π_2, respectively, we denote decoding into first and second arguments of pairing, respectively. Whenever we consider tuples of natural numbers as input to $f \in \mathfrak{P}$, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to (left-associatively) code the tuples into a single natural number (but we will not necessarily state the pairing explicitly).

For any $g \in \mathfrak{P}$ and $x \in \mathbb{N}$, we let $g[x]$ denote the sequence of the numbers $g(0), \ldots, g(x-1)$, if all are defined, and $\uparrow$ otherwise.

A partial function $f \in \mathfrak{P}$ is *partial computable* iff there is a deterministic, multi-tape Turing machine which, on input x, returns $f(x)$ if $f(x)\downarrow$, and loops infinitely if $f(x)\uparrow$. $\mathcal{P}$ and $\mathcal{R}$ denote, respectively, the set of all partial computable and the set of all (total) computable functions $\mathbb{N} \to \mathbb{N}$. The functions in $\mathcal{R}$ are called *computable functions*.

We let φ be any fixed acceptable programming system for the partial computable functions $\mathbb{N} \to \mathbb{N}$ with associated complexity measure Φ [19]. Further, we let φ_p denote the partial computable function computed by the φ-program with code number p, and we let Φ_p denote the partial computable *complexity* function of the φ-program with code number p.

[6] $f(x)$ converges should not be confused with f converges *to*.
[7] For a linear-time computable and invertible example, see [23, Section 2.3].

Whenever we consider sequences or finite sets as input to functions, we assume these objects to be appropriately coded as natural numbers. Similarly, when functions are defined to give non-numeric output, for example, when the outputs are in $\mathbb{N} \cup \{?\}$, we implicitly assume $\mathbb{N} \cup \{?\}$ to be appropriately coded onto the natural numbers.

We use complexity theoretic notions as introduced in [23]. We let **LinF** be the set of all linear time computable functions. A function g is called *linlin* iff g is computable in linear time and there is a linear time computable function g^{-1} such that $g^{-1} \circ g = \lambda x.x$. We let **LL** be the set of all linlin functions.

Learning in the Limit

A *learner* is a partial computable function. A *target* is a total computable function g; a *learning task* is a set of targets $\mathcal{S} \subseteq \mathcal{R}$.

A learning criterion consists of three parts which, together, determine whether a given learner is successful on a given learning task.

Firstly, the learning criterion has to specify what learners are allowed. This is called a *learner admissibility restriction*, and is modeled as a set $\mathcal{C} \subseteq \mathcal{P}$, the set of all admissible learners.

Secondly, the learning criterion has to specify how learner and target interact. This part is modeled as a *sequence generating operator*, which is an operator β taking as arguments a learner h and a target g and that outputs a function p. We call p the *learning sequence* of h given g. For this paper, we think of p as the sequence of conjectured programs of h on g.

Thirdly, the learning criterion has to specify which learning sequences are to be considered "successful" on a given target. This is done with a *sequence acceptance criterion*, a total binary predicate δ on a learning sequence and a target function.[8]

For $\mathcal{C}$ a learner admissibility restriction, β a sequence generating operator, δ a sequence acceptance criterion and h a learner, we call $(\mathcal{C}, \beta, \delta)$ a learning criterion. For every learning criterion I with $I = (\mathcal{C}, \beta, \delta)$ we let $\mathcal{C}_I = \mathcal{C}$, $\beta_I = \beta$ and $\delta_I = \delta$. Let $I = (\mathcal{C}, \beta, \delta)$ be a learning criterion. We proceed by giving definitions for I-learning.

We say that h *I-learns a learning task* $\mathcal{S}$ iff, $h \in \mathcal{C}$ and, for all $g \in \mathcal{S}$, with $p = \beta(h, g)$, $(p, g) \in \delta$. We denote by $\mathfrak{S}(I)$ and also by $\mathcal{C}\beta\delta$ the set of all I-learnable learning tasks.[9] With an abuse of notation, we sometimes also use $\mathcal{C}\beta\delta$ to denote I.

Any set of complexity-bounded functions is an example learner admissibility restriction, as are $\mathcal{R}$ and $\mathcal{P}$. We omit mentioning $\mathcal{C}$, if $\mathcal{C} = \mathcal{P}$ (no restriction on the learner).

We give the following three examples for sequence generating operators. Let **G** be defined thus.[10]

$$\forall h, g, i : \mathbf{G}(h, g)(i) = h(g[i]). \tag{1}$$

Let **It** be defined thus.[11]

$$\forall h, g : \mathbf{It}(h, g)(0) =? \ \wedge \ \forall i : \mathbf{It}(h, g)(i+1) = h(\mathbf{It}(h, g)(i), \langle i, g(i)\rangle). \tag{2}$$

Lastly, we give *transductive* learning [7].

$$\forall h, g : \mathbf{Td}(h, g)(0) =? \ \wedge \ \forall i : \mathbf{Td}(h, g)(i+1) = h(\langle i, g(i)\rangle). \tag{3}$$

---

[8] Herein, our βs and δs can essentially be modeled as multi-argument variants of Rogers' [22] *recursive operators*.

[9] Note that "$\mathcal{C}\beta\delta$" is the classical way to denote this, while we use "$\mathfrak{S}(I)$" whenever $\mathcal{C}$, β and δ are not explicit.

[10] **G** stands for Gold identification [16].

[11] **It** stands for iterative [14, 24].

For sequence acceptance criteria, we give the following five examples. We define *explanatory* learning [16] as follows.

$$\mathbf{Ex} = \{(p, g) \mid p \text{ total and } \exists e : p \to e \text{ and } \varphi_e = g\}.$$

Finite learning is given by

$$\mathbf{Fin} = \{(p, g) \mid p \text{ total and } \exists e \in \mathbb{N} : e \in \text{range}(p) \subseteq \{e, ?\} \text{ and } \varphi_e = g\}.$$

Further, we define a sequence acceptance criterion corresponding to *postdictively complete* learning[12] [2, 4, 24].

$$\mathbf{Pcp} = \{(p, g) \mid p \text{ total and } \forall i : g[i] \subseteq \varphi_{p(i+1)}\}.$$

For *conservative* learning [1] we give the following sequence acceptance criterion.

$$\mathbf{Conv} = \{(p, g) \mid p \text{ total and } \forall i : p(i+1) \neq p(i) \Rightarrow g[i] \not\subseteq \varphi_{p(i)}\}.$$

Finally, behaviorally correct learning as given by [3, 11] is associated with

$$\mathbf{Bc} = \{(p, g) \mid p \text{ total and } \forall^\infty i : \varphi_{p(i)} = g\}.$$

Any two sequence acceptance criteria δ and δ' can be combined by intersecting them. For ease of notation we write $\delta\delta'$ instead of $\delta \cap \delta'$.

For more examples of the above concepts, see [20].

3 Concept Definitions

In this section, we give the key concepts used in this paper in Definition 3.3. Before we can get to that, we define pre-orders and associated notions, as well as several sets of operators, for which we give examples and remark on some easy properties.

The main theorem of this section is Theorem 3.6, which shows the important connections between complete sets and epitomizers.

Let S be a set and $\preceq$ a binary relation on that set. We call $\preceq$ a *pre-order* iff $\preceq$ is reflexive and transitive. For $s \in S$ and $T \subseteq S$, we say that s is $\preceq$*-complete for* T iff, $s \in T$ and, for all $t \in T$, $t \preceq s$.

Next we define several sets of operators. For illustration, see Example 3.2.

▶ **Definition 3.1.** A function $\Theta : \mathcal{P} \to \mathcal{P}$ is called an *operator*. We define the following sets of operators.

- Let $\mathfrak{O}_{\text{eff}}$ be the set of all effective operators [22], i.e., all operators Θ such that there is a computable function $s \in \mathcal{R}$ with $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$.[13]
- Let $\mathcal{C} \subseteq \mathcal{P}$. Let $\mathfrak{O}^{\mathcal{C}}_{\text{loc}}$ be the set of all $\Theta \in \mathfrak{O}_{\text{eff}}$ such that, for all g, x, $\Theta(g)[x]$ depends only on $g[x]$, i.e., if there is a function f with $\forall g \in \mathcal{P}, \forall x : \Theta(g)[x] = f(g[x])$, and $f \in \mathcal{C}$. Note that $\Theta \in \mathfrak{O}_{\text{loc}}$ is equivalent to

$$\forall g, g' \in \mathcal{P} \forall x : g[x] = g'[x] \Rightarrow \Theta(g)[x] = \Theta(g')[x].^{14} \tag{4}$$

[12] Also called *consistent* learning.
[13] Note that, without loss of generality, we can take s to be linlin.

- Let $\mathcal{C} \subseteq \mathcal{P}$. Let $\mathfrak{O}^{\mathcal{C}}_{\text{elemWise}}$ be the set of all $\Theta \in \mathfrak{O}_{\text{eff}}$ such that there is a function $f \in \mathcal{C}$ such that $\forall g \in \mathcal{P} : \Theta(g) = \lambda x . f(g(x), x)$.[15] We write $\mathfrak{O}_{\text{elemWise}}$ for $\mathfrak{O}^{\mathcal{P}}_{\text{elemWise}}$.

Next, we define sets of *left-invertible* operators. For any set of operators $\mathfrak{O}$ and $\mathcal{S} \subseteq \mathcal{R}$, we let $\text{LInv}(\mathfrak{O}; \mathcal{S}) = \{\Theta \in \mathfrak{O} \mid \Theta(\mathcal{S}) \subseteq \mathcal{R} \wedge \exists \hat{\Theta} \in \mathfrak{O}\ \forall g \in \mathcal{S} : (\hat{\Theta} \circ \Theta)(g) = g\}$.

Clearly, $\mathfrak{O}_{\text{elemWise}} \subset \mathfrak{O}_{\text{loc}} \subset \mathfrak{O}_{\text{eff}}$.

▶ **Example 3.2.** For illustration, we give the following example operators.

- The operator Θ such that $\forall g \in \mathcal{P} \forall x : \Theta(g)(x) = g(x+1)$ is in $\mathfrak{O}_{\text{eff}}$, but not in $\mathfrak{O}_{\text{loc}}$ or $\mathfrak{O}_{\text{elemWise}}$; furthermore, Θ is not in $\text{LInv}(\mathfrak{O}_{\text{eff}})$ (Θ is not 1-1, hence cannot be left-inverted).
- The operator Θ such that

$$\forall g \in \mathcal{P} \forall x : \Theta(g)(x) = \begin{cases} 0, & \text{if } x = 0; \\ g(x-1), & \text{otherwise,} \end{cases} \tag{5}$$

  is in $\mathfrak{O}_{\text{eff}}$ and in $\mathfrak{O}_{\text{loc}}$, but not in $\mathfrak{O}_{\text{elemWise}}$; furthermore, Θ is in $\text{LInv}(\mathfrak{O}_{\text{eff}})$, but not in $\text{LInv}(\mathfrak{O}_{\text{loc}})$ (Θ has a computable left-inverse, but not a local one).
- The operator Θ such that $\forall g \in \mathcal{P} \forall x : \Theta(g)(x) = x + g(x)$ is in $\mathfrak{O}_{\text{eff}}$, $\mathfrak{O}_{\text{loc}}$ and also $\mathfrak{O}_{\text{elemWise}}$; furthermore, Θ is even in $\text{LInv}(\mathfrak{O}_{\text{elemWise}})$.

Now we give the definition of the central notions of this paper.

▶ **Definition 3.3.** Let $\mathfrak{O}$ be a set of operators and I a learning criterion. Let $\mathcal{S}, \mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{R}$.

(i) I is called $\mathfrak{O}$-*robust* iff, for all $\mathcal{S} \subseteq \mathcal{R}$ and $\Theta \in \text{LInv}(\mathfrak{O}; \mathcal{S})$,

$$\mathcal{S} \in \mathfrak{S}(I) \Rightarrow \Theta(\mathcal{S}) \in \mathfrak{S}(I).^{16} \tag{6}$$

(ii) We say that $\mathcal{S}$ *epitomizes* I with respect to a set of learning criteria $\mathcal{I}$, iff $\mathcal{S} \in \mathfrak{S}(I)$ and

$$\forall I' \in \mathcal{I} : [\mathcal{S} \in \mathfrak{S}(I') \Leftrightarrow \mathfrak{S}(I) \subseteq \mathfrak{S}(I')]\,.^{17} \tag{7}$$

(iii) If $\mathcal{S}$ epitomizes I with respect to the set of all $\mathfrak{O}$-robust learning criteria, then we say $\mathcal{S}$ $\mathfrak{O}$-*epitomizes* I.

(iv) We say that $\mathcal{S}$ $\mathfrak{O}$-*generates* I iff $\{\Theta(\mathcal{S}') \mid \mathcal{S}' \subseteq \mathcal{S}, \Theta \in \text{LInv}(\mathfrak{O}; \mathcal{S}')\}$ is the set of all I-learnable functions.

(v) $\mathcal{S}_0 \leq^{\mathfrak{O}} \mathcal{S}_1$ iff there is an operator $\Theta \in \text{LInv}(\mathfrak{O}; \mathcal{S}_0)$ such that $\Theta(\mathcal{S}_0) \subseteq \mathcal{S}_1$.

As an interesting first observation on epitomizers, we make the following remark regarding separations from unions of learning criteria.

▶ **Proposition 3.4.** Let I be a learning criterion and $\mathcal{I}$ a set of learning criteria with $\forall I' \in \mathcal{I} : \mathfrak{S}(I) \setminus \mathfrak{S}(I') \neq \emptyset$. Suppose there is a set epitomizing I with respect to $\mathcal{I}$. Then

$$\mathfrak{S}(I) \setminus \bigcup_{I' \in \mathcal{I}} \mathfrak{S}(I') \neq \emptyset.$$

[15] We call these operators *element wise*.

Theorem 5.7 provides very strong existence results for epitomizers which can be used to satisfy the corresponding hypothesis of Proposition 3.4.[18]

We can use Proposition 3.4 to give cases where epitomizers do not exist: For example, let I be reliable learnability and let $\mathcal{I}$ be the set of all learning criteria of delayed postdictive completeness; then $\forall I' \in \mathcal{I} : \mathfrak{S}(I) \setminus \mathfrak{S}(I') \neq \emptyset$ and $\mathfrak{S}(I) = \bigcup_{I' \in \mathcal{I}} \mathfrak{S}(I')$, which shows that there is no epitomizer of I with respect to $\mathcal{I}$ (see [8] for the definitions; the result will be included in an extension of [8]).[19]

The following theorem gives a number of general observations regarding the concepts introduced in Definition 3.3.

▶ **Proposition 3.5.** Let $\mathfrak{O}, \mathfrak{O}'$ be sets of operators, I a learning criterion.

(i) Let I be $\mathfrak{O}$-robust. Then, for all $\mathcal{S} \subseteq \mathcal{R}$, $\Theta \in \mathrm{LInv}(\mathfrak{O}; \mathcal{S})$ with $\Theta(\mathcal{S}) \subseteq \mathcal{R}$,

$$\mathcal{S} \in \mathfrak{S}(I) \Leftrightarrow \Theta(\mathcal{S}) \in \mathfrak{S}(I).$$

(ii) If $\mathfrak{O} \in \{\mathfrak{O}_{\mathrm{eff}}, \mathfrak{O}_{\mathrm{loc}}, \mathfrak{O}_{\mathrm{elemWise}}\}$, then $\leq^{\mathfrak{O}}$ is a pre-order.

(iii) Suppose I is $\mathfrak{O}$-robust. Suppose $\mathcal{S} \in \mathfrak{S}(I)$. Then, for all $\mathcal{S}' \subseteq \mathcal{R}$,

$$\mathcal{S}' \leq^{\mathfrak{O}} \mathcal{S} \Rightarrow \mathcal{S}' \in \mathfrak{S}(I).$$

Next is the central theorem of this section, showing that, for important sets of operators $\mathfrak{O}$ and certain learning criteria I, $\leq^{\mathfrak{O}}$-completeness for I *characterizes* $\mathfrak{O}$-epitomization of I.

▶ **Theorem 3.6.** Let $\mathfrak{O}$ be a set of operators containing the identity and which is closed under composition. Let I be a $\mathfrak{O}$-robust learning criterion. Let $\mathcal{S} \subseteq \mathcal{R}$. The following are equivalent.

(i) $\mathcal{S}$ $\mathfrak{O}$-epitomizes I.

(ii) $\mathcal{S}$ $\mathfrak{O}$-generates I.

(iii) $\mathcal{S}$ is $\leq^{\mathfrak{O}}$-complete for I.

4 Connection to Intrinsic Complexity

We will now give the definitions of intrinsic complexity. Some version thereof was introduced in [12], here we give an interpretation that fits our formalism. After that we will give theorems regarding the shortcomings of intrinsic complexity, as discussed in the introduction.

In particular, Theorem 4.2 gives an example $\leq^{\mathbf{GEx}}$-complete set of functions which is nonetheless learnable in much more restricted criteria. Then, in Theorem 4.3, we give two natural learning criteria I for which the learnable sets are not downward closed with respect to $\leq^I$. For the two criteria, the cause of this failure of closure is different: in one case it is a local restriction on the conjectures, in the other a memory restriction on the learner.

Finally, we show the equivalence of $\leq^{\mathbf{GEx}}$ with one of our reducibility notions in Theorem 4.5.

[18] We give the following example for illustration. Let, for all $a \in \mathbb{N}$, $\mathbf{Ex}^a$ be like $\mathbf{Ex}$, but, for success, the final program is allowed to be incorrect on up to a places. Further, let $\mathbf{Ex}^*$ be like $\mathbf{Ex}$, but the final program is allowed to be incorrect on up to finitely many places. From [11] we know that, for all $a \in \mathbb{N}$, $\mathbf{GEx}^a \subset \mathbf{GEx}^{a+1}$; hence, $\mathbf{GEx}^a \subset \mathbf{GEx}^*$. Theorem 5.7 provides an appropriate epitomizer for $\mathbf{GEx}^*$, so that we can deduce, with Proposition 3.4, $\bigcup_{a \in \mathbb{N}} \mathbf{GEx}^a \subset \mathbf{GEx}^*$ (which was shown in [11]).

[19] We are thankful to an anonymous referee who pointed out a (different) example for the nonexistence of epitomizers.

▶ **Definition 4.1.** Let $I = (\mathcal{C}, \beta, \delta)$ be a learning criterion, and g a function. We say that a sequence p is *I-admissible* for g iff $(p, g) \in \delta$.[20]

Let $\mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{R}$, and an identification criterion I be given. We say that $\mathcal{S}_0 \leq^I \mathcal{S}_1$ iff there exist recursive operators Θ and Ψ such that, for any function $g \in \mathcal{S}_0$,

(i) $\Theta(g) \in \mathcal{S}_1$, and
(ii) for any I-admissible sequence p for $\Theta(g)$, $\Psi(p)$ is an I-admissible sequence for g.

We say that a set $\mathcal{S} \subseteq \mathcal{R}$ is $\leq^I$-complete, iff $\mathcal{S}$ is $\leq^I$-complete for $\mathfrak{S}(I)$.

▶ **Theorem 4.2** ([12] & [3, 4])**.** $\mathcal{S}_{\text{FinSup}}$ is $\leq^{\mathbf{GEx}}$-complete, but $\mathcal{S}_{\text{FinSup}} \in \mathbf{GPcpEx} \subset \mathbf{GEx}$.

The following theorem shows two criteria classes to be not closed downwards with respect to their respective intrinsic reducibility notions.

▶ **Theorem 4.3.**

(i) There are sets $\mathcal{S}_0$ and $\mathcal{S}_1 \subseteq \mathcal{R}$ such that $\mathcal{S}_0 \leq^{\mathbf{ItEx}} \mathcal{S}_1$ and $\mathcal{S}_1 \in \mathbf{ItEx}$, but $\mathcal{S}_0 \notin \mathbf{ItEx}$.
(ii) There are sets $\mathcal{S}_0$ and $\mathcal{S}_1 \subseteq \mathcal{R}$ such that $\mathcal{S}_0 \leq^{\mathbf{GPcpEx}} \mathcal{S}_1$ and $\mathcal{S}_1 \in \mathbf{GPcpEx}$, but $\mathcal{S}_0 \notin \mathbf{GPcpEx}$.

In order to characterize the reducibility of intrinsic complexity in terms of our notions, we give the following definitions, extending notions from Section 3.

▶ **Definition 4.4.** Let $\mathfrak{O}, \mathfrak{O}'$ be sets of operators and $\mathcal{S} \subseteq \mathcal{R}$.

(i) Let $\text{LInv}(\mathfrak{O}, \mathfrak{O}'; \mathcal{S}) = \{\Theta \in \mathfrak{O} \mid \exists \hat{\Theta} \in \mathfrak{O}' \, \forall g \in \mathcal{S} : (\hat{\Theta} \circ \Theta)(g) = g\}$.
(ii) For two sets $\mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{R}$, we write $\mathcal{S}_0 \leq^{(\mathfrak{O}, \mathfrak{O}')} \mathcal{S}_1$ iff there is an operator $\Theta \in \text{LInv}(\mathfrak{O}, \mathfrak{O}'; \mathcal{S}_0)$ such that $\Theta(\mathcal{S}_0) \subseteq \mathcal{S}_1$.
(iii) Furthermore, let $\mathfrak{O}_{\text{limPEff}}$ be the set of all partial operators Θ such that there is a computable function $s \in \mathcal{R}$ with

$$\forall e : \Theta(\varphi_e) = \begin{cases} \varphi_p, & \text{if } \lambda t . s(e,t) \text{ converges to some } p; \\ \uparrow, & \text{otherwise.}^{21} \end{cases} \tag{8}$$

Now we show the equivalence of one particular reducibility notion of intrinsic complexity with one of our extended variants from Definition 4.4. Note that a similar characterization can be made for **GBc**.

▶ **Theorem 4.5.** We have

$$\leq^{(\mathfrak{O}_{\text{eff}}, \mathfrak{O}_{\text{limPEff}})} \quad \text{equals} \quad \leq^{\mathbf{GEx}}.$$

We get the following Corollary from Theorem 4.5.

▶ **Corollary 4.6.** Let I be a learning criterion with $\delta_I = \mathbf{Ex}$ as sequence acceptance criterion. We have that $\leq^{\mathfrak{O}_{\text{eff}}}$ is a subrelation of $\leq^{\mathbf{GEx}}$; in particular, for all $\mathcal{S} \subseteq \mathcal{R}$

$$\mathcal{S} \text{ is } \leq^{\mathfrak{O}_{\text{eff}}}\text{-complete for } \mathfrak{S}(I) \Rightarrow \mathcal{S} \text{ is } \leq^I\text{-complete.}$$

[20] I-admissibility is not to be confused with learner admissibility restrictions.
[21] Note that the operators from $\mathfrak{O}_{\text{limPEff}}$ resemble those from [21] and [13].

5 Specific Function Learning Epitomizers

In this section we present several epitomizers, starting with the very natural class $\mathcal{S}_{\text{FinSup}}$ in Theorem 5.2. After noting an immediate corollary, we show a self-*describing* class to be an epitomizer. Finally, we give a construction for self-*learning* classes and show them to be very powerful epitomizers in Theorem 5.7, i.e., epitomizing with respect to a very wide range of learning criteria, much wider than for self-descibing classes (which is in turn wider than for $\mathcal{S}_{\text{FinSup}}$).

Note that all proofs rely implicitly on Theorem 3.6, as they show completeness and derive epitomization from that.

But first, we give a table of examples of which learning criteria are robust with respect to which set of operators.

▶ **Example 5.1.** Let $\mathcal{C} \subseteq \mathcal{P}$ contain all linlin functions. Let $\mathcal{F} \subseteq \mathcal{P}$ be closed under $\mathcal{C}$-composition and contain all linear time computable functions.

The following table states several kinds of robustness of learning criteria with respect to certain sets of operators.

$\mathfrak{O}_{\text{eff}}$	$\mathcal{F}$**GEx**, $\mathcal{F}$**GBc**, $\mathcal{F}$**GFin**
$\mathfrak{O}^{\mathcal{C}}_{\text{loc}}$	$\mathcal{F}$**GPcpEx**, $\mathcal{F}$**GConvEx**, $\mathcal{F}$**GPcpConvEx**
$\mathfrak{O}^{\mathcal{C}}_{\text{elemWise}}$	$\mathcal{F}$**ItEx**, $\mathcal{F}$**ItConvEx**, $\mathcal{F}$**ItPcpConvEx**, $\mathcal{F}$**TdEx**, $\mathcal{F}$**TdBc**

Note that each criterion in any given row just above could also be listed in any lower row.

Next, we show $\mathcal{S}_{\text{FinSup}}$ to epitomize various *particular* learning criteria with respect to various *sets* of learning criteria.

▶ **Theorem 5.2.** We have

(i) $\mathcal{S}_{\text{FinSup}}$ $\mathfrak{O}_{\text{eff}}$-epitomizes **GEx**;
(ii) $\mathcal{S}_{\text{FinSup}}$ does *not* $\mathfrak{O}_{\text{loc}}$-epitomize **GEx**;
(iii) $\mathcal{S}_{\text{FinSup}}$ $\mathfrak{O}_{\text{loc}}$-epitomizes **GPcpEx**.
(iv) $\mathcal{S}_{\text{FinSup}}$ does *not* $\mathfrak{O}_{\text{elemWise}}$-epitomize **GPcpEx**;

From Theorem 5.2 we can directly get the following corollary, showing an identity of learning criteria power. The very short proof shows the power of our notions of epitomizers and robustness.

▶ **Corollary 5.3.** We have

GPcpEx = **GPcpConvEx**.

Next we analyze a set of self-describing functions which was first given in [11] and used a lot in [19]. Theorem 5.4 shows that this set is not a very good epitomizer.

▶ **Theorem 5.4.** Let $\mathcal{S}_0 = \{g \in \mathcal{R} \mid \varphi_{g(0)} = g\}$. Then $\mathcal{S}_0$ $\mathfrak{O}_{\text{eff}}$-epitomizes **GFin**. However, $\mathcal{S}_0$ does not $\mathfrak{O}_{\text{loc}}$-epitomize any learning criterion that can be built from components given in this paper as $\mathcal{S}_0 \equiv^{\mathfrak{O}_{\text{loc}}} \{g \in \mathcal{R} \mid \forall x > 0 : g(x) = 0\}$.

Next we analyze a set of self-describing functions which was used in [11] to show the separation of **GBc** and (a stronger version of) **GEx**. Theorem 5.5 shows that this set necessarily shows the separation of **GEx** and **GBc**, if any set does.

▶ **Theorem 5.5.** Let $\mathcal{S}_0 = \{g \in \mathcal{R} \mid \forall^\infty x : \varphi_{g(x)} = g\}$. Then $\mathcal{S}_0$ $\mathfrak{O}^{\mathbf{LL}}_{\mathrm{loc}}$-epitomizes **GBc**. However, $\mathcal{S}_0$ does not $\mathfrak{O}_{\mathrm{elemWise}}$-epitomize **GBc**, as $\mathcal{S}_{\mathrm{FinSup}} \not\leq^{\mathfrak{O}_{\mathrm{elemWise}}} \mathcal{S}_0$.[22]

Theorem 5.7 gives examples for *self-learning* classes of functions, which turn out to be very strong epitomizers. In order to define these classes, we need the following notions of *computable robustness* and *data normality*.

▶ **Definition 5.6.** Let I be a learning criterion.

We call I *computably element-wise robust* iff I is element-wise robust in a constructive way, i.e., there is an effective operator $\Psi \in \mathfrak{O}_{\mathrm{eff}}$ such that, for all $h, e \in \mathcal{P}$, $e \circ I(h) \subseteq I(\Psi(h, e))$.

We call I *data normal* iff (a) – (c) below.

(a) There is $f_I \in \mathcal{R}$ such that

$$\forall h \in \mathcal{P} \forall g \in \mathcal{R} \forall i : \beta_I(h, g)(i) = h(f_I(g[i], \beta_I(h, g)[i])).[23] \tag{9}$$

(b) There is a function $d_I \in \mathcal{R}$ such that

$$\forall h \in \mathcal{P} \forall g \in \mathcal{R} \forall i : d_I(f_I(g[i], \beta_I(h, g)[i])) = \begin{cases} ?, & \text{if } i = 0; \\ g(i-1), & \text{otherwise.}[24] \end{cases} \tag{10}$$

(c) There is an effective operator $\hat{\Psi} \in \mathfrak{O}_{\mathrm{eff}}$ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I(\hat{\Psi}(h))$ and $\hat{\Psi}(h)(f_I(\emptyset, \emptyset)) =?$.[25]

▶ **Theorem 5.7.** Let I be a computably element-wise robust learning criterion with $\mathcal{C}_I = \mathcal{P}$ (i.e., I does not impose global restrictions on the learner). Suppose I is data normal as witnessed by f and d. Let h_0 be such that

$$\forall x : h_0(x) = \begin{cases} ?, & \text{if } d(x) =?; \\ \varphi_{d(x)}(x), & \text{otherwise.} \end{cases} \tag{11}$$

Further, let $\mathcal{S}_0 = I(h_0)$.[26] Then $\mathcal{S}_0$ $\mathfrak{O}^{\mathbf{LL}}_{\mathrm{elemWise}}$-epitomizes I.

Theorem 5.7 provides epitomizing sets for the learning criteria $\beta\delta$ with β and δ as explicitly given in this paper, and many more. Furthermore, $\mathcal{S}_0$ of Theorem 5.7 epitomizes with respect to all learning criteria that can be built from the example components given in this paper (including the ones with learner admissibility restrictions), as they are all $\mathfrak{O}^{\mathbf{LL}}_{\mathrm{elemWise}}$-robust. In particular, Theorem 5.7 provides a superior epitomizer for **GBc** than the epitomizer of Theorem 5.5.[27]

---

[22] This uses Theorem 3.6.

[24] Intuitively, in the ith round the learner has access to the $(i-1)$st data item.

[25] Intuitively, without loss of generality, the output based on no data of a learner equals ?.

[26] $\mathcal{S}_0$ is a *self-learning class*. Roughly, the learner (h_0 in Theorem 5.7) defining such a set ($\mathcal{S}_0$ in Theorem 5.7) just *runs* each input datum as coding a program to determine its corresponding conjecture to output [9, 10].
Note that h_0 is not total, hence, not polynomial time computable, etc. It is work in progress to extend self-learning classes for criteria separations to cases which cover associated learners interestingly more restricted than simply being partial computable, e.g., restricted to being linear time computable.

[27] The first author of the present paper, when he was co-creating [11], had the intuition that, for any criterion I, if $(\mathbf{GBc} \setminus \mathfrak{S}(I)) \neq \emptyset$, then the $\mathcal{S}_0$ of Theorem 5.5 above would witness that separation. Consider $I = \mathbf{TdBc}$. Clearly, $\mathcal{S}_{\mathrm{FinSup}}$ separates **GBc** from **TdBc**. However, the epitomizer $\mathcal{S}_0$ of Theorem 5.5 clearly *is* **TdBc**-learnable—disproving the present first author's old intuition. Nicely, though, from $\mathfrak{O}^{\mathbf{LL}}_{\mathrm{elemWise}}$-robustness of **TdBc** (Example 5.1), the epitomizer $\mathcal{S}_0$ of Theorem 5.7 *does* separate **GBc** from **TdBc**.

References

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

2 J. Bārzdiņš. Inductive inference of automata, functions and programs. In *Proceedings of the 20th International Congress of Mathematicians, Vancouver, Canada*, pages 455–560, 1974. English translation in, *AMS Translations*: Series 2 109 (1977), pp. 107-112.

3 J. Bārzdiņš. Two theorems on the limiting synthesis of functions. *In Theory of Algorithms and Programs, Latvian State University, Riga*, 210:82–88, 1974.

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

5 J. Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8:15–32, 1974.

6 J. Case. Infinitary self-reference in learning theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:3–16, 1994.

7 J. Case and T. Kötzing. Dynamic modeling in inductive inference. In *Proceedings of ALT*, pages 404–418, 2008.

8 J. Case and T. Kötzing. Dynamically delayed postdictive completeness and consistency in learning. In *Proceedings of ALT*, pages 389–403. Springer, 2008.

9 J. Case and T. Kötzing. Solutions to open questions for non-U-shaped learning with memory limitations. In *Proceedings of ALT*, pages 285–299. Springer, 2010.

10 J. Case and T. Kötzing. Strongly non-U-shaped learning results by general techniques. In *Proceedings of COLT*, 2010. http://www.colt2010.org/papers/011koetzing.pdf.

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

12 R. Freivalds, E. Kinber, and C. Smith. On the intrinsic complexity of learning. *Information and Computation*, 123(1):64–71, 1995.

13 R. Freivalds, E. Kinber, and R. Wiehagen. Connections between identifying functionals, standardizing operations, and computable numberings. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 30:145–164, 1984.

14 M. Fulk. *A Study of Inductive Inference Machines*. PhD thesis, SUNY at Buffalo, 1985.

15 M. Fulk. Robust separations in inductive inference. In *Proceedings of FOCS*, pages 405–410, St. Louis, Missouri 1990.

16 E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

17 S. Jain. The intrinsic complexity of learning: A survey. *Fundamenta Informaticae*, 57(1):17–37, 2003.

18 S. Jain, E. Kinber, C. Papazian, C. Smith, and R. Wiehagen. On the intrinsic complexity of learning recursive functions. *Information and Computation*, 184(1):45 – 70, 2003.

19 S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Mass., second edition, 1999.

20 T. Kötzing. *Abstraction and Complexity in Computational Learning in the Limit*. PhD thesis, University of Delaware, 2009. http://pqdtopen.proquest.com/#viewpdf?dispub=3373055.

21 M. Pour-El. A comparison of five "computable" operators. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:325–340, 1960.

22 H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967. Reprinted by MIT Press, Cambridge, Massachusetts, 1987.

23 J. Royer and J. Case. *Subrecursive Programming Systems: Complexity and Succinctness*. Research monogr. in *Progress in Theoretical Computer Science*. Birkhäuser Boston, 1994.

24 R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationverarbeitung und Kybernetik*, 12:93–99, 1976.