

Inductive Inference with Procrastination: Back to Definitions ^{*}

Andris Ambainis
Computer Science Division
University of California
Berkeley, CA 94720-1776 USA
e-mail: ambainis@cs.berkeley.edu

Rūsiņš Freivalds
Institute of Mathematics and Computer Science
University of Latvia
Raina bulv. 29, LV-1459 Riga, Latvia
e-mail: rusins@mii.lu.lv

Carl H. Smith [†]
Department of Computer Science
University of Maryland
College Park, MD 20742 USA
e-mail: smith@cs.umd.edu

Abstract

In this paper, we reconsider the definition of procrastinating learning machines. In the original definition of Freivalds and Smith [FS93], constructive ordinals are used to bound mindchanges. We investigate possibility of using arbitrary linearly ordered sets to bound mindchanges in similar way. It turns out that using certain ordered sets it is possible to define inductive inference types different from the previously known ones. We investigate properties of the new inductive inference types and compare them to other types.

^{*}This research was supported by Latvian Science Council Grant No.93.599 and NSF Grant 9421640. Some of the results from this paper were presented earlier [AFS96].

[†]The third author was supported in part by NSF Grant 9301339.

1 Introduction

We study inductive inference using the model developed by Gold [Gol67]. There is a well known hierarchy of larger and larger classes of learnable sets of phenomena based on the number of time a learning machine is allowed to change its conjecture. All of these bounds were specified a priori [CS83]. Procrastination is a technique introduced in [FS93] and studied further in [Amb95, Aps94, JS97] to specify the finite bound on the number of mind-changes dynamically during the learning process. It turned out that all the classes of learning sets of phenomena based on the dynamically determined mind change bound were larger than all the classes in the known mind change hierarchy. All classes, with either a predetermined or a dynamically determined bound are strictly smaller than the class obtained by considering inference in the limit as originally studied by Gold.

Inductive inference with procrastination is obtained from Gold's traditional model of inductive inference [Gol67] by allowing inductive inference machines to procrastinate about number of times they change their output conjecture. To formally describe such procrastination, Freivalds and Smith [FS93] used constructive ordinals. The use of constructive ordinals in this context provided a very general and flexible model with which to describe many different behaviors of learning machines within the same framework. In this paper we shall return to definitions of inductive inference with procrastination and consider the case of using an arbitrary ordered set instead of the constructive ordinals. We prove that there are particular ordered sets, with constructive descriptions, such that, if these sets are used instead of constructive ordinals, they give rise to new types of inductive inference with procrastination that cannot be described using only constructive ordinals. We investigate properties of such ordered sets and relations between sets which can be identified in the new model and those which can be learned in the previous model. Some of our results were independently rediscovered in [SSV97].

2 Preliminaries

2.1 Inductive inference of recursive functions

As is traditional in the study of inductive inference, we will use the basic model of learning in the limit developed by Gold [Gol67]. As targets of the learning procedure, we will use the recursive (totally computable) functions. Several authors have argued that this paradigm is sufficiently general to model, via suitable encodings, a large variety of real world learning situations [AS83].

In the sequel, we will consider the learning of a particular recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$. An Inductive Inference Machine (abbreviated as IIM) is an algorithmic device that receives as input the values of function $f(0), f(1), \dots$ and tries to guess some program computing the function f . An IIM outputs a sequence of conjectures, each computing some partial recursive function. The function f is identified by an IIM, if sequence of conjectures produced when using the range of f as input converges to a program computing f . A set of recursive functions U is identified by an IIM M if M identifies each function in U . The set U is *identifiable* if there exists IIM M which identifies this set. The collection of all sets which are identifiable will be denoted as EX .

2.2 Inductive inference with procrastination

An IIM makes a *mindchange* when it outputs a conjecture that is syntactically different from the previous one. Two cases have been considered. The first, called *learning the limit* is when the number of mindchanges is finite, but not bounded a priori. This notion is precisely the same as EX -identification as defined above. The second case arises when the number of allowed mindchanges is bounded a priori by some constant n . In this case, the learning process is assumed to stop when (and if) the n^{th} mind change is made. This restricted form of inference is called EX_n -identification. The classes EX_0, EX_1, \dots are defined analogously with the class EX . A well known result from [CS83] is that $EX_0 \subset EX_1 \subset \dots \subset EX_n \subset EX_{n+1} \subset \dots \subset EX$. Freivalds and Smith [FS93] proposed a model of identification that gives rise to classes contained in EX , but strictly larger than EX_n , for any n . Our new model, described below, is a generalization of the model used in [FS93].

We now proceed to define our general notion of inductive inference with procrastination. Consider a linearly ordered set A with ordering relation $<$. For the sake of simplicity, assume that A has a maximal element. In the case where A is well-ordered, it is also a constructive ordinal and the definition below coincides precisely with the one from [FS93]. Suppose that each inductive inference machine has a counter containing an element of A . With out loss of generality, we will assume that initially, the IIM's counter contains a maximal element of A . Each time the IIM makes a mindchange, the element in its counter is algorithmically replaced by a smaller element of A . If there exists no smaller element, then the IIM is blocked from making any mindchanges. The function f is identified by such IIM (we shall say: EX_A -identified) if the sequence of output conjectures converges to correct program. Notice, the due to the well-ordering of A , convergence is guaranteed. The collection of all sets of functions which are EX_A -identifiable will be denoted by EX_A .

We proceed to consider three simple examples of particular sets A . For the first example, consider $A_1 = \{1, 2, 3\}$ with the ordering relation $1 < 2 < 3$. For this example, the IIM can make only two mindchanges. Initially, the value 3 will be in its counter. At the time of the first mindchange the 3 is replaced by 2. At the second mindchange 2 is replaced by 1. No elements smaller than 1 are in A_1 , hence the IIM cannot replace the element in its counter and cannot make more mindchanges. It can be easily seen that each set of functions which can be identified with two mindchanges can be EX_{A_1} -identified, too. Hence, $EX_{A_1} = EX_2$.

For the second example, let A_2 be the set of all integers which are smaller than 0. For an ordering relation, use the standard less than relation on integers. Any IIM using this ordering will start with 0 in its counter. In this case, the number of mind changes made by the IIM is not restricted at all since $-n$ can always be replaced by $-(n + 1)$. Hence, $EX_{A_2} = EX$. This example illustrates what can happen if the set which is used for counting contains infinite descending chains.

For the final, more interesting example, consider $A_3 = \mathbb{N} \cup \{\star\}$. Again, use the standard less than relation on natural numbers augmented by $n < \star$ for all $n \in \mathbb{N}$. At the beginning \star is in the counter. At the time of the first conjecture the IIM has to replace the \star with some natural number n in the counter. Subsequently, each time the IIM makes a mindchange, it has to diminish element in its counter. This can be done at most n times.

Consequently, the IIM can make at most n mindchanges.

Notice that an IIM using values from A_3 in its counter behaves by first selecting how many mind changes it will make at the point when it outputs its first conjecture. A similar behavior is shown by some pattern language learning algorithms. More complicated examples of procrastination structure similar to the third example were given in [FS93]. Inductive inference with procrastination allows us to describe bounds on number of mindchanges. Using various ordered sets we can describe many different types of bounds on number of mindchanges.

2.3 Systems of notations

To effectively manipulate the elements of some infinite set we must assign notations to them. The notations should be assigned so that they allow us to algorithmically extract information from the notation. Sometimes, the natural choice of assigning notations is very simple. However, there can be cases when finding the most natural system is difficult, or even impossible.

Church and Kleene [Chu38, CK37, Kle38] proposed calling system of notations *acceptable* if it satisfied certain constraints. They defined an acceptable systems of notations for constructive ordinals. It should be noted that almost every natural way of algorithmically describing the constructive ordinals can be turned into an acceptable system of notations. Let Ω denote the set of all constructive ordinals. Below we modify the definition of Church and Kleene slightly so as to describe arbitrary ordered sets. Suppose that we have some linearly ordered set A . All elements of set A can be classified into three groups:

1. The minimal element: An element x is *minimal* if for all $y \in A$, if $y \neq x$ then $y < x$. There is at most one minimal element. Sometimes there will be none.
2. Successor elements: An element x is a *successor* if there exists an element y such that $y < x$ and there does not exist a z such that $y < z < x$. Successor elements have an element immediately preceding them in ordering $<$. For elements y and its successor x , we shall write $x = y + 1$.
3. Limit elements: All other elements of the set A are *limit* elements.

We inductively define a *system of notations* for A . This system mirrors the one defined in [Kle38].

Definition 1 *A system of notations S for a set A is a mapping v_S from a set of integers D_S onto the set A such that*

1. *There exists a partial recursive function k_S such that*
 - (a) *If $v_S(x)$ is the unique minimal element, then $k_S(x) = 0$;*
 - (b) *If $v_S(x)$ is a successor element then $k_S(x) = 1$;*
 - (c) *If $v_S(x)$ is a limit element then $k_S(x) = 2$.*
2. *There exists a partial recursive function p_S such that, if $v_S(x)$ is successor then $p_S(x)$ is defined and $v_S(x) = v_S(p_S(x)) + 1$.*
3. *There exists a partial recursive function q_S such that, if $v_S(x)$ is limit element then $q_S(x)$ is defined, $\varphi_{q_S(x)}$ is total function and $v_S(x) = \lim_n v_S(\varphi_{q_S(x)}(n))$.*

An important property of such systems is that there may be several different notations possible for the same element of A .

All of the previous papers on inductive inference with procrastination [Amb95, Aps94, FS93] considered only the case of using well-ordered sets (constructive ordinals) as mindchange counters. Freivalds and Smith [FS93] and Apsītis [Aps94] studied inductive inference with procrastination within one, fixed system of notations. The influence of a particular choice of a system of notations was investigated in [Amb95]. Case, Suraj and Jain [CJS95] used similar methods to bound the number of mindchanges for functions computable in limit.

In this paper, we consider the case when the set which is used as the mindchange counter is not well-ordered, e.g. it contains infinite descending chains. There are two possible cases. In the first case, the sequence of notations for some infinite descending chain can be generated algorithmically. In this case, an arbitrary EX -identifiable set of functions can be identified. (See the second example, above.) In the second case, there exist infinite descending chains of elements but notations for the chains cannot be generated algorithmically. Using ordered sets in such systems of notations we can define new inductive inference types which are different from previously known ones. This is the subject of section 3.

For the sake of brevity we call such ordered sets with a system of notations as described above *constructive pseudo-ordinals*. More precisely, a constructive pseudo-ordinal is an ordered set containing infinite descending chains of elements that can be described in some system of notation so that there is no recursive (i.e. computable) sequence of notations for any infinite descending chain of elements.

In all the theorems below, “system of notations for constructive pseudo-ordinal” means “system of notations for constructive pseudo-ordinals such that there is no recursive sequence of notations for any infinite descending chain of elements.”

3 Main results

In this section we give affirmative answers to three fundamental questions. Firstly, we show that there exist constructive pseudo-ordinals. Secondly, we show that using pseudo-ordinals as mindchange counters, we can identify some set that cannot be identified using only constructive ordinals for counting mindchanges. Finally, we show that there exist sets which are *EX*-identifiable but cannot be identified using constructive pseudo-ordinals to count mindchanges.

Theorem 1 *There exists a linearly ordered set A and a system of notations S such that:*

1. *There exists an infinite descending chain of elements in A ;*
2. *There is no infinite descending chain of elements of A for which notations in the system S can be generated algorithmically.*

Proof. Consider a set A consisting of all elements of the form either $a\omega + b$ or $\omega^2 - a\omega + b$ for $a, b \in \mathbb{N}$. Define an ordering on the set A so that

1. $a_1\omega + b_1 < a_2\omega + b_2$ if $a_1 < a_2$ or ($a_1 = a_2$ and $b_1 < b_2$);
2. $\omega^2 - a_1\omega + b_1 < \omega^2 - a_2\omega + b_2$ if $a_1 > a_2$ or ($a_1 = a_2$ and $b_1 < b_2$);
3. $a_1\omega + b_1 < \omega^2 - a_2\omega + b_2$ for all a_1, a_2, b_1, b_2 .

Next, we define a system of notations S for A . S will consist of notations α and $\alpha_{i_1, \dots, i_k} + n$ for all possible $k, i_1, \dots, i_k, n \in \mathbb{N}$. The notations are assigned to elements of A as follows:

1. α denotes ω^2 ;
2. $\alpha_{i_1, \dots, i_k} + n$ denotes
 - (a) n , if for some $j \in \{1, \dots, k\}$ the computation of $\varphi_j(j)$ terminates within k steps and $\varphi_j(j) \geq i_j$.
 - (b) $l\omega + n$, where l is smallest number such that for some $j \in \{1, \dots, k\}$ the computation of $\varphi_j(j)$ terminates in exactly $k + l$ steps and $\varphi_j(j) \geq i_j$.
 - (c) $\omega^2 - k\omega + n$, if, for all $j \in \{1, \dots, k\}$ $\varphi_j(j) < i_j$ or $\varphi_j(j)$ is undefined.

The next task is to define the functions k_S, p_S and q_S :

1. $k_S(x)$ is defined to be equal to
 - (a) 0, if $x = \alpha_{i_1, \dots, i_k} + 0$ and, for some $j \in \{1, \dots, k\}$, the computation of $\varphi_j(j)$ terminates within k steps and $i_j \leq \varphi_j(j)$.
 - (b) 1, if $x = \alpha_{i_1, \dots, i_k} + n$ and $n \neq 0$.
 - (c) 2, otherwise.
2. $p_S(\alpha_{i_1, \dots, i_k} + n)$ will be equal to $\alpha_{i_1, \dots, i_k} + (n - 1)$.
3. $q_S(\alpha_{i_1, \dots, i_k} + 0)$ will be a program computing sequence $\alpha_{i_1, \dots, i_k, 1} + 1, \alpha_{i_1, \dots, i_k, 2} + 2, \dots$

It is easy to see that definitions of k_S, p_S and q_S are consistent with the descriptions of the notations for elements of A . Each element of the set A has at least one notation. Hence, there exists infinite descending chains of elements. For example, we can take the chain of elements $\omega^2, \omega^2 - \omega, \omega^2 - 2\omega, \dots$ and the corresponding chain of notations $\alpha, \alpha_{i_1}, \alpha_{i_1, i_2}, \dots$ where i_1, i_2, \dots are such that, for all j , if $\varphi_j(j)$ is defined, then $i_j > \varphi_j(j)$. Hence, the first part of the theorem is proved. It remains to show that there does not exist a

recursive function g such that the sequence of notations $g(1), g(2), \dots$ denotes a descending chain of elements $\beta_1 > \beta_2 > \dots$ from the set A .

For a contradiction, assume that such a recursive function g exists.

First, notice that none of the elements of the infinite descending sequence denotes 0 or $l\omega + n$. (Otherwise, all elements after that would be of form $l\omega + n$ as well and there is no infinite descending sequence consisting of elements of this form.) This implies that, if a notation $\alpha_{i_1, \dots, i_k} + n$ appears in the sequence $g(0), g(1), \dots$ then, for every $j \in \{1, \dots, k\}$, either $\varphi_j(j) < i_j$ or $\varphi_j(j)$ is undefined.

Second, for every k , there must be an element of the sequence $\omega^2 - k'\omega + n$ with $k' > k$. (Otherwise, there would be the maximum k such that $\omega^2 - k\omega + n$ appears in the sequence. Then, all the elements in the sequence after $\omega^2 - k\omega + n$ must be smaller than $\omega^2 - k\omega + n$. However, they cannot be $\omega^2 - k'\omega + n'$ with $k' > k$. The only other elements of S that are smaller than $\omega^2 - k\omega + n$ are $\omega^2 - k\omega + n - 1, \omega^2 - k\omega + n - 2, \dots, \omega^2 - k\omega$ but there is only a finite number of them and, therefore, they cannot form an infinite decreasing sequence.)

From the definition of the system of notation S , we see that all notations for $\omega^2 - k'\omega + n$ are of the form $\alpha_{i_1, \dots, i_{k'}} + n$. Therefore, for every k , there must be an element of $g(0), g(1), \dots$ of the form $\alpha_{i_1, \dots, i_{k'}} + n$ with $k' > k$.

Now, consider the algorithm, which, given k , searches $g(0), g(1), \dots$, finds the first element $\alpha_{i_1, \dots, i_{k'}} + n$ with $k' \geq k$ and then outputs i_k . This algorithm computes a recursive function $h(k)$. Let m be the index for h , i.e. $\varphi_m = h$. Then, $\varphi_m(m) = i_m$ with i_m taken from the first element of the form $\alpha_{i_1, \dots, i_{k'}} + n$ with $k' > m$. However, this element must denote $\omega^2 - k'\omega + n$ and, therefore, we must also have $i_m > \varphi_m(m)$ or $\varphi_m(m)$ undefined. A contradiction. \square

We have proved that constructive pseudo-ordinals exist. The next two theorems show that, using constructive pseudo-ordinals, we can describe inductive inference types different from the previously known ones.

Theorem 2 *If A is a constructive pseudo-ordinal and S is a system of notations for A , then*

$$EX_S \not\subseteq \bigcup_{\alpha \in \Omega} EX_\alpha$$

Proof. Recall that we are assuming that A has no effective infinite descending chains. We will define a set U_0 that is in EX_S to prove the theorem. No

neat set theoretic definition of U_0 is known. Instead, we define an IIM M_0 that EX_S identifies a subset of the functions of finite support. This subset will be our U_0 . M_0 behaves as follows.

M_0 starts by inputting $f(0)$ and then producing its first conjecture, a program computing function

$$h(x) = \begin{cases} f(0) & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

M_0 continues reading input in the natural domain order $f(0), f(1), \dots$ until it finds $f(n) \neq 0$. M_0 then modifies the notation in its counter according to the following rules:

1. If the counter contains the notation x denoting a successor element, replace it by $p_S(x)$ (the notation for the preceding element);
2. If the counter contains the notation x denoting a limit element, replace it by $\varphi_{q_S(x)}(n)$.

After modifying the counter, M_0 outputs as a conjecture a program computing the function

$$h(x) = \begin{cases} f(x) & \text{if } x \leq n \\ 0 & \text{otherwise} \end{cases}$$

We prove by way of contradiction that U_0 cannot be identified by any IIM using constructive ordinal α to regulate mind changes. Suppose that M is such an IIM. We find a function from U_0 which is not identified by M .

We start by noneffectively choosing some infinite descending chain of elements of the set A , $\alpha_1, \alpha_2, \dots$. Suppose that α_1 is a maximal element of the set A . (If it is not so, we can replace α_1 with the maximal element of set A and $\alpha_1, \alpha_2, \dots$ will remain an infinite descending chain.) Next we noneffectively construct two sequences x_1, x_2, \dots and n_1, n_2, \dots . The sequence x_1, x_2, \dots will consist of notations in the system S . n_1, n_2, \dots will be a monotonically increasing sequence of natural numbers. Simultaneously, we construct a sequence of functions, f_1, f_2, \dots . x_1 will be some notation for α_1 . Let $n_1 = 0$. Define

$$f_1(x) = \begin{cases} 1 & \text{if } x = n_1, \\ 0 & \text{otherwise.} \end{cases}$$

We inductively define x_i, n_i and f_i for $i > 1$. We start by first defining a value m as follows.

1. If x_{i-1} denotes successor element, then set m equal to $n_{i-1} + 1$ and
2. If x_{i-1} denotes limit element, then let j denote the smallest number such that x_{i-1} denotes an element greater than α_j . Take as m the smallest number such that $m > n_{i-1}$ and $\varphi_{q_S(x_{i-1})}(m)$ denotes an element greater than α_j .

Consider the operation of M trying to learn f_{i-1} . If M does not identify f_{i-1} , then x_i, x_{i+1}, \dots and n_i, n_{i+1}, \dots will be undefined. If M identifies f_{i-1} , let n_i be such that $n_i \geq m$ and, after reading $f_{i-1}(0), \dots, f_{i-1}(n_i)$, M outputs a correct conjecture and does not change it later. x_i will be equal to $p_S(x_{i-1})$ if x_{i-1} denotes successor element and $\varphi_{q_S(x_{i-1})}(n_i)$ if x_{i-1} denotes limit element. Finally, define

$$f_i(x) = \begin{cases} 1 & \text{if } x = n_j \text{ for some } j \in \{1, \dots, i\} \\ 0 & \text{otherwise.} \end{cases}$$

The above procedure for choosing x_1, x_2, \dots and n_1, n_2, \dots is nonconstructive because we cannot effectively determine which elements of sequence $\varphi_{q_S(x_{i-1})}(0), \dots$ are greater than α_j .

Next, we show that M_0 can learn $\{f_i \mid i > 0\}$ and that this set cannot be learned by any IIM using a constructive ordinal to regulate mind changes. Toward this goal, we consider what happens when M_0 attempts to learn f_i . Using an induction argument, we show that:

After reading $f_i(0), \dots, f_i(n_i)$, the IIM M_0 outputs a correct conjecture and the notation in its counter, at that moment is equal to x_i .

If $i = 1$ we have $n_i = 0$. According to the definition of our IIM M_0 we have that, after reading the first value $f(0)$ M_0 outputs a program computing the function h such that $h(0) = f(0)$ and $h(x) = 0$ for $x > 0$. This program computes f_0 and is a correct conjecture. After issuing the first conjecture, the counter contains the notation for the maximal element of A , i.e. x_1 .

Suppose inductively that, after reading $f_i(0), \dots, f_i(n_i)$ the IIM M_0 outputs a correct conjecture and puts x_i in its counter. Consider what happens when M_0 next inputs f_{i+1} . All the values of f_{i+1} are the same as values of f_i except $f_{i+1}(n_{i+1})$. Hence, after reading $f_{i+1}(0), \dots, f_{i+1}(n_i)$ M_0 outputs a program computing f_i and puts x_i in its counter. According to the

definition of f_{i+1} , $f_{i+1}(n_i + 1), \dots, f_{i+1}(n_{i+1} - 1)$ are all zeros. So, M_0 will not change either its conjecture or the notation in counter while reading $f_{i+1}(n_i + 1), \dots, f_{i+1}(n_{i+1} - 1)$.

From the definition of f_{i+1} we know that $f_{i+1}(n_{i+1}) = 1$. Hence, after reading $f_{i+1}(n_{i+1})$, M_0 changes the value in its counter. If the notation in counter (i.e. x_i) denotes a successor element, then it is replaced by notation for the preceding element $p_S(x_i)$, i.e. x_{i+1} . If the notation x_i denotes a limit element, then it is replaced by $\varphi_{q_S(x_i)}(n_{i+1})$, i.e. x_{i+1} . After the change of notation in the counter, M_0 outputs a program computing the function f_{i+1} .

We have proved that M_0 after reading $f_{i+1}(0), \dots, f_{i+1}(n_{i+1})$ puts the notation x_{i+1} in its counter and outputs a correct conjecture, completing the induction.

To see that M_0 identifies f_i , for each $i \in \mathbb{N}$ notice that after reading $f_i(0), \dots, f_i(n_i)$, M_0 outputs a correct conjecture. If $x > n_i$, then $f_i(x) = 0$. Hence, M_0 does not change its conjecture later.

To complete that proof, we assume by way of contradiction that an IIM M , using a constructive ordinal to regulate mind changes, can learn all the functions f_i for all $i \in \mathbb{N}$. Consider the nonrecursive function

$$f(x) = \begin{cases} 1 & \text{if } x = n_i \text{ for some } i \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}$$

and assume that M receives $f(0), f(1), \dots$ as input. From the definition of n_i we have that for each $i \in \mathbb{N}$, after reading $f_{i-1}(0), \dots, f_{i-1}(n_i - 1)$ i.e. $f(0), \dots, f(n_i - 1)$ the IIM M outputs as conjecture a program computing f_{i-1} .

Hence, if M receives $f(0), f(1), \dots$ as input, it outputs infinitely many different conjectures: a program computing f_1 , a program computing f_2 , and so on. This means that M makes infinitely many changes of conjecture and, consequently, infinitely many modifications to the ordinal on its counter. However, infinite descending sequences of ordinals do not exist and, hence, M cannot make infinitely much modifications of the ordinal, even if it receives as input a nonrecursive function. This is a contradiction, completing the proof of the theorem. \square

Theorem 3 *If A is a constructive pseudo-ordinal and S is a system of notations for A , then*

$$EX_S \neq EX$$

Proof. We consider the set FS consisting of all functions of finite support (all functions f such that there is at most finite number of x satisfying $f(x) \neq 0$). It is well known that $FS \in EX$ [Gol67]. We prove that $FS \notin EX_S$ for any system of notations S for a constructive pseudo-ordinal. Let M be an IIM using S to regulate mind changes. Consider giving M input from the everywhere 0 function in the natural domain increasing order. Choose the least k such that, after reading $f_0(0), \dots, f_0(k)$, M issues its first conjecture.

We now define a partial recursive function f in effective stages of finite extension. f^s denotes the finite initial segment of f defined prior to stage s and x^s denotes the least value not in the domain of f^s . By way of initialization, set $f^0 = \{(x, 0) | x \leq k\}$. Hence, $x^0 = k + 1$.

Stage s : Simulate M on two functions $g_1(x)$ and $g_2(x)$:

$$g_1^s(x) = \begin{cases} f_1(x) & \text{if } x < x^s \\ 0 & \text{if } x \geq x^s \end{cases}$$

$$g_2^s(x) = \begin{cases} f_1(x) & \text{if } x < x^s \\ 1 & \text{if } x = x^s \\ 0 & \text{if } x > x^s \end{cases}$$

Continue the simulations until an $i \in \{1, 2\}$ and $n \geq x^s$ are found with $M(g_i^s(0), \dots, g_i^s(n)) \neq M(g_i^s(0), \dots, g_i^s(x^s - 1))$, a change of conjecture. If the i found is 1, then set $f(x^s) = 0$, if $i = 2$, set $f(x^s) = 1$. For all x such that $x^s < x \leq n$, set $f(x) = 0$.

End Stage s .

Clearly, f is a partial recursive function. Either f is recursive, or the domain of f is finite because a suitable i and n were not found during some stage.

Case 1. The domain of f is finite. Suppose that stage s is the least stage that does not terminate. Then the domain of f is $\{x | x < x^s\}$. Notice that both g_1^s and g_2^s are recursive functions. Furthermore, M on input from either of these two functions converges to $M(f(0), \dots, f(x^s - 1))$, as otherwise an i and n would have been found in stage s and stage s would have terminated. Both g_1^s and g_2^s are functions of finite support and M fails to identify at least one of them.

Case 2. f is a recursive function. Then each stage terminates. Hence, for each s , $M(f(0), \dots, f(x^s - 1)) \neq M(f(0), \dots, f(x^{s+1} - 1))$. In other words, M

on input from f makes infinitely many mind changes. However, if M makes infinitely many mindchanges on a recursive function, this means that there is an infinite computable sequence of descending notations. This contradicts the definition of constructive pseudo-ordinal. Therefore, case 2 is impossible. \square

In fact, by using similar methods, we can prove stronger result.

Definition 2 *A class U of total recursive functions is dense if for any initial segment $g(0), \dots, g(n)$ there exists a function $f \in U$ such that $f(0) = g(0), \dots, g(n) = f(n)$.*

Evidently, the class FS is dense. We can prove

Theorem 4 *If U is dense, A is constructive pseudo-ordinal, and S system of notations for A , then*

$$U \notin EX_S$$

Proof. Similar to Theorem 3. \square

Next, we compare our new identification types EX_S with known identification types EX_α for various constructive ordinals α . From Theorem 2 we already know that $EX_S \not\subset EX_\alpha$ for any α . The next theorem shows that the inclusion $EX_\alpha \subset EX_A$ is possible in some cases.

Theorem 5 *If $\alpha = a\omega + b$ for some integers a, b , then, for any ordered set A containing infinite descending chains and any system of notations S for elements of A*

$$EX_\alpha \subset EX_S$$

Proof. It follows from Theorem 6 below that

$$EX_\alpha \subset EX_{\omega^2} \subset EX_S$$

if the system P is used for ordinals ω^2 and α . The theorem follows from the result in [Amb95] that the identification type EX_α for $\alpha = a\omega + b$ does not depend on the system of notations that is used. \square

For larger ordinals, the inclusion $EX_\alpha \subset EX_A$ depends on the system of notations used for ordinals.

Let P be the system of notations for ordinals up to ω^2 consisting of notations $i\omega + j$, for $i, j \in \mathbb{N}$ and ω^2 , with k_P , v_P and q_P defined in the

natural way ($k_P(x) = 0$ for $x = 0$, $k_P(x) = 2$ for $x = i\omega$ and $x = \omega^2$ and $k_P(x) = 1$ otherwise). $v_P(i\omega + j) = i\omega + j - 1$. $q_P(\omega^2)$ is a program computing $\omega, 2\omega, \dots$ and $q_P(i\omega)$ is a program computing $(i - 1)\omega, (i - 1)\omega + 1, \dots$.

Theorem 6 *If the system of notations P is used for constructive ordinals then*

$$EX_{\omega^2} \subset EX_A$$

Proof. The proof is a generalization of the proof that every class of functions that is EX_α -identifiable for some $\alpha < \omega^2$ in the system P is also EX_α -identifiable in any other system of notations for constructive ordinals[Amb95]. First, we introduce some definitions that we need to generalize the result of [Amb95] to constructive pseudoordinals.

Call an element x of the ordered set S *large* if there exists an infinite descending chain of elements which are all less than x . x is *small* if it is not large. For each small element x , the set of all elements smaller than x does not contain an infinite descending chain, e.g. it is well-ordered. Hence, it is order-isomorphic to some constructive ordinal. We shall denote this ordinal with $u_S(x)$. We note that, if x is a minimal element, then there are no smaller elements and $u_S(x) = 0$.

Lemma 1 *There is an algorithm, which, given a notation a from the system A and a notation $i\omega + j$ from P , produces a notation b in the system A such that b denotes a smaller element than a and*

1. *If a is a notation for a large element, then b denotes either a large element or a small element such that $u_S(b) \geq i\omega + j$.*
2. *If a is a notation for a small element and $u_S(a) > i\omega + j$, then b denotes a small element and $u_S(b) \geq i\omega + j$.*

Proof. The algorithm is as follows:

1. Enumerate all finite sequences a_1, \dots, a_n of notations in A such that $a_1 = a$ and, for every $k \in \{2, \dots, n\}$, either $a_k = p_A(a_{k-1})$ (if a_{k-1} denotes a successor element) or $a_k = \varphi_{q_A(a_{k-1})}(m)$ for some $m \in \mathbb{N}$ (if a_{k-1} denotes a limit element).

2. When a sequence a_1, \dots, a_n satisfying one of the following conditions (a) and (b) is found, output a_2 .
 - (a) There are at least $i + 1$ notations for a limit element among a_2, \dots, a_n .
 - (b) There are exactly i notations for a limit element among a_2, \dots, a_n but none of a_2, \dots, a_{j+1} denotes a limit element.

Lemma 2 *If the algorithm terminates, then a_2 is either a large element or a small element greater than or equal to $i\omega + j$.*

Proof. We have to show that a_2 cannot denote a small element which is less than $i\omega + j$. Assume that a_2 denotes a small element. Then, a_3, a_4, \dots, a_n all denote small elements as well.

Let a_{k_1}, a_{k_2}, \dots be the notations for limit elements in the sequence a_2, \dots, a_n . Then, the last of them must be at least ω , the one before that must be at least 2ω and so on. Hence, if there are $i + 1$ limit elements, the first of them (a_{k_1}) is at least $(i + 1)\omega$. We have $k_1 \geq 2$. Therefore, a_2 denotes an element which is at least a_{k_1} (and, therefore, at least $(i + 1)\omega$).

If there are i notations for limit elements, a_{k_1} can denote $i\omega$ but $k_1 \geq j + 2$ and, therefore, the element denoted by a_2 is at least $i\omega + j$. (The element denoted by a_{j+2} is at least $i\omega$. This implies that the element denoted by a_{j+1} is at least $i\omega + 1$, the element denoted by a_j is at least $i\omega + 2$ and so on, the element denoted by a_2 is at least $i\omega + j$.) \square

Next, we show that the algorithm always terminates.

Lemma 3 *If a_1 is a notation for a large element or a small element that is more than $i\omega + j$, then there is a sequence satisfying one of conditions (a) and (b).*

Proof. We show a non-effective way of constructing such a sequence.

Let $a_1 = a$. Each next element, x_k is chosen as follows. If a_{k-1} denotes a successor element, we set $a_k = p_A(a_{k-1})$. If a_{k-1} denotes a limit element, we do:

1. If a_{k-1} denotes a large element, there is a m such that $\varphi_{q_A(a_{k-1})}(m)$ is large. (Otherwise, all elements smaller than a_{k-1} would be small and, then, a_{k-1} would be small as well.) We set $a_k = \varphi_{q_A(a_{k-1})}(m)$.

2. If a_{k-1} denotes a small element that is w^2 or larger, then we find m such that $\varphi_{q_A(a_{k-1})}(m)$ denotes $(i+1)\omega$ or larger ordinal and set $a_k = \varphi_{q_A(a_{k-1})}(m)$.
3. If a_{i-1} denote a small element equal to $i'\omega$ for some $i' \in \mathbb{N}$, we find m such that $\varphi_{q_A(a_{k-1})}(m)$ denotes $(i' - 1)\omega + j$ or larger ordinal and set $a_k = \varphi_{q_A(a_{k-1})}(m)$.

We stop if a_k becomes equal to the minimum element or if there are $i+1$ limit elements in the sequence. Note that one of these two things always happens. (Otherwise, the elements after the last limit element would form an infinite sequence where each next element is the predecessor of the previous element. The predecessor can be computed efficiently. Therefore, this would be a computable infinite descending sequence of notations in A . However, A is a constructive pseudoordinal and this means that such sequences do not exist.)

If a_1 denotes a large element, then all other elements of the sequence denote large elements as well. Therefore, a_k never becomes equal to the minimum element (which is small) and we get a sequence with $i+1$ limit elements.

If a_1 is a small element that is $(i+1)\omega + 1$ or greater, then the first limit element in the sequence is at least $(i+1)\omega$. The next limit element is at least $i\omega$ and so on. This means that we get a sequence with $i+1$ limit elements.

Finally, if a_1 is $(i+1)\omega$ or $i\omega + j'$ for $j' > j$, then the first limit element among a_2, \dots, a_n is $i\omega$, the next limit element is $(i-1)\omega$ and so on. This means that there are i limit elements among a_2, \dots, a_n : $i\omega, (i-1)\omega, (i-2)\omega, \dots, \omega$.

To prove that the condition (b) is true, it remains to show that none of a_2, \dots, a_{j+1} is a limit element. To prove this, notice that a_2 denotes $i\omega + j''$ for some $j'' \geq j$. (If a_1 denotes $i\omega + j'$, then a_2 is just $i\omega + j' - 1$. If a_1 denotes $(i+1)\omega$, the rules for choosing a_2 from the sequence $\varphi_{q_P(a_1)}(m)$ guarantee that a_2 is at least $i\omega + j$.)

This implies that a_3 denotes $i\omega + j'' - 1$, a_4 denotes $i\omega + j'' - 2$ and so on, with a_{j+1} denoting $i\omega + j'' - j + 1$. All of these are successor elements because $j'' \geq j$. \square

The set of all finite sequences a_1, a_2, \dots, a_n such that $a_1 = a$ and a_k is $p_A(a_{k-1})$ or $\varphi_{q_A(a_{k-1})}(m)$ for all $k \in \{2, \dots, n\}$ is recursively enumerable. Therefore, the algorithm can enumerate all such sequences and, at some

point, it finds the sequence of Lemma 3. Then, it terminates and outputs a_2 (which is correct answer by Lemma 2). \square

Lemma 1 allows us to transform an arbitrary EX_{ω^2} - identification algorithm (working in system of notations P) into EX_A - identification algorithm. All the conjectures of algorithm remain the same. Notations in the counter are transformed as follows:

1. If the EX_{ω^2} - algorithm puts ω^2 on the counter, then the EX_A - algorithm puts the notation for the maximal element on its counter.
2. When the EX_{ω^2} algorithm changes the ordinal notation on its counter to $i\omega + j$, the EX_A algorithm runs the subroutine from Lemma 1 with $i\omega + j$ and the notation a that is on the counter of EX_A -algorithm. The subroutine finds a notation b that denotes a smaller element than a . The EX_A -algorithm changes the notation on its counter from a to b .

Lemma 1 guarantees that the counter of the EX_A algorithm will always contain either a notation for large element or a notation for a small element a such that $u_S(a)$ is greater or equal to the ordinal in the counter of the EX_{ω^2} algorithm. Hence, each time the EX_{ω^2} algorithm changes the notation in its counter, the EX_A algorithm also makes a change. This means that the number of allowed mindchanges for the EX_A algorithm will be at least the same as for the EX_{ω^2} algorithm.

We have proven that the constructed EX_A algorithm will be equivalent to the given EX_{ω^2} algorithm. \square

Theorem 7 *There exists a system of notations R for constructive ordinals such that in system R ,*

$$EX_{\omega^2} \not\subseteq EX_A$$

Proof. Define

$$U_h = \{f \text{ is total recursive and there are at most } h(f(0)) - 1 \\ \text{values such that } x > 0 \text{ that } f(x) \neq 0\}$$

Definition 3 *The total function $f : \mathbb{N} \rightarrow \mathbb{N}$ is called computable as the limit of a monotone function if there exists total recursive function $g(x, y)$ such that for each fixed x , $g(x, y)$ is nondecreasing and converges to $f(x)$.*

The following lemma was proved in [Amb95].

Lemma 4 [Amb95] *If h is computable as the limit of a monotone function then*

$$U_h \in EX_{\omega^2}$$

for some system of notations R .

It remains to construct function h such that $U_h \notin EX_A$. Our construction is similar to the construction of such function for constructive ordinals in the proof of Lemma 7 in [Amb95]. We give it below.

Lemma 5 *There exists a function h which is computable as a limit of a monotone function such that*

$$U_h \notin EX_A$$

Proof. Let M_1, M_2, \dots be a list of all IIMs that use a system of notations S for the ordered set A to regulate mind changes. We suppose that each of these machines starts with a maximal element of A in its counter. Next we construct a function $g(x, i)$ so that it converges in the limit to $h(x)$ and $U_h \notin EX_A$. Our construction will guarantee that for each i , IIM M_i does not identify some function f such that $f \in U_h$ and $f(0) = i$. The following algorithm computes $g(i, x)$ and simultaneously constructs two functions f_1 and f_2 such that M_i can not identify both of them.

1. Set $j = k = 0$ and $m = 1$. Define

$$f_1(x) = \begin{cases} i & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

2. Simulate the first j steps of the computation of M_i on input from the function f_1 . If M_i does not output any conjectures, then set $g(i, j) = m$, $j = j + 1$ and repeat. If a conjecture is produced, go to the next step.

3. Let t be the conjecture produced in step 2 and k the number of values of the function f_1 that M_i needed to first produce the conjecture. Set $m = m + 1$, $g(i, j) = m$, and $j = j + 1$. Define:

$$f_2(x) = \begin{cases} f_1(x) & \text{if } x < k \\ 1 & \text{if } x \geq k \end{cases}$$

Go to the next step.

4. Simulate j steps of the computations of M_i on functions f_1 and f_2 , extending the domain of f_2 by adding points $(x, 0)$ to its graph as needed. If M_i does not change its conjecture from t on either one of these functions, then set $g(i, j) = m$ and $j = j + 1$. Repeat this step. If no mind change is found, then f_2 will be defined to be a total function in this step.
5. Suppose M_i changed its conjecture to t on input from $f_l (l \in \{1, 2\})$. Let k be the number of values of the function f_l that were input in order to produce the new conjecture. Define the new $f_1(x)$ and $f_2(x)$ as extensions of $f_l(x)$. Namely, let

$$f_1(x) = \begin{cases} f_l(x) & \text{if } x < k \\ 0 & \text{if } x \geq k \end{cases}$$

$$f_2(x) = \begin{cases} f_l(x) & \text{if } x < k \\ 1 & \text{if } x \geq k \end{cases}$$

Set $m = m + 1$, $g(i, j) = m$, $j = j + 1$, and go to 4.

If Step 5 is executed infinitely many times, we can construct a recursive function f on which M_i makes infinitely many mindchanges. This is impossible. (If there was such a function f , the notations that appear on the counter of M_i while reading f , would form an infinite sequence of descending notations and this sequence would be recursive because f is recursive. However, there is no recursive infinite descending sequences in the system of notations A .)

Hence, Step 5 is executed only a finite number of times. So, m is incremented a finite number of times and $h(i) = \lim_{x \rightarrow \infty} g(i, x)$ always exists.

Consider the last time when Step 5 is executed. After this time $f_1(x)$ and $f_2(x)$ are two different functions on which M_i outputs the same conjecture t and never later changes from t . Hence, M_i cannot identify both of these functions. It can be checked that while the algorithm works, $g(i, j) = m$ remains greater than number of x 's such that $f_1(x) \neq 0$ or $f_2(x) \neq 0$ and $x > 0$. Hence, the number of x 's such that $f_1(x) \neq 0$ or $f_2(x) \neq 0$ and $x > 0$ is less than $h(i)$ and $f_1, f_2 \in U_h$.

We have proved that, for arbitrary IIM M_i there exists a function which belongs to U_h but is not EX_S identified by M_i . \square

From Lemmas 4 and 5 the theorem follows. \square

References

- [AFS96] A. Ambainis, R. Freivalds, and C. Smith. General inductive inference types based on linearly ordered sets. In C. Puech and R. Reischuk, editors, *Proceedings of the 13th Symposium on the Theoretical Aspects of Computer Science, Lecture Notes in Computer Science*, volume 1046, pages 243–253. Springer, 1996.
- [Amb95] A. Ambainis. The power of procrastination in inductive inference: how it depends on used ordinal notations. In P. Vitányi, editor, *Proceedings of EuroCOLT'95, Lecture Notes in Artificial Intelligence*, volume 904, pages 99–111. Springer, 1995.
- [Aps94] K. Apsītis. Derived sets and inductive inference. In S. Arikawa and K. Jantke, editors, *Proceedings of AII'94, Lecture Notes in Artificial Intelligence*, volume 872, pages 26–39. Springer, 1994.
- [AS83] D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.
- [Chu38] A. Church. The constructive second number class. *Bulletin of the AMS*, 44:224–232, 1938.
- [CJS95] J. Case, S. Jain, and M. Suraj. Not-so-nearly-minimal-size program inference. In K. Jantke, editor, *Algorithmic Learning for Knowledge-Based Systems, Lecture Notes in Artificial Intelligence*, volume 961, 76–95, Springer, 1995.

- [CK37] A. Church and S. Kleene. Formal definitions in the theory of ordinal numbers. *Fundamenta Mathematicae*, 28:11–21, 1937.
- [CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25(2):193–220, 1983.
- [FS93] R. Freivalds and C. Smith. On the power of procrastination for machine learning. *Information and Computation*, 107:237–271, 1993.
- [Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [JS97] S. Jain and A. Sharma. Elementary formal systems, intrinsic complexity, and procrastination. *Information and Computation*, 132:65–84, 1997.
- [Kle38] S. Kleene. On notation for ordinal numbers. *Journal of Symbolic Logic*, 3:150–155, 1938.
- [SSV97] A. Sharma, F. Stephan, and Y. Ventsov. Generalized notions of mind chance complexity. In *Proceedings of 10th Annual ACM Conference on Computational Learning Theory*, pages 96–108, 1997.