# Learning and Consistency

Rolf Wiehagen
University of Kaiserslautern
Department of Computer Science
P.O. Box 3049
67653 Kaiserslautern
Germany
wiehagen@informatik.uni-kl.de

Thomas Zeugmann
Research Institute of
Fundamental Information Science
Kyushu University 33
Fukuoka 812
Japan
thomas@rifis.sci.kyushu-u.ac.jp

**Abstract**

In designing learning algorithms it seems quite reasonable to construct them in such a way that all data the algorithm already has obtained are correctly and completely reflected in the hypothesis the algorithm outputs on these data. However, this approach may totally fail. It may lead to the unsolvability of the learning problem, or it may exclude any efficient solution of it.
Therefore we study several types of consistent learning in recursion-theoretic inductive inference. We show that these types are not of universal power. We give "lower bounds" on this power. We characterize these types by some versions of decidability of consistency with respect to suitable "non-standard" spaces of hypotheses.
Then we investigate the problem of learning consistently in polynomial time. In particular, we present a natural learning problem and prove that it can be solved in polynomial time if and only if the algorithm is allowed to work inconsistently.

## 1. Introduction

The phenomenon of learning has attracted much attention of researchers in various fields. When dealing with learning computer scientists are mainly interested in studying the question whether or not learning problems may be solved algorithmically. Nowadays, algorithmic learning theory is a rapidly emerging science, cf. Angluin and Smith (1983, 1987), Osherson, Stob and Weinstein (1986). Nevertheless, despite the enormous progress having been made since the pioneering papers of Solomonoff (1965) and of Gold (1965, 1967), there are still many problems that deserve special attention. The global question we shall deal with may be posed as follows: Are all data of equal importance a learning algorithm is fed?

First we study this question in the setting of inductive inference. Then we ask whether the insight obtained may be important when one has to solve learning problems that are in a sense closer to potential applications. For that purpose we consider a domain that has recently attracted attention, namely the learnability of indexed families (cf., e.g., Zeugmann, Lange and Kapur, 1995, and the references therein). We consider the particular indexed family of all the pattern languages and show the superiority of inconsistent learning strategies over consistent ones for this particular family. Next, we want to explain all this in some more detail.

One main problem of algorithmic learning theory consists in synthesizing "global descriptions" for the objects to be learned from examples. Thus, one goal is the following. Let $f$ be any computable function from $\mathbb{N}$ into $\mathbb{N}$. Given more and more examples $f(0), f(1), ..., f(n), ...$ a learning strategy is required to produce a sequence of hypotheses $h_0, h_1, ..., h_n, ...$ the limit of which is a correct global description of the function $f$, i.e., a program that computes $f$. Since at any stage $n$ of this learning process the strategy knows exclusively the examples $f(0), f(1), ..., f(n)$, it seems reasonable to construct the hypothesis $h_n$ in a way such that for any $x \leq n$ the "hypothesis function" $g$ described by $h_n$ is defined and computes the value $f(x)$. Such a hypothesis is called *consistent.* In other words, a hypothesis is consistent if and only if all information obtained so far about the unknown object is completely and correctly encoded in this hypothesis. Otherwise, a hypothesis is said to be *inconsistent.* Consequently, if the hypothesis $h_n$ above is inconsistent, then there must be an $x \leq n$ such that $g(x) \neq f(x)$. Note that there are two possible reasons for $g$ to differ from $f$ on argument $x$; namely, $g(x)$ may be not defined, or the value $g(x)$ is defined and does not equal $f(x)$. Hence, if a hypothesis is inconsistent then it is not only wrong but it is wrong on an argument for which the learning strategy does already know the correct value. At first glance we are tempted to totally exclude strategies producing inconsistent hypotheses from our considerations. It might seem that *consistent strategies*, i.e., strategies that produce always consistent hypotheses, are the only reasonable learning devices.

Surprisingly enough this is a misleading impression. As it turns out, in a sense learning seems to be *the art of knowing what to overlook.* Barzdin (1974a) first announced that there are classes of recursive functions that can be learned in the limit but only by strategies working inconsistently. This result directly yields the following questions:

(1) Why does it make sense to output inconsistent hypotheses?
(2) What kind of data, if any, the strategy should overlook?

As we shall see below the first question finds its preliminary answer in the fact that, in general, there is no algorithm detecting whether or not a hypothesis is consistent. Consequently, in general, a strategy has no chance to effectively verify the consistency of its previous guess with the new data it has been fed. On the other hand, a strategy cannot overcome this drawback by simply searching any consistent hypothesis, since it has to converge in the limit, too. Therefore, in order to be really successful in the limit a strategy cannot take care whether all the information it is provided with is actually correctly reflected by its current hypotheses. Answering the second question is more complicated. However, an intuitively satisfying answer is provided by a characterization of identification in the limit in terms of computable numberings (cf. Wiehagen (1978), Theorem 8). This theorem actually states that a class $U$ of recursive functions can be learned in the limit iff there are a space of hypotheses containing for each function at least one program, and a computable "discrimination" function $d$ such that for any two programs $i$ and $j$ the value $d(i,j)$ is an upper bound for an argument $x$ on which program $i$ behaves differently than program $j$ does. The key observation used in constructing the strategy that infers any function from $U$ is the following. Let $i$ be the strategy's last guess and let $f(0),...,f(n)$ be the data now fed to it. If the strategy finds a program $j$ such that for all inputs $x \leq d(i,j)$ its output equals $f(x)$, then program $i$ *cannot be a correct one* for function $f$. Then the strategy changes its mind from $i$ to $i+1$. In other words, the strategy uses the data up to $d(i,j)$ for the purpose to find a *proof* for the possible *incorrectness* of its actual hypothesis $i$ via some global property the space of all hypotheses possesses. Just for achieving this reasonable purpose the learning strategy may ignore all the data $f(x)$ where $d(i,j) < x \leq n$, thus trading off consistency of the new hypothesis (unachievable, in general, anyway) for an incorrectness proof concerning the former hypothesis.

Summarizing the above discussion we see that the main reason for the superiority of inconsistent strategies in the setting of inductive inference of recursive functions is caused by the undecidability of consistency. As we shall see further, consistent learning is also sensitive with respect to additional requirements that a learning strategy might be supposed to fulfil, e.g. the order in which the input data are provided or the domain on which a strategy is required to behave consistently.

However, it remained open whether this inconsistency phenomenon is of epistemological importance, only, but of almost no practical relevance. Dealing with the latter problem requires a different approach. It might be well conceivable that consistency is often decidable if one restricts itself to learning problems that are of interest with respect to potential applications. Consequently, the superiority of inconsistent strategies in this setting, if any, can only be established in terms of complexity theory. What we present in the sequel is a partial solution to this problem. As it turned out, there are natural learning problems having the following property: Though consistency is decidable in the corresponding domain, these learning problems can be solved by a polynomial-time strategy if and only if the strategy may work inconsistently.

Hence the inconsistency phenomenon does survive also in domains where consistency *is* decidable. Moreover, the reason for the eventual superiority of inconsistent strategies is in both settings in some sense the same. In both cases the learning algorithms cannot handle the problem of finding/constructing consistent hypotheses. On the one hand, this inability has been caused by the provable absence of any algorithm solving it, while, on the other hand, this problem may be computationally intractable.

As far as we know the result above is the first one formally proving the existence of learning problems that cannot be solved in polynomial time by any consistent strategy in a setting where consistency is decidable. Moreover, in our opinion it strongly recommends to take *inconsistent* learning strategies seriously into consideration. This requires both, the elaboration of "intelligent" inconsistent techniques as well as finding criteria with the help of which one can decide whether or not fast consistent strategies are unavailable. The inconsistency technique we have used just consists in ignoring data unnecessary for the strategy in order to fulfil its learning task. Of course, this might not be the only such technique.

The paper is structured as follows. Section 2 presents notation and definitions. Section 3 deals with the inconsistency phenomenon in the setting of inductive inference of recursive functions. The problem whether the inconsistency phenomenon is of any relevance in the world of polynomial time learning is affirmatively exemplified in Section 4. In Section 5 we discuss the results obtained and present open problems. All references are given in Section 6. Note that Wiehagen and Zeugmann (1992) and Wiehagen (1992) dealt already with the inconsistency phenomenon in inductive inference and in exact learning in polynomial time. Furthermore, part of the present paper has been published in Wiehagen and Zeugmann (1994).

## 2. Preliminaries

Unspecified notations follow Rogers (1967). $\mathbb{N} = \{0, 1, 2, ...\}$ denotes the set of all natural numbers. The set of all finite sequences of natural numbers is denoted by $\mathbb{N}^*$. The classes of all partial recursive and recursive functions of one, and two arguments over $\mathbb{N}$ are denoted by $P$, $P^2$, $R$, and $R^2$, respectively. $R_{0,1}$ denotes the set of all $0-1$ valued recursive functions. Sometimes it will be suitable to identify a recursive function with the sequence of its values, e.g., let $\alpha = (a_0, ..., a_k) \in \mathbb{N}^*$, $j \in \mathbb{N}$, and $p \in R_{0,1}$; then we write $\alpha jp$ to denote the function $f$ for which $f(x) = a_x$, if $x \leq k$, $f(k+1) = j$, and $f(x) = p(x-k-2)$, if $x \geq k+2$. Furthermore, let $g \in P$ and $\alpha \in \mathbb{N}^*$; we write $\alpha \sqsubset g$ iff $\alpha$ is a prefix of the sequence of values associated with $g$, i.e., for any $x \leq k$, $g(x)$ is defined and $g(x) = a_x$. If $U \subseteq R$, then we denote by $[U]$ the set of all prefixes of functions from $U$.

Any function $\psi \in P^2$ is called a numbering. Moreover, let $\psi \in P^2$, then we write $\psi_i$ instead of $\lambda x \psi(i, x)$ and set $P_\psi = \{\psi_i \mid i \in \mathbb{N}\}$ as well as $R_\psi = P_\psi \cap R$. Consequently, if $f \in P_\psi$, then there is a number $i$ such that $f = \psi_i$. If $f \in P$ and $i \in \mathbb{N}$ are such that $\psi_i = f$, then $i$ is called a $\psi$–program for $f$. A numbering $\varphi \in P^2$ is called a Gödel numbering (cf. Rogers (1967)) iff $P_\varphi = P$, and for any numbering $\psi \in P^2$, there is a $c \in R$ such that $\psi_i = \varphi_{c(i)}$ for all $i \in \mathbb{N}$. *Göd* denotes the set of all Gödel numberings.

Using a fixed encoding $\langle ... \rangle$ of $\mathbb{N}^*$ onto $\mathbb{N}$ we write $f^n$ instead of $\langle (f(0), ..., f(n)) \rangle$, for any $n \in \mathbb{N}$, $f \in R$. Furthermore, the set of all permutations of $\mathbb{N}$ is denoted by $\Pi(\mathbb{N})$. Any element $X \in \Pi(\mathbb{N})$ can be represented by a unique sequence $(x_n)_{n\in\mathbb{N}}$ that contains each natural number precisely ones. Let $X \in \Pi(\mathbb{N})$, $f \in P$ and $n \in \mathbb{N}$. Then we write $f^{X,n}$ instead of $\langle (x_0, f(x_0), ..., x_n, f(x_n)) \rangle$ provided $f(x_k)$ is defined for all $k \leq n$. Finally, a sequence $(j_n)_{j\in\mathbb{N}}$ of natural numbers is said to *converge* to the number $j$ iff all but finitely many numbers of it are equal to $j$. A sequence $(j_n)_{j\in\mathbb{N}}$ of natural numbers is said to *finitely converge* to the number $j$ iff it converges in the limit to $j$ and for all $n \in \mathbb{N}$, $j_n = j_{n+1}$ implies $j_k = j$ for all $k \geq n$. Now we are ready to define some concepts of learning.

**Definition 1. (Gold, 1965)** *Let $U \subseteq R$ and let $\psi \in P^2$. The class $U$ is said to be learnable in the limit with respect to $\psi$ iff there is a strategy $S \in P$ such that for each function $f \in U$,*

(1) *for all $n \in \mathbb{N}$, $S(f^n)$ is defined,*

(2) *there is a $j \in \mathbb{N}$ such that $\psi_j = f$ and the sequence $(S(f^n))_{n\in\mathbb{N}}$ converges to $j$.*

*If $U$ is learnable in the limit with respect to $\psi$ by a strategy $S$, we write $U \in LIM_\psi(S)$. Let $LIM_\psi = \{U \mid U$ is learnable in the limit w.r.t. $\psi\}$, and let $LIM = \bigcup_{\psi\in P^2} LIM_\psi$.*

Some remarks are mandatory here. Let us start with the semantics of the hypotheses produced by a strategy $S$. If $S$ is defined on input $f^n$, then we always interpret the number $S(f^n)$ as a $\psi$–number. This convention is adopted to all the definitions below. Furthermore, note that $LIM_\varphi = LIM$ for any Gödel numbering $\varphi$. In the above definition $LIM$ stands for "limit." Moreover, in accordance with the definition of convergence, only finitely many data of the graph of a function $f$ were available to the strategy $S$ up to the unknown point of convergence. Therefore, some form of learning must have taken place. Thus, the use of the term "learn" in the above definition is indeed justified. As we have mentioned, in general it is not decidable whether or not a strategy has already converged when successively fed some graph of a function. With the next definition we consider a special case where it has to be decidable whether or not a strategy has already learned its input function.

**Definition 2. (Gold, 1967; Trakhtenbrot and Barzdin, 1970)** *Let $U \subseteq R$ and let $\psi \in P^2$. The class $U$ is said to be finitely learnable with respect to $\psi$ iff there is a strategy $S \in P$ such that for any function $f \in U$,*

(1) *for all $n \in \mathbb{N}$, $S(f^n)$ is defined,*

(2) *there is a $j \in \mathbb{N}$ such that $\psi_j = f$ and the sequence $(S(f^n))_{n\in\mathbb{N}}$ finitely converges to $j$.*

*If $U$ is finitely learnable with respect to $\psi$ by a strategy $S$, we write $U \in FIN_\psi(S)$. Let $FIN_\psi = \{U \mid U$ is finitely learnable w.r.t. $\psi\}$, and let $FIN = \bigcup_{\psi\in P^2} FIN_\psi$.*

Next we formally define different models of consistent learning.

**Definition 3. (Barzdin, 1974a)** *Let $U \subseteq R$ and let $\psi \in P^2$. The class $U$ is called consistently learnable in the limit with respect to $\psi$ iff there is a strategy $S \in P$ such that*

(1) *$U \in LIM_\psi(S)$,*

(2) *$\psi_{S(f^n)}(x) = f(x)$ for all $f \in U$, $n \in \mathbb{N}$ and $x \leq n$.*

*$CONS_\psi(S)$, $CONS_\psi$ and $CONS$ are defined analogously as above.*

Intuitively, a consistent strategy does correctly reflect all the data it has already seen. If a strategy does not always work consistently, we call it inconsistent.

Next, we add a requirement to the definition of the learning type $CONS_\psi$ that is often implicitly assumed in applications, namely, that the strategy is defined on every input, cf. Michalski et al. (1984, 1986).

**Definition 4. (Jantke and Beick, 1981)** *Let $U \subseteq R$ and let $\psi \in P^2$. The class $U$ is called $R$–consistently learnable in the limit with respect to $\psi$ iff there is a strategy $S \in R$ such that $U \in CONS_\psi(S)$.*

*$R{-}CONS_\psi(S), R{-}CONS_\psi$ and $R{-}CONS$ are defined analogously as above.*

The latter definition has a peculiarity that should be mentioned. Although the strategy is required to be recursive, consistency is only demanded for inputs that correspond to some function $f$ from the class to be learned. With the next definition we model the scenario in which consistency is required on all inputs. In order to distinguish the resulting learning type from the latter defined one, we use the prefix $T$. Informally, $T$ points to *total* consistency.

**Definition 5. (Wiehagen and Liepe, 1976)** *Let $U \subseteq R$ and let $\psi \in P^2$. The class $U$ is called $T$–consistently learnable in the limit with respect to $\psi$ iff there is a strategy $S \in R$ such that*

(1) $U \in CONS_\psi(S)$,

(2) $\psi_{S(f^n)}(x) = f(x)$ *for all* $f \in R$, $n \in \mathbb{N}$ *and* $x \leq n$.

*$T{-}CONS_\psi(S)$, $T{-}CONS_\psi$ and $T{-}CONS$ are defined in the same way as above.*

Finally, looking at potential applications it is often highly desirable to make no assumptions concerning the *order* in which input data should be presented. Therefore, we sharpen Definitions 3 through 5 by additionally demanding a strategy to behave consistently independently of the order of the input.

**Definition 6. (Blum and Blum, 1975)** *Let $U \subseteq R$ and let $\psi \in P^2$. $U \in T{-}CONS_\psi^{arb}$ iff there is a strategy $S \in R$ such that*

(1) *for all $f \in U$ and every $X \in \Pi(\mathbb{N})$, there is a $j \in \mathbb{N}$ such that $\psi_j = f$, and $(S(f^{X,n}))_{n\in\mathbb{N}}$ converges to $j$,*

(2) $\psi_{S(f^{X,n})}(x_m) = f(x_m)$ *for every permutation $X \in \Pi(\mathbb{N})$, $f \in R$, $n \in \mathbb{N}$, and $m \leq n$.*

*$T{-}CONS_\psi^{arb}(S)$ as well as $T{-}CONS^{arb}$ are defined in analogy to the above.*

Furthermore, appropriately incorporating the requirement to learn from arbitrary input directly yields the learning types $LIM^{arb}$, $FIN^{arb}$, $CONS^{arb}$, and $R$–$CONS^{arb}$. Therefore, the formal definition of these learning models is omitted here. Note that $LIM = LIM^{arb}$ as well as $FIN = FIN^{arb}$, cf. Jantke and Beick (1981). Moreover, for all learning types $LT \in \{FIN,\ FIN^{arb},\ T{-}CONS,\ T{-}CONS^{arb},\ R{-}CONS,\ R{-}CONS^{arb},\ CONS,\ CONS^{arb}\}$ we have $LT_\varphi = LT$ for every Gödel numbering $\varphi$. In the following section we aim to compare the learning power of the different models of consistent learning to one another as well as to the other models of learning that we have defined. Note that in the following $\subseteq$ denotes subset and $\subset$ denotes *proper* subset. Finally, incomparability of sets is denoted by #.

## 3. The Inconsistency Phenomenon in Inductive Inference

The main goal of this section is a thorough study of the learning power of the different models of consistent learning. Our first subsection deals with the learning type $CONS$ and compares it to learning in the limit.

### 3.1. The General Inconsistency Phenomenon

The inconsistency phenomenon has been discovered independently by Barzdin (1974a) and the Blums (1975). They observed that there are classes of recursive functions that are inferable in the limit but which cannot be learned by any consistent strategy. Since both papers do not contain a proof of this assertion, we present here a proof from Wiehagen (1976) actually showing a somewhat stronger result than the one formulated in the next theorem. We shall discuss this issue below.

**Theorem 1. (Barzdin, 1974a)** $CONS \subset LIM$

*Proof.* Let $U = \{f \in R |\ f = \alpha jp,\ \alpha \in \mathbb{N}^*,\ j \geq 2,\ p \in R_{0,1},\ \varphi_j = f\}$, where $\varphi \in Göd$. Obviously, $U \in LIM(S)$, where $S(f^n)$ is equal to the last value $f(x) \geq 2$ from $(f(0), ..., f(n))$ and 0, if no such value exists. For the purpose to prove that $U \notin CONS$ we need the following claim.

*Claim.* For every $\alpha \in \mathbb{N}^*$, there is an $f \in U$ such that $\alpha \sqsubset f$.

Indeed, by an implicit use of the Recursion Theorem, cf. Rogers (1967), it is easy to see that for every $\alpha \in \mathbb{N}^*$ and every $p \in R_{0,1}$, there is a $j \geq 2$ such that $\varphi_j = \alpha jp$.

Now, suppose that there is a strategy $S \in P$ such that $U \in CONS_\varphi(S)$. By the claim, we get $S \in R$ and for every $\alpha \in \mathbb{N}^*$, $\alpha \sqsubset \varphi_{S(\alpha)}$. Thus, on every $\alpha \in \mathbb{N}^*$, $S$ always produces a consistent guess. Then, again by an implicit use of the Recursion Theorem, let $j \geq 2$ be any $\varphi$–number of the function $f$ defined as follows: $f(0) = j$, and for any $n \in \mathbb{N}$,

$$f(n+1) = \begin{cases} 0, & \text{if} \quad S(f^n 0) \neq S(f^n) \\ 1, & \text{if} \quad S(f^n 0) = S(f^n) \text{ and } S(f^n 1) \neq S(f^n). \end{cases}$$

In accordance with the claim and the assumption that $S$ works consistently one straightforwardly verifies that $S(f^n 0) \neq S(f^n)$ or $S(f^n 1) \neq S(f^n)$ for any $n \in \mathbb{N}$. Therefore the function $f$ is everywhere defined and we have $f \in U$. On the other hand, the strategy $S$ changes its mind infinitely often when successively fed $f$, a contradiction to $U \in CONS_\varphi(S)$. q.e.d.

Note that the class from the proof of Theorem 1 is even *iteratively* learnable in the limit. We call a class $U$ of recursive functions iteratively learnable iff there is a strategy that learns any $f \in U$ as follows: In step $n$ the strategy exclusively gets its previous guess produced in step $n-1$ as well as the new value $f(n)$. By IT we denote the collection of all classes $U$ which can be learned iteratively. In Wiehagen (1976) $CONS \subset IT \subset LIM$ has been proved. Recent papers give new evidence for the power of iterative learning (cf., e.g., Porat and Feldman (1988), Lange and Wiehagen (1991), Lange and Zeugmann (1992)).

A closer look to the proof above shows that, in general, a strategy attempting to learn functions consistently has to overcome two difficulties. First, it should avoid to change too often its current guess that is eventually no longer consistent, to a definitely consistent hypothesis, since this behavior may force the strategy to diverge. Second, trusting that its current guess is consistent may eventually lead to an actually inconsistent hypothesis, since the strategy cannot effectively prove consistency. Indeed, it turns out that a class $U$ is consistently learnable iff there is a suitable space of hypotheses $\psi$ such that the consistency problem restricted to $U$ and $\psi$ is effectively decidable. More precisely, let $U \subseteq R$ and let $\psi$ be any numbering. We say that $U$–consistency is decidable with respect to $\psi$ iff there is a predicate $cons \in P^2$ such that for every $\alpha \in [U]$ and all $i \in \mathbb{N}$, $cons(\alpha, i)$ is defined, and $cons(\alpha, i) = 1$ if and only if $\alpha \sqsubset \psi_i$.

**Theorem 2.** *$U \in CONS$ iff there is a numbering $\psi \in P^2$ such that*

(1) *$U \subseteq P_\psi$,*

(2) *$U$–consistency with respect to $\psi$ is decidable.*

Theorem 2 is a consequence of Theorem 9 from Wiehagen (1978). Note that for an arbitrary Gödel numbering, $U$-consistency is undecidable for any non-empty class $U \subseteq R$.

Next we provide deeper insight to the problem whether or not consistent learning is sensitive with respect to the domain of the allowed strategies.

## 3.2. Consistent Learning in Dependence on the Domain of the Strategies

As already mentioned, in machine learning it is often assumed that learning algorithms are defined on all inputs. On the one hand, this requirement is partially justified by a result of Gold (1967). He proved that learning in the limit is insensitive with respect to the requirement to learn exclusively with recursive strategies, i.e., if $U \in LIM(S)$, then there is a strategy $\hat{S} \in R$ such that $U \in LIM(\hat{S})$. One the other hand, consistency is a common requirement in machine learning. Therefore, it is natural to ask whether or not the power of consistent learning algorithms further decreases if one restricts itself to recursive strategies. The answer to this question is provided by our next theorem.

**Theorem 3.** *$T\!-\!CONS \subset R\!-\!CONS \subset CONS$.*

*Proof.* By definition, $T-CONS \subseteq R-CONS \subseteq CONS$. In order to show $R-CONS \setminus T-CONS \neq \emptyset$ let $U = \{f \mid f \in R,\ \varphi_{f(0)} = f\}$ where $\varphi \in Göd$. Obviously, $U \in R\!-\!CONS_\varphi(S)$ by the strategy $S(f^n) = f(0)$ for all $n \in N$.

Now assume that $U \in T\!-\!CONS_\varphi(S)$. Hence $S \in R$ and $\varphi_{S(f^n)}(x) = f(x)$ for any $f \in R, n \in \mathbb{N}$ and $x \leq n$. By an implicit use of the Recursion Theorem, let $f = \varphi_i$ be the following function.

$$f(0) = i,$$

$$f(n+1) = \begin{cases} 0, & \text{if } S(f^n 0) \neq S(f^n) \\ 1, & \text{if } S(f^n 0) = S(f^n) \text{ and } S(f^n 1) \neq S(f^n). \end{cases}$$

Clearly, $f \in U$ (note that one of the two cases in the definition of $f$ must happen for all $n \geq 1$). On the other hand, $S(f^n) \neq S(f^{n+1})$ for all $n \in \mathbb{N}$, contradicting $U \in T\text{–}CONS_\varphi(S)$. Hence $U \notin T\text{–}CONS$. This completes the proof of $T\text{–}CONS \subset R\text{–}CONS$.

In order to prove $CONS \setminus R-CONS \neq \emptyset$ we use a class similar to the class above, namely $U = \{f \mid f \in R,$ either $\varphi_{f(0)} = f$ or $\varphi_{f(1)} = f\}$. First we show that $U \in CONS$. The wanted strategy is defined as follows. Let $f \in R$ and $n \in \mathbb{N}$.

$S(f^n) =$ "Compute in parallel $\varphi_{f(0)}(x)$ and $\varphi_{f(1)}(x)$ for all $x \leq n$ until (A) or (B) happens.

(A) $\varphi_{f(0)}(x) = f(x)$ for all $x \leq n$.

(B) $\varphi_{f(1)}(x) = f(x)$ for all $x \leq n$.

If (A) happens first, then output $f(0)$. If (B) happens first, then output $f(1)$. If neither (A) nor (B) happens, then $S(f^n)$ is not defined."

By the definition of $U$, it is obvious that $S(f^n)$ is defined for all $f \in U$ and all $n \in \mathbb{N}$. Moreover, $S$ is clearly consistent. Hence, it suffices to prove that $(S(f^n))_{n\in\mathbb{N}}$ converges for all $f \in U$. But this is also an immediate consequence of the definition of $U$, since either $\varphi_{f(0)} \neq f$ or $\varphi_{f(1)} \neq f$. Hence $S$ cannot oscillate infinitely often between $f(0)$ and $f(1)$. Consequently, $U \in CONS_\varphi(S)$.

Next we show that $U \notin R{-}CONS$. Suppose there is a strategy $S \in R$ such that $U \in R{-}CONS_\varphi(S)$. Applying Smullyan's Recursion Theorem, cf. Smullyan (1961), we construct a function $f \in U$ such that either $S(f^n) \neq S(f^{n+1})$ for all $n \in \mathbb{N}$ or $\varphi_{S(f^x)}(y) \neq f(y)$ for some $x, y \in \mathbb{N}$ with $y \leq x$. Since both cases yield a contradiction to the definition of $R{-}CONS$, we are done. The wanted function $f$ is defined as follows. Let $h$ and $s$ be two recursive functions such that for all $i, j \in \mathbb{N}, \varphi_{h(i,j)}(0) = \varphi_{s(i,j)}(0) = i$ and $\varphi_{h(i,j)}(1) = \varphi_{s(i,j)}(1) = j$. For any $i, j \in \mathbb{N}, x \geq 2$ we proceed inductively.

Suspend the definition of $\varphi_{s(i,j)}$. Try to define $\varphi_{h(i,j)}$ for more and more arguments via the following procedure.

**(T)** Test whether or not (A) or (B) happens (this can be effectively checked, since $S \in R$):

(A) $S(\varphi^x_{h(i,j)}0) \neq S(\varphi^x_{h(i,j)})$,

(B) $S(\varphi^x_{h(i,j)}1) \neq S(\varphi^x_{h(i,j)})$.

If (A) happens, then let $\varphi_{h(i,j)}(x+1) = 0$, let $x := x+1$, and goto (T).
In case (B) happens, set $\varphi_{h(i,j)}(x+1) = 1$, let $x := x+1$, and goto (T).
If neither (A) nor (B) happens, then define $\varphi_{h(i,j)}(x') = 0$ for all $x' > x$, and goto $(*)$.

$(*)$ Set $\varphi_{s(i,j)}(n) = \varphi_{h(i,j)}(n)$ for all $n \leq x$, and $\varphi_{s(i,j)}(x') = 1$ for all $x' > x$.

By Smullyan's Recursion Theorem, there are numbers $i$ and $j$ such that $\varphi_i = \varphi_{h(i,j)}$ and $\varphi_j = \varphi_{s(i,j)}$. Now we distinguish the following cases.

*Case* 1. The loop in (T) is never left.

Then we directly obtain that $\varphi_i \in U$, since $\varphi_j = ij$, and hence a finite function. Moreover, in accordance with the definition of the loop (T), on input $\varphi^n_i$ the strategy $S$ changes its mind for all $n > 0$.

*Case* 2. The loop in (T) is left.

Then there exist an $x$ such that $S(\varphi^x_{h(i,j)}0) = S(\varphi^x_{h(i,j)}1)$. Hence $S(\varphi^{x+1}_i) = S(\varphi^{x+1}_j)$, since $\varphi_{h(i,j)} = \varphi_i$, $\varphi_{s(i,j)} = \varphi_j$, $\varphi_i(n) = \varphi_j(n)$ for all $n \leq x$ by $(*)$, as well as $\varphi_i(x+1) = 0$ and $\varphi_j(x+1) = 1$. Furthermore, $\varphi_i, \varphi_j \in R$. Since $\varphi_i(x+1) \neq \varphi_j(x+1)$, we get $\varphi_i \neq \varphi_j$. On the other hand, $\varphi_i(0) = i$ and $\varphi_j(1) = j$. Consequently, both functions $\varphi_i$ and $\varphi_j$ belong to $U$. But $S(\varphi_i^{x+1}) = S(\varphi_j^{x+1})$ and $\varphi_i(x+1) \neq \varphi_j(x+1)$, hence $S$ does not work consistently on input $\varphi_i^{x+1}$ or $\varphi_j^{x+1}$. This contradiction completes the proof. q.e.d.

Note that even $T-CONS$, the most stringent type of consistent identification considered above, is of remarkable power. Therefore, let $NUM = \{U \mid \text{there is } \psi \in R^2 \text{ such that } U \subseteq P_\psi\}$ denote the set of all recursively enumerable classes of recursive functions. Then in Wiehagen and Liepe (1976) the following result was proved.

**Theorem 4.** $NUM \subset T-CONS$

*Proof.* Every class $U \in NUM$ can be learned T-consistently by a slightly modified version of Gold's identification-by-enumeration strategy.

A class witnessing $T-CONS \setminus NUM \neq \emptyset$ can be defined as follows. Let $\varphi \in$ *Göd*. Let $\Phi$ be any complexity measure associated with $\varphi$, cf. Blum (1967). Then $\{\Phi_i \mid i \in \mathbb{N},\ \varphi_i \in R\} \in T-CONS \setminus NUM$. We omit the details. q.e.d.

The next result provides a more subtle insight into the different power of $T$–$CONS$ and $R-CONS$.

**Theorem 5.**

(1) $FIN \# T-CONS$

(2) $FIN \subset R-CONS$

*Proof.* (1) $T-CONS \setminus FIN \neq \emptyset$ is proved in Wiehagen and Liepe (1976). Note that the class $\{\Phi_i \mid i \in \mathbb{N},\ \varphi_i \in R\}$ where $\Phi$ is a complexity measure associated with $\varphi \in$ *Göd* in the sense of Blum (1967) also witnesses $T-CONS \setminus FIN \neq \emptyset$.

Let now $U = \{f \mid f \in R,\ \varphi_{f(0)} = f\}$. Obviously, $U \in FIN$. On the other hand, $U \notin T-CONS$, cf. proof of Theorem 3. Hence $T-CONS \setminus FIN \neq \emptyset$. This proves (1).

Next, we prove Assertion (2). Since $T$–$CONS \subset R$–$CONS$, $R$–$CONS \setminus FIN \neq \emptyset$ is an immediate consequence of (1). The proof of $FIN \subseteq R-CONS$ mainly relies on the decidability of convergence of any finite learning algorithm. Let $U \in FIN$, and let $S$ be any strategy witnessing $U \in FIN_\varphi(S)$. Furthermore, let $s \in R$ be any function such that $\varphi_{s(\alpha)} = \alpha 0^\infty$ for all $\alpha \in \mathbb{N}^*$. The wanted strategy $\hat{S}$ is defined as follows. Let $f \in R$ and $n \in \mathbb{N}$. Then

$\hat{S}(f^n) =$ "In parallel, try to compute $S(f^0), ..., S(f^n)$ for precisely $n$ steps. Let $k \geq 1$ be the least number such that all values $S(f^0), ..., S(f^k)$ turn out to be defined, and $S(f^{k-1}) = S(f^k)$.
In case this $k$ is found, output $S(f^k)$. Otherwise, output $s(f^n)$."

It remains to show that $U \in R-CONS(\hat{S})$. Obviously, $\hat{S} \in R$. Now, let $f \in U$. We have to show that $\hat{S}$ consistently learns $f$.

*Claim* 1. $\hat{S}$ learns $f$.

Since $f \in U$, the strategy $S$ is defined for all inputs $f^n$, $n \in \mathbb{N}$. Moreover, since $S$ finitely learns $f$, the sequence $(S(f^n))_{n \in \mathbb{N}}$ finitely converges to a $\varphi$–program of $f$.

Hence, $\hat{S}$ eventually has to find the least $k$ such that $S(f^{k-1}) = S(f^k)$, and all values $S(f^0), ..., S(f^k)$ are defined. By the definition of $FIN$, $\varphi_{S(f^k)} = f$. Hence, $\hat{S}$ learns $f$.

*Claim* 2. For all $f \in U$ and $n \in \mathbb{N}$, $\hat{S}(f^n)$ is a consistent hypothesis.

Clearly, as long as $\hat{S}$ outputs $s(f^n)$, it is consistent. Suppose, $\hat{S}$ outputs $S(f^k)$ for the first time. Then it has verified that $S(f^{k-1}) = S(f^k)$. Since $f \in U$, and $U \in FIN_{\varphi}(S)$, this directly implies $\varphi_{S(f^k)} = f$. Therefore, $\hat{S}$ again outputs a consistent hypothesis. Since this hypothesis is repeated in any subsequent learning step, the claim is proved. q.e.d.

The next result points out another difference between the types $CONS$, $R$–$CONS$, on the one hand, and $T-CONS$, on the other hand.

**Theorem 6.**

(1) *$CONS$ and $R-CONS$ are not closed under finite union.*

(2) *$T-CONS$ is closed under recursively enumerable union.*

*Proof.* (1) is a direct consequence of a more general result of Barzdin (1974b). Let $U = \{f \mid f \in R,\ \varphi_{f(0)} = f\}$ and let $V = \{\alpha 0^\infty \mid \alpha \in \mathbb{N}^*\}$. It is easy to see that $U,\ V \in R-CONS$, and hence $U,\ V \in CONS$. On the other hand, $U \cup V \notin LIM$ as shown in Barzdin (1974b).

In order to prove (2), we restate this assertion more formally. Let $(S_i)_{i \in \mathbb{N}}$ be a recursive enumeration of $T$–consistent strategies. Then there exists a strategy $S$ such that $T-CONS(S) = \bigcup_{i \in \mathbb{N}} T-CONS(S_i)$.

Without loss of generality, we may assume that all the strategies $S_i$ output as hypotheses programs in some fixed Gödel numbering $\varphi$. Note that the following proof mainly uses a proof technique of Minicozzi (1976). Let $f \in R$. Then two cases are possible. Either there is a strategy $S_i$ that learns $f$ or all strategies fail to learn it. However, in the latter case each of the strategies $S_i$ has to change its mind infinitely often. On the other hand, if $f$ is learned by some $S_i$ then at least one strategy stabilizes its output. The wanted strategy $S$ searches for an enumerated machine that might learn $f$ as follows.

The strategy $S$ dovetails the computation of more and more outputs of the enumerated strategies. For each strategy $S_i$ that is already included in its dovetailed computations, it counts the number of equal outputs. This number is called weight. As long as a strategy repeats its actual guess on the next input, the weight increments. If a strategy performs a mind change, its weight reduces to zero. After having read the initial segment $f^k$ of the function $f$ the strategy $S$ favors from the first $k + 1$ strategies $S_0, ..., S_k$ that one which actually has the greatest weight. In case there are two strategies $S_i$ and $S_j$ taking the greatest weight the strategy $S$ chooses that one having the smallest index.

We formally define $S$ as follows. Let $f \in R$, and $k \in \mathbb{N}$.

$S(f^k) =$ "Compute in parallel

$S_0(f^0),\ ...,\ S_0(f^k),$

$S_1(f^0)$, ..., $S_1(f^k)$,

–

–

–

$S_k(f^0)$, ..., $S_k(f^k)$,

and assign to each strategy $S_i$, $i \le k$, its weight, i.e., the greatest number $m \le k-i$ satisfying the condition that $S_i(f^{k-i-m}) = S_i(f^{k-i-m+1}) = ... = S_i(f^{k-i})$. Note that we calculate the weights in a triangular fashion. This is necessary in order to achieve convergence of $S$. Choose $w(k)$ to be the smallest $i \le k$ such that the strategy $S_i$ has the greatest weight.

In case all considered machines have weight zero, output a $\varphi$–program of $f(0)\cdots f(k)0^\infty$.
If $w(k) = w(k-1)$, then output $S_{w(k)}(f^k)$. Otherwise, output a $\varphi$–program of $f(0)\cdots f(k)0^\infty$ that is different from $S(f^{k-1})$."

It remains to show that $T\text{–}CONS_\varphi(S) = \bigcup_{i\in\mathbb{N}} T\text{–}CONS_\varphi(S_i)$. Obviously, $S$ works consistently on any initial segment it is fed, since all of the strategies $S_i$, $i \in \mathbb{N}$, do so. Now, let $f \in R$ and suppose that $f$ is learned by some strategy $S_i$. Consequently, there exists numbers $j$, $n_0$ such that $S_i(f^n) = j$ for all $n \ge n_0$, and $\varphi_j = f$. Hence, for any strategy that learns $f$ its weight increases after some point in each step of $S$'s computation. Therefore, for almost all $k$ the strategy $S$ must favor exactly one of the strategies $S_i$ that learns $f$ and, after some point, $S$ outputs always $S_i(f^k)$. Note that the computation of weights in a triangular fashion really ensures the desired convergence, since any new strategy included in $S$'s computation initially gets weight zero.

On the other hand, if none of the strategies $S_i$, $i \in \mathbb{N}$, learns $f$, then each strategy $S_i$ has to perform infinitely many mind changes. This is ensured by our assumption that each $S_i$ is $T$–consistent. Hence, the case $w(k) \ne w(k-1)$ occurs infinitely often. But each occurrence of this case forces $S$ to perform a mind change. Consequently, $S$ cannot converge. q.e.d.

We finish this subsection with characterizations of $T\text{–}CONS$ and $R\text{–}CONS$ which are similar to that of $CONS$ presented in Theorem 2. Therefore let $\psi \in P^2$ be any numbering. Then we say that consistency with respect to $\psi$ is decidable iff there is a predicate $cons \in R^2$ such that for every $\alpha \in \mathbb{N}^*$ and all $i \in \mathbb{N}$, $cons(\alpha, i) = 1$ if and only if $\alpha \sqsubset \psi_i$.

**Theorem 7.** *$U \in T{-}CONS$ iff there is a numbering $\psi \in P^2$ such that*

(1) *$U \subseteq P_\psi$,*

(2) *consistency with respect to $\psi$ is decidable.*

*Proof.* Necessity. Let $U \in T{-}CONS_\varphi(S)$ where $\varphi \in P^2$ is any numbering and $S$ is a $T$–consistent strategy. Let $M = \{(z,n) |\ z, n \in \mathbb{N},\ \varphi_z(x)$ is defined for any $x \le n,\ S(\varphi_z^n) = z\}$ be recursively enumerated by a function $e$. Then define a numbering $\psi$ as follows. Let $i, x \in \mathbb{N}$, $e(i) = (z, n)$.

$$\psi_i(x) = \begin{cases} \varphi_z(x), & \text{if } x \le n \\ \varphi_z(x), & \text{if } x > n \text{ and, for any } y \in \mathbb{N} \text{ such that } n < y \le x, \\ & \varphi_z(y) \text{ is defined and } S(\varphi_z^y) = z \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

In order to show (1) let $f \in U$ and $n, z \in \mathbb{N}$ be such that for any $m \ge n,\ S(f^m) = z$. Clearly, $\varphi_z = f$. Furthermore, $(z, n) \in M$. Let $i \in \mathbb{N}$ be such that $e(i) = (z, n)$. Then, by definition of $\psi$, $\psi_i = \varphi_z = f$. Hence $U \subseteq P_\psi$.

In order to prove (2) we define *cons* $\in R^2$ such that for any $\alpha \in \mathbb{N}^*$, $i \in \mathbb{N}$, $cons(\alpha, i) = 1$ iff $\alpha \sqsubset \psi_i$. Let $\alpha = (\alpha_0, \dots, \alpha_x) \in \mathbb{N}^*$ and $i \in \mathbb{N}$. Let $e(i) = (z, n)$. Then define

$$cons(\alpha, i) = \begin{cases} 1, & \text{if } x \le n \text{ and, for any } y \le x, \alpha_y = \psi_i(y) \\ 1, & \text{if } x > n, \alpha_y = \psi_i(y) \text{ for any } y \le n, \text{ and,} \\ & \text{for any } y \in \mathbb{N} \text{ such that } n < y \le x, S(\alpha_0, \dots, \alpha_y) = z \\ 0, & \text{otherwise.} \end{cases}$$

It is not hard to see that *cons* $\in R^2$ and, for every $\alpha \in \mathbb{N}^*$, $i \in \mathbb{N}$, we have $cons(\alpha, i) = 1$ iff $\alpha \sqsubset \psi_i$.

Sufficiency. Let $\psi \in P^2$ be any numbering. Let *cons* $\in R^2$ be such that for all $\alpha \in \mathbb{N}^*$, $i \in \mathbb{N}$, $cons(\alpha, i) = 1$ iff $\alpha \sqsubset \psi_i$. Let $U \subseteq P_\psi$. In order to consistently learn any function $f \in U$ it suffices to define $S(f^n) = min\{i \mid cons(f^n, i)\}$. However, $S$ is undefined if, for $f \notin U$, $n \in \mathbb{N}$, there is *no* $i \in \mathbb{N}$ such that $f^n \sqsubset \psi_i$. The following more careful definition of $S$ will circumvent this difficulty. Let $\varphi \in Göd$. Let $aux \in R$ be such that for any $\alpha \in \mathbb{N}^*, \varphi_{aux(\alpha)} = \alpha 0^\infty$. Finally, let $c \in R$ be such that for all $i \in \mathbb{N}, \psi_i = \varphi_{c(i)}$. Then, for any $f \in R$, $n \in \mathbb{N}$, define a strategy $S$ as follows.

$$S(f^n) = \begin{cases} c(j), & \text{if } I = \{i \mid i \le n, \text{ cons } (f^n, i) = 1\} \ne \emptyset \text{ and } j = min\ I \\ aux(f^n), & I = \emptyset. \end{cases}$$

Clearly, $S \in R$ and $S$ outputs only consistent hypotheses. Now let $f \in U$. Then, obviously, $(S(f^n))_{n \in \mathbb{N}}$ converges to $c(min\{i \mid \psi_i = f\})$. Hence, $S$ witnesses $U \in T{-}CONS_\varphi$. q.e.d.

Finally, let $U \subseteq R$ and $\psi \in P^2$ be any numbering. Then we say that $U$–consistency with respect to $\psi$ is $R$–decidable iff there is a predicate *cons* $\in R^2$ such that for every $\alpha \in [U]$ and all $i \in \mathbb{N}$, $cons(\alpha, i) = 1$ if and only if $\alpha \sqsubset \psi_i$.

**Theorem 8.** *$U \in R{-}CONS$ iff there is a numbering $\psi \in P^2$ such that*

(1) *$U \subseteq P_\psi$,*

(2) *$U$–consistency with respect to $\psi$ is $R$–decidable.*

The proof of Theorem 8 is similar to that of Theorem 7.

The characterizations of $T-CONS$, $R-CONS$ and $CONS$ give rise to point out some relationship between the problem of deciding consistency and the halting problem. As it follows from Theorem 3 and Theorem 4, for any of the learning types $LT \in \{T{-}CONS,\ R{-}CONS,\ CONS\}$, we have $NUM \subset LT$. On the other hand, as it was shown in Wiehagen and Zeugmann (1994), for any class $U \subseteq R$ and any numbering $\psi \in P^2$, if $U \notin NUM$ and $U \subseteq P_\psi$, then the halting problem with respect to $\psi$ is undecidable, i.e., there is no $h \in R^2$ such that for any $i, x \in \mathbb{N}, h(i,x) = 1$ iff $\psi_i(x)$ is defined.

Consequently, for any $U \in LT \setminus NUM$ and any numbering $\psi \in P^2$ from the corresponding theorem above characterizing $LT$, the halting problem with respect to $\psi$ is undecidable. On the other hand, the corresponding version of consistency with respect to $\psi$ is decidable. Hence this version of consistency *cannot* be decided by firstly deciding the halting problem and secondly, if possible, computing the desired values of the function under consideration in order to compare these values with the given ones. More formally, in general, $cons(f^n, i)$ *cannot* be evaluated by deciding whether or not $\psi_i(x)$ is defined for all $x \leq n$ and, if it is, then computing $\psi_i^n$ and comparing it with $f^n$.

Informally, though consistency is decidable in the "characteristic" numberings of Theorems 2, 7, 8 it is not decidable in a "straightforward way."

## 3.3. Consistent Learning on Arbitrary Input Order

In applications it is often highly desirable to make no assumption concerning the *order* in which input data should be presented. Therefore, we now state a characterization of the type $T{-}CONS^{arb}$ due to Blum and Blum (1975). This characterization easily yields a "lower bound" on the power of $T{-}CONS^{arb}$.

**Definition 7.** *A numbering $\psi \in P^2$ is said to be measurable iff the predicate "$\psi_i(x) = y$" is uniformly recursive in $i, x, y$.*

**Theorem 9. (Blum and Blum, 1975)**

*$U \in T{-}CONS^{arb}$ iff there is a measurable numbering $\psi \in P^2$ such that $U \subseteq P_\psi$.*

Note that the measurability of $\psi$ allows deciding consistency with respect to $\psi$ in a "straightforward way."

**Corollary 10.** $NUM \subset T{-}CONS^{arb}$

*Proof.* $NUM \subseteq T{-}CONS^{arb}$ can easily be proved by a slight modification of Gold's identification by enumeration.
For proving the proper inclusion, let $\varphi \in G\ddot{o}d$ and $\Phi \in P^2$ be a complexity measure associated with $\varphi$, cf. Blum (1967). Let $U = \{\Phi_i \mid i \in \mathbb{N},\ \varphi_i \in R\}$. Then $U \notin NUM$, since otherwise $R \in NUM$ could be proved, a contradiction. On the other hand, $U \in T{-}CONS^{arb}$ via Theorem 9, since $\Phi \in P^2$ is a measurable numbering. q.e.d.

As far as we now, it is still an open problem whether $T{-}CONS^{arb}$ is a *proper* subset of $T{-}CONS$. But, of course, any solution to this problem will not influence the "lower bound" on the power of $T{-}CONS^{arb}$ by Corollary 10.

The reader is encouraged to consult Jantke and Beick (1981), Zeugmann (1983) and Fulk (1989) for further investigation concerning consistent identification.

## 4. Exact Learning in Polynomial Time and Inconsistency

The results presented in the previous section may lead to the impression that the inconsistency phenomenon may be far beyond any practical relevance, since it has been established in a setting where the consistency problem is undecidable in general. Therefore now we ask whether the inconsistency phenomenon does survive in settings where consistency is decidable. Moreover, we additionally restrict ourselves to deal exclusively with learning strategies that are polynomial time computable. Then, of course, the superiority of inconsistent strategies, if any, has to be established in terms of complexity theory. We present a learning problem which is generally consistently solvable but which *cannot be solved consistently in polynomial time* unless $\mathcal{P} \neq \mathcal{NP}$. The desired goal is achieved by elaborating an algorithm *inconsistently* solving the same learning problem in *polynomial time*. As far as we know this is the first learning problem for which the announced properties are rigorously proved. In our opinion, this result gives strong evidence of seriously taking inconsistent learning strategies into consideration.

The setting we want to deal with is the learnability of pattern languages introduced by Angluin (1980). Subsequently, Shinohara (1982) dealt with polynomial time learnability of subclasses of pattern languages. Nix (1983) outlined interesting applications of pattern inference algorithms. Recently, Jantke (1991) and Lange and Zeugmann (1993) as well as Zeugmann, Lange and Kapur (1995) dealt with the learnability of pattern languages under monotonicity constraints. Moreover, Kearns and Pitt (1989), Ko, Marron and Tzeng (1990) and Schapire (1990) intensively studied the learnability of pattern languages in the PAC–learning model; thus, Schapire (1990) proved that the class *PAT* of all pattern languages is not PAC-learnable unless $\mathcal{P}_{/poly} = \mathcal{NP}_{/poly}$.

So let us define pattern languages. Let $\Sigma = \{a, b, ..\}$ be any non–empty finite alphabet containing at least two elements. Furthermore, let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\Sigma \cap X = \emptyset$. *Patterns* are non–empty strings from $\Sigma \cup X$, e.g., $ab$, $ax_1ccc$, $bx_1x_1cx_2x_2$ are patterns. $L(p)$, the language generated by pattern $p$ is the set of strings which can be obtained by substituting non–null strings from $\Sigma^*$ for the variables of the pattern $p$. Thus $aabbb$ is generable from pattern $ax_1x_2b$, while $aabba$ is not. *Pat* and *PAT* denote the set of all patterns and of all pattern languages over $\Sigma$, respectively. In order to deal with the learnability of pattern languages we have to specify from what information the learning strategies should do their task. Following Gold (1967) we distinguish between learning from text and from informant. Formally, let $L \subseteq \Sigma^*$; we call a mapping $I : \mathbb{N} \to \Sigma^* \times \{+, -\}$ *informant* of $L$ iff

(1) For every $w \in \Sigma^*$, there are an $n \in \mathbb{N}$ and $\lambda \in \{+, -\}$ such that $I(n) = (w, \lambda)$,

(2) for every $n \in \mathbb{N}$, $w \in \Sigma^*$, and $\lambda \in \{+, -\}$, if $I(n) = (w, \lambda)$ then $w \in L$ iff $\lambda = +$.

Let *Info(L)* denote the set of all informants of $L$. Furthermore, for $I \in$ *Info(L)* and $n \in \mathbb{N}$, let $I^n = cod(I(0), ..., I(n))$, where *cod* denotes an effective and bijective mapping from the set of all finite sequences of elements from $\Sigma^* \times \{+, -\}$ onto $\mathbb{N}$.

Finally, we set $I_n = \{w \in \Sigma^* \mid$ there are $i \le n,\ \lambda \in \{+,-\}$ s.t. $I(i) = (w,\lambda)\}$, and $I_n^+ = I_n \cap L,\ I_n^- = I_n \setminus I_n^+$.

Any mapping $T$ from $\mathbb{N}$ onto $L$ is called a *text* for L. By *Text(L)* we denote the set of all texts for $L$. The sets $T_n$, $T_n^+$ as well as $T_n^-$ are analogously defined as above.

Intuitively, a text for $L$ generates the language $L$ without any information concerning the complement of $L$, whereas an informant of $L$ decides $L$ by informing the strategy whether or not any word from $\Sigma^*$ belongs to $L$. Note that we allow a text and an informant to be non–effective.

**Definition 8.** *PAT is called learnable in the limit from informant (abbr.* $PAT \in LIM-INF$*) iff there is an effective strategy* $S$ *from* $\mathbb{N}$ *into Pat such that for all* $L \in PAT$ *and every* $I \in$ *Info(L),*

(1) *for all* $n \in \mathbb{N}$, $S(I^n)$ *is defined,*

(2) *there is a* $p \in Pat$ *such that* $L(p) = L$ *and for almost all* $n \in \mathbb{N}$, $S(I^n) = p$.

**Definition 9.** *PAT is called consistently learnable in the limit from informant (abbr.* $PAT \in CONS–INF$*) iff there is an effective strategy* $S$ *from* $\mathbb{N}$ *into Pat such that*

(1) $PAT \in LIM-INF$ *by* $S$,

(2) *for all* $L \in PAT$, $I \in$ *Info(L) and* $n \in \mathbb{N}$, $I_n^+ \subseteq L(S(I^n))$ *and* $I_n^- \cap L(S(I^n)) = \emptyset$.

Note that a consistent learning strategy is required to correctly reflect both the positive as well as the negative data it has already seen. Next we sharpen Definition 8 and 9 by additionally requiring polynomial time computability of $S$.

**Definition 10.** *PAT is called (consistently) learnable in the limit from informant in polynomial time (abbr.* $PAT \in Poly-LIM-INF$ $(PAT \in Poly-CONS-INF)$*) iff there are a strategy* $S$ *and a polynomial pol such that*

(1) $PAT \in LIM-INF$ $(PAT \in CONS-INF)$ *by* $S$,

(2) *for all* $L \in PAT$, $I \in$ *Info(L) and* $n \in \mathbb{N}$,
*time to compute* $S(I^n) \le pol(length(I^n))$.

Learning from text is analogously defined in replacing everywhere "informant" by "text." However, one point should be stated more precisely, namely that consistent learning from text does only require consistency with the data contained in the text. In order to have an example illuminating the difference we could define a strategy that initially outputs $x_1$. Since $L(x_1)$ contains every string over $\Sigma$ but the empty one, this hypothesis is consistent on text for every finite input. However, since the strategy has to converge, it cannot maintain this hypothesis *ad infinitum.* Finally, we use $LIM-TXT$, $CONS-TXT$, $Poly-LIM-TXT$ as well as $Poly-CONS-TXT$ to denote the corresponding learning types from text.

Now we can state the result mentioned above.

**Theorem 11.**

(1) $PAT \in CONS{-}INF$,

(2) $PAT \notin Poly{-}CONS{-}INF$, *provided* $\mathcal{P} \neq \mathcal{NP}$,

(3) $PAT \in Poly{-}LIM{-}INF$.

*Proof.* Assertion (1) is proved in applying Gold's (1967) enumeration technique. Therefore, let $(p_i)_{i\in\mathbb{N}}$ be any fixed effective enumeration of *Pat*. Let $L \in PAT$, let $I \in$ *Info(L)* be any informant, and let $n \in \mathbb{N}$. Define $S(I^n)$ to be the first pattern $p$ in the enumeration of *Pat* satisfying $I_n^+ \subseteq L(p)$ and $I_n^- \cap L(p) = \emptyset$. Since membership for pattern languages is uniformly decidable, cf. Angluin (1980), $S$ is computable. Due to the definition of $S$, consistency is obvious. Moreover, the strategy converges to the first pattern in the enumeration that generates the language $L$ to be learned. Note that $S$ cannot be computed in polynomial time, unless $\mathcal{P} = \mathcal{NP}$, since membership for pattern languages is $\mathcal{NP}$–complete, cf. Angluin (1980).

Next we have to show that there is no strategy at all consistently learning *PAT* from informant that is computable in polynomial time, if $\mathcal{P} \neq \mathcal{NP}$. This part of the proof is done by showing the $\mathcal{NP}$–hardness of an appropriate problem defined below. For any information concerning reducibility as well as $\mathcal{NP}$–complete problems the reader is referred to Garey and Johnson (1979). First we define the following decision problem *SEP*. Let $W^+,\ W^- \subseteq \Sigma^*$. We say that $W^+,\ W^-$ are *separable* iff there is a pattern $p$ such that $W^+ \subseteq L(p)$ and $W^- \cap L(p) = \emptyset$. *SEP* denotes the problem of *deciding* whether any $W^+,\ W^- \subseteq \Sigma^*$ are separable. Moreover, by *CSEP* we denote the problem of *constructing* a separating pattern $p$ for any given $W^+,\ W^-$ that are separable. The proof of Assertion (2) is completed via the following lemmata.

**Lemma A. (Ko, Marron, Tzeng, 1990)**
$3{-}SAT$ *is polynomial time reducible to SEP.*

**Lemma B.** *CSEP is $\mathcal{NP}$–hard.*

*Proof of Lemma B.* Let $C3-SAT$ denote the problem to construct a satisfying assignment to any satisfiable instance from $3{-}SAT$.

*Claim* 1. $C3{-}SAT \in \mathcal{P}$ implies $3{-}SAT \in \mathcal{P}$.

Assume there is an algorithm $\mathcal{A}$ having a running time that is bounded by some polynomial *pol* in the length of its input, and that, moreover, on input $C$ returns a satisfying assignment of $C$, if $C$ is satisfiable. Now let $C$ be any instance of $3-SAT$. Start $\mathcal{A}$ on input $C$. Since any polynomial is time constructible, we may combine $\mathcal{A}$ with a clock, i.e., we can efficiently stop $\mathcal{A}$ on input $C$ after at most $pol(length(C))$ steps of computation. Then two cases are possible. Either $\mathcal{A}$ returns nothing. Consequently, $C$ cannot be satisfiable. Otherwise $\mathcal{A}$ outputs an assignment *ass* within the given time bound. Then one can check in polynomial time whether or not *ass* indeed satisfies $C$. In case it does, we know that $C$ is satisfiable. In case it does not $C$ is again not satisfiable, since otherwise $\mathcal{A}$ would fail.

Note that we *cannot* prove the $\mathcal{NP}$–hardness of *CSEP* in the same manner as in showing Claim 1, since membership for pattern languages is $\mathcal{NP}$–complete. Hence, one cannot check in polynomial time whether a pattern eventually returned on input

$(W^+, W^-)$ does indeed separate these sets. However, we overcome this difficulty by showing the following claim.

*Claim* 2. $CSEP \in \mathcal{P}$ implies $C3{-}SAT \in \mathcal{P}$.

In accordance with Lemma A let *red* be any polynomial time reduction of $3{-}SAT$ to $SEP$. Suppose, there is an algorithm $\mathcal{B}$ solving $CSEP$ in polynomial time. Now let $C$ be any satisfiable instance of $3{-}SAT$. The wanted satisfying assignment may be computed as follows. First, compute $red(C) = (W^+, W^-)$. Since $C$ is satisfiable, we get that $(W^+, W^-)$ are separable. Next compute $p = \mathcal{B}(W^+, W^-)$. Finally, let *ass* be the assignment constructed to $p$ in the proof of the "only–if" direction of Lemma A. Since *red* is computable in time bounded by a polynomial in the length of $C$, the length of $(W^+, W^-)$ is bounded by a polynomial in the length of $C$, too. Consequently, *ass* is polynomial time computable. Hence, $C3{-}SAT \in \mathcal{P}$, if $CSEP \in \mathcal{P}$.

Finally, Claim 1 and Claim 2 directly yield Lemma B.

The proof of Assertion (2) is completed by showing the next claim.

*Claim* 3. $PAT \in Poly{-}CONS{-}INF$ implies $CSEP \in \mathcal{P}$.

Suppose $PAT \in Poly{-}CONS{-}INF$ by some strategy $S$. Let $W^+$, $W^-$ be any two separable sets, and let $p$ be any pattern separating them. Let $I \in Info(L(p))$ be an arbitrary informant such that, for some $n$, $I_n^+ = W^+$ and $I_n^- = W^-$. In accordance with the definition of separability, $I$ obviously exists. Consequently, $S(I^n)$ has to be defined, and furthermore, $q = S(I^n)$ has to be a pattern separating $W^+$, $W^-$. Finally, $S$ is polynomial time computable. Hence we get $CSEP \in \mathcal{P}$.

It remains to prove Assertion (3). In Lange and Wiehagen (1991) $PAT \in Poly{-}LIM{-}TXT$ has been shown. The corresponding strategy witnessing $PAT \in Poly{-}LIM{-}TXT$ works by "overlooking" data, namely it ignores all but the actually shortest strings of the language to be learned. It turns out that sufficiently many really shortest strings of a pattern language do suffice to learn it. From these remaining strings a hypothesis is generated in time that is even polynomial in the length of these strings. However, this hypothesis may be inconsistent, while being correct in the limit. Let $S$ denote the strategy from Lange and Wiehagen (1991) proving $PAT \in Poly{-}LIM{-}TXT$. We define a strategy $\tilde{S}$ witnessing $PAT \in Poly{-}LIM{-}INF$ as follows. On any input $I^n$ we set $\tilde{S}(I^n) = S(I_n^+)$. This proves (3) and hence the theorem.

Note that $\tilde{S}$ even works semi–consistently, since $I_n^- \cap L(\tilde{S}(I^n)) = \emptyset$ is valid for all $n \in \mathbb{N}$. Moreover, $\tilde{S}$ works iteratively as $S$ does. q.e.d.

At this point some remarks are mandatory. It should be mentioned that any consistent strategy $S$, independently of how complex it is, may be trivially converted into an inconsistent one that works in quadratic time. This is done as follows. On input $I^n$, one simulates $S$ on input $I^1$, $I^2$,...,$I^n$ no more than $n$ steps, and outputs $S(I^k)$, where $k$ is the largest number $y \leq n$ for which $S(I^y)$ is computable within at most $n$ steps.

However, it is obvious that this simulation technique does not yield any advantage. It does neither increase the efficiency of the learning algorithm, if one sums up all steps of computation until the learning task is successfully solved; nor does it enlarge the learning power. What we are looking for are "intelligent" inconsistent techniques.

In our opinion, Lange and Wiehagen's (1991) refined strategy behaves thus by the following reasons. First, it avoids membership tests at all. Second, it iteratively computes its current hypothesis. Third, the test whether or not it should eventually change its mind is extremely simple and may be executed in linear time. Moreover, the algorithm yielding an eventually new hypothesis performs exclusively syntactical or formal manipulations over strings.

Finally, let *Poly*$-CONS-LEXINF$ (*Poly*$-CONS-LEXTXT$) be the learning type obtained from *Poly*$-CONS-INF$ ( *Poly*$-CONS-TXT$ ) by restricting the information presentation from any informant $I \in$ *Info(L)* (any text $T \in$ *Text(L)*) to the lexicographically ordered one. Furthermore, let $S$ be a strategy such that $PAT \in LIM-INF$ ($LIM-TXT$) by $S$. Then, for any $L \in PAT$ and $D \in$ *Info(L)* $\cup$ *Text(L)*, let

$$Conv(S, D) = \text{the least number } m \text{ such that for all } n \geq m,\ S(D^n) = S(D^m)$$

denote the *stage of convergence* of $S$ on $D$.

The following theorem actually states that the inconsistent learning strategy of Lange and Wiehagen (1991) may behave both consistently and efficiently, if it receives the crucial information on the language to be learned in an appropriate order.

**Theorem 12.**

(1) *There are a strategy* $\tilde{S}$ *and a polynomial pol such that*

   *(i)* $PAT \in$ *Poly*$-CONS-LEXINF$ *by* $\tilde{S}$,

   *(ii) for every* $p \in Pat$, *there are uncountably many informants* $I \in$ Info$(L(p))$ *such that*

   – $\tilde{S}$ *works consistently on* $I$,

   – $\Sigma_{n=0}^{Conv(\tilde{S},I)}$*time to compute* $\tilde{S}(I^n) \leq pol(length(p))$.

(2) *There are a strategy* $S$ *and a polynomial pol such that*

   *(i)* $PAT \in$ *Poly*$-CONS-LEXTXT$ *by* $S$,

   *(ii) for every* $p \in Pat$, *there are uncountably many texts* $T \in$ Text$(L(p))$ *such that*

   – $S$ *works consistently on* $T$,

   – $\Sigma_{n=0}^{Conv(S,T)}$*time to compute* $S(T^n) \leq pol(length(p))$.

*Sketch of proof.* Assertion (1), part (i) is proved using the strategy $\tilde{S}$ from the proof of Theorem 11, Assertion (3) above. Then part (i) follows by Lemma 2 of Lange and Wiehagen (1991). Part (ii) directly follows from Theorem 2, Assertion (3) of Lange and Wiehagen (1991).

The first part of Assertion (2) is an immediate consequence of the proof of Theorem 1 in Lange and Wiehagen (1991), while the second follows as above. q.e.d.

We conjecture that Theorem 11 remains valid after replacing $INF$ by $TXT$. The corresponding Assertion (1) has been proved by Angluin (1980). As already mentioned above, the strategy from Lange and Wiehagen (1991) directly yields (3). Consequently, the only part not yet proved is (2). One reason why (2) seems to be easier provable for informant than for text is the following. A strategy working consistently on informant has to work "hard" at *any* step of the learning process in order to build its actual "biconsistent" hypothesis, while a consistent strategy on text may output the trivially consistent hypothesis $x_1$ for an "unbounded" number of steps.

## 5. Conclusions

We have investigated the problem of consistent versus inconsistent learning. In spite of the remarkable power of consistent learning it turns out that this power is not universal. There are learning problems which can exclusively be solved by inconsistent strategies, i.e., by strategies that do temporarily incorrectly reflect the behavior of the unknown object on data for which the correct behavior of the object is already known at the current stage of the learning process. This phenomenon has been investigated in a "highly theoretical" setting, namely in inductive inference of recursive functions. In this setting the seemingly senseless work of inconsistent strategies could be completely explained by the undecidability of consistency.

However, it turned out that the inconsistency phenomenon is also valid in more realistic situations, namely in domains where consistency is always decidable and the learning strategies have to work in polynomial time. The reason is quite analogous to that in the setting of arbitrary recursive functions. Providing $\mathcal{P} \neq \mathcal{NP}$, the $\mathcal{NP}$-hardness of problems can prevent learning strategies from producing consistent hypotheses in polynomial time. Note that the validity of our main result, Theorem 11, crucially depends on the fact that membership testing for pattern languages (hence also deciding consistency for hypotheses patterns, in general) is $\mathcal{NP}$-complete.

On the other hand, inspired by Wiehagen and Zeugmann (1992), Kummer (1992) has proved that the inconsistency phenomenon also holds in domains where the membership problem is even decidable in polynomial time for any single object to be learned, but *not uniformly* decidable in polynomial time with respect to the whole class of objects to be learned.

Finally, one can easily show that in domains where the membership problem is *uniformly* decidable in polynomial time, under weak assumptions any polynomial time learning algorithm can be effectively transformed into a polynomial time algorithm possessing at least the same learning power and being even consistent. Nevertheless, just in these domains we conjecture that there are learning problems solvable consistently in polynomial time, but solvable inconsistently (much) faster.

Moreover, we conjecture that our results may be extended to incremental learning of *finite* classes of *finite* objects such as Boolean functions, too.

In any case, we regard the results obtained as giving strong evidence to take fast *inconsistent* strategies seriously into account.

Finally, the presented results do suggest directions of further research such as

– finding fast inconsistent learning techniques,

– deriving conditions yielding that a given learning problem has no fast consistent solution, but it has a fast inconsistent one.

**Acknowledgements**

The first author has been supported by the German Ministry for Research and Technology (BMFT) within the joint project (BMFT–Verbundprojekt) GOSLER under grant 01 IW 101 B9 and E7.

Substantial part of this work has been performed while the first author was visiting TH Darmstadt. He is grateful indebted to Rüdiger Reischuk and Wolfgang Bibel for providing inspiring working conditions.

Both authors heartily acknowledge many enlightening discussions with Steffen Lange (HTWK Leipzig) and with Andreas Jakoby and Christian Schindelhauer (TH Darmstadt).

## 6. References

Angluin, D. (1980), Finding patterns common to a set of strings, *Journal of Computer and System Sciences* **21**, 46 – 62.

Angluin, D., and Smith, C.H. (1983), Inductive inference: theory and methods, *Computing Surveys* **15**, 237 – 269.

Angluin, D., and Smith, C.H. (1987), Formal inductive inference, *in* "Encyclopedia of Artificial Intelligence" (St.C. Shapiro, Ed.), Vol. 1, pp. 409 – 418, Wiley-Interscience Publication, New York.

Barzdin, J. (1974a), Inductive inference of automata, functions and programs, *in* "Proceedings International Congress of Math.," Vancouver, pp. 455 – 460.

Barzdin, J. (1974b), Две теоремы о предельном синтезе функций, *in* "Теория Алгоритмов и Программ," (J. Barzdin, Ed.), Vol.1, pp.82 – 88, Latvian State University.

Blum, L., and Blum, M. (1975), Toward a mathematical theory of inductive inference, *Information and Control* **28**, 122 – 155.

Blum, M. (1967), Machine independent theory of complexity of recursive functions, *Journal of the Association for Computing Machinery* **14**, 322 – 336.

Fulk, M. (1988), Saving the phenomena: requirements that inductive inference machines not contradict known data, *Information and Computation* **79**, 193 – 209.

Garey, M.R., and Johnson, D.S. (1979), "Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$–completeness," San Francisco, Freeman and Company.

Gold, M.E. (1965), Limiting recursion, *Journal of Symbolic Logic* **30**, 28 – 48.

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

Jantke, K.P. (1991a), Monotonic and non-monotonic inductive inference, *New Generation Computing* **8**, 349 – 360.

Jantke, K.P., and Beick, H.R. (1981), Combining postulates of naturalness in inductive inference, *Journal of Information Processing and Cybernetics (EIK)* **8/9**, 465 – 484.

Kearns, M., and Pitt, L. (1989), A polynomial-time algorithm for learning $k$–variable pattern languages from examples, *in* "Proceedings 1st Annual Workshop on Computational Learning Theory," (D. Haussler and L. Pitt, Eds.), pp. 196 –205, Morgan Kaufmann Publishers Inc., San Mateo.

Ko, Ker-I, Marron, A., and Tzeng, W.G. (1990), Learning string patterns and tree patterns from examples, *in* "Proceedings 7th Conference on Machine Learning," (B.W. Porter, and R.J. Mooney, Eds.), pp. 384 – 391, Morgan Kaufmann Publishers Inc., San Mateo.

Kummer, M. (1992), personal communication to T. Zeugmann.

Lange, S., and Wiehagen, R. (1991), Polynomial-time inference of arbitrary pattern languages, *New Generation Computing* **8**, 361 – 370.

Lange, S., and Zeugmann, T. (1992), Types of monotonic language learning and their characterization, *in* "Proceedings 5th Annual ACM Workshop on Computational Learning Theory," (D. Haussler, Ed.), pp. 377 – 390, ACM Press, New York.

Lange, S., and Zeugmann, T. (1993), Monotonic versus non-monotonic language learning, *in* "Proceedings 2nd International Workshop on Nonmonotonic and Inductive Logic," (G. Brewka, K.P. Jantke and P.H. Schmitt, Eds.), Lecture Notes in Artificial Intelligence Vol. 659, pp. 254 – 269, Springer-Verlag, Berlin.

Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (1984), "Machine Learning, An Artificial Intelligence Approach," Vol. 1, Springer-Verlag, Berlin.

Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (1986), "Machine Learning, An Artificial Intelligence Approach," Vol. 2, Morgan Kaufmann Publishers Inc., San Mateo.

Minicozzi, E. (1976), Some natural properties of strong-identification in inductive inference, *Theoretical Computer Science* **2**, 345 – 360.

Nix, R.P. (1983), Editing by examples, Yale University, Dept. Computer Science, Technical Report 280.

Osherson, D., Stob, M., and Weinstein, S. (1986), "Systems that Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists," MIT-Press, Cambridge, Massachusetts.

Porat, S., and Feldman, J.A. (1988), Learning automata from ordered examples, *in* “Proceedings 1st Workshop on Computational Learning Theory,” (D. Haussler and L. Pitt, Eds.), pp. 386 – 396, Morgan Kaufmann Publishers Inc., San Mateo.

Rogers, H.Jr. (1967), “Theory of Recursive Functions and Effective Computability,” McGraw–Hill, New York.

Schapire, R.E. (1990), Pattern languages are not learnable, *in* “Proceedings 3rd Annual Workshop on Computational Learning Theory,” (M.A. Fulk and J. Case, Eds.), pp. 122 – 129, Morgan Kaufmann Publishers, Inc., San Mateo.

Shinohara, T. (1982), Polynomial time inference of extended regular pattern languages, *in* “Proceedings RIMS Symposia on Software Science and Engineering,” Lecture Notes in Computer Science 147, pp. 115 – 127, Springer-Verlag, Berlin.

Smullyan, R.M. (1961), Theory of formal systems, *Annals of Math. Studies* **47**.

Solomonoff, R. (1964), A formal theory of inductive inference, *Information and Control* **7**, 1 – 22, 234 – 254.

Trakhtenbrot, B.A., and Barzdin, J. (1970) “Конечные Автоматы (Поведение и Синтез),” Наука, Москва,
English translation: “Finite Automata–Behavior and Synthesis, Fundamental Studies in Computer Science 1,” North-Holland, Amsterdam, 1973.

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

Wiehagen, R. (1978), Characterization problems in the theory of inductive inference, *in* “Proceedings 5th Colloquium on Automata, Languages and Programming,” (G. Ausiello and C. Böhm, Eds.), Lecture Notes in Computer Science 62, pp. 494 – 508, Springer-Verlag, Berlin.

Wiehagen, R. (1992), From inductive inference to algorithmic learning theory, *in* “Proceedings 3rd Workshop on Algorithmic Learning Theory,” (S. Doshita, K. Furukawa, K.P. Jantke and T. Nishida, Eds.), Lecture Notes in Artificial Intelligence 743, pp. 3 – 24, Springer-Verlag, Berlin.

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

Wiehagen, R., and Zeugmann, T. (1992), Too much information can be too much for learning efficiently, *in* “Proceedings 3rd International Workshop on Analogical and Inductive Inference,” ( K.P. Jantke, Ed.), Lecture Notes in Artificial Intelligence 642, pp. 72 – 86, Springer-Verlag, Berlin.

Wiehagen, R., and Zeugmann, T. (1994), Ignoring data may be the only way to learn efficiently, *Journal of Theoretical and Experimental Artificial Intelligence* **6**, 131 – 144.

Zeugmann, T. (1983), A–posteriori characterizations in inductive inference of recursive functions, *Journal of Information Processing and Cybernetics (EIK)* **19**, 559 – 594.

Zeugmann, T., Lange, S., and Kapur, S. (199x), Characterizations of monotonic and dual monotonic language learning, *Information & Computation* **120**, 1995, 155 – 173.