PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

*Phil. Trans. R. Soc. A* (2012) **370**, 3570–3596
doi:10.1098/rsta.2011.0320

# Computability-theoretic learning complexity

By John Case[1,*] and Timo Kötzing[2]

[1]*Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586, USA*
[2]*Max-Planck Institute for Informatics, 66123 Saarbrücken, Germany*

Initially discussed are some of Alan Turing's wonderfully profound and influential ideas about mind and mechanism—including regarding their connection to the main topic of the present study, which is within the field of computability-theoretic learning theory. Herein is investigated the part of this field concerned with the algorithmic, trial-and-error inference of eventually correct programs for functions from their data points. As to the main content of this study: in prior papers, beginning with the seminal work by Freivalds *et al.* in 1995, the notion of *intrinsic complexity* is used to analyse 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 that are learnable under a given learning criterion, but not under other criteria that 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 sets of computable 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 the so-called *self-learning* sets, previously applied by Case & Kötzing in 2010. These strong epitomizers can be *easily* generated and employed in a myriad of settings to witness *with certainty* the strict separation in learning power between the criteria so epitomized and other not as powerful criteria!

**Keywords: computability-theoretic learning; self-reference; epitomizing success criteria; reducibilities and completeness; minds and machines; inductive inference**

## 1. Introduction

The present study is within the field of computability-theoretic learning theory.[1]

We provide, as part of this introduction to the actual subject of the present study, rhetoric to connect some of Alan Turing's wonderfully profound and influential ideas about mind and mechanism to the primary content of the study.

*Author for correspondence (case@cis.udel.edu).

[1]Sometimes simply called inductive inference.

One contribution of 18 to a Theme Issue 'The foundations of computation, physics and mentality: the Turing legacy'.

 

Turing [1] gave the world, among other things, an analysis of algorithmic calculation by an (importantly) human clerk that led Turing to the first convincing mathematically rigorous and complete definition of what is computable by algorithm.[2] The importance of Turing's analysis to theoretical psychology was recognized, for example, by Myhill [3, p. 149], where he says:

> ... in the author's view, the theorems of Church and Gödel are psychological laws. Mr. E. H. Galanter of the Department of Psychology, University of Pennsylvania, described them in conversation with the author as 'the only known psychological laws comparable in exactitude with the laws of physics'.[3]

Of course, Turing himself initiated earlier the part of artificial intelligence that is concerned with computer simulations of human intelligence [4]. A little later, Turing carried his mechanization program over to a mathematical model of the very complex biological process of embryonic development [5]. This model has been influential in both developmental biology and chemistry.

Next we will segue into the topic of the present study—but with asides of relevance to treating mind as (algorithmic) mechanism (which, as noted above, Turing began).

In the present study, we analyse the problem of algorithmically learning a description (our descriptions will also be algorithmic) for an infinite sequence (a function from the natural numbers into the natural numbers) when presented with larger and larger initial segments of that sequence.[4] For example, a learner $h$ might be presented with more and more of the sequence $g = 0, 1, 4, 9, 16, \ldots$. After each new datum of $g$, $h$ may output a description of a function as its conjecture. For example, $h$ might output a computer program for the constantly 0 function after seeing the first element of this sequence $g$, a program for the sequence of all natural numbers after seeing the first two elements of $g$, and a program for the squaring function on the further elements from $g$.

In computability-theoretic learning theory, Gold, in his seminal paper [6], mathematically models human child language learning as algorithmic.[5] Case [7,8] posits that the universe and hence its component humans are algorithmic at least in (quantum mechanically) *expected* behaviour, and claims that, then, some theorems about algorithmic learnability admit of interpretations for cognitive science and for philosophy of science. Regarding the relevance for philosophy of science, Gold [6] hints at connections between algorithmic learnability and (scientific) inductive inference. Blum and Blum [9] clarified this connection

[2]There were previous attempts by Church to give such a definition, but it was Turing's analysis and definition that ultimately convinced Gödel as to the definitional correctness [2]. Nicely too Turing showed his definition was constructively equivalent to Church's. In computer science terms, this means that Turing's formalism and Church's are intercompilable.

[3]Myhill, at least in his later years, was not in favour of the idea that humans are algorithmic mechanisms.

[4]For us, the natural numbers are the non-negative integers.

[5]Gold's sequences to be learned represent the correct utterances of a language, and his successful conjectures for such are programs or grammars for the *set* of utterances, instead of, as above (and as in the present study), for the *sequence* itself.

with essentially the following example (where successful conjectures are for the sequence/function itself).

> Consider the physicist who looks for a law to explain a growing body of physical data. His data consist of a set of pairs $(x, y)$, where $x$ describes a particular experiment, e.g., a high-energy physics experiment, and $y$ describes the results obtained, e.g., the particles produced and their respective properties. The law he seeks is essentially an algorithm for computing the function $f(x) = y$.

As noted in [10,11], such an algorithm is a *predictive explanation*: if one has the good fortune to *have* such an algorithm, one can use it to predict the outcomes of the associated experiments. Finding programs for such functions (also known as sequences) is what one hopes to do in science—modelled as the search for predictive explanations. The present study does not itself involve direct applications to philosophy of science, whereas Case [7,8] surveys a number of them. Such applications essentially involve a slight extension of Turing's mechanization of mind program to apply to the (collections of) minds of human scientists (over historical time) attempting to provide predictive explanations for phenomena.

Many criteria for saying whether learner $h$ is *successful* on sequence/function $g$ have been proposed in the literature. Gold [6] gave a first, simple learning criterion, later called [10,11] *Ex-learning*,[6] where a learner is *successful* if and only if (iff) it eventually stops changing its conjectures, and its final conjecture is a correct description of the input sequence.[7]

Trivially, each *single*, describable sequence $g$ has a suitable constant function as an Ex-learner (this learner constantly outputs an algorithmic description for $g$). Thus, we are interested in knowing for which *sets of functions* $\mathcal{S}$ there is 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 (algorithmic) descriptions for the functions.[8] This framework is known as *function learning in the limit* and has been studied extensively, employing a wide range of learning criteria similar to Ex-learning [12].

Freivalds *et al.* [13] considered how to define the *learning complexity* of sets of learnable functions. They introduced the seminal notion of *intrinsic complexity* and defined, for learning criterion $I$, 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, Freivalds *et al.* [13] show 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 [13–15].

From [13], 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

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

[7] Later in the study, after providing more formal definitions of this and other learning criteria, for uniformity of terminology, we change the term *Ex-learning* to **GEx**-*learning*. This is to show that this criterion is 'Gold-like' as in [6].

[8] One could, for example, think of the programming system as one of Java, C, Turing machines, etc.

the most difficult to $I$-learn. Freivalds *et al.* [13] show that the set $\mathcal{S}_{\mathrm{FinSup}}$ of all computable functions of finite support[9] is $\leq^{\mathrm{Ex}}$-complete. These notions from [13] are structural analogues, 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 Freivalds *et al.* [13].

(a) For some learning criteria $I$, the relation $\leq^I$ is not very fine-grained. In particular, there are $\leq^I$-complete sets of functions that are also learnable with respect to much more restricted learning criteria (theorem 4.3).
(b) 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 (theorem 4.4).

In this study, we quantify the difficulty of learning a given set of functions in a *new* way. First, we consider the following concept, essentially from [13]. A set of functions $\mathcal{S}$ *epitomizes* a learning criterion $I$ *with respect to* some set 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.[10]

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 sets of learning criteria $\mathcal{I}$. We give epitomizers with respect to sets of all learning criteria that are *robust* with respect to certain sets of (computable) operators (operating on functions). Essentially, a learning criterion $I$ is *robust* with respect to a given set of operators $\mathfrak{C}$ iff, for each $I$-learnable task $\mathcal{S}$ and each operator $\Theta \in \mathfrak{C}$, $\Theta(\mathcal{S})$ is also $I$-learnable, i.e. the set of $I$-learnable sets of functions is closed under operators from $\mathfrak{C}$.[11]

Furthermore, for any set of operators $\mathfrak{C}$, we define a reducibility $\leq^{\mathfrak{C}}$ 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 $\mathfrak{C}$-robust learning criteria iff $\mathcal{S}$ is $\leq^{\mathfrak{C}}$-complete for all $I$-learnable sets (theorem 3.7)! The benefits of this theorem, which we call the fundamental epitomization theorem, are twofold.

First, since, as noted above, we believe that epitomization captures the complexity of learning, the fundamental epitomization theorem 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.

[9]A (total) function has *finite support* iff only finitely many arguments have a function value other than 0.

[10]Note that Freivalds *et al.* [13] called epitomizing sets *characteristic*. To the best of our knowledge, neither this term nor the concept caught on in the later literature—until now.

[11]In the previous literature, a set $\mathcal{S}$ was called *robustly $I$-learnable* iff, for *all recursive operators* [16] $\Theta$, the set of total functions in $\Theta(\mathcal{S})$ is $I$-learnable [17,18]. The motivation behind 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 set of operators.

First, we can use epitomizers to show the identity of the learning power of two learning criteria. For example, theorem 6.1 establishes $\mathcal{S}_{\mathrm{FinSup}}$ to be epitomizing with respect to various learning criteria and certain sets of operators. In corollary 6.2, we use this to show a learning criterion $I$ to be as powerful as one of the epitomized learning criteria by showing that $\mathcal{S}_{\mathrm{FinSup}}$ can be $I$-learned. With classic methods, the proof of this result might have required tedious work with attention to detail, whereas we can conclude it to be 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, the so-called *self-learning* sets (theorem 6.6) are particularly useful epitomizers and have recently been used in the literature to prove particularly difficult separations (see [19,20], which solve two previously open problems using this technique, and see also [21,22]). Intuitively, the defining learner for a *self-learning set* merely runs each input function value as a program (on inputs relevant for that kind of learner).[12]

Freivalds *et al.* [13] noted that their $\leq^{\mathrm{Ex}}$-completeness does not give epitomization with respect to even the set of learning criteria considered in [13].

Thus, we believe that our approach to complexity of learning is both more *comprehensive* and more *useful* than the notion of intrinsic complexity from [13].

We present mathematical preliminaries in §2. The notions discussed above and some first theorems about them are given in §3—including the mentioned fundamental epitomization theorem characterizing epitomizers as complete sets (theorem 3.7).

Section 4 gives definitions and results regarding the notion of intrinsic complexity introduced in [13]. We have already mentioned our theorems 4.3 and 4.4 that witness drawbacks of this older notion; furthermore, in theorem 4.6, we characterize $\leq^{\mathrm{Ex}}$ in terms of one of our reducibility notions, and conclude in corollary 4.7 that all sets complete with respect to a central one of our reducibility notions are $\leq^{\mathrm{Ex}}$-complete.

Finally, in §6, we present a series of tasks and we state which learning criteria they epitomize *at what strength*. Note that epitomizers with respect to larger sets of learning criteria are stronger. As indicated above, we give each epitomization result, for some set of operators $\mathfrak{C}$, and with respect to the corresponding set of $\mathfrak{C}$-robust learning criteria. It will be seen that, the smaller the set of operators $\mathfrak{C}$, the larger the set of $\mathfrak{C}$-robust learning criteria. Theorem 6.1 entails that $\mathcal{S}_{\mathrm{FinSup}}$, the set of functions of finite support introduced above, is a very weak epitomizer. Some so-called self-*describing* sets (from the earlier literature) are also surprisingly weak epitomizers (theorem 6.3); some are of considerably greater strength (theorem 6.4); but, interestingly, the strongest are the (to be defined below) *self-learning* sets (theorem 6.6).

Some of our proofs involve subtle infinitary program self-reference arguments employing (variants of) Case's operator recursion theorem (ORT) [24,25].

We are working on extending the present study to employ self-learning sets that work for those learning criteria, *unlike those herein*, which require the learners

[12]Not explored herein is the connection between machine self-reference (which, as we will see, is needed to exploit the just-mentioned self-learning sets) and the weaker phenomenon of human (conscious) self-modelling. Regarding this connection, see [23].

to be total on *all* inputs. Some preliminary success re-learning grammars for languages (as noted above, not the topic of the present study) appear in [20] and involve the introduction of a hybrid operator recursion theorem (HORT).[13]

The present study is an extension of [26].

## 2. Mathematical preliminaries

### (*a*) *Notation and definitions*

Unintroduced computability-theoretic notions follow Rogers [16].

$\mathbb{N}$ 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, and $\mathrm{Pow}(A)$ 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, the 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) by $\lambda x_1, \ldots, x_n . f(x_1, \ldots, x_n)$. 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.[14]

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 one-to-one and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[15] With $\pi_1$ and $\pi_2$, 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 all (total) computable functions $\mathbb{N} \to \mathbb{N}$. The functions in $\mathcal{R}$ are called *computable functions*.

[13]The HORT permits infinitary self-and-other program reference between certain highly restricted complexity-bounded programming systems and a general-purpose programming system.

[14]$f(x)$ converges should not be confused with $f$ converges *to*.

[15]For a linear time computable and invertible example, see [27, §2.3].

We let $\varphi$ be any fixed acceptable programming system for the partial computable functions $\mathbb{N} \to \mathbb{N}$ with the associated complexity measure $\Phi$ [12,28]. Further, we let $\varphi_p$ denote the partial computable function computed by the $\varphi$-program with code number $p$, and we let $\Phi_p$ denote the partial computable *complexity* function of the $\varphi$-program with code number $p$.

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 [27]. We let **LinF** be the set of all linear time deterministic multi-tape Turing machine 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.

Some of our proofs will make use of s-m-n [16]. It is easy to see from [27] that, for a natural choice of $\varphi$ corresponding to multi-tape Turing machines, we can get the following S-m-n theorem:[16]

$$\exists s \in \mathbf{LL}\ \forall e, x, y : \varphi_{s(e,x)}(y) = \varphi_e(x, y). \tag{2.1}$$

For this study, we will assume (2.1). As a direct consequence (see [29] and the proof of [30, theorem 2.3.5]), we get a linlin version of **ORT**, given as follows ($\mathfrak{C}_{\text{eff}}$ is the set of all effective operators, see definition 3.2).

$$\forall \Theta \in \mathfrak{C}_{\text{eff}}\ \exists e \in \mathbf{LL}\ \forall x, y : \varphi_{e(x)}(y) = \Theta(e)(x, y). \tag{2.2}$$

### (*b*) *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 that, together, determine whether a given learner is successful on a given learning task.

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

Secondly, the learning criterion has to specify how the learner and the target interact. This part is modelled as a *sequence generating operator*, which is an operator $\beta$ taking as arguments a learner $h$ and a target $g$ and that outputs a function $p$. We denote by $p$ the *learning sequence* of $h$ given $g$. For this study, 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 $\delta$ on a learning sequence and a target function.[17]

For $\mathcal{C}$ a learner admissibility restriction, $\beta$ a sequence generating operator, $\delta$ 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

[16]We are grateful to Jim Royer for remarks providing an S-m-n theorem also linear time *invertible*.
[17]Herein, our $\beta$s and $\delta$s can essentially be modelled as multi-argument variants of Rogers' [16] *recursive operators*.

$\delta_I = \delta$. 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.[18] 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 make use of the special symbols '?' and '#', where '?' is used to denote 'don't know' conjectures, and '#' is used to denote a pause in a text.

We give the following three examples for sequence generating operators. Let **G** be defined thus:[19]

$$\forall h, g, i : \mathbf{G}(h, g)(i) = h(g[i]). \tag{2.3}$$

Let **It** be defined thus:[20]

$$\forall h, g : \mathbf{It}(h, g)(0) =? \ \wedge \ \forall i : \mathbf{It}(h, g)(i+1) = h(i, g(i), \mathbf{It}(h, g)(i)). \tag{2.4}$$

Finally, we give *transductive* learning [29].

$$\forall h, g : \mathbf{Td}(h, g)(0) =? \ \wedge \ \forall i : \mathbf{Td}(h, g)(i+1) = h(\langle i, g(i)\rangle). \tag{2.5}$$

For sequence acceptance criteria, we give the following five examples. We define *explanatory* learning [6] 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[21] [9,32,33].

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

For *conservative* learning [34], 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, behaviourally correct learning as given by [11,35] is associated with

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

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

[18]Note that '$\mathcal{C}\beta\delta$' is the classical way to denote this, while we use '$\mathfrak{S}(I)$' whenever $\mathcal{C}$, $\beta$ and $\delta$ are not explicit.
[19]**G** stands for Gold identification [6].
[20]**It** stands for iterative [31,32].
[21]Also called *consistent* learning.

Note that Ex-learning from the introduction is captured in this notation as **GEx**. *Bc-learning* from the prior literature, e.g. [11,35], requires for success that, for the learner's sequence of conjectured programs, all but finitely many of these programs correctly compute the input sequence/function. Bc-learning is similarly captured by **GBc**. For more examples of the above concepts, see [30].

We will use the following proposition later.

**Proposition 2.1.** *Let $\mathfrak{S} \subseteq \mathrm{Pow}(\mathcal{R})$ be $\subseteq$-downward closed. Then there is a learning criterion $I$ with $\mathfrak{S}(I) = \mathfrak{S}$.*

*Proof.* We let $p$ and $p'$ be two different computable functions. We associate with each element $\mathcal{S} \in \mathfrak{S}$ a learner $h_{\mathcal{S}}$. Let $\beta$ be such that, for all $\mathcal{S} \in \mathfrak{S}$ and $g \in \mathcal{S}$, $\beta(h_{\mathcal{S}}, g) = p$; for all other pairs $(h, g) \in \mathcal{P} \times \mathcal{R}$, let $\beta(h, g) = p'$. Let $\delta$ accept, for all $g \in \mathcal{R}$, $(p, g)$. Obviously, $\mathfrak{S}(\beta\delta) = \mathfrak{S}$. ∎

## 3. Concept definitions

In this section, we give the key concepts used in this study in definition 3.4. 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 the fundamental epitomization theorem (theorem 3.7), which shows the important connections between complete sets and epitomizers.

**Definition 3.1.** Let $S$ be a set and $\preceq$ a binary relation on $S$. 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 3.3.

**Definition 3.2.** A function $\Theta : \mathcal{P} \to \mathcal{P}$ is called an *operator*. We define the following sets of operators.

— Let $\mathfrak{C}_{\mathrm{eff}}$ be the set of all effective operators [16], i.e. all operators $\Theta$ such that there is a computable function $s \in \mathcal{R}$ with $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$.[22]

— Let $\mathcal{C} \subseteq \mathcal{P}$. Let $\mathfrak{C}^{\mathcal{C}}_{\mathrm{loc}}$ be the set of all $\Theta \in \mathfrak{C}_{\mathrm{eff}}$ such that there is a function $f \in \mathcal{C}$ with $\forall g \in \mathcal{P}, \forall x : \Theta(g)[x] = f(g[x])$.[23] We write $\mathfrak{C}_{\mathrm{loc}}$ for $\mathfrak{C}^{\mathcal{P}}_{\mathrm{loc}}$; note that $\Theta \in \mathfrak{C}_{\mathrm{loc}}$ is equivalent to

$$\forall g, g' \in \mathcal{P}\ \forall x : g[x] = g'[x] \Rightarrow \Theta(g)[x] = \Theta(g')[x]. \tag{3.1}$$

— Let $\mathcal{C} \subseteq \mathcal{P}$. Let $\mathfrak{C}^{\mathcal{C}}_{\mathrm{inj}}$ be the set of all $\Theta \in \mathfrak{C}_{\mathrm{eff}}$ such that[24]

$$\forall g, g' \in \mathcal{P}\ \forall x : \Theta(g)[x] = \Theta(g')[x] \Rightarrow g[x] = g'[x], \tag{3.2}$$

and there is a function $f \in \mathcal{C}$ with $\forall g \in \mathcal{P}, \forall x : g[x] = f(\Theta(g)[x])$. We write $\mathfrak{C}_{\mathrm{inj}}$ for $\mathfrak{C}^{\mathcal{P}}_{\mathrm{inj}}$.[25]

[22]Note that, without loss of generality, we can take $s$ to be linlin (this can be proven using linlin s-m-n).

[23]Intuitively, for all $g, x$, $\Theta(g)[x]$ depends only on $g[x]$. We call these operators *local*.

[24]'inj' stands for *injective*, but note that the operators from $\mathfrak{C}_{\mathrm{inj}}$ are not merely injective, but have a stronger requirement that could be called *locally injective*.

[25]Note that all effective operators satisfying (3.2) are in $\mathfrak{C}_{\mathrm{inj}}$.

— Let $\mathcal{C} \subseteq \mathcal{P}$. Let $\mathfrak{C}_{\text{ew}}^{\mathcal{C}}$ be the set of all $\Theta \in \mathfrak{C}_{\text{eff}}$ such that there is a function $f \in \mathcal{C}$ such that $\forall g \in \mathcal{P} : \Theta(g) = \lambda x . f(g(x), x)$.[26] We write $\mathfrak{C}_{\text{ew}}$ for $\mathfrak{C}_{\text{ew}}^{\mathcal{P}}$.

Clearly, $\mathfrak{C}_{\text{ew}} \subset \mathfrak{C}_{\text{loc}} \subset \mathfrak{C}_{\text{eff}}$. We define sets of *left-invertible* operators as follows. For any set of operators $\mathfrak{C}$ and $\mathcal{S} \subseteq \mathcal{R}$, we let

$$\text{LInv}(\mathfrak{C}; \mathcal{S}) = \{\Theta \in \mathfrak{C} \mid \Theta(\mathcal{S}) \subseteq \mathcal{R} \wedge \exists \hat{\Theta} \in \mathfrak{C}\, \forall g \in \mathcal{S} : (\hat{\Theta} \circ \Theta)(g) = g\}. \tag{3.3}$$

**Example 3.3.** For illustration, we give the following example operators.

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

$$\forall g \in \mathcal{P}\, \forall x : \Theta(g)(x) = \begin{cases} 0, & \text{if } x = 0, \\ g(x-1), & \text{otherwise,} \end{cases} \tag{3.4}$$

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

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

**Definition 3.4.** Let $\mathfrak{C}$ be a set of operators and $I$ a learning criterion. Let $\mathcal{S}, \mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{R}$.

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

$$\mathcal{S} \in \mathfrak{S}(I) \Rightarrow \Theta(\mathcal{S}) \in \mathfrak{S}(I). \tag{3.5}$$

Intuitively, $I$ is $\mathfrak{C}$-robust iff the set of $I$-learnable sets is closed under operators from $\mathfrak{C}$ that are left-invertible *on the set to which they are applied* (by an operator from $\mathfrak{C}$).
(b) 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[27]

$$\forall I' \in \mathcal{I} : [\mathcal{S} \in \mathfrak{S}(I') \Leftrightarrow \mathfrak{S}(I) \subseteq \mathfrak{S}(I')]. \tag{3.6}$$

(c) If $\mathcal{S}$ epitomizes $I$ with respect to the set of all $\mathfrak{C}$-robust learning criteria, then we say that $\mathcal{S}$ $\mathfrak{C}$-*epitomizes* $I$.
(d) We say that $\mathcal{S}$ $\mathfrak{C}$-*generates* $I$ iff $\{\Theta(\mathcal{S}') \mid \mathcal{S}' \subseteq \mathcal{S}, \Theta \in \text{LInv}(\mathfrak{C}; \mathcal{S}')\}$ is the set of all $I$-learnable functions.
(e) $\mathcal{S}_0 \leq^{\mathfrak{C}} \mathcal{S}_1$ iff there is an operator $\Theta \in \text{LInv}(\mathfrak{C}; \mathcal{S}_0)$ 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.

[26] We call these operators *element-wise*.
[27] Equation (3.6) is equivalent to $\mathcal{S} \notin \mathfrak{S}(I') \Leftrightarrow \mathfrak{S}(I) \setminus \mathfrak{S}(I') \neq \emptyset$. Thus, epitomizers give canonical candidates for learning criteria separations.

**Proposition 3.5.** *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 that there is a set epitomizing $I$ with respect to $\mathcal{I}$. Then*

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

Theorem 6.6 provides very strong existence results for epitomizers that can be used to satisfy the corresponding hypothesis of proposition 3.5.[28]

We can use proposition 3.5 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;[29] 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 [36] for the definitions; the result will be included in an extension of [36]).[30]

The following theorem gives a number of general observations regarding the concepts introduced in definition 3.4.

**Proposition 3.6.** *Let $\mathfrak{C}, \mathfrak{C}'$ be sets of operators, and $I$ a learning criterion.*

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

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

(b) *If $\mathfrak{C} \in \{\mathfrak{C}_{\mathrm{eff}}, \mathfrak{C}_{\mathrm{loc}}, \mathfrak{C}_{\mathrm{ew}}\}$, then $\preceq^{\mathfrak{C}}$ is a pre-order.*

(c) *Suppose that $I$ is $\mathfrak{C}$-robust. Then $\mathfrak{S}(I)$ is downward closed under $\preceq^{\mathfrak{C}}$, i.e. for all $\mathcal{S} \in \mathfrak{S}(I)$ and $\mathcal{S}' \subseteq \mathcal{R}$,*

$$\mathcal{S}' \preceq^{\mathfrak{C}} \mathcal{S} \Rightarrow \mathcal{S}' \in \mathfrak{S}(I).$$

*Proof.* Regarding (a): Let $\mathcal{S} \subseteq \mathcal{R}$, $\Theta \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S})$ as witnessed by $\hat{\Theta}$ and $\Theta(\mathcal{S}) \subseteq \mathcal{R}$. The direction '$\Rightarrow$' is obvious.

Suppose $\Theta(\mathcal{S}) \in \mathfrak{S}(I)$. We have $\hat{\Theta} \in \mathrm{LInv}(\mathfrak{C}; \Theta(\mathcal{S}))$ (as witnessed by $\Theta$); hence, as $\mathcal{S} = \hat{\Theta}(\Theta(\mathcal{S})) \in \mathfrak{S}(I)$, $\mathcal{S} \in \mathfrak{S}(I)$.

Item (b) is obvious.

Regarding (c): Suppose $\Theta \in \mathfrak{C}$ witnesses $\mathcal{S}' \preceq \mathcal{S}$. As $\Theta(\mathcal{S}') \subseteq \mathcal{S} \in \mathfrak{S}(I)$, $\Theta(\mathcal{S}) \in \mathfrak{S}(I)$. Hence, from $I$ being $\mathfrak{C}$-robust and (a), we get $\mathcal{S}' \in \mathfrak{S}(I)$. ∎

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

[28] 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 6.6 provides an appropriate epitomizer for $\mathbf{GEx}^*$, so that we can deduce, with proposition 3.5, $\bigcup_{a \in \mathbb{N}} \mathbf{GEx}^a \subset \mathbf{GEx}^*$ (which was shown in [11]).

[29] Intuitively, the definition of delayed postdictive completeness asks for the learner to start a *countdown* for each datum seen; as soon as the countdown has run out, the learner has to correctly postdict the associated datum.

[30] We are grateful to an anonymous referee who pointed out a (different) example for the non-existence of epitomizers.

**Theorem 3.7 (Fundamental epitomization theorem).** *Let $\mathfrak{C}$ be a set of operators containing the identity and which is closed under composition. Let $I$ be a $\mathfrak{C}$-robust learning criterion. Let $\mathcal{S} \subseteq \mathcal{R}$. The following are equivalent.*

(a) *$\mathcal{S}$ $\mathfrak{C}$-epitomizes $I$.*
(b) *$\mathcal{S}$ $\mathfrak{C}$-generates $I$.*
(c) *$\mathcal{S}$ is $\leq^{\mathfrak{C}}$-complete for $I$.*

*Proof.* '(a)⇒(b)': By proposition 2.1, there is a learning criterion $I'$ such that exactly all of $\mathfrak{S} = \{\Theta(\mathcal{S}') \mid \mathcal{S}' \subseteq \mathcal{S}, \Theta \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S}')\}$ are $I'$-learnable. We need to show that $\mathfrak{S}(I) = \mathfrak{S}$. As $I$ is $\mathfrak{C}$-robust, $\mathfrak{S} \subseteq \mathfrak{S}(I)$. To show the other direction, we first show that $I'$ is $\mathfrak{C}$-robust. Any $I'$-learnable set is of the form $\Theta(\mathcal{S}')$ where $\mathcal{S}' \subseteq \mathcal{S}$ and $\Theta \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S}')$ as witnessed by $\hat{\Theta}$. Pick such $\mathcal{S}', \Theta, \hat{\Theta}$. Suppose $\Theta' \in \mathrm{LInv}(\mathfrak{C}; \Theta(\mathcal{S}'))$ as witnessed by $\hat{\Theta}'$. As $\mathfrak{C}$ is closed under composition, $\Theta' \circ \Theta \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S}')$ as witnessed by $\hat{\Theta} \circ \hat{\Theta}'$. Thus, $\Theta'(\Theta(\mathcal{S}')) \in \mathfrak{S}$. Therefore, $I'$ is $\mathfrak{C}$-robust.

Now we show that $\mathfrak{S}(I) \subseteq \mathfrak{S}$. As $\mathfrak{C}$ contains the identity, $\mathcal{S} \in \mathfrak{S}$. Thus, as $\mathcal{S}$ $\mathfrak{C}$-epitomizes $I$, $\mathfrak{S}(I) \subseteq \mathfrak{S}(I') = \mathfrak{S}$.

'(b)⇒(c)': Let $\mathcal{S}' \in \mathfrak{S}(I)$. As $\mathcal{S}$ is an $\mathfrak{C}$-generator for $I$, let $\mathcal{S}'' \subseteq \mathcal{S}$ and $\Theta \in \mathrm{LInv}(\mathfrak{C}, \mathcal{S}')$ as witnessed by $\hat{\Theta}$ be such that $\Theta(\mathcal{S}'') = \mathcal{S}'$. Now we have that $\hat{\Theta}$ is left-invertible on $\mathcal{S}' \subseteq \Theta(\mathcal{S})$ (by $\Theta$). Thus, $\hat{\Theta} \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S}')$ and $\hat{\Theta}(\mathcal{S}') = \mathcal{S}'' \subseteq \mathcal{S}$, which shows that $\mathcal{S}' \leq^{\mathfrak{C}} \mathcal{S}$ as desired.

'(c)⇒(a)': Let $I'$ be $\mathfrak{C}$-robust. Suppose there is $\mathcal{S}' \in \mathfrak{S}(I) \setminus \mathfrak{S}(I')$. From $\mathcal{S}$ being $\leq^{\mathfrak{C}}$-complete for $\mathfrak{S}(I)$, we get $\Theta \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S})$ such that $\Theta(\mathcal{S}') \subseteq \mathcal{S}$. As $I'$ is $\mathfrak{C}$-robust and $\mathcal{S}'$ not $I'$-learnable, we have that $\Theta(\mathcal{S}')$ is not $I'$-learnable; thus, $\mathcal{S}$ $(\supseteq \Theta(\mathcal{S}'))$ is not $I'$-learnable. ∎

## 4. Connection to intrinsic complexity

We will now give the definitions of intrinsic complexity. Some version thereof was introduced in [13]; here we give an interpretation that fits our formalism. After that we will give theorems regarding the shortcomings of intrinsic complexity, as discussed in §1.

In particular, theorem 4.3 gives an example $\leq^{\mathbf{GEx}}$-complete set of functions that is nonetheless learnable in much more restricted criteria. Then, in theorem 4.4, 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, and 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 4.6.

**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 $(p, g) \in \delta$.[31]

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 $\Theta$ and $\Psi$ such that, for any function $g \in \mathcal{S}_0$,

[31] $I$-admissibility is not to be confused with learner admissibility restrictions.

(a) $\Theta(g) \in \mathcal{S}_1$ and
(b) 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 $\mathfrak{S}(I)$.

Note that, for any learning criterion $I$, the definition of $\leq^I$ is not sensitive to the learner admissibility restriction or the sequence generating operator of $I$. We formalize this as follows.

**Remark 4.2.** Let $I_0$ and $I_1$ be learning criteria with identical sequence acceptance criteria. Then $\leq^{I_0}$ and $\leq^{I_1}$ coincide.

**Theorem 4.3 ([9,13,35]).** *$\mathcal{S}_{\text{FinSup}}$ is $\leq^{\mathbf{GEx}}$-complete, but $\mathcal{S}_{\text{FinSup}} \in \mathbf{GPcpEx} \in \mathbf{GEx}$.*[32]

*Proof.* Completeness was proven in [13]; $\mathbf{GPcpEx} \subset \mathbf{GEx}$ is a well-known result [9,12,35]; the rest is straightforward. ∎

The following theorem shows the learnable sets of two learning criteria to be not downward closed with respect to their respective intrinsic reducibility notions.

**Theorem 4.4.**

(a) *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 $\mathcal{S}_0 \notin \mathbf{ItEx}$.*
(b) *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 $\mathcal{S}_0 \notin \mathbf{GPcpEx}$.*

*Proof.* Regarding (a): The intuitive idea of this proof is that $\leq^{\mathbf{ItEx}}$ does not depend on the severe memory limitation imposed by **It**. Clearly, $\leq^{\mathbf{ItEx}}$ equals the relation $\leq^{\mathbf{GEx}}$. Let $\mathcal{S} \in \mathbf{GEx} \backslash \mathbf{ItEx}$ (the existence of such sets is well known [12]). As $\mathcal{S}_{\text{FinSup}}$ is $\leq^{\mathbf{GEx}}$-complete, we have $\mathcal{S} \leq^{\mathbf{GEx}} \mathcal{S}_{\text{FinSup}}$; thus, $\mathcal{S} \leq^{\mathbf{ItEx}} \mathcal{S}_{\text{FinSup}}$. Clearly, $\mathcal{S}_{\text{FinSup}} \in \mathbf{ItEx}$, which concludes the proof of (a).

Regarding (b): The idea of this proof is that $\leq^{\mathbf{GPcpEx}}$ allows 'looking into the future' by the operators involved.

We use a set of functions from [37, corollary 15] (see also [9, p. 140 ff]). Let $\mathcal{S}_0$ be such that

$$\mathcal{S}_0 = \mathcal{S}_{\text{FinSup}} \cup \{g \in \mathcal{R} \mid \varphi_{g(0)} = g \wedge \forall x : \Phi_{g(0)}(x) \leq g(x+1)\}. \tag{4.1}$$

Let $\Theta \in \mathfrak{C}_{\text{eff}}$ be such that

$$\forall g, x : \Theta(g)(x) = \begin{cases} \langle g(x), 1\rangle, & \text{if } \Phi_{g(0)}(x) \leq g(x+1) \wedge \varphi_{g(0)}(x) = g(x),\text{[33]} \\ \langle g(x), 0\rangle, & \text{otherwise.} \end{cases} \tag{4.2}$$

We argue the following three claims, which will complete the proof.

— $\mathcal{S}_0 \leq^{\mathbf{GPcpEx}} \Theta(\mathcal{S}_0)$;

[32] An anonymous referee pointed out that an even more interesting result would be to find a set in **GPcpEx** that is $\leq^{\mathbf{GPcpEx}}$-complete for the **GEx** learnable sets; this way, the more fine-grained reduction $\leq^{\mathbf{GPcpEx}}$ would be shown to be too coarse.
[33] Intuitively, $\Theta$ 'looks into the future', as $\Theta(g)(x)$ depends on $g(x+1)$.

— $\Theta(\mathcal{S}_0) \in \mathbf{GPcpEx}$;
— $\mathcal{S}_0 \notin \mathbf{GPcpEx}$.

The last item has been shown in [37] (without the set of finite functions already in [9]); the second is straightforward to see by a learner $h$ as follows. On no data, $h$ outputs anything. Furthermore, as long as all data seen by $h$ have a 1 as second component, $h$ outputs the first component of the first datum. Otherwise, $h$ outputs a canonical program for an extension of the seen data with $\langle 0,0\rangle$s.

By s-m-n, there is $s \in \mathcal{R}$ such that $\forall e : \varphi_{s(e)} = \pi_1 \circ \varphi_e$. Let $\Psi$ be such that

$$\forall p : \Psi(p) = s \circ p. \tag{4.3}$$

To show that $\Theta, \Psi$ witness $\mathcal{S}_0 \leq^{\mathbf{GPcpEx}} \Theta(\mathcal{S}_0)$, it suffices to show that, for all $g \in \mathcal{S}_0$ and $p$ **GPcpEx**-admissible for $\Theta(g)$, $\Psi(p)$ is **GPcpEx**-admissible for $g$. Let $g \in \mathcal{S}_0$ and $p$ be **GPcpEx**-admissible for $\Theta(g)$. Therefore, for all $x$ and all $w < x$, $\varphi_{p(x)}(w) = \Theta(g)(w)$; thus,

$$\varphi_{s(p(x))}(w) = \pi_1(\varphi_{p(x)}(w)) = \pi_1(\Theta(g)(w)) = g(w). \tag{4.4}$$

This shows that $\Psi(p)$ is a postdictively complete sequence of conjectures for $g$. Convergence of $p$ to a correct index is straightforward. ■

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

**Definition 4.5.** Let $\mathfrak{C}$ and $\mathfrak{C}'$ be sets of operators and $\mathcal{S} \subseteq \mathcal{R}$.

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

$$\forall e : \Theta(\varphi_e) = \begin{cases} \varphi_p, & \text{if } \lambda t . s(e,t) \text{ converges to some } p, \\ \uparrow, & \text{otherwise.}^{34} \end{cases} \tag{4.5}$$

Now we show the equivalence of one particular reducibility notion of intrinsic complexity to one of our extended variants from definition 4.5. Note that a similar characterization can be made for **GBc**.

**Theorem 4.6.** *We have*

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

*Proof.* Let $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{R}$. Suppose $\mathcal{S} \leq^{(\mathfrak{C}_{\mathrm{eff}}, \mathfrak{C}_{\mathrm{limPEff}})} \mathcal{S}'$ as witnessed by $\Theta \in \mathfrak{C}_{\mathrm{eff}}$ and $\hat{\Theta} \in \mathfrak{C}_{\mathrm{limPEff}}$. Then $\Theta$ is as required for $\mathcal{S} \leq^{\mathbf{GEx}} \mathcal{S}'$. Let $s \in \mathcal{R}$ be as given by $\hat{\Theta} \in \mathfrak{C}_{\mathrm{limPEff}}$. Let $\Psi$ be such that

$$\forall p, t : \Psi(p)(t) = s(p(t), t). \tag{4.6}$$

Clearly, $\Psi$ is as required for $\mathcal{S} \leq^{\mathbf{GEx}} \mathcal{S}'$.

[34]Note that the operators from $\mathfrak{C}_{\mathrm{limPEff}}$ resemble those from [38,39].

For the converse, suppose $\mathcal{S} \leq^{\mathbf{GEx}} \mathcal{S}'$ as witnessed by $\Theta$ and $\Psi$. Then $\Theta$ is as required for $\mathcal{S} \leq^{\mathfrak{C}} \mathcal{S}'$. By s-m-n, there is an $f \in \mathcal{R}$ such that

$$\forall e : \varphi_{f(e)} = \Psi(\lambda x . e). \tag{4.7}$$

Let $s \in \mathcal{R}$ be such that

$$\forall e, t : s(e,t) = \begin{cases} 0, & \text{if } \forall t' \leq t : \Phi_{f(e)}(t') > t, \\ \varphi_{f(e)}(t'), & \text{otherwise, with } t' \text{ maximal such that} \\ & t' \leq t \text{ and } \Phi_{f(e)}(t') \leq t. \end{cases} \tag{4.8}$$

Obviously, $s$ is total. Furthermore, for all $e$ with $\varphi_e \in \Theta(\mathcal{S})$, from the choice of $\Psi$ we have that $\varphi_{f(e)}$ is total and converges to a program number for the pre-image of $\varphi_e$ with respect to $\Theta$. We let $\hat{\Theta}$ be as in (4.5). It is easy to see that $\hat{\Theta}$ is also as required. ■

We get the following corollary from theorem 4.6.

**Corollary 4.7.** *Let $I$ be a learning criterion with $\delta_I = \mathbf{Ex}$ as the sequence acceptance criterion. We have that $\leq^{\mathfrak{C}_{\text{eff}}}$ is a subrelation of $\leq^{\mathbf{GEx}}$; in particular, for all $\mathcal{S} \subseteq \mathcal{R}$*

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

*Proof.* Clearly, $\leq^I$ and $\leq^{\mathbf{GEx}}$ are identical. Note that, for all sets of operators $\mathfrak{C}_0, \mathfrak{C}_0', \mathfrak{C}_1$ and $\mathfrak{C}_1'$ with $\mathfrak{C}_0 \subseteq \mathfrak{C}_0'$ and $\mathfrak{C}_1 \subseteq \mathfrak{C}_1'$ we have that $\leq^{(\mathfrak{C}_0,\mathfrak{C}_1)}$ is a subrelation of $\leq^{(\mathfrak{C}_0',\mathfrak{C}_1')}$. Further, for all sets of operators $\mathfrak{C}$, it is easy to see that $\leq^{\mathfrak{C}}$ equals $\leq^{(\mathfrak{C},\mathfrak{C})}$. Thus, $\leq^{\mathfrak{C}_{\text{eff}}}$ is a subrelation of $\leq^{(\mathfrak{C}_{\text{eff}},\mathfrak{C}_{\text{limPEff}})}$. The rest follows from theorem 4.6. ■

## 5. Robustness

In this section, we give a number of examples for robustness of learning criteria with respect to various sets of operators. These results are summarized in table 1.

**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.

Table 1 states several kinds of robustness of learning criteria with respect to certain sets of operators. In particular, in each row, the learning criteria in the right column are robust with respect to the set of operators in the left column.

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

The remaining section deals with proving the statement of example 5.1.

First, we define some properties for sequence acceptance criteria.

**Definition 5.2.** Let $\mathfrak{C} \subseteq \mathfrak{C}_{\text{eff}}$ be a set of effective operators. A sequence acceptance criterion $\delta$ is called $\mathfrak{C}$-*robust* iff, for all $\Theta \in \mathfrak{C}$ and $s \in \mathcal{R}$ with $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$,

$$\forall p \in \mathfrak{P}, g \in \mathcal{R} : (p,g) \in \delta \Rightarrow (s \circ p, \Theta(g)) \in \delta.^{35} \tag{5.1}$$

[35]Note that, for robustness of learning criteria, we restricted ourselves to certain left-invertible operators, whereas, for robustness of sequence acceptance criteria, we make no such restriction.

Table 1. Learning criteria and their robustness.

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

**Definition 5.3.** Let $\mathcal{R}^{\infty} = \{r \in \mathcal{R} \mid r \text{ is non-decreasing and } \forall x, \forall^{\infty} t : r(t) \geq x\}$.[36] A sequence acceptance criterion $\delta$ is called a *delayable* iff

$$\forall p \in \mathcal{P}, g \in \mathcal{R}, r \in \mathcal{R}^{\infty} : (p, g) \in \delta \Rightarrow (p \circ r, g) \in \delta.^{37} \tag{5.2}$$

The following examples are straightforward.

**Example 5.4.**

(a) **Ex** and **Bc** are delayable and $\mathfrak{C}_{\text{eff}}$-robust.
(b) **Pcp** is *not* delayable, whereas **Conv** is.

Next, we make a side remark and use Pitt-style delaying tricks (see [40]) to show that, for learning criteria with **G** as a sequence generating operator and a delayable sequence acceptance criterion, all learners can be assumed linear time computable.

**Proposition 5.5.** *Let $\delta$ be delayable. Then*

$$\mathcal{P}\alpha\mathbf{G}\delta = \mathbf{LinF}\alpha\mathbf{G}\delta. \tag{5.3}$$

*Proof.* The inclusion '$\supseteq$' is trivial. We fix linear time computable functions $T$ and $S$ as shown existent in [27, theorem 3.20] such that, for all $e$ and $x$, if $\varphi_e(x)\downarrow$, then, for all but finitely many $t$, $T(e, x, t) = 1$; and, for all $e, x$ and $t$, if $T(e, x, t) = 1$, then $S(e, x, t) = \varphi_e(x)$.

Let $\mathcal{S} \in \mathcal{P}\mathbf{G}\delta$ as witnessed by $h \in \mathcal{P}$. Let $e$ be such that $\varphi_e = h$. Let $f, h' \in \mathbf{LinF}$ be such that, for all $\sigma$,

$$f(\sigma) = \max_{\{t \mid \sigma[t] \leq \log(|\sigma|)\}} T(e, \sigma[t], \log(|\sigma|)) = 1 \tag{5.4}$$

and

$$h'(\sigma) = \begin{cases} 0, & \text{if, for all } t \text{ with } \sigma[t] \leq \log(|\sigma|), \\ & \quad T(e, \sigma[t], \log(|\sigma|)) = 0, \\ S(e, \sigma[f(\sigma)], \log(|\sigma|)), & \text{otherwise.} \end{cases} \tag{5.5}$$

It is easy to see that $h'$ is linear time computable and learns all functions learned by $h$. ∎

[36]Note that $\mathcal{R}^{\infty}$ is equal to the set of all non-decreasing and unbounded total computable functions.
[37]Intuitively, if $p$ is a valid (with respect to $\delta$) sequence of conjectures for $g$, then any delayed variant of $p$ is.

We now characterize $\mathfrak{C}$-robustness of **Pcp** and **Conv** in the following two theorems.

**Theorem 5.6.** *Let $\mathfrak{C} \subseteq \mathfrak{C}_{\text{eff}}$ be a set of effective operators. We have*

$$\textbf{Pcp} \text{ is } \mathfrak{C}\text{-robust} \Leftrightarrow \mathfrak{C} \subseteq \mathfrak{C}_{\text{loc}}. \tag{5.6}$$

*Proof.* We show the equivalence operator-wise. Suppose $\Theta \in \mathfrak{C}$ and $s$ is any element $\mathcal{R}$ with $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$.

'$\Leftarrow$': Suppose $(p, g) \in \textbf{Pcp}$, let $x \in \mathbb{N}$. We have $\varphi_{p(x)}[x] = g[x]$. Thus, as $\Theta \in \mathfrak{C}_{\text{loc}}$,

$$\Theta(g)[x] = \Theta(\varphi_{p(x)})[x] = \varphi_{(s \circ p)(x)}[x]. \tag{5.7}$$

'$\Rightarrow$': Suppose $g, g' \in \mathcal{R}$ and $x \in \mathbb{N}$ with $g[x+1] = g'[x+1]$. Suppose, by way of contradiction, $\Theta(g)(x) \neq \Theta(g')(x)$. Let $e$ and $e'$ be such that $\varphi_e = g$ and $\varphi_{e'} = g'$. Let $p \in \mathcal{R}$ be such that

$$\forall y : p(y) = \begin{cases} e', & \text{if } y \leq x+1, \\ e, & \text{otherwise.} \end{cases} \tag{5.8}$$

Then $(p, g) \in \textbf{Pcp}$. As

$$\varphi_{(s \circ p)(x+1)}(x) = \varphi_{s(e')}(x) = \Theta(g')(x) \neq \Theta(g)(x), \tag{5.9}$$

we get $(s \circ p, \Theta(g)) \notin \textbf{Pcp}$. ∎

**Theorem 5.7.** *Let $\mathfrak{C} \subseteq \mathfrak{C}_{\text{eff}}$ be a set of operators. We have*

$$\textbf{Conv} \text{ is } \mathfrak{C}\text{-robust} \Leftrightarrow \mathfrak{C} \subseteq \mathfrak{C}_{\text{inj}}. \tag{5.10}$$

*Proof.* We show the equivalence operator-wise. Let $\Theta \in \mathfrak{C}$ and $s \in \mathcal{R}$ with $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$.

'$\Leftarrow$': Suppose $(p, g) \in \textbf{Conv}$, let $x \in \mathbb{N}$ with $p(x) \neq p(x+1)$. Thus, $g[x+1] \neq \varphi_{p(x)}[x+1]$. Hence, as $\Theta \in \mathfrak{C}_{\text{inj}}$,

$$\Theta(g)[x+1] \neq \Theta(\varphi_{p(x)})[x+1] = \varphi_{(s \circ p)(x)}[x+1]. \tag{5.11}$$

'$\Rightarrow$': Without loss of generality, suppose $s$ is one-to-one. Let $g, g' \in \mathcal{R}$ and $x$ be such that $g[x] = g'[x]$ and $g(x) \neq g'(x)$. Suppose, by way of contradiction, that $\Theta(g)[x+1] = \Theta(g')[x+1]$. Let $e$ and $e'$ be such that $\varphi_e = g$ and $\varphi_{e'} = g'$. Let $p \in \mathcal{R}$ be such that

$$\forall y : p(y) = \begin{cases} e', & \text{if } y \leq x, \\ e, & \text{otherwise.} \end{cases} \tag{5.12}$$

Then $(p, g) \in \textbf{Conv}$. We have $s(e) \neq s(e')$, as $s$ is one-to-one and $g \neq g'$. As

$$\varphi_{(s \circ p)(x)}[x+1] = \varphi_{s(e')}[x+1] = \Theta(g')[x+1] = \Theta(g)[x+1], \tag{5.13}$$

we get $(s \circ p, \Theta(g)) \notin \textbf{Conv}$ (a mind change was made from $x$ to $x+1$, even though the previous hypothesis $(s \circ p)(x)$ was correct on known data $\Theta(g)[x+1]$). ∎

**Definition 5.8.**

— An operator $\Theta$ is called $\mathcal{S}$-bijective iff, for all $g \in \mathcal{S}$ and $g' \in \mathcal{R}$,

$$\forall x : g[x] = g'[x] \Leftrightarrow \Theta(g)[x] = \Theta(g')[x]. \tag{5.14}$$

— A sequence acceptance criterion $\delta$ is called *local bijection robust* iff, for all $\mathcal{S}$, $\Theta$ $\mathcal{S}$-bijective and $s$ total (not necessarily computable) with $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$,

$$\forall p \in \mathfrak{P}, g \in \mathcal{S} : (p, g) \in \delta \Rightarrow (s \circ p, \Theta(g)) \in \delta. \tag{5.15}$$

As a corollary to the proofs of the directions '$\Leftarrow$' of theorems 5.6 and 5.7, we see that **Pcp** and **Conv** are local bijection robust. Furthermore, local bijection robust criteria are closed under intersection.

**Proposition 5.9.** *Let $\Theta \in \mathfrak{C}_{\mathrm{eff}}$. Then there is an $f \in$ **LinF** such that, for all sequences $\sigma$, $f(\sigma)$ is a sequence with*

$$(\forall g \in \mathcal{R} \mid \sigma \subseteq g)\, f(\sigma) \subseteq \Theta(g),$$

*and, for all $g \in \mathcal{R}$ with $\Theta(g) \in \mathcal{R}$, there is $r \in \mathcal{R}^{\infty}$ such that*[38]

$$\forall t : f(g[t]) = \Theta(g)[r(t)].^{39}$$

*Proof.* We argue informally. We view $\Theta$ as a recursive operator. Let $\sigma$ be a sequence. We define $f(\sigma)$ as follows. We let $\Theta$ compute $\Theta(\sigma)(x)$ for all $x < \log(1 + \#\mathrm{elets}(\sigma))$ for $\log(1 + |\sigma|)$ steps. Let $x_0$ be maximal such that, for all $x < x_0$, this computation halted with output $z_x$. Then $f(\sigma)$ is the sequence $z_0, \ldots, z_{x_0 - 1}$. For all $g \in \mathcal{R}$ with $\Theta(g) \in \mathcal{R}$, we let $r$ be such that, for all $t$, $r(t)$ is the $x_0$ obtained while computing $f(g[t])$ as above. It is straightforward to check that $f$ is as desired. ■

After having introduced various properties of sequence acceptance criteria and establishing them for some examples, we are now ready to give our main theorem to establish robustness of learning criteria with respect to certain sets of operators. Example 5.1 provides corollaries to theorem 5.10.

**Theorem 5.10.** *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. Further, let $\delta, \delta'$ be sequence acceptance criteria and $\mathfrak{C} \subseteq \mathfrak{C}_{\mathrm{eff}}$.*

(a) *Suppose $\delta$ is $\mathfrak{C}$-robust and delayable. Then $(\mathcal{F}, \mathbf{G}, \delta)$ is $\mathfrak{C}$-robust.*
(b) *Suppose $\delta$ is local bijection robust. Then $(\mathcal{F}, \mathbf{G}, \delta)$ is $\mathfrak{C}^{\mathcal{C}}_{\mathrm{loc}}$-robust.*
(c) *Suppose $\delta$ is $\mathfrak{C}^{\mathcal{C}}_{\mathrm{ew}}$-robust. Then $(\mathcal{F}, \mathbf{It}, \delta)$ is $\mathfrak{C}^{\mathcal{C}}_{\mathrm{ew}}$-robust.*

*Proof.* Regarding (a): Let $\mathcal{S} \in \mathcal{F}\mathbf{G}\delta$ as witnessed by $h \in \mathcal{F}$ and let $\Theta \in \mathrm{LInv}(\mathfrak{C}; \mathcal{S})$ as witnessed by $\hat{\Theta}$. Let $s \in \mathcal{R}$ be such that $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$. Let $f \in \mathcal{R}$

[38] We gave the set $\mathcal{R}^{\infty}$ in definition 5.3.
[39] Note that, in *this* study, we do not use that $f$ is linear time computable, but merely that it is computable.

be as claimed existent for $\hat{\Theta}$ in proposition 5.9. Let $h' \in \mathcal{P}$ be such that

$$h' = s \circ h \circ f. \tag{5.16}$$

It suffices to show that $h'$ $(\mathcal{P}, \mathbf{G}, \delta)$-learns $\Theta(\mathcal{S})$, as we then get (a) by the application of proposition 5.5. Obviously, $h' \in \mathcal{P}$. Let $g \in \mathcal{S}$. Let $p = \mathbf{G}(h, g)$. By the choice of $f$, for all $g'$ with $\hat{\Theta}(g')$ total, there is $r \in \mathcal{R}^{\infty}$ such that $\forall t : f(g'[t]) = \hat{\Theta}(g')[r(t)]$. Let $r$ be witnessing this for $\Theta(g)$ (as we have that $\hat{\Theta}(\Theta(g))$ is total). By the choice of $\hat{\Theta}$ we have $\forall t : f(\Theta(g)[t]) = \hat{\Theta}(\Theta(g))[r(t)] = g[r(t)]$. Thus,

$$\begin{aligned}\mathbf{G}(h', \Theta(g)) &= \lambda t.\, h'(\Theta(g)[t]) = \lambda t.\, (s \circ h)(f(\Theta(g)[t])) = \lambda t.\, (s \circ h)(g[r(t)]) \\ &= s \circ (\lambda t.\, h(g[t])) \circ r = s \circ p \circ r.\end{aligned}$$

By $\delta$ being $\mathfrak{C}$-robust, we have $(s \circ p, \Theta(g)) \in \delta$. As $\delta$ is delayable, we have $(s \circ p \circ r, \Theta(g)) \in \delta$ as desired.

Regarding (b): Let $\mathcal{S} \in \mathcal{F}\mathbf{G}\delta$ as witnessed by $h \in \mathcal{F}$; furthermore, let $\Theta \in \mathrm{LInv}(\mathfrak{C}^{\mathcal{C}}_{\mathrm{loc}}; \mathcal{S})$ as witnessed by $\hat{\Theta} \in \mathfrak{C}^{\mathcal{C}}_{\mathrm{loc}}$. Let $s$ be linlin such that $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$. Let $f \in \mathcal{C}$ be as given by $\hat{\Theta} \in \mathfrak{C}^{\mathcal{C}}_{\mathrm{loc}}$, that is, such that $\forall g \in \mathcal{R}, \forall x : f(g[x]) = \hat{\Theta}(g)[x]$. In particular, $\forall g \in \mathcal{S}, \forall x : f(\Theta(g)[x]) = g[x]$. Let $h' \in \mathcal{P}$ be such that

$$h' = s \circ h \circ f. \tag{5.17}$$

We show that $h'$ $(\mathcal{F}, \mathbf{G}, \delta)$-learns $\Theta(\mathcal{S})$. Obviously, $h' \in \mathcal{F}$, as $\mathcal{F}$ is closed under $\mathcal{C}$-composition. Let $g \in \mathcal{S}$ and $p = \mathbf{G}(h, g)$. By the choice of $f$,

$$\mathbf{G}(h', \Theta(g)) = \lambda t.\, h'(\Theta(g)[t]) = s \circ \lambda t.\, h(f(\Theta(g)[t])) = s \circ \lambda t.\, h(g[t]) = s \circ p, \tag{5.18}$$

similarly as in the proof of (a). By $\delta$ being local bijection robust, it suffices to show $\Theta$ to be $\{g\}$-bijective. The direction '$\Rightarrow$' in (5.14) of $\{g\}$-bijectivity follows directly from $\Theta \in \mathfrak{C}_{\mathrm{loc}}$, the other from $\hat{\Theta} \in \mathfrak{C}_{\mathrm{loc}}$ being an inverse for $\Theta$ on $g$.

Regarding (c): Let $\mathcal{S} \in \mathcal{F}\mathbf{It}\delta$ as witnessed by $h \in \mathcal{F}$; furthermore, let $\Theta \in \mathrm{LInv}(\mathfrak{C}^{\mathcal{C}}_{\mathrm{ew}}; \mathcal{S})$ as witnessed by $\hat{\Theta}$. Let $s$ be linlin such that $\forall e : \Theta(\varphi_e) = \varphi_{s(e)}$. Let $f \in \mathcal{C}$ be such that $\forall g : \hat{\Theta}(g) = \lambda x.\, f(x, g(x))$. Let $h' \in \mathcal{P}$ be such that

$$h' = \lambda x, y, e.\, (s \circ h)(x, f(x, y), s^{-1}(e)).^{40} \tag{5.19}$$

The remainder of the proof is analogous to (a). ■

## 6. Specific function learning epitomizers

In this section we present several epitomizers, starting with the very natural set of functions $\mathcal{S}_{\mathrm{FinSup}}$ in theorem 6.1. After noting an immediate corollary, we show a self-*describing* set to be an epitomizer. Finally, we give a construction for self-*learning* sets and show them to be very powerful epitomizers in theorem 6.6, i.e. epitomizing with respect to a very wide range of learning criteria, much wider than for self-descibing sets (which is in turn wider than for $\mathcal{S}_{\mathrm{FinSup}}$).

[40]Recall that the input to an iterative learner on function $g$ in iteration $x$ is the triple $x$, $g(x)$ and prior conjecture.

Note that all proofs rely implicitly on the fundamental epitomization theorem (theorem 3.7), as they show completeness and derive epitomization from that.

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

**Theorem 6.1.** *We have*

(a) $\mathcal{S}_{\text{FinSup}}$ $\mathfrak{C}_{\text{eff}}$*-epitomizes* **GEx**;
(b) $\mathcal{S}_{\text{FinSup}}$ *does not* $\mathfrak{C}_{\text{loc}}$*-epitomize* **GEx**;
(c) $\mathcal{S}_{\text{FinSup}}$ $\mathfrak{C}_{\text{loc}}$*-epitomizes* **GPcpEx**;
(d) $\mathcal{S}_{\text{FinSup}}$ *does not* $\mathfrak{C}_{\text{ew}}$*-epitomize* **GPcpEx**.

*Proof.* We prove (a) at the end, since the proof of (c) gives valuable intuition for the proof of (a).

The claim of (b) follows from $\mathcal{S}_{\text{FinSup}} \in \mathbf{GPcpEx} \subset \mathbf{GEx}$; see [9,12,35].

The claim of (d) follows from the fact that $\mathcal{S}_{\text{FinSup}}$ is identifiable by a learner outputting only programs for total functions; the associated learning criterion, called *Popperian* identification, is strictly weaker than **GPcpEx** and $\mathfrak{C}_{\text{ew}}$-robust, as only subsets of uniformly recursive functions are learnable [11,41].

We prove (c). We use theorem 3.7 and show $\mathcal{S}_{\text{FinSup}}$ to be $\mathfrak{C}_{\text{loc}}$-complete for **GPcpEx**. Let $\mathcal{S} \in \mathbf{GPcpEx}$ as witnessed by $h$. Let $\Theta$ and $\hat{\Theta}$ be such that, for all $g \in \mathcal{P}$ and $x \in \mathbb{N}$,

$$\Theta(g)(x) = \begin{cases} \uparrow, & \text{if } h(g[x])\uparrow \vee h(g[x+1])\uparrow, \\ 0, & \text{else if } x > 0 \text{ and } h(g[x]) = h(g[x+1]), \\ 1 + h(g[x+1]), & \text{otherwise} \end{cases} \tag{6.1}$$

and

$$\hat{\Theta}(g)(x) = \varphi_{g(t)-1}(x), \quad \text{where } t = \max_{t \le x}(g(t) \neq 0).^{41} \tag{6.2}$$

Obviously, $\Theta, \hat{\Theta} \in \mathfrak{C}_{\text{loc}}$. Let $g \in \mathcal{S}$ and $x \in \mathbb{N}$. We show $(\hat{\Theta} \circ \Theta)(g)(x) = g(x)$. We have $\Theta(g)(0) \neq 0$. Let $t \le x$ be maximal such that $\Theta(g)(t) \neq 0$. By the definition of $\Theta$ and the maximality of $t$, $\Theta(g)(t) = 1 + h(g[x+1])$. Thanks to $h$ learning $g$ postdictively completely, we have

$$\hat{\Theta}(\Theta(g))(x) = \varphi_{\Theta(g)(t)-1}(x) = \varphi_{h(g[x+1])}(x) = g(x). \tag{6.3}$$

Furthermore, for all $g \in \mathcal{S}$, as $h$ converges on $g$, $\Theta(g) \in \mathcal{S}_{\text{FinSup}}$.

Regarding (a), we use a similar but more complex construction. The main difference is that we need to insert more information as long as the learner is not converged to ensure invertibility of the operator to be defined.

[41] By convention, $\max \emptyset = 0$ and $(0-1)\uparrow$.

Let $\mathcal{S} \in \mathbf{GEx}$ as witnessed by $h$. Without loss of generality, we assume $h$ to be total [12]. Let $\Theta$, $\hat{\Theta}$, $\Psi_0$, $\Psi_1$ and $c$ be such that, for all $j \in \{1,2\}$, $g \in \mathcal{P}$, $x \in \mathbb{N}$,

$$c(x) = 1 + \langle h(g[x+1]), g[x+1]\rangle, \tag{6.4}$$

$$\Theta(g)(x) = \begin{cases} \uparrow, & \text{if } h(g[x])\uparrow \vee h(g[x+1])\uparrow, \\ 0, & \text{else if } x > 0 \wedge h(g[x]) = h(g[x+1]) \wedge \\ & \quad \forall y \leq x : (\Phi_{h(g[x])}(y) > x \vee \varphi_{h(g[x])}(y) = g(y)), \\ c(x), & \text{otherwise,} \end{cases} \tag{6.5}$$

$$\Psi_j(g,x) = \pi_j(g(\max_{t \leq x}(g(t) \neq 0)) - 1) \tag{6.6}$$

and

$$\hat{\Theta}(g)(x) = \begin{cases} \uparrow, & \text{if } (\Psi_1(g,x)\uparrow \vee \varphi_{\Psi_1(g,x)}(x)\uparrow) \wedge \forall t \geq x : g(t) = 0, \\ \Psi_2(g,t)(x), & \text{else if there is minimal } t \leq \Phi_{\Psi_1(g,x)}(x) :^{42} \\ & \quad t \geq x \wedge g(t)\downarrow \neq 0, \\ \varphi_{\Psi_1(g,x)}(x), & \text{otherwise.} \end{cases} \tag{6.7}$$

Obviously, $\Theta, \hat{\Theta} \in \mathfrak{C}_{\text{eff}}$. Let $g \in \mathcal{S}$ and $x \in \mathbb{N}$. We show $(\hat{\Theta} \circ \Theta)(g)(x) = g(x)$.

Let $g' = \Theta(g)$. Obviously, for each $j \in \{0,1\}$, $\Psi_j(g',x)\downarrow$.

*Case 1*: There is $t \leq \Phi_{\Psi_1(g',x)}(x)$ with $t \geq x$ and $g'(t) \neq 0$.

Fix the minimal such $t$. Then the first case in the definition of $\hat{\Theta}(g')(x)$ does not hold, but the second case does. Therefore,

$$\hat{\Theta}(\Theta(g))(x) = \Psi_2(g',t)(x) = \pi_2(g'(t))(x) = g[t+1](x) = g(x). \tag{6.8}$$

*Case 2*: $\forall t \geq x : g'(t) = 0$.

Then $h$ on $g$ is converged after $x$ steps (by the definition of $\Theta$) and $h(g[x])$ is a program for $g$. Thus, $\varphi_{\Psi_1(g,x)}(x)\downarrow = g(x)$, which shows that the first and second cases in the definition of $\hat{\Theta}(g')(x)$ do not hold and $\hat{\Theta}(\Theta(g))(x) = g(x)$.

*Case 3*: Otherwise.

As case 1 does not hold, for all $t$ with $x \leq t \leq \Phi_{\Psi_1(g',x)}(x)$, $g'(t) = 0$. As case 2 does not hold, $\Phi_{\Psi_1(g',x)}(x)\downarrow$. Let $t = \Phi_{\Psi_1(g',x)}(x)$. We have $g'(t) = 0$. Let $t' = \max_{t \leq x}[g'(t) \neq 0]$. Note that $\Psi_1(g',t) = h(g[t']) = h(g[t])$.

From the definition of $\Theta$, as $\Theta(g)(t) = 0$, $\varphi_{h(g[t])}(x) = g(x)$. Hence, the first and second cases in the definition of $\hat{\Theta}(g')(x)$ do not hold and

$$\hat{\Theta}(\Theta(g))(x) = \varphi_{\Psi_1(g,x)}(x) = \varphi_{h(g[t])}(x) = g(x). \tag{6.9}$$

This finishes the different cases. We also have, for all $g \in \mathcal{S}$, as $h$ converges on $g$, $\Theta(g) \in \mathcal{S}_{\text{FinSup}}$. ∎

From theorem 6.1 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.

[42]Note that there are several ways in which $\Phi_{\Psi_1(g,x)}(x)$ can be undefined, in which case $\hat{\Theta}$ makes an unbounded search.

**Corollary 6.2.** *We have*

$$\mathbf{GPcpEx} = \mathbf{GPcpConvEx}.$$

*Proof.* Clearly, $\mathcal{S}_{\mathrm{FinSup}}$, the set of all computable functions with finite support, is **GPcpConvEx**-learnable. By example 5.1, **GPcpConvEx** is $\mathfrak{C}_{\mathrm{loc}}$-robust. We obtain the result using theorem 6.1(c). ∎

Next we analyse a set of self-describing functions that was first given in [11] and used extensively in [12]. Theorem 6.2 shows that this set is not a very good epitomizer.

**Theorem 6.3.** *Let* $\mathcal{S}_0 = \{g \in \mathcal{R} \mid \varphi_{g(0)} = g\}$. *Then* $\mathcal{S}_0$ $\mathfrak{C}_{\mathrm{eff}}$*-epitomizes* **GFin**. *However,* $\mathcal{S}_0$ *does not* $\mathfrak{C}_{\mathrm{loc}}$*-epitomize any learning criterion that can be built from components given in this study as* $\mathcal{S}_0 \equiv^{\mathfrak{C}_{\mathrm{loc}}} \{g \in \mathcal{R} \mid \forall x > 0 : g(x) = 0\}$.

*Proof.* We show $\mathcal{S}_0$ to be $\leq^{\mathfrak{C}_{\mathrm{eff}}}$-complete for **GFin** and apply theorem 3.7. Let $\mathcal{S} \in \mathbf{GFin}$ as witnessed by $h \in \mathcal{R}$ (we can assume, without loss of generality, $h \in \mathcal{R}$). By the one-to-one **ORT**,[43] there is a one-to-one $s \in \mathcal{R}$ such that

$$\forall e, x : \varphi_{s(e)}(x) = \begin{cases} s(e), & \text{if } x = 0, \\ \langle \varphi_e(0), \varphi_e(x) \rangle, & \text{otherwise.} \end{cases} \tag{6.10}$$

Let $\Theta$, $\hat{\Theta}$ be such that

$$\forall g \in \mathcal{P}, x : \Theta(g)(x) = \begin{cases} s(h(g[\mu t . h(g[t]) \neq ?])), & \text{if } x = 0, \\ \langle g(0), g(x) \rangle, & \text{otherwise} \end{cases} \tag{6.11}$$

and

$$\forall g \in \mathcal{P}, x : \hat{\Theta}(g)(x) = \begin{cases} \pi_1(g(1)), & \text{if } x = 0, \\ \pi_2(g(x)), & \text{otherwise.} \end{cases} \tag{6.12}$$

Clearly, $\Theta \in \mathrm{LInv}(\mathfrak{C}_{\mathrm{eff}})$ as witnessed by $\hat{\Theta}$. Furthermore, $\Theta(\mathcal{S}) \subseteq \mathcal{S}_0$. This finishes the proof of the first assertion of the theorem.

We get $\mathcal{S}_0 \leq^{\mathfrak{C}_{\mathrm{loc}}} \mathcal{S}_1$ by letting (a different) $\Theta$ be such that

$$\forall g \in \mathcal{P}, x : \Theta(g)(x) = \begin{cases} g(0), & \text{if } x = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6.13}$$

We have $\Theta \in \mathrm{LInv}(\mathfrak{C}_{\mathrm{loc}}; \mathcal{S}_0)$, as, for all $g' \in \Theta(\mathcal{S}_0)$, $g'(0)$ contains all the information to compute the pre-image of $g'$. Clearly, $\Theta(\mathcal{S}_0) \subseteq \mathcal{S}_1$.

By the one-to-one **ORT**,[44] there is a (different from above) one-to-one $s \in \mathcal{R}$ such that

$$\forall e, x : \varphi_{s(e)}(x) = \begin{cases} s(e), & \text{if } x = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6.14}$$

Now we get $\mathcal{S}_1 \leq^{\mathfrak{C}_{\mathrm{loc}}} \mathcal{S}_0$ by letting (a different) $\Theta$ be such that

$$\forall g \in \mathcal{P}, x : \Theta(g)(x) = \begin{cases} s(g(0)), & \text{if } x = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6.15}$$

[43]Note that using the one-to-one parametric recursion theorem suffices.
[44]Note that, again, using the one-to-one parametric recursion theorem suffices.

We have $\Theta \in \mathrm{LInv}(\mathfrak{C}_{\mathrm{loc}}; \mathcal{S}_1)$, as, for all $g' \in \Theta(\mathcal{S}_1)$, $g'(0)$ contains all the information to compute the pre-image of $g'$. Clearly, $\Theta(\mathcal{S}_1) \subseteq \mathcal{S}_0$. ■

Next we analyse a set of self-describing functions that was used in [11] to show the separation of **GBc** and (a stronger version of) **GEx**. Theorem 6.4 shows that this set necessarily shows the separation of **GEx** and **GBc**, if any set does.

**Theorem 6.4.** *Let* $\mathcal{S}_0 = \{g \in \mathcal{R} \mid \forall^\infty x : \varphi_{g(x)} = g\}$. *Then* $\mathcal{S}_0$ $\mathfrak{C}_{\mathrm{loc}}^{\mathbf{LL}}$*-epitomizes* **GBc**. *However,* $\mathcal{S}_0$ *does not* $\mathfrak{C}_{\mathrm{ew}}$*-epitomize* **GBc**, *as* $\mathcal{S}_{\mathrm{FinSup}} \not\leq^{\mathfrak{C}_{\mathrm{ew}}} \mathcal{S}_0$.[45]

*Proof.* We show $\mathcal{S}_0$ to be $\leq^{\mathfrak{C}_{\mathrm{loc}}^{\mathbf{LL}}}$-complete for **GBc** and apply theorem 3.7. Let $\mathcal{S} \in \mathbf{GBc}$ as witnessed by $h \in \mathbf{LinF}$ (without loss of generality, we assume $h$ to be linear time computable).

By linlin **ORT**, there is a linlin $s \in \mathbf{LL}$ such that

$$\forall e, x, y : \varphi_{s(e,x)}(y) = s(h(\varphi_e[y+1]), \varphi_e[y+1]). \tag{6.16}$$

Let $\Theta$ be such that

$$\forall g \in \mathcal{P}, x : \Theta(g)(x) = s(h(g[x+1]), g[x+1]). \tag{6.17}$$

Clearly, $\Theta \in \mathrm{LInv}(\mathfrak{C}_{\mathrm{loc}}^{\mathbf{LL}})$. We show $\Theta(\mathcal{S}) \subseteq \mathcal{S}_0$. Let $g \in \mathcal{S}$. We have, for all but finitely many $x$ and all $y$,

$$\varphi_{\Theta(g)(x)}(y) \underset{(6.17)}{=} \varphi_{s(h(g[x+1]),g[x+1])}(y) \tag{6.18}$$

$$\underset{(6.16)}{=} s(h(\varphi_{h(g[x+1])}[y+1]), \varphi_{h(g[x+1])}[y+1]) \tag{6.19}$$

$$\underset{\forall^\infty x}{=} s(h(g[y+1]), g[y+1]) \tag{6.20}$$

and

$$\underset{(6.17)}{=} \Theta(g)(y). \tag{6.21}$$

Thus, $\forall^\infty x : \varphi_{\Theta(g)(x)} = \Theta(g)$. Hence, $\Theta(\mathcal{S}) \subseteq \mathcal{S}_0$. This finishes the proof of the first assertion.

We show $\mathcal{S}_{\mathrm{FinSup}} \not\leq^{\mathfrak{C}_{\mathrm{ew}}} \mathcal{S}_0$. Let $\Theta \in \mathrm{LInv}(\mathfrak{C}_{\mathrm{ew}}; \mathcal{S}_{\mathrm{FinSup}})$ and suppose, by way of contradiction, $\Theta(\mathcal{S}_{\mathrm{FinSup}}) \subseteq \mathcal{S}_0$. Let $g, g' \in \mathcal{S}_{\mathrm{FinSup}}$ with $g \neq g'$. We have that $\Theta(g)$ and $\Theta(g')$ are finite variants of one another; thus, for large enough $x$, $\Theta(g)(x) = \Theta(g')(x)$ is a program for both $\Theta(g)$ and $\Theta(g')$. Therefore, $\Theta(g) = \Theta(g')$. Hence, $\Theta$ is not one-to-one on $\mathcal{S}_{\mathrm{FinSup}}$ and thus not left-invertible on $\mathcal{S}_{\mathrm{FinSup}}$. ■

Theorem 6.6 gives examples for *self-learning* sets of functions, which turn out to be very strong epitomizers. In order to define these sets, we need the following notions of *computable robustness* and *data normality*.

**Definition 6.5.** 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 $\Psi \in \mathfrak{C}_{\mathrm{eff}}$ such that, for all $h, e \in \mathcal{P}$, $e \circ I(h) \subseteq I(\Psi(h, e))$.

[45]This uses the fundamental epitomization theorem (theorem 3.7).

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

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

$$\forall h \in \mathcal{P}\, \forall g \in \mathcal{R}\, \forall i : \beta_I(h,g)(i) = h(f_I(g[i], \beta_I(h,g)[i])).^{46} \tag{6.22}$$

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

$$\forall h \in \mathcal{P}\, \forall g \in \mathcal{R}\, \forall i : d_I(f_I(g[i], \beta_I(h,g)[i])) = \begin{cases} ?, & \text{if } i = 0, \\ g(i-1), & \text{otherwise.}^{47} \end{cases} \tag{6.23}$$

(c) There is an effective operator $\hat{\Psi} \in \mathfrak{C}_{\text{eff}}$ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I(\hat{\Psi}(h))$ and $\hat{\Psi}(h)(f_I(\emptyset,\emptyset)) = ?$.[48]

**Theorem 6.6.** *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*

$$\forall x : h_0(x) = \begin{cases} ?, & \text{if } d(x) = ?, \\ \varphi_{d(x)}(x), & \text{otherwise.} \end{cases} \tag{6.24}$$

*Further, let $\mathcal{S}_0 = I(h_0)$.*[49] *Then $\mathcal{S}_0$ $\mathfrak{C}_{\text{ew}}^{\mathbf{LL}}$-epitomizes $I$.*

*Proof.* We show $\mathcal{S}_0$ to be $\leq^{\mathfrak{C}_{\text{ew}}^{\mathbf{LL}}}$-complete for $\mathfrak{S}(I)$ and apply theorem 3.7. Let $\mathcal{S} \in \mathfrak{S}(I)$ as witnessed by $h \in \mathcal{P}$.

Let $\Psi$ be as given by $I$ being computably element-wise robust. Let $\hat{\Psi}$ be as given by (c) of $I$ being data normal. By linlin **ORT**, there is a strictly monotonic increasing $e \in \mathbf{LL}$ such that

$$\forall w, x : \varphi_{e(w)}(x) = \hat{\Psi}(\Psi(h,e))(x). \tag{6.25}$$

It now suffices to show that $e \circ \mathcal{S} \subseteq \mathcal{S}_0$, as $(\lambda g\,.\, e \circ g) \in \text{LInv}(\mathfrak{C}_{\text{ew}}^{\mathbf{LL}})$. Suppose $I = (\mathcal{P}, \beta, \delta)$.

Let

$$g \in (e \circ \mathcal{S}). \tag{6.26}$$

We show, by induction on $i$,

$$\forall i : \beta(h_0, g)(i) = \beta(\hat{\Psi}(\Psi(h,e)), g)(i). \tag{6.27}$$

[46]Intuitively, the $i$th conjecture of $h$ on $g$ depends only on some information (as specified by $f_I$) about the first $i$ data points and conjectures.

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

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

[49]Note that $\mathcal{S}_0$ is a *self-learning set*. Roughly, the learner ($h_0$ in theorem 6.6) defining such a set ($\mathcal{S}_0$ in theorem 6.6) just *runs* each input datum as coding a program to determine its corresponding conjecture to output [19,21]. Note that $h_0$ is not total and hence not, for example, polynomial time computable. As noted in §1, [20] introduce a new tool (HORT) to begin handling some complexity-bounded examples of total learners.

This is clear for $i=0$ (both sides of the equality are ?). Let $i>0$, and suppose (inductively) $\beta(h_0,g)[i]=\beta(\hat{\Psi}(\Psi(h,e)),g)[i]$. Let

$$x=f(g[i],\beta(h_0,g)[i])\underset{\text{IH}}{=}f(g[i],\beta(\hat{\Psi}(\Psi(h,e)),g)[i]). \tag{6.28}$$

Note that $d(x)\underset{(6.23)}{=}g(i-1)\underset{(6.26)}{\in}\text{range}(e)$. We have

$$\beta(h_0,g)(i)\underset{(6.22)\ \&\ (6.28)}{=}h_0(x) \tag{6.29}$$

$$\underset{(6.24)}{=}\varphi_{d(x)}(x) \tag{6.30}$$

$$\underset{(6.25)}{=}\hat{\Psi}(\Psi(h,e))(x) \tag{6.31}$$

$$\underset{(6.28)}{=}\hat{\Psi}(\Psi(h,e))(f(g[i],\beta(\hat{\Psi}(\Psi(h,e)),g)[i])) \tag{6.32}$$

and

$$\underset{(6.22)}{=}\beta(\hat{\Psi}(\Psi(h,e)),g)(i). \tag{6.33}$$

This concludes the induction proof of (6.27). Thus, $h_0$ on any function $g\in(e\circ\mathcal{S})$ makes the same conjectures as $\hat{\Psi}(\Psi(h,e))$ on $g$. By the choice of $\Psi$ and $\hat{\Psi}$, $\hat{\Psi}(\Psi(h,e))$ is an $I$-learner for $(e\circ\mathcal{S})$; thus, $(e\circ\mathcal{S})\subseteq I(h_0)=\mathcal{S}_0$. ■

Theorem 6.6 provides epitomizing sets for the learning criteria $\beta\delta$ with $\beta$ and $\delta$ as explicitly given in this study, and many more. Furthermore, $\mathcal{S}_0$ of theorem 6.6 epitomizes with respect to all learning criteria that can be built from the example components given in this study (including the ones with learner admissibility restrictions), as they are all $\mathfrak{C}^{\mathbf{LL}}_{\text{ew}}$-robust. In particular, theorem 6.6 provides a superior epitomizer for **GBc** than the epitomizer of theorem 6.4.[50]

Timo Kötzing was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant NE 1182/5-1. The authors are grateful for the many suggestions of reviewers for the current version as well as for a previous (conference) version [26].

## References

1 Turing, A. M. 1936 On computable numbers with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **42**, 230–265. (doi:10.1112/plms/s2-42.1.230)
2 Davis, M. 1982 Why Gödel didn't have Church's thesis. *Inf. Control* **54**, 3–24.
3 Myhill, J. 1952 Some philosophical implications of mathematical logic. I. Three classes of ideas. *Rev. Metaphys.* **6**, 165–198.
4 Turing, A. M. 1950 Computing machinery and intelligence. *Mind* **59**, 433–460. (doi:10.1093/mind/LIX.236.433)
5 Turing, A. M. 1952 The chemical basis of morphogenesis. *Phil. Trans. R. Soc. Lond. B* **237**, 37–72. (doi:10.1098/rstb.1952.0012)

[50]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 6.4 above would witness that separation. Consider $I=\mathbf{TdBc}$. Clearly, $\mathcal{S}_{\text{FinSup}}$ separates **GBc** from **TdBc**. However, the epitomizer $\mathcal{S}_0$ of theorem 6.4 clearly *is* **TdBc**-learnable—disproving the present first author's old intuition. Nicely, though, from $\mathfrak{C}^{\mathbf{LL}}_{\text{ew}}$-robustness of **TdBc** (example 5.1), the epitomizer $\mathcal{S}_0$ of theorem 6.6 *does* separate **GBc** from **TdBc**.
An aside: it is argued in [42] that self-referential examples may portend more natural examples.

6 Gold, E. 1967 Language identification in the limit. *Inf. Control* **10**, 447–474. (doi:10.1016/S0019-9958(67)91165-5)
7 Case, J. 2007 Directions for computability theory beyond pure mathematical. In *Mathematical problems from applied logic II. Logics for the XXIst century* (eds D. Gabbay, S. Goncharov & M. Zakharyaschev). International Mathematical Series, vol. 5, pp. 53–98. New York: Springer.
8 Case, J. In press. Algorithmic scientific inference: within our computable expected reality. Invited talk and paper at *Proc. 3rd Int. Workshop on Physics and Computation, Nile, Egypt, 30 August–6 September 2010.* (To appear in *Int. J. Unconvent. Comput.*)
9 Blum, L. & Blum, M. 1975 Toward a mathematical theory of inductive inference. *Inf. Control* **28**, 125–155. (doi:10.1016/S0019-9958(75)90261-2)
10 Case, J. & Smith, C. 1978 Anomaly hierarchies of mechanized inductive inference. In *Proc. Symp. on the Theory of Computation*, pp. 314–319. New York, NY: ACM Press.
11 Case, J. & Smith, C. 1983 Comparison of identification criteria for machine inductive inference. *Theor. Comput. Sci.* **25**, 193–220. (doi:10.1016/0304-3975(83)90061-0)
12 Jain, S., Osherson, D., Royer, J. & Sharma, A. 1999 *Systems that learn: an introduction to learning theory*, 2nd edn. Cambridge, MA: MIT Press.
13 Freivalds, R., Kinber, E. & Smith, C. 1995 On the intrinsic complexity of learning. *Inf. Comput.* **123**, 64–71. (doi:10.1006/inco.1995.1158)
14 Jain, S., Kinber, E., Papazian, C., Smith, C. & Wiehagen, R. 2003 On the intrinsic complexity of learning recursive functions. *Inf. Comput.* **184**, 45–70. (doi:10.1016/S0890-5401(03)00059-2)
15 Jain, S. 2003 The intrinsic complexity of learning: a survey. *Fundam. Inf.* **57**, 17–37.
16 Rogers, H. 1967 *Theory of recursive functions and effective computability.* New York: McGraw-Hill. (Reprinted by MIT Press, Cambridge, MA, 1987.)
17 Fulk, M. A. 1990 Robust separations in inductive inference. In *Proc. 31st Annu. Symp. on Foundations of Computer Science, St. Louis, MO, 22–24 October 2010*, pp. 405–410. New York, NY: IEEE Press. (doi:10.1109/FSCS.1990.89560)
18 Fulk, M. 2011 Robust separations in inductive inference. *J. Symbolic Logic* **76**, 368–376. (doi:10.2178/jsl/1305810752)
19 Case, J. & Kötzing, T. 2010 Solutions to open questions for non-U-shaped learning with memory limitations. In *Proc. 21st Int. Conf. on Algorithmic Learning Theory, Canberra, Australia, 6–8 October 2010* (eds M. Hutter, F. Stephan, V. Vovk & T. Zeugmann). Lecture Notes in Artificial Intelligence, no. 6331, pp. 285–299. Berlin, Germany: Springer.
20 Case, J. & Kötzing, T. In press. Memory-limited non-U-shaped learning with solved open problems. *Theor. Comput. Sci.* (To appear in *Algorithmic Learning Theory* special issue.)
21 Case, J. & Kötzing, T. 2010 Strongly non-U-shaped learning results by general techniques. In *Proc. 23rd Annu. Conf. on Learning Theory, Haifa, Israel, 27–29 June 2010* (eds A. T. Kalai & M. Mohri), pp. 181–193. Madison, WI: Omnipress.
22 Kötzing, T. 2011 Iterative learning from positive data and counters. In *Proc. 22nd Int. Conf. on Algorithmic Learning Theory, Espoo, Finland, 5–7 October 2011* (eds J. Kivinen, C. Szepesvári, E. Ukkonen & T. Zeugmann). Lecture Notes in Artificial Intelligence, no. 6925, pp. 40–54. Berlin, Germany: Springer.
23 Case, J. 1999 Machine self-reference and consciousness. In *Proc. Abstracts of the Third Annual Meeting of the Association for the Scientific Study of Consciousness*, London, Ontario.
24 Case, J. 1974 Periodicity in generations of automata. *Math. Syst. Theory* **8**, 15–32. (doi:10.1007/BF01761704)
25 Case, J. 1994 Infinitary self-reference in learning theory. *J. Exp. Theor. Artif. Intell.* **6**, 3–16. (doi:10.1080/09528139408953778)
26 Case, J. & Kötzing, T. 2011 Measuring learning complexity with criteria epitomizers. In *Proc. 28th Int. Symp. on Theoretical Aspects of Computer Science, Dortmund, Germany, 10–12 March 2011*, pp. 320–331. Dagstuhl, Germany: Schloss Dagstuhl, Leibniz-Zentrum für Informatik.
27 Royer, J. & Case, J. 1994 *Subrecursive programming systems: complexity and succinctness.* Progress in Theoretical Computer Science. Boston, MA: Birkhäuser.
28 Blum, M. 1967 A machine independent theory of the complexity of recursive functions. *J. Appl. Math. Comput.* **14**, 322–336. (doi:10.1145/321386.321395)

29 Case, J. & Kötzing, T. 2008 Dynamic modeling in inductive inference. In *Proc. 19th Int. Conf. on Algorithmic Learning Theory, Budapest, Hungary, 13–16 October 2008* (eds Y. Freund, L. Györfi, G. Turán & T. Zeugmann). Lecture Notes in Artificial Intelligence, no. 5254, pp. 404–418. Berlin, Germany: Springer.
30 Kötzing, T. 2009 Abstraction and complexity in computational learning in the limit. PhD thesis, University of Delaware, Newark. See http://pqdtopen.proquest.com/#viewpdf?dispub=3373055.
31 Fulk, M. 1985 A study of inductive inference machines. PhD thesis, State University of New York, Buffalo.
32 Wiehagen, R. 1976 Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektron. Informationverarb. Kybernetik* **12**, 93–99.
33 Bārzdiņš, J. 1974 Inductive inference of automata, functions and programs. In *Proc. Int. Congress of Mathematicians*, pp. 455–560. (Transl. in Anosov D. V. *et al.* 1977 *Twenty lectures delivered at the International Congress of Mathematicians in Vancouver, 1974*. American Mathematical Society Translations, Series 2, no. 109, pp. 107–112. Providence, RI: American Mathematical Society.)
34 Angluin, D. 1980 Inductive inference of formal languages from positive data. *Inf. Control* **45**, 117–135. (doi:10.1016/S0019-9958(80)90285-5)
35 Bārzdiņš, J. 1974 Two theorems on the limiting synthesis of functions. In *Theory of algorithms and programs*, vol. 1, issue 210, pp. 82–88. Riga, Latvia: Latvian State University. [In Russian.]
36 Case, J. & Kötzing, T. 2008 Dynamically delayed postdictive completeness and consistency in learning. In *Proc. 19th Int. Conf. on Algorithmic Learning Theory, Budapest, Hungary, 13–16 October 2008* (eds Y. Freund, L. Györfi, G. Turán & T. Zeugmann). Lecture Notes in Artificial Intelligence, no. 5254, pp. 389–403. Berlin, Germany: Springer.
37 Akama, Y. & Zeugmann, T. 2008 Consistent and coherent learning with $\delta$-delay. *Inf. Comput.* **206**, 1362–1374. (doi:10.1016/j.ic.2008.06.005)
38 Pour-El, M. B. 1960 A comparison of five 'computable' operators. *Math. Logic Q.* **6**, 325–340. (doi:10.1002/malq.19600061510)
39 Freivalds, R., Kinber, E. & Wiehagen, R. 1984 Connections between identifying functionals, standardizing operations, and computable numberings. *Math. Logic Q.* **30**, 145–164. (doi:10.1002/malq.19840300904)
40 Pitt, L. 1989 Inductive inference, DFAs, and computational complexity. In *Proc. Int. Workshop on Analogical and Inductive Inference, Reinhardsbrunn Castle, Germany, October 1989* (ed. K. P. Jantke). Lecture Notes in Artificial Intelligence, no. 397, pp. 18–44. Berlin, Germany: Springer.
41 Barzdiņš, J. & Freivalds, R. 1972 On the prediction of general recursive functions. *Sov. Math. Dokl.* **13**, 1224–1228.
42 Case, J. 1999 The power of vacillation in language learning. *SIAM J. Comput.* **28**, 1941–1969. (doi:10.1137/S0097539793249694)