A Solution to Wiehagen's Thesis*

Timo Kötzing

Friedrich-Schiller-Universität Jena, Jena, Germany
timo.koetzing@uni-jena.de

Abstract

Wiehagen's *Thesis in Inductive Inference* (1991) essentially states that, for each learning criterion, learning can be done in a normalized, enumerative way. The thesis was not a formal statement and thus did not allow for a formal proof, but support was given by examples of a number of different learning criteria that can be learned enumeratively.

Building on recent formalizations of learning criteria, we are now able to formalize Wiehagen's Thesis. We prove the thesis for a wide range of learning criteria, including many popular criteria from the literature. We also show the limitations of the thesis by giving four learning criteria for which the thesis does not hold (and, in two cases, was probably not meant to hold). Beyond the original formulation of the thesis, we also prove stronger versions which allow for many corollaries relating to *strongly decisive* and *conservative* learning.

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases Algorithmic Learning Theory, Wiehagen's Thesis, Enumeration Learning

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.494

1 Introduction

In Gold-style learning [10] (also known as *inductive inference*) a *learner* tries to learn an infinite sequence, given more and more finite information about this sequence. For example, a learner h might be presented longer and longer initial segments of the sequence $g = 1, 4, 9, 16, \ldots$. After each new datum of g, h may output a description of a function (for example a Turing machine program computing that function) as its conjecture. h might output a program for the constantly-1 function after seeing the first element of this sequence g, and then, as soon as more data is available, a program for the squaring function. Many criteria for saying whether h is successful on g have been proposed in the literature. Gold, in his seminal paper [10], gave a first, simple learning criterion, later called *Ex-learning*[1], where a learner is successful iff it eventually stops changing its conjectures, and its final conjecture is a correct program (computing the input sequence).

Trivially, each single, describable sequence g has a suitable constant function as an Ex-learner (this learner constantly outputs a description for g). Thus, we are interested in sets of total computable functions $\mathcal{S}$ for which there is a single learner h learning each member of $\mathcal{S}$ (those sets $\mathcal{S}$ are then called *Ex-learnable*).

Gold [10] showed an important class of sets of functions to be Ex-learnable:[2] each

* We would like to thank Sandra Zilles for bringing Wiehagen's Thesis in connection with the approach of abstractly defining learning criteria, as well as the anonymous reviewers for their friendly and helpful suggestions.

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

[2] We let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers and we fix a coding for programs based on Turing machines letting, for any program (code) $p \in \mathbb{N}$, φ_p be the function computed by the Turing machine coded to p.

uniformly computable set of total functions is Ex-learnable; a set of functions $\mathcal{S}$ is uniformly computable iff there is a computable function e such that $\mathcal{S} = \{\varphi_{e(n)} \mid n \in \mathbb{N}\}$. The corresponding learner learns by enumeration: in every iteration, it finds the first index n such that $\varphi_{e(n)}$ is consistent with all known data, and outputs $e(n)$ as the conjecture.

However, it is well-known that there are sets which are not uniformly computable, yet Ex-learnable. Blum and Blum [6] gave the following example. Let e be a total computable listing of programs such that the predicate $\varphi_{e(n)}(x) = y$ is decidable in n, x and y. Crucially, some of the $\varphi_{e(n)}$ may be undefined on some arguments; these functions are not required to be learned, but the set of all the *total* functions enumerated is Ex-learnable. This uses the same strategy as for uniformly computable sets of functions, but this learning already goes *beyond enumeration* of all and only the learned functions, as there are sets which are so learnable, but not uniformly computable. The price is that the learner may give intermediate conjectures $e(n)$ which are programs for partial functions; this is necessarily so, as noted in [9].

As already shown by Wiehagen [16], there are Ex-learnable sets of functions that cannot be learned while always having a hypothesis that is consistent with the known data. Thus, the above strategy for learning employed by Blum and Blum [6] is not applicable for all learning tasks. In [17, 18] Wiehagen was looking for whether there is a more general strategy which also enumerates a list of candidate conjectures and is applicable to *all* Ex-learnable sets. He showed that this is indeed possible, giving an insightful characterization of Ex-learning.

A main focus of the research in inductive inference defines learning criteria that are different from (but usually similar in flavor to) Ex-learning. For example, *consistent* learning requires that each conjecture is consistent with the known data; *monotone* learning requires the sequence of conjectures to be monotone with respect to inclusion of the graphs of the computed functions. Wiehagen also gives characterizations for these learning criteria and more. Other researchers give similar characterizations; recent work in this area includes, for example, [1]. For any learning criterion I we are again interested in sets of total computable functions $\mathcal{S}$ for which there is a single learner h which learns every function in $\mathcal{S}$ in the sense specified by I; we call such $\mathcal{S}$ *I-learnable.*

Wiehagen was inspired by his work to conjecture a general structure of learning, as stated in his *Thesis in Inductive Inference* [18], which we rephrase in the language of this paper:

> Let $\mathcal{I}$ be any learning criterion. Then for any $\mathcal{I}$-learnable class $\mathcal{S}$, an enumeration of programs e can be constructed such that $\mathcal{S}$ is $\mathcal{I}$-learnable with respect to e by an enumerative learner.

Note that [18] called a learning criterion an “inference type” and a learner an “inference strategy”. About his thesis, Wiehagen [18] wrote that “We do not exclude that one nice day a formal proof of this thesis will be presented. This would require ‘only’ to formalize the notions of ‘inference type’ and ‘enumerative inference strategy’ which does not seem to be hopeless. But up to this moment we prefer ‘verifying’ our thesis analogously as it has been done with ‘verifying’ Church’s thesis, namely by formally proving it for ‘real’, reasonable, distinct inference types.”

Recently, the notion of a learning criterion was formalized in [13] (see Section 2.1 for the formal notions relevant to this paper). Our first contribution in this paper is a formalization of “enumeration learner” in Definition 2. It is in the nature of the very general thesis that any formalization may be too broad in some respects and too narrow in other. For example, our formalizations exclude some learning criteria, such as finite learning, learning by non-total learners, and criteria featuring global restrictions on the learner. However, for the scope of our definitions, we already get very strong and insightful results in this paper.

In Theorem 3 we discuss four different learning criteria in which the thesis does *not* hold. The first one is *prediction*, which attaches a totally different meaning to the "conjectures" than Ex-learning (the thesis was probably never meant to hold for such learning criteria). The second criterion involves mandatory oscillation between (correct) conjectures, which is in immediate contradiction to enumerative learning. The third learning criterion is *transductive learning*, where the learner has very little information in each iteration. The fourth is learning in a non-standard hypothesis space. The last two learning criteria do not contradict enumerative learning directly, but still demand too much for learning by enumeration.

In Section 4 we show that there is a broad core of learning criteria for which Wiehagen's Thesis holds. For this we introduce the notion of a *pseudo-semantic restriction*, where only the semantics of conjectures and possibly the occurrence of mind changes matter, but not other parts of their syntax. Theorem 10 shows that Wiehagen's Thesis holds in the case of *full information learning* (like in Ex-learning given above, where the learner only gets more information in each iteration) when all restrictions are pseudo-semantic, and in Theorem 16 we see that the same holds in the case of *iterative learning* (a learning model in which a learner has a restricted memory). Note that these two theorems already cover a very wide range of learning criteria from the literature, including all given by Wiehagen [18].

Finally, going beyond the scope of Wiehagen's Thesis, we show that we can assume the enumeration e of programs to be *semantically 1-1* (each $e(n)$ codes for a different function) if we assume a little bit more about the learning criteria, namely that their restrictions allow for *patching* and *erasing* (see Definition 11). This is formally shown in Theorem 13 (for the case of full information learning) and in Theorem 17 (for the case of iterative learning). Example criteria to which these theorems apply include Ex-learning, as well as consistent and monotone learning. Wiehagen [18] already pointed out in special cases that one can get such semantically 1-1 enumerations. From these results on learning with a semantically 1-1 enumeration we can derive corollaries to conclude that the learning criteria, to which the theorems apply, allow for *strongly decisive* and *conservative* learning (see Definition 1); for example, for plain Ex-learning, this proves (a stronger version of) a result from [15] (which showed that Ex-learning can be done decisively). Note that all positive results are sufficient conditions for enumerative learnability; except for the (weak) condition given in Remark 9, we could not find interesting *necessary* conditions.

The benefits of this work are threefold. First, we address a long-open problem in its essential parts. Second, we derive results about (strongly) decisive and conservative learning in many different settings. Finally, we further develop general techniques to derive powerful theorems applicable to many different learning criteria, thanks to general notions such as "pseudo-semantic restriction".

Note that we omit a number of nontrivial proofs due to space constraints.

2 Mathematical Preliminaries

We fix any computable 1-1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$; Whenever we consider tuples of natural numbers as input to a function, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to code the tuples into a single natural number. We similarly fix a coding for finite sets and sequences, so that we can use those as input as well. We use $\emptyset$ to denote the empty sequence; for every non-empty sequence σ we let σ^- denote the sequence derived from σ by dropping the last listed element.

If a function f 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 $f(x)\downarrow = p$. For any total computable predicate P, we use $\mu x\,.\,P(x)$ to denote the minimal x such that $P(x)$ (undefined, if no such x exists). The special symbol ? is used as a possible hypothesis (meaning "no change of hypothesis").

Unintroduced notation for computability theory follows [14]. $\mathcal{P}$ and $\mathcal{R}$ denote, respectively, the set of all partial computable and the set of all computable functions (mapping $\mathbb{N} \to \mathbb{N}$). For any function $f : \mathbb{N} \to \mathbb{N}$ and all i, we use $f[i]$ to denote the sequence $f(0), \ldots, f(i-1)$ (undefined, if any one of these values is undefined).

We will use a number of basic computability-theoretic results in this paper. First, we fix a *padding function*, a 1-1 function pad $\in \mathcal{R}$ such that $\forall p, n, x : \varphi_{\mathrm{pad}(p,n)}(x) = \varphi_p(x)$. Intuitively, pad generates infinitely many syntactically different copies of the semantically same program. We require that pad is monotone increasing in both arguments. The *S-m-n Theorem* states that there is a 1-1 function $s \in \mathcal{R}$ such that $\forall p, n, x : \varphi_{s(p,n)}(x) = \varphi_p(n, x)$. Intuitively, s-m-n allows for "hard-coding" arguments to a program.

2.1 Learning Criteria

In this section we formally introduce our setting of learning in the limit and associated learning criteria. We follow [13] in its "building-blocks" approach for defining learning criteria. A *learner* is a partial computable function from $\mathbb{N}$ to $\mathbb{N} \cup \{?\}$. A *sequence generating operator* is a function β taking as arguments a function h (the learner) and a function g (the learnee) and that outputs a function p. We call p the *conjecture sequence* of h given g. Intuitively, β defines how a learner can interact with a given learnee to produce a sequence of conjectures.

The most important sequence generating operator is $\mathbf{G}$ (which stands for "Gold", who first studied it [10]), which gives the learner full information about the learning process so far; this corresponds to the examples of learning criteria given in the introduction. Formally, $\mathbf{G}$ is defined such that

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

We define two additional sequence generating operators $\mathbf{It}$ (*iterative learning*, [16]) and $\mathbf{Td}$ (*transductive learning*, [8]) as follows. For all learners h, learnees g and all i,

$$\mathbf{It}(h,g)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0;^{3} \\ h(\mathbf{It}(h,g)(i-1), i-1, g(i-1)), & \text{otherwise;} \end{cases}$$

$$\mathbf{Td}(h,g)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0; \\ \mathbf{Td}(h,g)(i-1), & \text{else, if } h(i-1, g(i-1)) = ?; \\ h(i-1, g(i-1)), & \text{otherwise.} \end{cases}$$

For both of iterative and transductive learning, the learner is presented with a new datum each turn (argument/value pair from the learnee in complete and argument-increasing order). Furthermore, in iterative learning, the learner has access to the previous conjecture, but not so in transductive learning; however, in transductive learning, the learner can *implicitly* take over the previous conjecture by outputting "?".

Successful learning requires the learner to observe certain restrictions, for example convergence to a correct index. These restrictions are formalized in our next definition. A

[3] $h(\emptyset)$ denotes the *initial conjecture* (based on no data) made by h.

sequence acceptance criterion is a predicate δ on a learning sequence and a learnee. The most important sequence acceptance criterion is denoted **Ex** (which stands for "Explanatory"), already studied by Gold [10]. The requirement is that the conjecture sequence converges (in the limit) to a correct hypothesis for the learnee (we met this requirement already in the introduction). Formally, for any programming system[4] ψ, we define $\mathbf{Ex}_\psi$ as a predicate such that

$$\mathbf{Ex}_\psi = \{(p,g) \in \mathcal{R}^2 \mid \exists n_0, q : \forall n \geq n_0 : p(n) = q \wedge \psi_q = g\}.$$

Standardly we use $\mathbf{Ex} = \mathbf{Ex}_\varphi$. We will meet many more sequence acceptance criteria below. We combine any two sequence acceptance criteria δ and δ' by intersecting them; we denote this by juxtaposition (for example, the sequence acceptance criteria given below are meant to be always used together with **Ex**).

For any set $\mathcal{C} \subseteq \mathcal{P}$ of possible learners, any sequence generating operator β and any sequence acceptance criterion δ, $(\mathcal{C}, \beta, \delta)$ (or, for short, $\mathcal{C}\beta\delta$) is a *learning criterion.* A learner $h \in \mathcal{C}$ *$\mathcal{C}\beta\delta$-learns* the set $\mathcal{C}\beta\delta(h) = \{g \in \mathcal{R} \mid \delta(\beta(h,g),g)\}$. A set $\mathcal{S} \subseteq \mathcal{R}$ of possible learnees is called *$\mathcal{C}\beta\delta$-learnable* iff there is a function $h \in \mathcal{C}$ which $\mathcal{C}\beta\delta$-learns all elements of $\mathcal{S}$ (possibly more). Abusing notation, we also use $\mathcal{C}\beta\delta$ to denote the set of all $\mathcal{C}\beta\delta$-learnable sets (learnable by some learner).

Next we define a number of further sequence acceptance criteria which are of interest for this paper.

► **Definition 1.** With **Cons** we denote the restriction of *consistent learning* [4, 6] (being correct on all known data); with **Conf** the restriction of *conformal learning* [17] (being correct or divergent on known data); with **Conv** we denote the restriction of *conservative learning* [2] (never abandoning a conjecture which is correct on all known data); with **Mon** we denote the restriction of *monotone learning* [12] (conjectures make all the outputs that previous conjectures made – monotonicity in the graphs); finally, with **PMon** we denote the restriction of *pseudo-monotone learning* [18] (conjectures make all the *correct* outputs that previous conjectures made). The following definitions formalize these restrictions.

$$\begin{aligned}
\mathbf{Conf} &= \{(p,g) \in \mathcal{R}^2 \mid \forall n \forall x < n : \varphi_{p(n)}(x)\downarrow \Rightarrow \varphi_{p(n)}(x) = g(x)\};\\
\mathbf{Cons} &= \{(p,g) \in \mathcal{R}^2 \mid \forall n \forall x < n : \varphi_{p(n)}(x) = g(x)\};\\
\mathbf{Conv} &= \{(p,g) \in \mathcal{R}^2 \mid \forall n : p(n) \neq p(n+1) \Rightarrow \exists x < n+1 : \varphi_{p(n)}(x) \neq g(x)\};\\
\mathbf{Mon} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \ \forall x : \varphi_{p(i)}(x)\downarrow \Rightarrow \varphi_{p(j)}(x)\downarrow = \varphi_{p(i)}(x)\};\\
\mathbf{PMon} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \ \forall x : \varphi_{p(i)}(x)\downarrow = g(x) \Rightarrow \varphi_{p(j)}(x)\downarrow = \varphi_{p(i)}(x)\}.
\end{aligned}$$

An example of a well-studied learning criterion is $\mathcal{R}$**GConsEx**, requiring convergence of the learner to a correct conjecture, as well as consistent conjectures along the way.

Furthermore, we are interested in a number of restrictions which disallow certain kinds of returning to abandoned conjectures. We say that a learner exhibits a *U-shape* when it first outputs a correct conjecture, abandons this, and then returns to a correct conjecture. We distinguish between *syntactic* U-shapes (returning to the syntactically same conjecture), *semantic* U-shapes (returning to the semantically same conjecture, after semantically abandoning it; note that we drop the qualifier "semantic" in this case) and *strong* U-shapes (outputting a semantically same conjecture after syntactically abandoning it; this is called strong, because it leads to the stronger restriction). Forbidding these kinds of U-shapes leads

[4] We call ψ a *programming system* iff, for all p, ψ_p is a computable function, and the function mapping any p and x to $\psi_p(x)$ is also (partial) computable.

to the respective non-U-shapedness restrictions **SynNU**, **NU** and **SNU**. If we consider forbidding returning to abandoned conjectures more generally, we get three corresponding restrictions of *decisiveness*. We give the formal definitions here.

$$\begin{aligned}
\mathbf{SynNU} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \leq k : (\varphi_{p(i)} = g \wedge p(i) = p(k)) \Rightarrow p(j) = p(i)\};\\
\mathbf{NU} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \leq k : \varphi_{p(i)} = g = \varphi_{p(k)} \Rightarrow \varphi_{p(j)} = \varphi_{p(i)}\};\\
\mathbf{SNU} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \leq k : \varphi_{p(i)} = g = \varphi_{p(k)} \Rightarrow p(j) = p(i)\};\\
\mathbf{SynDec} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \leq k : p(i) = p(k) \Rightarrow p(j) = p(i)\};\\
\mathbf{Dec} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \leq k : \varphi_{p(i)} = \varphi_{p(k)} \Rightarrow \varphi_{p(j)} = \varphi_{p(i)}\};\\
\mathbf{SDec} &= \{(p,g) \in \mathcal{R}^2 \mid \forall i \leq j \leq k : \varphi_{p(i)} = \varphi_{p(k)} \Rightarrow p(j) = p(i)\}.
\end{aligned}$$

Of these variations of disallowing returning to abandoned conjectures, mostly **NU** [3] and **Dec** [15] are well-studied, but also **SNU** [5, 18] drew some attention; however, almost all of this work was done for the case of learning of languages (with the exception of [15]).

Note that the literature knows many more learning criteria than those constructible from the parts given in this section (see the text book [11] or the survey [19] for an overview).

3 Learning by Enumeration

In this section we formally introduce our notions of *learning by enumeration* and derive some easy statements from these definitions. We start with the general definition of learning by enumeration.

▶ **Definition 2.** Let $\mathcal{I}$ be a learning criterion and let $\mathcal{S} \subseteq \mathcal{R}$ be $\mathcal{I}$-learnable by some learner $h \in \mathcal{R}$. We say that h *learns by enumeration* iff there is a 1-1 enumeration $e \in \mathcal{R}$ of possible conjectures such that, for each $g \in \mathcal{R}$, there is a monotonically non-decreasing function r such that $e \circ r$ is the conjecture sequence of h on g. We say that a learning criterion $\mathcal{I}$ *allows for learning by enumeration* iff each $\mathcal{I}$-learnable set is $\mathcal{I}$-learnable by a learner learning by enumeration. We call e the enumeration of conjectures.

Note that, since e is required to be 1-1 and r non-decreasing, in any learning sequence of an enumeration learner h, no once abandoned hypothesis will be returned to; such abandoned hypotheses we call *refuted*. This immediately gives the following remark.

▶ Remark. Let h learn by enumeration. Then h learns syntactically decisively. In particular, for any learning criterion $\mathcal{I}$ allowing for learning by enumeration, every $\mathcal{I}$-learnable set is $\mathcal{I}$-learnable by a syntactically decisive learner.

From the wealth of (theoretically possible) learning criteria we quickly see that there are learning criteria which do not allow for learning by enumeration. For example, the task of *prediction* is typically modeled by using the sequence acceptance criterion $\mathbf{M}$ (for *matching*) defined as $\{(p,g) \mid \exists n_0 \forall n \geq n_0 : p(n) = g(n)\}$; in this case, the output of the learner is interpreted as the prediction for the next element in the sequence, instead of as a program (we consider the learning criterion $\mathcal{R}\mathbf{GM}$). A relaxation of the strict convergence required by **Ex** is given by $\mathbf{Fex}_{\leq k}$ ($k > 0$), where a learner may oscillate between at most k different (but correct!) hypotheses in the limit; as a somewhat unnatural variant, we let $\mathbf{Fex}_{=k}$ require oscillation between *exactly* k different (and correct) hypotheses. With these definitions, we get the follwing theorem.

▶ **Theorem 3.** *The following learning criteria do* not *allow for learning by enumeration.*

1. $\mathcal{R}\mathbf{GM}$.
2. $\mathcal{R}\mathbf{GFex}_{=2}$.
3. $\mathcal{R}\mathbf{TdEx}$.
4. *For some programming systems* ψ, $\mathcal{R}\mathbf{GEx}_{\psi}$.

In fact, all these learning criteria do not even allow for syntactically decisive learning; in the case of all items except (3) not even for syntactically non-U-shaped learning.

At the side we remark that Theorem 3, (3) cannot be strengthened in the same way as the other items: $\mathcal{R}\mathbf{TdEx}$-learning *does* allow for strongly non-U-shaped learning, as the next theorem shows.

▶ **Theorem 4.** *We have that every* $\mathcal{R}\mathbf{TdEx}$*-learnable set is so learnable by a strongly non-U-shaped learner, i.e.*

$$\mathcal{R}\mathbf{TdSNUEx} = \mathcal{R}\mathbf{TdEx}.$$

We will see in Theorem 10 that many learning criteria allow for learning by enumeration because of a simple padding trick, by semantically (but not syntactically) repeating any relevant conjecture infinitely in the enumeration (see below for details). In the following definition we strengthen the definition of learning by enumeration by requiring the enumeration of hypothesis to never *semantically* repeat a hypothesis.

▶ **Definition 5.** A function $e \in \mathcal{R}$ is called *semantically 1-1* iff, for all i, j, $\varphi_{e(i)} = \varphi_{e(j)}$ implies $i = j$. That is (by taking the contrapositive), different pre-images under e not only give different images, but even *semantically* different images.

A learner h which learns by enumeration using some $e \in \mathcal{R}$ as the enumeration of conjectures is said to *learn by semantically 1-1 enumeration* iff e is semantically 1-1. For $\mathcal{I}$ a learning criterion, a set $\mathcal{S} \subseteq \mathcal{R}$ is said to be $\mathcal{I}$*-learnable by semantically 1-1 enumeration* iff there is an $\mathcal{I}$-learner h for $\mathcal{S}$ learning by semantically 1-1 enumeration. We say that a learning criterion $\mathcal{I}$ *allows for learning by semantically 1-1 enumeration* iff each $\mathcal{I}$-learnable set $\mathcal{S}$ is $\mathcal{I}$-learnable by semantically 1-1 enumeration.

Some of the power of learning by semantically 1-1 enumeration is shown in the following remark, strengthening the conclusion of Remark 3.

▶ Remark. Let h learn by semantically 1-1 enumeration. Then h learns strongly decisively. In particular, for any learning criterion $\mathcal{I}$ allowing for learning by semantically 1-1 enumeration, every $\mathcal{I}$-learnable set is $\mathcal{I}$-learnable by a strongly decisive learner.

4 The Power of Enumeration Learning

In this section we give our theorems confirming Wiehagen's Thesis for a wide range of learning criteria. First we look at the very important family of learning criteria which use $\mathbf{G}$ as their sequence generating operator (full information learning). Note that all examples given in [18] were from this family (but did not require total learners).

We start by giving a definition for enumerative learning in the $\mathbf{G}$-setting (Definition 6) and that of pseudo-semantic restrictions (Definition 8). After this we are ready for the first main theorem of the paper, Theorem 10, which shows that Wiehagen's Thesis holds for many learning criteria using $\mathbf{G}$ as their sequence generating operator. With Definition 11 we introduce *patching* and *erasing*, which will allow for us to give the second main theorem of the paper, Theorem 13, which shows that many learning criteria with $\mathbf{G}$ as their sequence generating operator even allow for semantically 1-1 enumeration.

At the end of this section, with Theorems 16 and 17 and Corollary 18 we show that all results carry over to **It** as sequence generating operator.

▶ **Definition 6.** A pair (R, e) where R is a total computable predicate over pairs of numbers and (finite) data sequences and $e \in \mathcal{R}$ is a 1-1 computable function, is called a **G**-*style enumeration by refutation pair* iff, for all i, σ, y, $R(i, \sigma)$ implies $R(i, \sigma y)$ (i.e., R is monotone in the second argument, any conjecture, once refuted, stays refuted) and, for all σ, there is i such that $R(i, \sigma)$. The associated *enumeration learner* $h_{(R,e)}$ is defined such that

$$\forall \sigma : h_{(R,e)}(\sigma) = e(\mu i \,.\, \neg R(i, \sigma)).$$

The following theorem is straightforward to verify.

▶ **Theorem 7.** *For every δ, if h is a learner $\mathcal{R}\mathbf{G}\delta$-learning by enumeration according to Definition 2 with some enumeration of hypotheses e, then there is some R such that $h = h_{(R,e)}$.*

In order to exclude the examples given in Theorem 3 we now make some definitions for learning criteria which allow for learning by enumeration. Intuitively, we focus our attention on learning criteria which consider all conjectures as φ-conjectures, and are only interested in syntactic properties as far as mind changes are concerned.

▶ **Definition 8.** For all $p \in \mathcal{R}$, we let

$$\begin{aligned} \mathrm{Sem}(p) &= \{p' \in \mathcal{R} \mid \forall i : \varphi_{p(i)} = \varphi_{p'(i)}\}; \\ \mathrm{Mc}(p) &= \{p' \in \mathcal{R} \mid \forall i : (p(i) = p(i+1) \Rightarrow p'(i) = p'(i+1))\}. \end{aligned}$$

A sequence acceptance criterion δ is said to be a *semantic restriction* iff, for all $(p, g) \in \delta$ and $p' \in \mathrm{Sem}(p)$, $(p', g) \in \delta$. A sequence acceptance criterion δ is said to be a *pseudo-semantic restriction* iff, for all $(p, g) \in \delta$ and $p' \in \mathrm{Sem}(p) \cap \mathrm{Mc}(p)$, $(p', g) \in \delta$.

Intuitively, semantic restrictions allow for arbitrarily changing the syntax of the conjectures, as long as the semantics stay the same. Pseudo-semantic restrictions further require that no additional mind changes are introduced this way.

▶ **Example 9.** Any intersection of two (pseudo-) semantic restrictions is a (pseudo-) semantic restriction. Example semantic restrictions include **Conf**, **Cons**, **Mon**, **PMon**, **NU**, **Dec**; pseudo-semantic restrictions include **Ex**, **Conv**, **SNU** and **SDec**. Many more learning criteria from the literature could be added to these lists.

Example sequence acceptance criteria which are *not* pseudo-semantic restrictions include **M** (prediction), **SynNU**, **SynDec** and several more from the literature.

With these definitions we can now formulate the first main theorem of the paper, confirming Wiehagen's Thesis for a large family of **G**-style learning criteria.

▶ **Theorem 10.** *Let δ be a pseudo-semantic restriction. Then $\mathcal{R}\mathbf{G}\delta$ allows for learning by enumeration.*

Proof. The proof is based on a "padding trick": we can safely refute any hypothesis as long as we make sure that a (syntactically different) copy of the refuted hypothesis is still available. Formally, let $\mathcal{S} \in \mathcal{R}\mathbf{G}\delta$, as witnessed by a learner $h \in \mathcal{R}$. We define a computable predicate R and $c, e \in \mathcal{R}$ such that

$$\begin{aligned} c(\emptyset) &= 0; \\ \forall \sigma \neq \emptyset : c(\sigma) &= \mu n \,.\, \langle h(\sigma^-), c(\sigma^-) \rangle \leq \langle h(\sigma), n \rangle; \\ \forall m, n : e(m, n) &= \mathrm{pad}(m, n);[5] \\ \forall i, \sigma : R(i, \sigma) &\Leftrightarrow \langle h(\sigma), c(\sigma) \rangle > i. \end{aligned}$$

Clearly, e is 1-1 and R is monotone in the second component (as c is monotone). Let $g \in \mathcal{S}$. Let $p = \mathbf{G}(h,g)$ be the conjecture sequence of h on g and $p' = \mathbf{G}(h_{(R,e)}, g)$ the conjecture sequence of $h_{(R,e)}$ on g. It remains to show that $p' \in \mathrm{Sem}(p) \cap \mathrm{Mc}(p)$. We start with $p' \in \mathrm{Sem}(p)$. For all j, x we have[6]

$$\begin{aligned}
\varphi(p'(j), x) &= \varphi(h_{(R,e)}(g[j]), x)\\
&= \varphi(e(\mu i\,\textbf{.}\,\neg R(i, g[j])), x)\\
&= \varphi(e(\mu i\,\textbf{.}\,\langle h(g[j]), c(g[j])\rangle \leq i), x)\\
&= \varphi(e(\langle h(g[j]), c(g[j])\rangle), x)\\
&= \varphi(\mathrm{pad}(h(g[j]), c(g[j])), x)\\
&= \varphi(h(g[j]), x)\\
&= \varphi(p(j), x).
\end{aligned}$$

Hence, $p' \in \mathrm{Sem}(p)$.

Suppose $j \in \mathbb{N}$ such that $h(g[j]) = h(g[j+1])$. Then, $c(g[j+1]) = c(g[j])$; hence, for all i, $R(i, g[j]) \Leftrightarrow R(i, g[j+1])$ (there are no new hypotheses rejected). Therefore, $h_{(R,e)}(g[j]) = h_{(R,e)}(g[j+1])$. This shows $p' \in \mathrm{Mc}(p)$. ◀

We are now interested in strengthening the conclusion of Theorem 10 by restricting the family of learning criteria under consideration. For this we introduce variations on the notion of a pseudo-semantic restriction.

▶ **Definition 11.** Let δ be a sequence acceptance criterion. We say that δ *allows for patching* iff, for all $(p,g) \in \delta$ and $p' \in \mathrm{Mc}(p)$ such that all conjectures of p' are just as the corresponding conjectures of p, only possibly corrected for some arguments (these corrections are called *patches*); we say δ *allows for monotone patching* if this holds for all $p' \in \mathrm{Mc}(p)$ which patch later conjectures in all the places that earlier conjectures are patched at. Formally, δ allows for monotone patching iff, for all $(p,g) \in \delta$ and all $p' \in \mathrm{Mc}(p)$ where there is $(A_n)_{n \in \mathbb{N}}$ such that

$$\begin{aligned}
\forall n < m: &\qquad A_n \subseteq A_m;\\
\forall n, x: &\qquad \varphi_{p'(n)}(x) = \begin{cases} \varphi_{p(n)}(x), & \text{if } x \notin A_n;\\ g(x), & \text{if } x \in A_n. \end{cases}
\end{aligned}$$

we have $(p', g) \in \delta$. If we drop the first requirement of monotonicity of $(A_n)_{n \in \mathbb{N}}$, we get the formal definition for δ allowing for patching.

We say that δ *allows for erasing* iff, for all $(p,g) \in \delta$ and $p' \in \mathrm{Mc}(p)$ such that all conjectures of p' are just as the corresponding conjectures of p, only possibly made divergent for some arguments for which no data is known (the arguments set to diverge are called *erased*); we say δ *allows for monotone erasing* if this holds for all $p' \in \mathrm{Mc}(p)$ which erase in later conjectures at at most the places that earlier conjectures were erased at (and only on unknown data). Finally, we say δ *allows for almost-monotone erasing* iff monotone erasing is violated only when the new conjecture corrects an earlier mistake; formally: δ allows for almost-monotone erasing iff for all $(p,g) \in \delta$ and $p' \in \mathrm{Mc}(p)$ we have $(p', g) \in \delta$ if there is a

[5] The function pad was defined in Section 2.
[6] For convenience we write, for all a, z, $\varphi(a,z)$ instead of $\varphi_a(z)$.

sequence $(A_n)_{n\in\mathbb{N}}$ of subsets of $\mathbb{N}$ such that the following hold.

$$\begin{aligned}
\forall n: &\quad A_n \cap \{0,\dots n-1\} = \emptyset;\\
\forall n,m: &\quad n \leq m \Rightarrow (A_m \subseteq A_n \vee \exists x : \varphi_{p(m)}(x) = g(x) \neq \varphi_{p(n)}(x){\downarrow});\\
\forall n,x: &\quad \varphi_{p'(n)}(x) = \begin{cases} \varphi_{p(n)}(x), & \text{if } x \notin A_n;\\ \uparrow, & \text{if } x \in A_n. \end{cases}
\end{aligned}$$

Intuitively, an almost-monotone erasing erases less and less except when the new conjecture corrects a *convergent* mistake, and only erases where no data is available.

For example, if some δ allows for (monotone) patching, then a $\mathcal{R}\mathbf{G}\delta$-learner can always patch all known data into the conjecture (up to the last mind change – otherwise $p' \in \mathrm{Mc}(p)$ will be violated). We use almost-monotone erasing in Theorem 13, where we need something more than just monotone erasing, and do not require full erasing power for the sake of generality. Note that any δ allowing for (monotone) patching or erasing is a pseudo-semantic restriction.

▶ **Example 12.** Any intersection of two sequence acceptance criteria allowing for (monotone) patching or erasing again allows for (monotone) patching or erasing, respectively; the same holds for almost-monotone erasing. Examples of sequence acceptance criteria allowing for patching and erasing are **Conf**, **Cons** and **Ex**; **Mon** and **PMon** allow for monotone patching and monotone erasing. **Mon**, but not **PMon**, allows for almost-monotone erasing.

With the definition of a patching and erasing we can now give another main theorem of the paper. The proof uses ideas from proofs in [18] (where, implicitly, special cases of this theorem have been proven), as well as from [7], which gives a general technique for avoiding U-shapes in language learning.

▶ **Theorem 13.** *Let δ allow for monotone patching and almost-monotone erasing. Then $\mathcal{R}\mathbf{G}\delta$ allows for learning by semantically 1-1 enumeration. Furthermore, there is an enumeration learner which learns conservatively.*

Proof. Let $\mathcal{S} \in \mathcal{R}\mathbf{G}\delta$ as witnessed by some learner $h \in \mathcal{R}$. As δ allows for monotone patching, we can assume, without loss of generality, that h patches each new conjecture with the known data at every mind change.

Let M be the set of all finite sequences σ such that either $\sigma = \emptyset$ or $h(\sigma^-) \neq h(\sigma)$. Note that M is a decidable set. We can assume, without loss of generality, that M is infinite; as otherwise we can introduce dummy members into M which will not invalidate the proof. Thus, there is a 1-1 total computable enumeration $(\tau_i)_{i\in\mathbb{N}}$ of all and only the elements in M respecting the order on finite sequences (i.e., for all i,j, if $\tau_i \subseteq \tau_j$, then $i \leq j$; in particular, $\tau_0 = \emptyset$). For all i, we let $z(i) = h(\tau_i)$ be the conjecture after the ith listed sequence and $n(i) = \mathrm{len}(\tau_i)$ the length of the ith sequence. We define e with s-m-n such that, for all i and x,

$$\varphi_{e(i)}(x) = \begin{cases} \varphi_{z(i)}(x), & \text{if } x < n(i) \text{ or, for all } y \text{ with } n(i) < y < x:\\ & \quad \varphi_{z(i)}(y){\downarrow} \text{ and } h(\varphi_{z(i)}[y+1]) = z(i);\\ \uparrow, & \text{otherwise.} \end{cases}$$

Note that, for all i, $\varphi_{z(i)}[n(i)]{\downarrow} = \tau_i$, as we assumed that h patches all known data into the new conjecture at each mind change. In particular, this shows that e is semantically 1-1. We define R as follows.

$$R(i,\sigma) \Leftrightarrow \tau_i \text{ and } \sigma \text{ are } \subseteq\text{-incomparable or } \tau_i \subset \sigma \text{ and } \exists \sigma' : \tau_i \subseteq \sigma' \subseteq \sigma \wedge h(\sigma') \neq h(\tau_i).$$

Note that R is total computable and monotone in its second argument. Intuitively, we always use the conjecture that h would have used, modified appropriately to ensure that the enumeration is semantically 1-1. Clearly, (R, e) is an $\mathbf{G}$-style enumeration by refutation pair. We show that $h_{(R,e)}$ $\mathcal{R}\mathbf{G}\delta$-learns $\mathcal{S}$.

Let $g \in \mathcal{S}$. Let $p = \mathbf{G}(h, g)$ be the conjecture sequence of h on g and $p' = \mathbf{G}(h_{(R,e)}, g)$ the conjecture sequence of $h_{(R,e)}$ on g. From the order of listing of the τ_i and the definition of R we get that, for all n, $p'(n) = e(i)$ for i such that τ_i is the $\subseteq$-maximal element of M with $\tau_i \subseteq g[n]$; this also gives that h made no mind change between τ_i and $g[n]$. Thus, we get $p' \in \mathrm{Mc}(p)$ and, for all n, $p(n)$ and $p'(n)$ are semantically equivalent apart from possibly erased arguments; let $(A_n)_{n\in\mathbb{N}}$ be the corresponding sequence of erased sets of arguments (which are thus exactly the arguments on which the corresponding conjecture $p'(n)$ is undefined). Let now $n < m$ be such that $A_m \not\subseteq A_n$. Without loss of generality, $n = 0$ or $p(n) \neq p(n-1)$ and $p(m) \neq p(m-1)$. Thus, $p'(n) = e(i)$ with $\tau_i = g[n]$ and $p'(m) = e(j)$ with $\tau_j = g[m]$. Then we get from patching that $\varphi_{p(m)}(m-1) = g(m-1)$. Furthermore, A_m contains only numbers $\geq m$, and since A_n is closed upwards and $A_m \not\subseteq A_n$, we get $m - 1 \notin A_n$. This shows that $\varphi_{p'(n)}(m-1) = \varphi_{e(i)}(m-1)$ converges; thus, it converges to the same value as $\varphi_{p(n)}$ on $m-1$. However, this value cannot equal $g(n)$, as this value leads to a mind change (this we get from $\tau_j \in M$), and any value leading to a mind change would be erased by the definition of e. ◀

We can see the deep power and versatility of Theorem 13 in connection with Remark 3 and the various examples of sequence acceptance criteria fulfilling the prerequisites of Theorem 13, which leads, for example, to the following corollary.

▶ **Corollary 14.** *The following learning criteria allow for learning strongly decisively and conservatively.* $\mathcal{R}\mathbf{GEx}$*;* $\mathcal{R}\mathbf{GConfEx}$*;* $\mathcal{R}\mathbf{GConsEx}$*;* $\mathcal{R}\mathbf{GMonEx}$*;* $\mathcal{R}\mathbf{GConsMonEx}$.

At the side we remark that Theorem 13 cannot be improved to apply also to pseudo-monotone learning, as the following Theorem shows.

▶ **Theorem 15.** *There is a* $\mathcal{R}\mathbf{GPMonEx}$*-learnable set of functions which cannot be so learned strongly non-U-shapedly.*

Finally, we show that analogous theorems can also be derived for iterative learning. Theorem 16 is analogous to Theorem 10, and Theorem 17 is analogous to Theorem 13; both proofs are also analogous, but different in some details.

▶ **Theorem 16.** *Let δ be a pseudo-semantic restriction. Then $\mathcal{R}\mathbf{It}\delta$ allows for learning by enumeration.*

▶ **Theorem 17.** *Let δ allow for monotone patching and almost-monotone erasing. Then $\mathcal{R}\mathbf{It}\delta$ allows for learning by semantically 1-1 enumeration. Furthermore, there is an enumeration learner which learns conservatively.*

Just as in the case of $\mathbf{G}$-style learning, were we got a powerful corollary (Corollary 14), we get the analogous corollary also for $\mathbf{It}$-style learning.

▶ **Corollary 18.** *The following learning criteria allow for learning strongly decisively and conservatively.* $\mathcal{R}\mathbf{ItEx}$*;* $\mathcal{R}\mathbf{ItConfEx}$*;* $\mathcal{R}\mathbf{ItConsEx}$*;* $\mathcal{R}\mathbf{ItMonEx}$*;* $\mathcal{R}\mathbf{ItConsMonEx}$.

References

1 Y. Akama and T. Zeugmann. Consistent and coherent learning with δ-delay. *Information and Computation*, 206:1362–1374, 2008.

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

3 G. Baliga, J. Case, W. Merkle, F. Stephan, and W. Wiehagen. When unlearning helps. *Information and Computation*, 206:694–709, 2008.

4 J. Bārzdiņš. Inductive inference of automata, functions and programs. In *Proc. of the International Congress of Mathematicians*, pages 455–560, 1974. English translation in, *American Mathematical Society Translations*: Series 2 109 (1977), pp. 107-112.

5 H.R. Beick. *Induktive Inferenz mit Höchster Inferenzgeschwindigkeit.* PhD thesis, Humboldt University of Berlin, 1984.

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

7 J. Case and T. Kötzing. Strongly non-U-shaped learning results by general techniques. In *Proc. of COLT (Conference on Learning Theory)*, pages 181–193, 2010.

8 J. Case and T. Kötzing. Learning secrets interactively. Dynamic modeling in inductive inference. *Information and Computation*, 220:60–73, 2012.

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

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

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

12 K. Jantke. Monotonic and non-monotonic inductive inference of functions and patterns. In J. Dix, K. Jantke, and P. Schmitt, editors, *Nonmonotonic and Inductive Logic*, volume 543 of *Lecture Notes in Computer Science*, pages 161–177. 1991.

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

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

15 G. Schäfer-Richter. *Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien.* PhD thesis, RWTH Aachen, 1984.

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

17 R. Wiehagen. *Zur Theorie der Algorithmischen Erkennung*, 1978. Dissertation B, Humboldt University of Berlin.

18 R. Wiehagen. A thesis in inductive inference. In *Proc. of the Workshop on Nonmonotonic and Inductive Logic*, pages 184–207, 1991.

19 Thomas Zeugmann and Sandra Zilles. Learning recursive functions: A survey. *Theoretical Computer Science*, 397:4–56, 2008.