Latvijas Universitāte•University of Latvia

Inductive Inference and Constructive Ordinals

Dr. Sc. Comp. Dissertation

Andris Ambainis
Institute of Mathematics and Computer Science
University of Latvia
Raina bulv.29, Riga, LV-1459
Latvia

Rīga, 1997

Acknowledgements.

Interaction with other people was crucial to the success of this research. First, a lot of thanks to my advisor, Rūsiņš Freivalds for bringing me into theoretical computer science research and helping whenever possible. Thanks to Agnis Andžāns for working with me while I was in high school and sending me to Rūsiņš Freivalds later.

My coauthors Sanjay Jain and Arun Sharma helped to improve the content of chapters 3 and 4 greatly. Research described in this thesis also benefited from discussions and comments by Kalvis Apsītis, John Case, Dick de Jongh, Frank Stephan, Mahe Velauthapillai and Manfred Warmuth. A lot of comments by anonomous referees at various conferences was very useful. Thanks to Mark Changizi, Martin Kummer and Carl Smith for sending me their papers and to Bob Daley and Balya Kalyanasundaram for writing [16] that inspired me and was the starting-point for chapter 6.

The financial support for this research was provided by Latvia Science Council Grants 93.599 and 96.0282 and fellowship "SWH Izglītībai, zinātnei un kultūrai" from Latvia Education Foundation. The results of this thesis were presented at EuroCOLT'95, COLT'96, EuroCOLT'97. Financial support for participiation in these conferences was provided by Soros Foundation Latvia (for EuroCOLT'95) and the organizers of conferences.

Finally, I thank all my colleagues, friends and my family for everything good that happened during these years.

Contents

Chapter 1

Introduction

The topic of this thesis is applications of well-orderings and ordinals in inductive inference. Inductive inference is a branch of theoretical computer science that studies the process of learning on a very general level[25, 7, 41].

Traditionally, inductive inference studies the learning of arbitrary recursive functions (or languages). A learning algorithm receives data about an unknown function (language) and outputs a sequence of conjectures about this fuction. Each conjecture is a program in a general programming language computing some function (or a grammar for a language). The learning algorithm succeeds when it outputs a correct program (grammar).

Various aspects of this model have been studied[41]. Ordinals have found two very different applications in inductive inference.

First, ordinals can be used as counters. Ordinal counters are generalizations of counters that use natural numbers.

Most frequently, they are used to count mindchanges. A mindchange is an event when the learning machine changes its conjecture (by outputting a program different from the previous conjecture). The number of mindchanges can be considered as a measure of complexity for inductive inference[11, 23].

Most of researchers consider only one type of complexity bounds on the number of mindchanges: constant bounds. Constant bounds are established by requiring that the learning algorithm makes at most c mindchanges where c is a constant that is the same for all functions. However, there are situations which cannot be described by constant bounds.

In particular, such bounds do not take into account scenarios in which a learning machine, after examining an element of the language is in a position to issue a bound

on the number of mind changes it will make before the onset of convergence. For example, consider the class

$$COINIT = \{L \mid (\exists n)[L = \{x \mid x \geq n\}]\}.$$

Intuitively, $COINIT$ is the collection of languages that contain all natural numbers except a finite initial segment. Clearly, a learning machine that, at any given time, finds the minimum element n in the data seen so far and emits a grammar for the language $\{x \mid x \geq n\}$ learns $COINIT$ in the limit from positive data. It is also easy to see that the class $COINIT$ cannot be identified by any machine that is required to converge within a constant number of mind changes. However, the machine identifying $COINIT$ can, after examining an element of the language, issue an upper bound on the number of mind changes.

In this example, the number of mindchanges is bounded but this bound is not a constant bound. An another example is the class of pattern languages ($PATTERN$), first introduced by Angluin [6]. Such scenarios can be modeled by the use of constructive ordinals as mind change counters introduced by Freivalds and Smith [24]. Use of constructive ordinals provides a very general and flexible model that can be used to describe a lot of different behaviours of learning machines.

This thesis investigates ordinal bounds on the number of mindchanges in three directions.

First, we give the ordinal mind change bounds for identification in the limit of unions of pattern languages from both positive and negative data (informants). We describe these results in chapter 3.

Second, we investigate conditions under which an ordinal mind change bound can be guaranteed. We first establish a useful technical result which states that if a learning machine makes a finite number of mind changes on any text, then the class of languages that can be identified by this machine has an ordinal mind change bound. This result allows us to derive various sufficient conditions for the existence of ordinal bounds. These conditions involve different notions like finite thickness, finite elasticity, conservativeness, etc.. These results are described in chapter 4.

Third, we investigate the dependence of ordinal bounds on the particular notation for ordinals. Usually, notation is regarded as something unimportant and it is expected that all results will be true (or false) no matter what notation is used. However, the situation is different with ordinals.

There exist many nonequivalent systems of notations for constructive ordinals.

The particular system of notations is very important in many computational models involving ordinals[13, 21]. The power of a machine can change dramatically when the system of notations changes. We show that the learning power is not influenced by the system of notations only when small ordinals (below ω^2) are used. For greater ordinals, the system of notations has very large influence. These and other results about systems of notations are described in chapter 5.

The second application of constructive ordinals is probabilistic and team learning (chapter 6). Here, ordinals are used to resolve difficult problems in an unexpected way.

We consider finite identification of recursive functions. In this model, the machine can output only one program and it must be correct. This seems much simpler than the identification in the limit where an unlimited number of conjectures is allowed. However, if we consider probabilistic and team learning[28, 51], the situation is just the opposite.

It is well known that teams of machines can identify larger classes of functions than single machines. Identification by probabilistic machines is closely related to identification by teams because any team can be simulated by a probabilistic machine. Teams of different size and probabilistic machines with different probabilities of success have different learning power. Previous research[22, 18, 17] has shown that relations between teams and probabilistic machines with different characteristics are very complicated. Finite identification by teams and probabilistic machines has been studied for 18 years. Still, we are far from the complete understanding of the situation.

In chapter 6, we consider **PFin**, a restricted version of finite identification. The structure of different **PFin**-teams and probabilistic **PFin**-machines is less complicated than the similar structure for unrestricted finite identification. However, it is complicated, too. Researchers have come to a conclusion that it is unlikely that it will be possible to determine all probabilities at which the learning power of probabilistic **PFin**-machines is different[16].

We propose a different approach. Instead of determining these probabilities explicitly, we study global properties of the probability structure. We prove that the set of different probabilities is well-ordered and has a system of notations. Then, we give an algorithm that receives two probabilities p_1 and p_2 and answers whether the any machine with probability of success p_1 can be simulated by a machine with probability of success p_2. Well-orderedness and the system of notations is crucial

for the construction of decision algorithm.

The precise ordering type of this set is ϵ_0, a huge ordinal that is order-equivalent to the set of all expressions possible in first-order arithmetic. This result shows that the probability structure is very complicated. Very general methods are required to deal with so complex structures. Ordinals and systems of notations give us such methods.

In the next chapter, we give precise definitions of problems analysed in this thesis. Then, in chapters 3, 4, 5 and 6, we give our results. At the beginning of each chapter, we give a more detailed survey of results in this chapter.

The results in chapters 3 and 4 were published in [5]. These results were obtained together with Sanjay Jain and Arun Sharma. The results in chapter 5 appeared in [1] and the results in chapter 6 appeared in [2].

Chapter 2

Technical preliminaries

2.1 Notation

We use standard recursion-theoretic concepts[45, 46, 52]. $\mathbb{N} = \{0, 1, \ldots\}$ denotes the set of all natural numbers, $\mathbb{N}^+ = \{1, 2, \ldots\}$ denotes the set of all positive integers, $\mathbb{Q}$ denotes the set of rational numbers and $\mathbb{R}$ denotes the set of real numbers. The symbols $\subseteq, \supseteq, \subset, \supset$, and $\emptyset$ denote subset, superset, proper subset, proper superset, and the emptyset, respectively. $\langle , \rangle$ denotes one-to-one and onto pairing function from $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.

$\varphi_0, \varphi_1, \ldots$ is a fixed acceptable programming system[38, 44]. In chapters about language identifications (chapters 3 and 4), it is an acceptable programming system for all recursively enumerable languages. In chapters considering function identification (chapters 5 and 6), it is an acceptable programming system for all partial recursive functions. φ_i is the partial recursive function computed by the i^{th} program in φ (or the language recognized by the i^{th} grammar in φ, if the language identification is considered).

2.2 Paradigms of inductive inference

In this thesis, we consider three identification paradigms: language identification in the limit (chapters 3 and 4), identification of recursive functions in the limit (chapter 5) and finite identification of recursive functions (chapter 6). Several authors have argued that these paradigms is sufficiently general to model, via suitable encodings, a large variety of real world learning situations [7, 11, 25, 41].

2.2.1 Language identification in the limit

L denotes a typical variable for a language. $\overline{L}$ denotes the complement of L, that is, $\overline{L} = \mathbb{N} - L$.

We first define the notion of texts for languages.

Definition 1

(a) A *text* for a language L is a mapping T from $\mathbb{N}$ into $(\mathbb{N} \cup \{\#\})$ such that L is the set of natural numbers in the range of T.

(b) $content(T)$ denotes the set of natural numbers in the range of T.

(c) The initial sequence of text T of length n is denoted $T[n]$.

(d) The set of all finite initial sequences of $\mathbb{N}$ and #'s is denoted SEQ.

Intuitively, a text T for a language L is a presentation of elements of L (possibly repeated) and no non-elements of L; #'s in the presentation may be thought of as modeling pauses in data input[1]. It is easy to see that there exists a computable bijection between SEQ and $\mathbb{N}$. Members of SEQ are inputs to machines that learn grammars (acceptors) for r.e. languages. We let σ and τ, with or without decorations[2], range over SEQ. Λ denotes the empty sequence. content(σ) denotes the set of natural numbers in the range of σ and length of σ is denoted $|\sigma|$. We say that $\sigma \subseteq \tau$ ($\sigma \subseteq T$) to denote that σ is an initial sequence of τ (T).

Definition 2 A language learning machine is an algorithmic mapping from SEQ into $\mathbb{N} \cup \{?\}$.

M denotes a typical variable for a language learning machine. We also fix an acceptable programming system and interpret the output of a language learning machine as the index of a program in this system. Then, a program conjectured by a machine in response to a finite initial sequence may be viewed as a candidate accepting grammar for the language being learned. $\mathbf{M}(\tau)$ is the program conjectured by **M** after reading τ.

A conjecture of "?" by a machine is interpreted as "no guess at this moment." This is useful to avoid biasing the number of mind changes of a machine. For this

[1] Note that the only text for the empty language is an infinite sequence of #'s.

[2] Decorations are subscripts, superscripts and the like.

paper, we assume, without loss of generality, that $\sigma \subseteq \tau$ and $\mathbf{M}(\sigma) \neq ?$ implies $\mathbf{M}(\tau) \neq ?$.

We say that $\mathbf{M}$ converges on text T to i (written: $\mathbf{M}(T)$ converges to i) just in case for all but finitely many n, $\mathbf{M}(T[n]) = i$. The following definition introduces Gold's criterion for successful identification of languages.

Definition 3 [25]

(a) **M TxtEx**-*identifies* a text T if $\mathbf{M}(T)$ converges to a grammar for content(T).

(b) **M TxtEx**-*identifies* an r.e. language L (written: $L \in \mathbf{TxtEx}(\mathbf{M})$) just in case **M TxtEx**-identifies each text T for L.

(c) **TxtEx** denotes the set of all collections $\mathcal{L}$ of r.e. languages such that some machine **TxtEx**-identifies each language in $\mathcal{L}$.

The next two definitions describe the notion of informants as a model of both positive and negative data presentation and identification in the limit from informants.

Definition 4 An informant for L is an infinite sequence (repetitions allowed) of ordered pairs such that for each $n \in \mathbb{N}$ either $(n, 1)$ or $(n, 0)$ (but not both) appear in the sequence and $(n, 1)$ appears only if $n \in L$ and $(n, 0)$ appears only if $n \notin L$.

I denotes a typical variable for informants. $I[n]$ denotes the initial sequence of informant I with length n. content(I) $= \{(x, y) \mid (x, y)$ appears in sequence $I\}$. content($I[n]$) is defined similarly.

$$\text{PosInfo}(I[n]) = \{x \mid (x, 1) \in \text{content}(I[n])\}.$$

$$\text{NegInfo}(I[n]) = \{x \mid (x, 0) \in \text{content}(I[n])\}.$$

We now define identification from both positive and negative data.

Definition 5 [25]

(a) **M InfEx**-*identifies* an r.e. language L just in case **M**, fed any informant for L, converges to a grammar for L. In this case we say that $L \in \mathbf{InfEx}(\mathbf{M})$.

(b) **M InfEx**-*identifies* a collection of languages, $\mathcal{L}$, just in case **M InfEx**-identifies each language in $\mathcal{L}$.

(c) **InfEx** denotes the set of all collections $\mathcal{L}$ of r.e. languages such that some machine **InfEx**-identifies $\mathcal{L}$.

2.2.2 Function identification in the limit

In this paradigm, the object that is learned by an IIM is a recursive (totally computable) function. IIM receives the values of function $f(0)$, $f(1)$, $\cdots$ as the input.

Definition 6 [25]

(a) **M Ex**-*identifies* an recursive function f just in case **M**, fed $f(0)$, $f(1)$, ..., converges to a program for f. In this case we say that $L \in \mathbf{Ex(M)}$.

(b) **M Ex**-*identifies* a collection of functions, U, just in case **M Ex**-identifies each function in U.

(c) **Ex** denotes the set of all collections U of recursive functions such that some machine **Ex**-identifies U.

2.2.3 Finite identification of functions

Identification in the limit allows an unlimited number of conjectures. On the contrary, finite identification allows only one conjecture on each input.

Definition 7 [22]

(a) **M Fin**-*identifies* an recursive function f just in case the first program issued by M on the input $f(0)$, $f(1)$, ... computes f. In this case we say that $L \in \mathbf{Fin(M)}$.

(b) **M Fin**-*identifies* a collection of functions, U, just in case **M Fin**-identifies each function in U.

(c) **Fin** denotes the set of all collections U of recursive functions such that some machine **Fin**-identifies U.

We consider **PFin**, a restricted version of **Fin**.

Definition 8 A machine M is *Popperian* iff all programs issued by M on al inputs compute total recursive functions.

Definition 9 (a) **M PFin**-*identifies* a collection of functions U iff **M** is Popperian and **M Fin**-identifies each function in U.

(b) **PFin** denotes the set of all collections U of recursive functions such that some machine **PFin**-identifies U.

2.3 Well-orderings and ordinals

A linear ordering is *well-ordering* if it does not contain infinite descending sequences. *Ordinals*[48] are standard representations of well-orderings.

The ordinal 0 represents the ordering type of the empty set, the ordinal 1 represents the ordering type of any 1 element set, the ordinal 2 represents the ordering type of any 2 element set and so on. The ordinal ω represents the ordering type of the set $\{0, 1, 2, \ldots\}$. The ordinal $\omega + 1$ represents the ordering type of $\{0, 1, 2, \ldots\}$ followed by element ω. The ordinal $\omega \cdot 2$ represents the ordering type $\{0, 1, 2, \ldots\}$ followed by $\{\omega, \omega + 1, \omega + 2, \ldots\}$. Greater ordinals can be defined similarly (cf.[48]). We use arithmetic operations on ordinals defined in two different ways.

Definition 10 [36] Let A and B be two disjoint sets, α be the ordering type of A and β be the ordering type of B.

(a) $\alpha + \beta$ is the ordering type of $A \cup B$ ordered so that $x < y$ for any $x \in A$, $y \in B$ and order is the same within A and B.

(b) $\alpha \cdot \beta$ is the ordering type of $A \times B$ ordered so that $(x_1, y_1) < (x_2, y_2)$ iff $x_1 < x_2$ or $x_1 = x_2$ and $y_1 < y_2$.

We note that both sum and product of ordinals are not commutative. For example, we have $1 + \omega = \omega \neq \omega + 1$ and $2 \cdot \omega = \omega \neq \omega \cdot 2$.

Definition 11 [36] $\alpha - \beta$, the difference of α and β is an ordinal γ such that $\alpha = \beta + \gamma$.

$\alpha - \beta$ always exists and is unique[36]. We also use the natural sum and the natural product of ordinals. These operations use the representation of ordinals as exponential polynomials. In this paper, we consider only ordinals which are less than or equal to

$$\epsilon_0 = \lim(\omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \ldots).$$

If $\alpha < \epsilon_0$, then

$$\alpha = \omega^{\alpha_1} \cdot c_1 + \ldots + \omega^{\alpha_n} \cdot c_n$$

where $\alpha_1, \ldots, \alpha_n$ are smaller ordinals and $c_1, c_2, \ldots, c_n$ are natural numbers. If we require that $\alpha_1 > \alpha_2 > \ldots$ and $c_1, \ldots, c_n$ are nonzero, this representation is unique.

Definition 12 [36] Let

$$\alpha = \omega^{\alpha_1} \cdot c_1 + \ldots + \omega^{\alpha_n} \cdot c_n$$

$$\beta = \omega^{\alpha_1} \cdot d_1 + \ldots + d_n\omega^{\alpha_n} \cdot d_n$$

(a) The natural sum of α and β is

$$\alpha(+)\beta = \omega^{\alpha_1} \cdot (c_1 + d_1) + \ldots + \omega^{\alpha_n} \cdot (c_n + d_n).$$

(b) $\alpha(\cdot)\beta$, the natural product of α and β is the product of base ω representations as polynomials. $\omega^{\alpha_i}(\cdot)\omega^{\alpha_j} = \omega^{\alpha_i(+)\alpha_j}$ and $\alpha(\cdot)\beta$ is the natural sum of $\omega^{\alpha_i(+)\alpha_j} \cdot c_i d_j$ for all i, j.

Natural sum and natural product are commutative. They can be used to bound the ordering type of unions.

Theorem 1 *Let $A_1, \ldots, A_s$ be arbitrary subsets of a well-ordered set A, $\alpha_1, \ldots, \alpha_s$ be the ordering types of $A_1, \ldots, A_s$ and α be the ordering type of $A_1 \cup \ldots \cup A_s$. Then,*

$$\alpha \leq \alpha_1(+)\alpha_2(+)\ldots(+)\alpha_s.$$

The difference between this theorem and Definition 10 is that Definition 10 requires $x < y$ for all $x \in A$, $y \in B$ but Theorem 1 has no such requirement. Next, we give a similar application of the natural product.

Theorem 2 *Let $A_1, \ldots, A_s$ and A be well-ordered sets with ordering types $\alpha_1, \ldots, \alpha_s$ and α, respectively. Assume that $f : A_1 \times A_2 \times \ldots \times A_s \rightarrow A$ is a strictly increasing function onto A, i.e.*

$$f(\alpha_1, \ldots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \ldots, \alpha_s) < f(\alpha_1, \ldots, \alpha_{i-1}, \alpha_i', \alpha_{i+1}, \ldots, \alpha_s)$$

for all $i \in \{1, \ldots, s\}$ and $\alpha_i < \alpha_i'$. Then

$$\alpha \leq \alpha_1(\cdot)\alpha_2(\cdot)\ldots(\cdot)\alpha_s.$$

Both Theorem 1 and 2 will be used in section 6.9. *Transfinite induction* is a generalization of the usual mathematical induction.

Theorem 3 *[36. Principle of transfinite induction] Let A be a well-ordered set and $P(x)$ be a predicate. If*

(a) $P(x)$ is true when x is the smallest element of A, and

(b) $P(y)$ for all $y \in A$ which are smaller than x implies $P(x)$,

then $P(x)$ for all $x \in A$.

2.4 Ordinals as mindchange counters

Definition 13 **F**. an algorithmic mapping from SEQ into constructive ordinals, is an *ordinal mind change counter function* just in case $(\forall \sigma \subseteq \tau)[\mathbf{F}(\sigma) \succeq \mathbf{F}(\tau)]$.

Definition 14 [24] Let α be a constructive ordinal.

(a) We say that **M**. with associated ordinal mind change counter function **F**, $\mathbf{TxtEx}_\alpha$-*identifies* a text T just in case the following three conditions hold:

(i) $\mathbf{M}(T)$ converges to a grammar for content(T),

(ii) $\mathbf{F}(\Lambda) = \alpha$ and

(iii) $(\forall n)[? \neq \mathbf{M}(T[n]) \neq \mathbf{M}(T[n+1]) \Rightarrow \mathbf{F}(T[n]) \succ \mathbf{F}(T[n+1])]$.

(b) **M**. with associated ordinal mind change counter function **F**, $\mathbf{TxtEx}_\alpha$-*identifies* L (written: $L \in \mathbf{TxtEx}_\alpha(\mathbf{M}, \mathbf{F})$) just in case **M**, with associated ordinal mind change counter function **F**, $\mathbf{TxtEx}_\alpha$-identifies each text for L.

(c) $\mathbf{TxtEx}_\alpha = \{\mathcal{L} \mid (\exists \mathbf{M}, \mathbf{F})[\mathcal{L} \subseteq \mathbf{TxtEx}_\alpha(\mathbf{M}, \mathbf{F})]\}$.

IIM with an ordinal mindchange counter can be defined in a formally different but equivalent manner. Namely, we can assume that the machine **M** operates a counter α containing an ordinal. Before each mindchange, M replaces the counter on the ordinal by a smaller one.

Essentially, this is the same definition expressed in a less formal way. The counter α is a counterpart to the function F of Definition 14. It is easy to see that both definitions are equivalent.

We use both definitions. More formal definition appears more appropriate for chapters 3 and (especially) 4. The second, informal definition, is used in chapter 5.

Similarly to the above definitions, we can define $\mathbf{InfEx}_\alpha$ to denote classes of languages that can be identified from informants with α as the ordinal mind change bound. We also define $\mathbf{Ex}_\alpha$ to denote classes of functions that can be identified with α as the ordinal mind change bound. The following simple lemma will be useful in our proofs.

Lemma 1 *If* **M** *is an IIM with the number of mindchanges bounded by an ordinal* α, *then* M *makes finitely many mindchanges on any (even nonrecursive) input.*

Proof. By the way of contradiction. If **M** makes infinitely many mindchanges, it decreases its ordinal counter infinitely many times. Let α_0 be the first ordinal on the counter, α_1 be the ordinal which appears on the counter after α_0 and so on. According to the definition of IIM

$$\alpha_0 > \alpha_1 > \alpha_2 > \ldots.$$

Hence, $\alpha_0, \alpha_1, \ldots$ is an infinite decreasing sequence of ordinals.

It is well known that infinite decreasing sequences of ordinals do not exist. A contradiction. ∎

2.5 Systems of notations

2.5.1 Definitions

Ordinal numbers can be classified into three types:

(a) The ordinal 0;

(b) Ordinals having an immediate predecessor in ordering of all ordinals, such ordinals are called *successor* ordinals;

(c) Ordinals having no immediate predecessor, such ordinals are called *limit* ordinals.

In this paper we consider only those ordinals which can be described in some constructive way (*constructive ordinals*).

A *system of notations* is a method of assigning notations to ordinals. A system of notations is considered to be acceptable if it allows to extract certain information from notations and to perform certain operations on notations. More formally,

Definition 15 A *system of notations* S is a mapping v_S from a set of integers D_S onto a segment of ordinal numbers such that

(a) There exists a partial recursive function k_S such that

(i) If $v_S(x) = 0$ then $k_S(x) = 0$;

(ii) If $v_S(x)$ is a successor ordinal then $k_S(x) = 1$;

(iii) If $v_S(x)$ is a limit ordinal then $k_S(x) = 2$.

(b) There exists a partial recursive function p_S such that, if $v_S(x)$ is a successor ordinal then $p_S(x)$ is defined and $v_S(x) = v_S(p_S(x)) + 1$.

(c) There exists a partial recursive function q_S such that, if $v_S(x)$ is a limit ordinal then $q_S(x)$ is defined. $\varphi_{q_S(x)}$ is a total function and $v_S(\varphi_{q_S(x)}(0))$, $v_S(\varphi_{q_S(x)}(1))$, ... denotes an increasing sequence of ordinals converging to $v_S(x)$.

The members of D_S are called *notations*.

Definition 16 An ordinal α is *constructive ordinal* if there is a system of notations which assigns at least one notation to α.

Definition 17 A system of notations S is *univalent* if v_S is a one-to-one function (each ordinal has at most one notation).

Next, we define some effective operations on ordinal notations which will be used further. These operations are defined for an arbitrary system of notations S.

Each ordinal α can be expressed as $\alpha' + n$ where α' is zero or a limit ordinal and n is a finite ordinal. We can extract notations for α' and n from a notation for α using the following functions $L(\alpha)$ and $N(\alpha)$:

$$L(\alpha) = \begin{cases} \alpha & \text{if } \alpha \text{ is zero or a limit ordinal} \\ L(p_S(\alpha)) & \text{otherwise} \end{cases}$$

$$N(\alpha) = \begin{cases} 0 & \text{if } \alpha \text{ is zero or a limit ordinal} \\ 1 + N(p_S(\alpha)) & \text{otherwise} \end{cases}$$

When IIM uses a particular system of notations S, its counter can contain only notations of this system. Replacements are also restricted:

(a) IIM can replace notation x for a successor ordinal by $p_S(x)$.

(b) IIM can replace notation x for a limit ordinal with $\varphi_{q_S(x)}(n)$ for any n.

Other replacements are not allowed. This means that, for each notation, the fact that it denotes a smaller ordinal than the previous notation must follow from the functions p_S and q_S of the system S. IIM cannot replace a notation x by a notation

to check whether IIM makes correct replacements. For notations x and y, $x \prec y$ denotes that x is smaller than y and this follows from the system of notations that is currently used.

$\mathbf{TxtEx}_\alpha^S$, $\mathbf{InfEx}_\alpha^S$, $\mathbf{Ex}_\alpha^S$ denote the collection of all sets that are $\mathbf{TxtEx}_\alpha$, $\mathbf{InfEx}_\alpha$, $\mathbf{Ex}_\alpha$-identifiable by IIM using the system of notations S.

Next, we describe two most frequently used systems of notations: P and O. Most of our theorems use one of these two systems. Exception is chapter 5 where we construct other systems ourselves. These systems are specific to results of this chapter.

2.5.2 The system P

The first system of notation is the system P. In this system the notations are expressions consisting of $0, 1, \ldots, \omega$, addition, multiplication and exponentiation. For example, $\omega + 17$, $\omega^3 \cdot 3$, $\omega^{\omega+2} + \omega^2 \cdot 6$ all are notations in P. For such expressions, the functions k_P, p_P and q_P can be easily defined.

There exist ordinals which have no notation in P. It does not have notations for ordinals which are greater than or equal to ϵ_0.

However, P is very natural system of notations and very easy to use. In general systems of notations it is difficult and even impossible to perform some operations with ordinals. (For example, to find $\alpha + \omega$ if α is given.) In P such operations are very easy.

When scientists speak about ordinals without using any system of notations explicitly, they usually use the system P.

2.5.3 The universal systems S_1 and O

We will frequently use universal systems S_1 and O[35, 45]. Notations of S_1 are defined as follows:

(a) 1 is a notation for the ordinal 0;

(b) If x is a notation for α, then 2^x is a notation for $\alpha + 1$;

(c) If $\varphi_y(0)$, $\varphi_y(1)$, ... is a sequence of notations for an increasing sequence of ordinals converging to α, then $3 \cdot 5^y$ is a notation for α.

The system O is obtained from S_1 by eliminating some notations so that the system remains universal and some new useful properties appear. These properties are not important for results in this thesis. Therefore, we omit them. Our proofs will not use these properties and everything that we prove for O is true both for S_1 and O. We shall use O for the universal system because this is more common in papers in this area and using S_1 may cause some confusion.

O (or S_1) embeds all possible systems of notations and anything which is expressible in some system of notations is expressible in O (or S_1).

Lemma 2 *[45] Given any system A, there is a partial recursive mapping φ such that, if $x \in D_A$ then $v_S(x) = v_O(\varphi(x))$.*

The system O allows various constructions including some that may be considered pathological. For example, it is possible to construct a recursive sequence of notations $\varphi(0)$, $\varphi(1)$, ... denoting a sequence of ordinals $\omega \cdot h(0)$, $\omega \cdot h(1)$, ... such that $h(x)$ grows faster than any recursive function. We will use some such constructions in chapter 5.

The system O is convenient because it embeds any possible construction. In chapter 4, we will use the possibility to construct a notation for the limit of $\varphi_x(0)$, $\varphi_x(1)$, ...

We will also use the fact that, given two notations is O, notations for the sum and the product of ordinals can be computed[45].

Chapter 3

Ordinal mind change complexity of unions of pattern languages

3.1 Overview

Pattern languages, unions of pattern languages and elementary formal systems are natural language classes for which ordinal mindchange bounds exist. Shinohara [49] showed that many rich concepts can be represented by unions of pattern languages; these languages have been applied to knowledge acquisition from amino acid sequences (see Arikawa et al. [10]).

Previously, Jain and Sharma[29] proved ordinal bounds for the identification of these classes from positive data(text).

This section contains counterparts of these results for identification from both positive and negative data (informant). The main result is that unions of at most $i+1$ pattern languages can be identified with at most $\omega \cdot i$ mindchanges.

We use the system of notations P in this chapter. However, our result is valid for any system because, in chapter 5, we show that $\omega \cdot i$-identification has the same power in all systems of notations.

3.2 Results

Let Σ and X be mutually disjoint sets. Σ is finite and its elements are referred to as *constant symbols*. Elements of X are referred to as *variables*. For the present section, we let $a, b, \ldots$ range over constant symbols and $x, y, z, x_1, x_2, \ldots$ range over

variables.

Definition 18 A *term* or a *pattern* is an element of $(\Sigma \cup X)^+$. A *ground term* (or a *word*, or a *string*) is an element of Σ^+.

A *substitution* is a homomorphism from terms to terms that maps each symbol $a \in \Sigma$ to itself. The image of a term π under a substitution θ is denoted $\pi\theta$. We next describe the language defined by a pattern. Note that there exists a recursive bijective mapping between elements of Σ^+ and $\mathbb{N}$. Thus we can name elements of Σ^+ with elements of $\mathbb{N}$. We implicitly assume such a mapping when we discuss languages defined using subsets of Σ^+ below. (We do not explicitly use such a bijective mapping for ease of notation).

Definition 19 [6] The language associated with the pattern π is defined as

$$\mathbf{Lang}(\pi) = \{\pi\theta \mid \theta \text{ is a substitution and } \pi\theta \in \Sigma^+\}.$$

We define the class $PATTERN = \{\mathbf{Lang}(\pi) \mid \pi \text{ is a pattern}\}$.

Angluin [6] showed that $PATTERN \in \mathbf{TxtEx}$. Shinohara [49] showed that pattern languages are not closed under union, and hence it is useful to study identification of languages that are unions of more than one pattern language, as they can be used to represent more expressive concepts. We next define unions of pattern languages.

Let S be a set of patterns. Then $\mathbf{Lang}(S)$ is defined as $\bigcup_{\pi \in S} \mathbf{Lang}(\pi)$. Intuitively, $\mathbf{Lang}(S)$ is the language formed by the union of languages associated with patterns in S.

Definition 20 [49, 54] Let $n \in \mathbb{N}$. $PATTERN^n = \{\mathbf{Lang}(S) \mid \text{card}(S) \leq n\}$.

Shinohara [49] and Wright [54] showed that for $n > 1$, $PATTERN^n \in \mathbf{TxtEx}$. Jain and Sharma [29] showed that $PATTERN^n \in \mathbf{TxtEx}_{\omega^n}$ (using the system of notations P) and $PATTERN^n \notin \mathbf{TxtEx}_\alpha$ for $\alpha \prec \omega^n$ (for any system of notations).

We now consider the ordinal mind change complexity of identifying unions of pattern languages from informants. Let PAT denote the set of all canonical patterns [6]. Let $PAT^i = \{S \mid S \subseteq PAT \ \wedge \ \text{card}(S) = i\}$.

Suppose Pos and Neg are disjoint finite sets such that $\text{Pos} \neq \emptyset$. Then let

$$X_i^{\text{Pos,Neg}} = \{S \in PAT^i \mid [\text{Pos} \subseteq \mathbf{Lang}(S)] \ \wedge \ [\text{Neg} \subseteq \overline{\mathbf{Lang}(S)}]\}$$

Lemma 3 *Suppose we are given finite disjoint sets* Pos, Neg, *where* Pos $\neq \emptyset$, *and a natural number* i, *such that* $(\forall j \leq i)[X_j^{\text{Pos,Neg}} = \emptyset]$. *Then, effectively in* Pos, Neg, *and* i, *we can determine* $X_{i+1}^{\text{Pos,Neg}}$. *(Note that* $X_{i+1}^{\text{Pos,Neg}}$ *must be finite in this case!)*

PROOF. Suppose Pos, Neg, and i are as given in the hypothesis of the lemma. Let

$$P = \{p \in PAT \mid [\text{Pos} \cap \mathbf{Lang}(p) \neq \emptyset] \ \wedge \ [\text{Neg} \cap \mathbf{Lang}(p) = \emptyset]\}$$

Let

$$X = \{S \in PAT^{i+1} \mid [\text{Pos} \subseteq \mathbf{Lang}(S)] \ \wedge \ [S \subseteq P]\}$$

It is easy to verify that $X = X_{i+1}^{\text{Pos,Neg}}$. Also note that X can be obtained effectively from Pos, Neg and i. ∎

Corollary 1 *Suppose* Pos *and* Neg *are disjoint finite sets such that* Pos $\neq \emptyset$. *Then effectively in* Pos, Neg, *one can find* i, *and corresponding* $X_i^{\text{Pos,Neg}}$ *(which must be finite) such that* $i = \min(\{j \mid X_j^{\text{Pos,Neg}} \neq \emptyset\})$.

PROOF. Note that $PATTERN^0$ contains only the empty language. The corollary now follows by repeated use of Lemma 3, until one finds an i such that $X_i^{\text{Pos,Neg}} \neq \emptyset$. ∎

Theorem 4 *(a)* $PATTERN^1 \in \mathbf{InfEx}_0$.
(b) $(\forall i \geq 1)[PATTERN^{i+1} \in \mathbf{InfEx}_{\omega \cdot i}]$.

PROOF. (a) Shown by Lange and Zeugmann [37]. Also follows from the proof of Part (b).

(b) Fix i. Let $\mathbf{M}(I[n]), \mathbf{F}(I[n])$ be defined as follows.

Let Pos = PosInfo($I[n]$) and Neg = NegInfo($I[n]$).

If Pos $= \emptyset$, then $\mathbf{M}(I[n]) =?$ and $\mathbf{F}(I[n]) = \omega \cdot i$.

If Pos $\neq \emptyset$, then let $j = \min(\{j' \mid X_{j'}^{\text{Pos,Neg}} \neq \emptyset\})$. Note that j (and corresponding $X_j^{\text{Pos,Neg}}$) can be found effectively in $I[n]$, using Corollary 1.

If $j = 1$ and $\text{card}(X_j^{\text{Pos,Neg}}) > 1$, then $\mathbf{M}(I[n]) =?$. $\mathbf{F}(I[n]) = \omega \cdot i$.

If $j > 1$ or $\text{card}(X_j^{\text{Pos,Neg}}) = 1$, then $\mathbf{M}(I[n])$ = lexicographically least element in $X_j^{\text{Pos,Neg}}$. $\mathbf{F}(I[n]) = \omega \cdot (i + 1 - j) + (\text{card}(X_j^{\text{Pos,Neg}}) - 1)$.

It is easy to verify that $\mathbf{M}, \mathbf{F}$ witness the theorem. ∎

3.3 Summary

Pattern languages can be identified from positive data only. Hence, it may seem that negative data are not necessary at all. Theorem 4 refutes this claim by showing that the complexity decreases considerably when negative data are available ($\omega \cdot i$ instead of ω^{i+1} mindchanges).

It is open at this stage whether we can do better than the $\omega \cdot i$ bound for $PATTERN^{i+1}$. However, if we consider unions of $i+1$ simple pattern languages[1], then it is easy to see that the mind change bound for identification from informants is simply i.

[1] A simple pattern language is formed by substituting, for each variable, strings of length exactly one.

Chapter 4

General conditions for existence of ordinal mindchange bounds

4.1 Overview

The existence of an ordinal mind change bound for a class can be considered as a reflection of its learning "tractability". Therefore, it is useful to investigate conditions under which an ordinal mind change bound can be guaranteed. We consider a number of possibilities, including identification by conservative strategies, topological properties like finite thickness, M-finite thickness, and finite elasticity, and monotonicity requirements. We preview some of our results.

We first establish a useful technical result which states that if a learning machine makes a finite number of mind changes on any text, then the class of languages that can be identified by this machine has an ordinal mind change bound. This result is used to show that if an indexed family of computable languages has finite elasticity and can be conservatively identified then there is an ordinal mind change bound for this class. We also show that the requirement of conservative identification can be sacrificed in the previous result for the purely topological requirement that the class have M-finite thickness in addition to finite elasticity. Since finite thickness implies finite elasticity and M-finite thickness, the above results imply that any indexed family of computable languages with finite thickness has an ordinal mind change bound.

The results discussed above give general sufficient conditions for identifiability with ordinal bound on mind changes. However, the mind change bound α may be

arbitrarily large. An interesting question to ask is whether the ordinal mind change bound remains arbitrarily large if some other constraints such as monotonicity are added. We show a negative result in this direction as for every constructive ordinal bound α, there exists an indexed family of computable languages that can be identified strong-monotonically and has finite thickness, but cannot be identified with the ordinal mind change bound of α. A similar result also holds for dual strong-monotonicity.

In this chapter, we use the universal system of notations O.

4.2 A characterization of ordinal bounds on the number of mindchanges

We first establish an important technical result.

Theorem 5 *Let* $\mathbf{M}$ *be a learning machine such that, for any text* T *(irrespective of whether* $\mathbf{M}$ *identifies* T *or not),* $\mathbf{M}$ *makes only finitely many mind changes on* T *as input. Let* $\mathcal{L}$ *denote the class of all languages* **TxtEx**-*identified by* $\mathbf{M}$*. Then, for some ordinal mind change counter function* $\mathbf{F}$*, and constructive ordinal* α*,* $\mathcal{L} \subseteq \mathbf{TxtEx}^{O}_{\alpha}(\mathbf{M}, \mathbf{F})$.

PROOF. We define a *conjecture tree* $\mathcal{T}_{\mathbf{M}}$ for machine $\mathbf{M}$. The root of $\mathcal{T}_{\mathbf{M}}$ corresponds to the empty sequence, Λ. Other nodes of the tree correspond to finite initial sequences of texts, $T[n+1]$, such that $\mathbf{M}(T[n]) \neq \mathbf{M}(T[n+1])$. Let $S = \{\Lambda\} \cup \{T[n+1] \mid n \in \mathbb{N}, T \text{ is a text and } \mathbf{M}(T[n]) \neq \mathbf{M}(T[n+1])\}$. For $\sigma \in S$, we use V_σ to denote the node corresponding to the sequence σ. Node V_{σ_1} is a descendent of node V_{σ_2} iff $\sigma_2 \subset \sigma_1$.

We will now define a constructive ordinal, α_σ, corresponding to each $\sigma \in S$. For $\sigma \in S$, let $S_\sigma = \{\tau \in S \mid \sigma \subset \tau\}$. Intuitively S_σ denotes the proper descendants of σ in the tree $\mathcal{T}_{\mathbf{M}}$. Note that S_σ is recursively enumerable (effectively in σ). Let S^s_σ denote the finite set enumerated in s steps in some, effective in σ, enumeration of S_σ.

α_σ is defined as follows. α_σ is the limit of $f_\sigma(0), f_\sigma(1), \ldots$, where f_σ is defined as follows.

$f_\sigma(0) = 0$. $f_\sigma(i+1) = f_\sigma + \alpha_{\tau_1} + \ldots + \alpha_{\tau_k} + 1$, where $\tau_1, \tau_2, \ldots, \tau_k$, are the elements of S^i_σ.

We first need to show that α_σ are correct notation.

Lemma 4 *(a) Let V_σ be a leaf of $\mathcal{T}_{\mathbf{M}}$. Then α_σ is a correct ordinal notation.*

(b) Suppose $\sigma \in S$. and α_τ is a correct ordinal notation for each $\tau \in S_\sigma$. Then α_σ is a correct ordinal notation.

(c) For any $\sigma \in S$, α_σ is a correct ordinal notation.

(d) If $\sigma \in S$ and $\tau \in S_\sigma$, then $\alpha_\tau \prec \alpha_\sigma$.

PROOF. (a) If V_σ is a leaf, then S_σ is empty. Hence,

$$f_\sigma(0) = 0, f_\sigma(1) = 0 + 1 = 1, \ldots, f_\sigma(n) = f_\sigma(n-1) + 1 = (n-1) + 1 = n, \ldots.$$

It follows that α_σ is a notation for ω.

(b) Since, α_σ is a limit of $f_\sigma(0), f_\sigma(1), \ldots$, it suffices to show that each $f_\sigma(i)$ is a correct ordinal notation. Now, for each $\tau \in S_\sigma$, α_τ is correct notation. Thus, since $f_\sigma(i+1)$ is defined using $f_\sigma(i)$, α_τ, 1 and + operation only, $f_\sigma(i+1)$ is a correct ordinal notation.

(c) Suppose by way of contradiction that α_σ is not a correct notation. We then construct an infinite sequence $\sigma_0 \subset \sigma_1 \subset \ldots$ such that, for each i, $\sigma_i \in S$ and α_{σ_i} is not a correct notation.

Let $\sigma_0 = \sigma$. Suppose σ_i has been defined. Let σ_{i+1} be such that $\sigma_{i+1} \in S_{\sigma_i}$ and $\alpha_{\sigma_{i+1}}$ is not a correct notation. The existence of such a σ_{i+1} follows from parts (a) and (b).

Consider the text $T = \bigcup_{i \in \mathbb{N}} \sigma_i$. Now, since each $\sigma_i \in S$, we have that $\mathbf{M}$ on T makes infinitely many mind changes (after reading last element of σ_1, after reading last element of σ_2, and so on). This yields a contradiction to hypothesis of the theorem.

(d) Note that $\alpha_\sigma \succ f_\sigma(i)$, for each i. Suppose $\tau \in S^s_\sigma$. Then it is easy to see that $f_\sigma(s+1) \succ \alpha_\tau$. Thus $\alpha_\tau \prec \alpha_\sigma$.

This proves the Lemma. ∎

Let $\alpha = \alpha_\Lambda$. We now construct an $\mathbf{F}$ such that $\mathcal{L} \subseteq \mathbf{TxtEx}^O_\alpha(\mathbf{M}, \mathbf{F})$. $\mathbf{F}$ is defined as follows.

$$\mathbf{F}(T[n]) = \begin{cases} \alpha_\Lambda, & \text{if } T[n]) = \Lambda; \\ \mathbf{F}(T[n]-1), & \text{if } n > 0, \text{ and } \mathbf{M}(T[n+1]) = \mathbf{M}(T[n]); \\ \alpha_{T[n]}, & \text{otherwise.} \end{cases}$$

From the definition of α_σ and Lemma 4, it is easy to verify that $\mathbf{TxtEx}(\mathbf{M}) \subseteq \mathbf{TxtEx}^O_\alpha(\mathbf{M}, \mathbf{F})$. ∎

This is precisely the converse of Lemma 1. So, we can add an ordinal mindchange counter to a machine **M** if and only if **M** makes finite number of mindchanges on any input. This is a nice characterization of machines with the number of minachanges bounded by an ordinal.

We proved our result for **TxtEx**. However, the same argument gives us similar results for **InfEx** and **Ex**. This result is especially useful for **TxtEx** because, in this case, it allows to derive several sufficient conditions for the existence of an ordinal mindchange bounds. So far, we do not know about similar applications for **InfEx** and **Ex**.

4.3 Ordinal complexity and conservativeness

Theorem 5 allows us to establish several sufficient conditions for the existence of ordinal bounds on mind changes in the context of identification of indexed families of computable languages. We first adapt learnability notions to the context of indexed families of computable languages.

A sequence of nonempty languages $L_0, L_1, \ldots$ is an indexed family just in case there exists a computable function f such that for each $i \in \mathbb{N}$ and for each $x \in \mathbb{N}$,

$$f(i,x) = \begin{cases} 1, & \text{if } x \in L_i, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, there is a uniform decision procedure for languages in the class. Here, i may be thought of as a grammar for the language L_i. It makes sense to learn an indexed family of computable languages in terms of a hypothesis space that also describes an indexed family. In the following we only consider hypothesis spaces which describe an indexed family. We will abuse the notation slightly and use $\mathcal{L}$ to refer to both the concept class and the hypothesis space; it will be clear from context which interpretation is intended. To differentiate the concept class $\mathcal{L} = \{L_i \mid i \in \mathbb{N}\}$ from the hypothesis space $\mathcal{L}$, we sometimes say that the class of languages $\{L_i \mid i \in \mathbb{N}\}$ is the range of the hypothesis space $\mathcal{L}$ (written: range($\mathcal{L}$)). The next definition adapts Gold's criterion of identification in the limit to the identification of indexed families with respect to a given hypothesis space.

Definition 21 Let $\mathcal{L}$ be an indexed family and let $\mathcal{L}' = \{L'_0, L'_1, \ldots\}$ be a hypothesis space.

(a) Let $L \in \mathcal{L}$. A machine $\mathbf{M}$ **TxtEx**-*identifies* L with respect to hypothesis space $\mathcal{L}'$ just in case for any text T for L, $\mathbf{M}(T)\downarrow = j$ such that $L = L'_j$.

(b) A machine $\mathbf{M}$ **TxtEx**-*identifies* $\mathcal{L}$ with respect to $\mathcal{L}'$ just in case for each $L \in \mathcal{L}$, $\mathbf{M}$ **TxtEx**-identifies L with respect to $\mathcal{L}'$.

There are three kinds of identification that have been studied in the literature: (a) class comprising; (b) class preserving; and (c) exact. If the indexed family $\mathcal{L}$ is identified with respect to a hypothesis space $\mathcal{L}'$ such that $\mathcal{L} \subseteq \text{range}(\mathcal{L}')$ then the identification is referred to as *class comprising*. However, if it is required that the indexed family be identifiable with respect to a hypothesis space $\mathcal{L}'$ such that $\mathcal{L} = \text{range}(\mathcal{L}')$ then the identification is referred to as *class preserving*. Finally, if the identification of the indexed family $\mathcal{L}$ is required to be with respect to $\mathcal{L}$ itself, then the identification is referred to as *exact*. The reader is directed to the excellent survey by Zeugmann and Lange [55] for discussion of these issues.

We can similarly define $\mathbf{TxtEx}_\alpha$-identification with respect to hypothesis space $\mathcal{L}'$. Note that Theorem 5 holds with respect to all hypothesis spaces.

We next describe certain topological conditions on language classes that yield sufficient conditions for identifiability of indexed families of computable languages. The following notion was introduced by Angluin [6].

Definition 22 [6] $\mathcal{L}$ has *finite thickness* just in case for each $n \in \mathbb{N}$, $\text{card}(\{L \in \mathcal{L} \mid n \in L\})$ is finite.

PATTERN has finite thickness. Angluin [6] showed that if $\mathcal{L}$ is an indexed family of computable languages and $\mathcal{L}$ has finite thickness then $\mathcal{L} \in$ **TxtEx**. A more interesting topological notion was introduced by Wright [54] (see also Motoki, Shinohara, and Wright [39]) described below.

Definition 23 [54, 39] $\mathcal{L}$ has *infinite elasticity* just in case there exists an infinite sequence of pairwise distinct numbers, $\{w_i \in \mathbb{N} \mid i \in \mathbb{N}\}$, and an infinite sequence of pairwise distinct languages, $\{A_i \in \mathcal{L} \mid i \in \mathbb{N}\}$, such that for each $k \in \mathbb{N}$, $\{w_i \mid i < k\} \subseteq A_k$, but $w_k \notin A_k$. $\mathcal{L}$ is said to have *finite elasticity* just in case $\mathcal{L}$ does not have infinite elasticity.

Wright [54] showed that if a class $\mathcal{L}$ has finite thickness then it has finite elasticity. He further showed that if a class $\mathcal{L}$ is an indexed family of computable languages and $\mathcal{L}$ has finite elasticity, then $\mathcal{L} \in$ **TxtEx**.

Finite elasticity is a sufficient condition for identification of indexed families of computable languages. Also, the property of finite elasticity is preserved under finite unions. As already noted, it was shown in [29] that for each $n > 0$, $PATTERN^n \in \mathbf{TxtEx}_{\mathcal{O}^n}$.

At the moment, we do not know whether any indexed family of computable languages with finite elasticity is identifiable with an ordinal mind change bound. However, we are able to show that an indexed family of computable languages with finite elasticity has an ordinal mind change bound if it can be identified conservatively. The next definition describes conservative identification.

Definition 24 Let $\mathcal{L} = \{L_0, L_1, \ldots\}$ be a hypothesis space. $\mathbf{M}$ is said to be a *conservative learning machine with respect to hypothesis space* $\mathcal{L}$ just in case for all σ and τ such that $\sigma \subseteq \tau$ and $\text{content}(\tau) \subseteq L_{\mathbf{M}(\sigma)}$, $\mathbf{M}(\sigma) = \mathbf{M}(\tau)$.

Intuitively, conservative machines do not change their hypothesis if the input is contained in the language conjectured.

Theorem 6 *Let $\mathcal{L}'$ be an indexed family of computable languages with finite elasticity. Assume that $\mathcal{L}$ is identifiable by a conservative learning machine with respect to the hypothesis space $\mathcal{L}'$. Then $\mathcal{L} \in \mathbf{TxtEx}^O_\alpha$ with respect to hypothesis space $\mathcal{L}'$, for some constructive ordinal α.*

PROOF. Let $\mathbf{M}$ be a conservative learning machine which identifies $\mathcal{L}$ with respect to hypothesis space $\mathcal{L}'$. We will describe a machine $\mathbf{M}'$ which identifies $\mathcal{L}$ with respect to $\mathcal{L}'$, and changes its mind at most finitely often on any text. Theorem 5 will then imply the theorem.

For a given text T, $n \in \mathbb{N}$, let $\text{lmc}(\mathbf{M}', T[n])$ be defined as follows:

$$\text{lmc}(\mathbf{M}', T[n]) = \max(\{m+1 \mid m < n \ \wedge \ \mathbf{M}'(T[m]) \neq \mathbf{M}'(T[m+1])\})$$

Intuitively, lmc denotes the last point where $\mathbf{M}'$ made a mind change. Note that if $\mathbf{M}'(T[0]) = \mathbf{M}'(T[1]) = \cdots = \mathbf{M}'(T[n])$, then $\text{lmc}(\mathbf{M}', T[n]) = 0$. $\mathbf{M}'$ is now defined as follows:

$$\mathbf{M}'(T[n]) = \begin{cases} ?, & \text{if } n = 0 \text{ or } \mathbf{M}(T[n]) = ?; \\ \mathbf{M}(T[n]), & \text{if } \text{content}(T[\text{lmc}(\mathbf{M}', T[n-1])]) \subseteq L'_{\mathbf{M}(T[n])}; \\ \mathbf{M}'(T[n-1]), & \text{otherwise.} \end{cases}$$

It is easy to verify that $\mathbf{M}'$ **TxtEx**-identifies with respect to $\mathcal{L}'$ any language which $\mathbf{M}$ **TxtEx**-identifies with respect to $\mathcal{L}'$. We prove that $\mathbf{M}'$ makes only finitely many mind changes on any text T. By Theorem 5, this implies that $\mathcal{L} \in \mathbf{TxtEx}^O_\alpha$ with respect to hypothesis space $\mathcal{L}'$, for some constructive ordinal α.

Suppose by way of contradiction that $\mathbf{M}'$ makes infinitely many mind changes on a text T. Let $n_1 < n_2 < \ldots$ be such that, for each i, $\mathbf{M}'(T[n_i]) \neq \mathbf{M}'(T[n_i + 1])$. Then, it is easy to verify from the construction of $\mathbf{M}'$ that, for all i, content$(T[n_i + 1]) \subseteq L'_{\mathbf{M}'(T[n_{i+2}])}$. Moreover, since $\mathbf{M}$ is conservative, we have content$(T[n_i + 1]) \not\subseteq L'_{\mathbf{M}'(T[n_i])}$. It follows that $\mathcal{L}'$ has infinite elasticity. A contradiction. ∎

Definition 25 L_j is a *minimal concept of* L *in* $\mathcal{L}$ just in case $L \subseteq L_j$, $L_j \in \mathcal{L}$, and there is no $L_i \in \mathcal{L}$ such that $L \subseteq L_i$ and $L_i \subset L_j$.

Definition 26 [47] $\mathcal{L}$ satisfies *MEF-condition* if for any finite set D and any $L_i \in \mathcal{L}$ with $D \subseteq L_i$ there is a minimal concept L_j of D within $\mathcal{L}$ such that $L_j \subseteq L_i$. $\mathcal{L}$ satisfies *MFF-condition* if for any nonempty finite set D, the cardinality of $\{L_i \in \mathcal{L} \mid L_i$ is a minimal concept of D within $\mathcal{L}\}$ is finite. $\mathcal{L}$ has *M-finite thickness* if $\mathcal{L}$ satisfies both MEF-condition and MFF-condition.

Theorem 7 *Let $\mathcal{L}$ be an indexed family of computable languages. Assume that $\mathcal{L}$ has M-finite thickness and finite elasticity. Then $\mathcal{L} \in \mathbf{TxtEx}^O_\alpha$ with respect to hypothesis space $\mathcal{L}$, for some constructive ordinal α.*

PROOF. Suppose T is an arbitrary text. We then describe a learning machine $\mathbf{M}$. Define $\mathbf{M}(T[n])$ as follows. Let $L_i^{(n)}$ denote $L_i \cap \{x \mid x < n\}$.

If $\emptyset \in \mathcal{L}$, then let $G_\emptyset$ denote a grammar for $\emptyset$ in $\mathcal{L}$; otheriwse let $G_\emptyset = 0$.

$\mathbf{M}(T[n])$

Let $C_n =$ content$(T[n])$.

If $C_n = \emptyset$ then output $G_\emptyset$.

Let $S_n = \{i \leq n \mid C_n \subseteq L_i \ \wedge \ \neg(\exists j \leq n)[C_n \subseteq L_j \ \wedge \ L_j^{(n)} \subset L_i^{(n)}]\}$.

If S_n is not empty then output $\min(S_n)$, else output $\mathbf{M}(T[n-1])$.

End

The above learning machine is a slight modification of the machine of Mukouchi [40].

Let T be an arbitrary text (for a language L). Assume without loss of generality that content$(T) \neq \emptyset$. We will show that $\mathbf{M}$ makes only finitely many mind changes on T. Suppose for contradiction, $\mathbf{M}$ changes its mind infinitely often on T. First note that, if $\mathbf{M}(T[n]) \neq \mathbf{M}(T[n+1])$ then content$(T[n+1]) \subseteq L_{\mathbf{M}(T[n+1])}$. Consider two cases:

Case 1. $\mathbf{M}$ outputs infinitely many distinct conjectures i such that content$(T) \not\subseteq L_i$. (That is, card$(\{\mathbf{M}(T[n]) \mid n \in \mathbb{N} \wedge \text{content}(T) \not\subseteq L_{\mathbf{M}(T[n])}\}) = \infty$.)

Let $n_1 < n_2 < n_3 < \cdots$ be such that $\mathbf{M}(T[n_i]) \neq \mathbf{M}(T[n_{i+1}])$, and content$(T[n_{i+1}]) \not\subseteq L_{\mathbf{M}(T[n_i])}$. Note that there exist such an n_i by the hypothesis of this case. Also, by construction, we have content$(T[n_i]) \subseteq L_{\mathbf{M}(T[n_{i+1}])}$ (since, any *new* hypothesis output by $\mathbf{M}$ is consistent with the input). By considering the languages $L_{\mathbf{M}(T[n_{2i}])}$, we see that content$(T[n_{2i+1}]) \subseteq L_{\mathbf{M}(T[n_{2i+2}])}$, but content$(T[n_{2i+1}]) \not\subseteq L_{\mathbf{M}(T[n_{2i}])}$. It follows that $\mathcal{L}$ has an infinite elasticity. A contradiction.

Case 2. $\mathbf{M}$ issues finitely many distinct conjectures i such that content$(T) \not\subseteq L_i$.

Then, for large enough n, $L_{\mathbf{M}(T[n])} \supseteq \text{content}(T) = L$ (since $\mathbf{M}$ changes its hypothesis infinitely often and if $\mathbf{M}(T[n]) \neq \mathbf{M}(T[n+1])$ then content$(T[n+1]) \subseteq L_{\mathbf{M}(T[n+1])}$).

Mukouchi [40] showed the following lemma.

Lemma 5 [40] *Let $\mathcal{L} = \{L_i \mid i \in \mathbb{N}\}$ be a class satisfying MEF-condition and having finite elasticity. Let L be a nonempty language. If for some n, $L \subseteq L_n$, then there is a minimal concept L_j of L within $\mathcal{L}$ such that $L_j \subseteq L_n$.*

Since, we have already shown that, for large enough n, $L_{\mathbf{M}(T[n])} \supseteq L$, Lemma 5 implies that there is a minimal concept L_j of L within $\mathcal{L}$.

Let $S = \{L_j \mid L_j$ is a minimal concept for L within $\mathcal{L}\}$. Let m be such that, for all $L' \in S$, there exists a $j < m$ such that $L_j = L'$ (that is, all minimal concepts of $L = \text{content}(T)$ are represented by an index $\leq m$). Let j_m be the minimum number such that $L_{j_m} \in S$.

For large enough n ($> m$), the following hold

(i) $L_{\mathbf{M}(T[n])} \supseteq L$.

(ii) For all $j < m$, either content$(T[n]) \not\subseteq L_j$, or $L_j \in S$, or there exists an $L' \in S$, such that $L_j^{(n)} \supset L'^{(n)}$.

(iii) For all minimal concepts $L' \in S$, such that $L' \neq L_{j_m}$, $L'^{(n)} - L_{j_m}^{(n)} \neq \emptyset$.

Note that (i) and (ii) imply that, $\mathbf{M}(T[n])$ will only output an index for one of the minimal concepts. And, (iii) implies that this index must be j_m. Hence, $\mathbf{M}$ converges to j_m on the text T, i.e., $\mathbf{M}$ makes only finitely many mind changes on T. A contradiction.

Thus, $\mathbf{M}$ must make only finitely many mind changes on any text T. Similarly to Case 2, we can show that on any text for a language L_j, $\mathbf{M}$ converges to the smallest index for L_j. So, $\mathbf{M}$ makes finitely many mind changes on any input and **TxtEx**-identifies $\mathcal{L}$ with respect to $\mathcal{L}$. Thus, Theorem 5 implies that $\mathcal{L} \in \mathbf{TxtEx}_\alpha^O$ with respect to $\mathcal{L}$. for some constructive ordinal α. ∎

Corollary 2 *Let $\mathcal{L}$ be an indexed family of computable languages with finite thickness. Then $\mathcal{L} \in \mathbf{TxtEx}_\alpha^O$ with respect to $\mathcal{L}$, for some constructive ordinal α.*

PROOF. If $\mathcal{L}$ has finite thickness, then $\mathcal{L}$ has finite elasticity (cf. Wright [54] and Shinohara [50]) and M-finite thickness (cf. Mukouchi [40]). Hence, by Theorem 7, $\mathcal{L} \in \mathbf{TxtEx}_\alpha^O$ with respect to $\mathcal{L}$, for some constructive ordinal α. ∎

A special case of Theorem 7 is the learnability of length-bounded elementary formal systems with ordinal-bounded mind changes. (Shinohara [50] has proved that $LBEFS^{(\leq n)}$, the class of languages defined by length-bounded elementary formal systems with at most n axioms, has finite elasticity and Sato and Moriyama [47] have proved that $LBEFS^{(\leq n)}$ has M-finite thickness.) The learnability of $LBEFS^{(\leq n)}$ was shown by Shinohara [50]. Jain and Sharma [29] proved that $LBEFS^{(\leq n)}$ is learnable with the number of mind changes bounded by ordinal ω^n.

The results discussed in the present paper give general sufficient conditions for identifiability with ordinal bound on mind changes. However, they do not give explicit ordinals α. In all these theorems we have "$\mathcal{L} \in \mathbf{TxtEx}_\alpha^O$ for some ordinal α." It appears that ordinal α can be arbitrarily large. An interesting question to ask is if the ordinal bound α is still arbitrarily large if attention is restricted to classes that are identifiable by strategies that obey stronger restrications than those in Theorems 6 and 7.

In next section, we show that even if we require that a class $\mathcal{L}$ has finite thickness and that it is identifiable by a strong-monotonic learning machine, the ordinal mind change bound can be arbitrarily large. The reader should however note that strong-monotonicity together with finite elasticity implies the existence of an ordinal bound because strong-monotonicity implies conservatism.

4.4 Ordinal complexity and monotonicity

Below we describe the notion of strong-monotonic identification.

Definition 27 (Jantke [31])

(a) Let $\mathcal{L}' = \{L'_0, L'_1, \ldots\}$ be a hypothesis space. A learning machine $\mathbf{M}$ is said to be *strong monotonic with respect to* $\mathcal{L}'$ just in case for all σ and τ such that $\sigma \subseteq \tau$, $L'_{\mathbf{M}(\sigma)} \subseteq L'_{\mathbf{M}(\tau)}$.

(b) A learning machine $\mathbf{M}$ is said to *strong-monotonically* **TxtEx**-*identify* L *with respect to* $\mathcal{L}'$ just in case $\mathbf{M}$ **TxtEx**-identifies L with respect to $\mathcal{L}'$ and $\mathbf{M}$ is strong monotonic with respect to $\mathcal{L}'$.

(c) $\mathbf{M}$ *strong-monotonically* **TxtEx**-*identifies* $\mathcal{L}$ *with respect to* $\mathcal{L}'$ just in case, for each $L \in \mathcal{L}$, $\mathbf{M}$ strong-monotonically **TxtEx**-identifies L with respect to $\mathcal{L}'$.

We use a technical lemma.

Lemma 6 *Fix a constructive ordinal* α. *There exists an r.e. sequence of pairs of learning machines and corresponding ordinal mind change counter functions,* $(\mathbf{M}_0, \mathbf{F}_0)$, $(\mathbf{M}_1, \mathbf{F}_1)$, ..., *such that*

(a) for all $\mathcal{L} \in \mathbf{TxtEx}^O_\alpha$, *there exists an* i *such that* $\mathcal{L} \subseteq \mathbf{TxtEx}^O_\alpha(\mathbf{M}_i, \mathbf{F}_i)$.

(b) for all i, $\mathbf{F}_i(\Lambda) = \alpha$.

(c) for all i, *for all texts* T, *for all* n, $\mathbf{M}_i(T[n]) \neq \mathbf{M}_i(T[n+1]) \Rightarrow \mathbf{F}_i(T[n]) \succ \mathbf{F}_i(T[n+1])$.

The above lemma can be proved on the lines of the proof of Lemma 4.2.2B in [41].

Theorem 8 *Let α be a constructive ordinal. There exists an indexed family $\mathcal{L}$ such that $\mathcal{L}$ can be* **TxtEx***-identified strong-monotonically with respect to hypothesis space $\mathcal{L}$. $\mathcal{L}$ has finite thickness, and $\mathcal{L} \notin \mathbf{TxtEx}^O_\alpha$ with respect to any hypothesis space.*

PROOF. Let $(\mathbf{M}_0, \mathbf{F}_0), (\mathbf{M}_1, \mathbf{F}_1) \ldots$ be an enumeration of pairs of learning machines and corresponding ordinal mind change counter functions as given by Lemma 6. Note that for each $i \in \mathbb{N}$, and for any text T, $\mathbf{M}_i$ makes only finitely many mind changes when fed T [24].

Let $L_i = \{\langle i, x\rangle \mid x \in \mathbb{N}\}$. Note that L_i is infinite, and for distinct i, j, L_i and L_j are disjoint. Let $L_i^s = \{\langle i, x\rangle \mid x \leq s\}$. We now give an algorithm which receives i and enumerates (effectively in i) a finite set of languages $\mathcal{L}_i$ such that:

(a) if $L \in \mathcal{L}_i$, then $L = L_i^s$ for some s;

(b) $\mathcal{L}_i$ is finite (note that one can effectively decide the membership problem for languages in $\mathcal{L}_i$);

(c) $\mathcal{L}_i$ is not **TxtEx**-identified by $\mathbf{M}_i$ with respect to any hypothesis space;

(d) There exists a machine, effective in i, that strong-monotonically **TxtEx**-identifies $\mathcal{L}_i$ with respect to the hypothesis space $\mathcal{L}_i$.

Now define $\mathcal{L} = \bigcup_{i \in \mathbb{N}} \mathcal{L}_i$, such that for $L_i^s \in \mathcal{L}$, one can effectively find an index (in $\mathcal{L}$) for L_i^s. We will show that $\mathcal{L}$ establishes the theorem. First, the algorithm enumerating $\mathcal{L}_i$ is as follows:

Enumeration of $\mathcal{L}_i$.

Initially, let $\mathcal{L}_i$ consists of just the language L_i^0.

Let $n = 0$ and let initial sequence σ_0 be such that content$(\sigma_0) = L_i^0$. Go to Stage 0.

Stage s

Add the language L_i^{s+1} to $\mathcal{L}_i$.

Search for a γ extending σ_s, such that content$(\gamma) \subseteq L_i^{s+1}$, and $\mathbf{M}_i(\sigma_s) \neq \mathbf{M}_i(\gamma)$.

If and when such a γ is found, let σ_{s+1} be an extension of γ such that content$(\sigma_{s+1}) = L_i^{s+1}$.

Go to Stage $s + 1$.

End Stage s

End Enumeration of $\mathcal{L}_i$

We now show that $\mathcal{L}_i$'s satisfy the properties claimed.

Lemma 7 *For each $i \in \mathbb{N}$, there are only finitely many stages in the enumeration procedure for $\mathcal{L}_i$. Hence, $\mathcal{L}_i$ is finite.*

PROOF. Suppose by way of contradiction there is an $i \in \mathbb{N}$ such that there are infinitely many stages in the constrution of $\mathcal{L}_i$. Then $\mathbf{M}_i$ on $\bigcup_{s\in\mathbb{N}} \sigma_s$ makes infinitely many mind changes. A contradiction. ∎

Lemma 8 *For each $i \in \mathbb{N}$, $\mathbf{M}_i$ fails to* **TxtEx**-*identify $\mathcal{L}_i$ with respect to any hypothesis space.*

PROOF. Let s be the stage in the enumeration of $\mathcal{L}_i$ which starts but does not terminate. Then $\mathbf{M}_i$ can **TxtEx**-identify at most one of L_i^s and L_i^{s+1}, both of which are in $\mathcal{L}_i$. ∎

Now define $\mathcal{L} = \bigcup_{i\in\mathbb{N}} \mathcal{L}_i$, such that for $L_i^s \in \mathcal{L}$, one can effectively find an index (in $\mathcal{L}$) for L_i^s. It is easy to verify that $\mathcal{L}$ can be strong monotonicaly identified with respect to hypothesis space $\mathcal{L}$. Also, $\mathcal{L} \notin \mathbf{TxtEx}_\alpha^O$, by Lemma 8. Moreover, note that L_i's are pairwise disjoint. Thus, since each language in $\mathcal{L}_i$ is a subset of L_i and $\mathcal{L}_i$ is finite, we have that $\mathcal{L}$ has finite thickness. ∎

The reader should note that a similar result in the sense of class-preserving or exact identification cannot hold for dual strong-monotonicity [32] because class preserving dual strong monotonic identification is the same as finite identification (see [37], [55]). However, we can establish a similar result for class comprising dual strong monotonic identification.

Definition 28 [32]

(a) Let $\mathcal{L}' = \{L'_0, L'_1, \ldots\}$ be a hypothesis space. A learning machine $\mathbf{M}$ is said to be *dual strong-monotonic with respect to hypothesis space $\mathcal{L}'$* just in case for all σ and τ such that $\sigma \subseteq \tau$, $L'_{\mathbf{M}(\sigma)} \supseteq L'_{\mathbf{M}(\tau)}$.

(b) A learning machine $\mathbf{M}$ is said to *dual strong-monotonically* **TxtEx**-*identify L with respect to hypothesis space $\mathcal{L}'$* just in case $\mathbf{M}$ **TxtEx**-identifies L with respect to hypothesis space $\mathcal{L}'$ and $\mathbf{M}$ is dual strong monotonic with respect to $\mathcal{L}'$.

(c) $\mathbf{M}$ *dual strong-monotonically* **TxtEx**-*identifies $\mathcal{L}$ with respect to hypothesis space $\mathcal{L}'$* just in case, for each $L \in \mathcal{L}$, $\mathbf{M}$ dual strong-monotonically **TxtEx**-identifies L with respect to $\mathcal{L}'$.

Theorem 9 *Let α be a constructive ordinal. There exists an indexed family $\mathcal{L}$ and a hypothesis space $\mathcal{L}'$ such that $\mathcal{L}$ can be* **TxtEx**-*identified dual strong-monotonically with respect to $\mathcal{L}'$, $\mathcal{L}'$ has finite thickness, and $\mathcal{L} \notin \mathbf{TxtEx}^O_\alpha$ with respect to any hypothesis space.*

PROOF. Let $(\mathbf{M}_0, \mathbf{F}_0), (\mathbf{M}_1, \mathbf{F}_1) \ldots$ be an enumeration of pairs of learning machines and corresponding ordinal mind change counter functions as given by Lemma 6. Note that for each $i \in \mathbb{N}$, and any text T, $\mathbf{M}_i$, fed T, makes only finitely many mind changes [24].

For each i, we will define a recursive function g_i (where a program for g_i can be found effectively in i). g_i will satisfy the following properties:

(A) $\{x \mid g_i(x) = 1\}$ is nonempty and finite. Moreover, $\{x \mid g_i(x) = 1\} \subseteq \{\langle i, y\rangle \mid y \in \mathbb{N}\}$.

(B) Let $L_i = \{2x, 2x+1 \mid g_i(x) = 1\}$. Let $\mathcal{L}_i = \{L \subseteq L_i \mid (\forall x \mid g_i(x) = 1)(\exists! b \in \{0,1\})[2x + b \in L]\}$. Then, $\mathcal{L}_i \notin \mathbf{TxtEx}^O_\alpha(\mathbf{M}_i, \mathbf{F}_i)$ (with respect to any hypothesis space)[1].

We take $\mathcal{L} = \bigcup_i \mathcal{L}_i$ (using the fact that $g_i^{-1}(1)$ is finite, one can easily construct such an indexed family $\mathcal{L}$). From (B) it follows that $\mathcal{L} \notin \mathbf{TxtEx}^O_\alpha$ with respect to any hypothesis space.

We let $\mathcal{L}'$ be an hypothesis space such that range$(\mathcal{L}') = \{L \mid (\exists i)[L \subseteq L_i]\}$, where an index for $L_i - D$, for any finite set D, can be obtained effectively from i and D. Note that such an hypothesis space $\mathcal{L}'$ can be easily constructed. Clearly, $\mathcal{L}'$ has finite thickness.

It remains to construct recursive functions g_i as claimed above and to show that $\mathcal{L}$ can be dual strong monotonically identified with respect to hypothesis space $\mathcal{L}'$.

We now define g_i.

Definition of g_i

For $x < \langle i, 0\rangle$, let $g_i(x) = 0$. Let $g_i(\langle i, 0\rangle) = 1$.

Let $x_i^0 = \langle i, 0\rangle$. Intuitively, x_i^s denotes the largest x such that $g_i(x)$ is defined to be 1 before stage s.

Let $\sigma_0 = \Lambda$.

Go to Stage 0.

Stage s

[1] *Notation*: $\exists!$ denotes "there exists a unique."

1. Dovetail steps 2 and 3, until step 2 succeeds. If and when step 2 succeeds, go to step 4.
2. Search for an extension τ of σ_s, and $z \in \{2x_i^s, 2x_i^s + 1\}$, such that
   (a) $\mathbf{M}_i(\tau) \neq \mathbf{M}_i(\sigma_s)$, and
   (b) content(τ) = content$(\sigma_s) \cup \{z\}$.
3. For $x = x_s + 1$ to ∞ do
   Let $g_i(x) = 0$.
   EndFor
4. If and when such τ, z are found, let $\sigma_{s+1} = \tau$. Let $x_i^{s+1} \in \{\langle i, y\rangle \mid y \in \mathbb{N}\}$, be such that $g_i(x_i^{s+1})$ has not been defined until now.
   Let $g_i(x_i^{s+1}) = 1$.
   For $x < x_i^{s+1}$, such that $g_i(x)$ has not been defined until now, let $g_i(x) = 0$.

End Stage s

End of definition of g_i.

Lemma 9 *For each $i \in \mathbb{N}$, there are only finitely many stages in the construction of g_i.*

PROOF. Suppose by way of contradiction there are infinitely many stages. Then, $\mathbf{M}_i$ on $\bigcup_{s\in\mathbb{N}} \sigma_s$ makes infinitely many mind changes. A contradiction. ∎

Fix i. Using the above lemma, it is easy to verify that g_i satisfies (A). We now show that g_i satisfies (B). Suppose s is the stage which starts but does not terminate. Let $L' = \text{content}(\sigma_s) \cup \{2x_i^{s+1}\}$. Let $L'' = \text{content}(\sigma_s) \cup \{2x_i^{s+1} + 1\}$. Let T', extending σ_s, be a text for L'. Let T'' extending σ_s be a text for L''. Since step 2 in stage s did not succeed, we have that $\mathbf{M}_i(T') = \mathbf{M}_i(T'') = \mathbf{M}_i(\sigma_s)$. It follows that $\mathbf{M}_i$ does not **TxtEx**-identify $\mathcal{L}_i$ with respect to any hypothesis space. Thus (B) is satisfied.

We now give a machine $\mathbf{M}$ which, for each $L \in \mathcal{L}$, dual strong monotonically identifies L with respect to hypothesis space $\mathcal{L}'$. Let gram be a recursive function such that $L'_{\mathbf{gram}(i,D)} = L_i - D$ (by construction of $\mathcal{L}'$ such a function gram clearly exists).

For $x \in \mathbb{N}$ and $b \in \{0, 1\}$, let mate$(2x + b) = 2x + 1 - b$.

$\mathbf{M}(T[n])$

If content$(T[n]) = \emptyset$, then let $\mathbf{M}(T[n]) =?$.

1. Let i be such that content$(T[n]) \subseteq \{2\langle i, y\rangle + b \mid y \in \mathbb{N} \wedge b \in \{0,1\}\}$.
   (If no such i exists, then let $\mathbf{M}(T[n]) = \mathbf{M}(T[n-1])$.)
2. Let $D = \{\text{mate}(z) \mid z \in \text{content}(T[n])\}$.
3. Output gram(i, D).

End

It is easy to verify from the definition of L_i, $\mathcal{L}_i$, $\mathcal{L}$, $\mathcal{L}'$ that $\mathbf{M}$ is dual strong monotonic and **TxtEx**-identifies $\mathcal{L}$ with respect to hypothesis space $\mathcal{L}'$. Theorem follows. ∎

4.5 Summary

This chapter linked together ordinal bounds on the number of mindchanges, monotonicity requirements and topological properties of language classes (finite thickness and finite elasticity). Intricate relations between these notions were revealed. Interestingly, ordinal bounds are also related to inference of nearly-minimal size programs[4].

Chapter 5

The influence of the system of ordinal notations

5.1 Overview

This chapter is devoted to the influence of a particular notation for ordinals on the power of $\mathbf{Ex}_\alpha$, $\mathbf{TxtEx}_\alpha$ and $\mathbf{InfEx}_\alpha$. We prove our results for $\mathbf{Ex}_\alpha$ (identification of recursive functions in the limit) only. However, all results can be proved for $\mathbf{TxtEx}_\alpha$ and $\mathbf{InfEx}_\alpha$, too (with minor modifications in the proofs).

We remind that there are many nonequivalent systems of notations for constructive ordinals. We defined the requirements for an acceptable system of notations in section 2.5. There is a large variety of systems satisfying these requirements. In [24] it remained open whether the system of notations influences the learning power. We resolve this problem.

In section 5.2 we show that the learning power is not influenced by the system of notations while only small ordinals (below ω^2) are used. In this case, any learning machine working with ordinals in one system of notations can be transformed to an equivalent learning machine working in any other system of notations.

In section 5.3 we consider the bounds on the number of mindchanges described by the ordinal ω^2. Here, the situation is completely different. Our results reveal very strong influence of the system of notations on the learning power. We construct two systems of notations such that for some learning problems the first system is better and for some other problems the second system is better (cf. Theorem 11).

In section 5.4 we consider two particular systems of notations: O and P(cf. section

2.5). We give results relating these two systems to other systems.

The learning power can be increased in two ways: by using larger ordinals and by using more expressive systems of notations. We compare these two methods in sction 5.5. We show that the use of larger ordinals cannot compensate the weakness of the system of notations and, conversely, the use of stronger system cannot replace the use of larger ordinals.

5.2 EX_α-identification for $\alpha < \omega^2$

For small ordinals α the power of EX^A_α does not depend on the system of notations A.

Theorem 10 *If A and B are two systems of ordinal notations and α is an ordinal smaller than ω^2, then $EX^A_\alpha = EX^B_\alpha$.*

PROOF. We show that any system of notations can be simulated by the system P and, conversely, P can simulate any other system.

Lemma 10 *For an arbitrary system of notations A and an ordinal $\alpha < \omega^2$*

$$EX^P_\alpha \subseteq EX^A_\alpha$$

PROOF. This is a special case of the simulation of EX^P_α-IIM by EX^A_α-IIM for $\alpha < \omega^2 + \omega \cdot 2$ in the proof of Theorem 13. ∎

Lemma 11 *For an arbitrary system of notations A and an ordinal $\alpha < \omega^2$*

$$EX^A_\alpha \subseteq EX^P_\alpha$$

PROOF. The proof is based on the following lemma.

Lemma 12 *There exists a partial recursive function $t(x, y, z)$ such that, if x and y are notations in the system A, and z is a notation in the system P and*

$$v_A(y) < v_A(x) \leq v_P(z) < \alpha,$$

then $t(x, y, z)$ is a notation in P, $v_A(y) \leq v_P(t(x, y, z))$ and $v_P(t(x, y, z)) < v_P(z)$.

PROOF. Let $z = \omega \cdot k + l$. We define

$$t(x, y, z) = \begin{cases} \omega \cdot k + (l-1) & \text{if } l \neq 0, \\ \omega \cdot (k-1) + L(y) & \text{if } l = 0 \end{cases}.$$

∎

Let M_A be an EX^A_α-IIM.

Consider the EX^P_α-IIM M_P which simulates M_A and outputs the same conjectures. If M_A changes the notation on its ordinal counter from x to y, then M_P changes the notation from z to $t(x, y, z)$, where z is the notation on M_P's counter before change.

At the beginning both M_A and M_P have a notation for α on the counter. Further, always when M_A changes the ordinal, M_P changes the ordinal, too. Lemma 12 guarantees that all the time the ordinal on the counter of M_P is greater than or equal to the ordinal on the counter of M_A. ∎

From these two lemmas theorem 10 follows. ∎

5.3 EX_{ω^2}-identification

However, for ω^2 and larger ordinals, the dependence is rather strong. We can construct two systems of notations such that one is stronger than another (there exists a set of functions that is identifiable using the first system but is not identifiable using the second system). More, we can show that there exist two systems of notations such that in some cases the first is better and, in some other cases, the second is better.

Theorem 11 *There exist systems of notations A and B such that $EX^A_{\omega^2} \not\subseteq EX^B_{\omega^2}$ and $EX^B_{\omega^2} \not\subseteq EX^A_{\omega^2}$.*

PROOF. First, we construct two lim-computable functions such that, for some x the first grows much faster than the second and, for some other x the second grows much faster than the first(subsection 5.3.1). Then, we use these two functions to define two systems of notations S_{g_1} and S_{g_2} (subsection 5.3.2). Finally, we prove that there is a set of functions which can be identified using S_{g_1} but cannot be identified using S_{g_2}(subsection 5.3.3).

5.3.1 Lemma about lim-computable functions

Definition 29 A function $h(x) : \mathbb{N} \to \mathbb{N}$ is lim-computable if there exists a total recursive function $g(x,y)$ such that $h(x) = \lim_{y\to\infty} g(x,y)$.

For functions $h(x)$ and $g(x,y)$ we say that $h(x)$ is lim-computable as witnessed by $g(x,y)$. We say that $g(x,y)$ is *monotonic* if it is nondecreasing in y and increasing in x.

Lemma 13 *If $\varphi_1(x), \phi_2(x), \ldots$ is a computable sequence of partial recursive functions, then there exist functions $h_1(x)$ and $h_2(x)$ such that*

1. *h_1 is* lim*-computable as witnessed by a monotonous function $g_1(x,y)$;*
2. *h_2 is* lim*-computable as witnessed by a monotonous function $g_2(x,y)$;*
3. *For each x there is an y_1 such that $h_1(\langle x,y_1\rangle) > h_2(\phi_x(\langle x,y_1\rangle))$ or $\phi_x(\langle x,y_1\rangle)$ is undefined.*
4. *For each x there is an y_2 such that $h_2(\langle x,y_2\rangle) > h_1(\phi_x(\langle x,y_2\rangle))$ or $\phi_x(\langle x,y_2\rangle)$ is undefined.*

PROOF. We give an algorithm computing $g_1(x,y)$ and $g_2(x,y)$.

The algorithm uses variables $m_1, m_2, \ldots$ and $m_1', m_2', \ldots$ to mark the possible values of y_1 and y_2. Also, it uses variables $n_1, n_2, \ldots$ and $n_1', n_2', \ldots$.

1^{st} *step* Set $g_1(1,1) = g_2(1,1) = 1$ and $m_1 = 0, m_1' = 0, n_1 = 0, n_1' = 0$.

k^{th} *step* $(k > 1)$

1. Define $g_1(k,i) = g_1(k-1,i)+1$ and $g_2(k,i) = g_2(k-1,i)+1$ for $i \in \{1,\ldots,k-1\}$. For each $i \in \{1,\ldots,k\}$ define $g_1(i,k)$ equal to

   (a) the maximum of $g_1(i,k-1)$, $g_1(i-1,k)+1$ and $g_2(\min(k-1,n_l),k-1)+1$, if $i = \langle l,m_l\rangle$ for some l;

   (b) the maximum of $g_1(i,k-1)$ and $g_1(i-1,k)+1$, otherwise.

   Similarly, define $g_2(i,k)$ equal to

   (a) the maximum of $g_2(i,k-1)$, $g_2(i-1,k)+1$ and $g_1(\min(k-1,n_l'),k-1)+1$, if $i = \langle l,m_l'\rangle$ for some l;

   (b) the maximum of $g_2(i,k-1)$ and $g_2(i-1,k+1)$, otherwise.

2. For each $i \in \{1, 2, \ldots, k-1\}$ do:

   (a) Simulate the first k steps in the computations of $\phi_i(\langle i, m_i \rangle)$ and $\phi_i(\langle i, m'_i \rangle)$.

   (b) If the computation of $\phi_i(\langle i, m_i \rangle)$ terminates within k steps, then for each $j \in \{i, i+1, \ldots, k-1\}$ compute the smallest number r_j such that $\phi_i(\langle i, m_i \rangle) < \langle j, r_j \rangle$ and set $m'_j = max(m'_j, r_j)$.

   (c) If the computation of $\phi_i(\langle i, m'_i \rangle)$ terminates within k steps, then for each $j \in \{i+1, \ldots, k-1\}$ compute the smallest r_j satisfying $\phi_i(\langle i, m'_i \rangle) < \langle j, r_j \rangle$ and set $m_j = max(m_j, r_j)$.

3. For each $j \in \{1, \ldots, k-1\}$:

   (a) Set n_j equal to the greatest $\phi_i(\langle i, m_i \rangle)$ such that $i \in \{1, \ldots, j\}$ and the computation of $\phi_i(\langle i, m_i \rangle)$ terminates within k steps;

   (b) Set n'_j equal to the greatest $\phi_i(\langle i, m'_i \rangle)$ such that $i \in \{1, \ldots, j\}$ and the computation of $\phi_i(\langle i, m'_i \rangle)$ terminates within k steps.

4. Let m_k be the smallest number such that $\langle k, m_k \rangle > n'_{k-1}$ and m'_k be the smallest number such that $\langle k, m'_k \rangle > n_{k-1}$.

Proposition 1 *For each j the values of m_j, m'_j, n_j and n'_j change only finitely many times.*

PROOF. It suffices to prove the proposition for m_j and m'_j because, if from some moment the values of $m_1, m'_1, \ldots, m_j, m'_j$ do not change then the values of n_j and n'_j can change only finitely many times. (This can happen only when the computation of $\phi_i(\langle i, m_i \rangle)$ or $\phi_i(\langle i, m'_i \rangle)$ terminates for some $i \in \{1, \ldots, j\}$.)

We prove by induction that the values of m_j and m'_j can change only finitely many times.

The value of m_1 never changes.

Further, if we know that the values of $m_1, \ldots, m_j$ change only finitely many times, there exists a moment after which the algorithm does not change them. After this moment the value of m'_j can change only j times: when the computation of $\phi_i(\langle i, m_i \rangle)$ terminates for some $i \in \{1, \ldots, j\}$. Hence, the value of m'_j changes only finitely many times.

Similarly, if we know that the values of $m'_1, m'_2, \ldots, m'_j$ change only finite number of times, we can conclude that after some moment the algorithm does not change

them. After this moment the value of m_{j+1} can change only j times: when the computation of $\phi_i(\langle i, m'_i \rangle)$ terminates for some $i \in \{1, \ldots, j\}$. ∎

Further, m_i, m'_i, n_i, n'_i denote the last values of these variables (the values which are not changed later).

Proposition 2 *For each x there are only finitely many y such that $g_1(x,y) \neq g_1(x,y+1)$ or $g_2(x,y) \neq g_2(x,y+1)$.*

PROOF. By the way of contradiction. Let x_1 be the smallest number such that $g_1(x_1,y) \neq g_1(x_1,y+1)$ for infinitely many y and x_2 be the smallest number such that $g_2(x_2,y) \neq g_2(x_2,y+1)$ for infinitely many y.

Let $x_1 = \langle i_1, j_1 \rangle$ and $x_2 = \langle i_2, j_2 \rangle$.

Proposition 3 *j_1 is equal to the last value of m_{i_1} during the computation.*

PROOF. By the way of contradiction, assume that the last value of m_{i_1} is different from j_1. Then, for some N and all $k > N$, $g_1(x_1,k)$ is computed as

$$\max(g_1(x_1,k-1), g_1(x_1-1,k)+1).$$

x_1 is the smallest number such that $g_1(x_1,y) \neq g_1(x_1,y+1)$ for infinitely many y. Hence, there exists an N_0 such that $g_1(x_1-1,N_0) = g_1(x_1-1,N_0+1) = \ldots$. Then $g_1(x_1,N_0) = g_1(x_1,N_0+1) = \ldots$. A contradiction. ∎

Similarly, j_2 is equal to the last value of m'_{i_2}.

Let $i_1 \leq i_2$. ($i_1 > i_2$ case is similar.)

If $j \in \{1, \ldots, i_2\}$, and the computation of $\phi_j(\langle j, m_j \rangle)$ terminates, the algorithm sets m'_{i_2} to such value that $\langle i_2, m'_{i_2} \rangle > \phi_j(\langle j, m_j \rangle)$. Hence

$$x_2 = \langle i_2, m'_{i_2} \rangle > \max\{\phi_j(\langle j, m_j \rangle) | j \in \{1, \ldots, i_2\}\} \geq$$

$$\geq max\{\phi_j(\langle j, m_j \rangle) | j \in \{1, \ldots, i_1\}\} = n_{i_1}$$

Starting from some step, m_{i_1} is equal to j_1 (Proposition 3). Then, $g_1(x_1,k)$ is computed as

$$\max(g_1(x_1,k-1), g_1(x_1-1,k)+1, g_2(n_{i_1},k-1)+1)$$

$x_2 > n_{i_1}$ implies that $g_2(n_{i_1},y) \neq g_2(n_{i_1},y+1)$ for finitely many y. Also, there exist only finitely many y such that $g_1(x_1-1,y) \neq g_1(x_1-1,y+1)$. Hence, there exists an

N such that $g_1(x_1-1,N) = g_1(x_1-1,N+1) = \ldots$ and $g_2(n_{i_1},N) = g_2(n_{i_1},N+1) = \ldots$. Then, $g_1(x_1,N+1) = g_1(x_1,N+2) = \ldots$.

Contradiction with the assumption that $g_1(x_1,y) \neq g_1(x_1,y+1)$ for infinitely many y. ∎

Hence, for each x there exists an N such that $g_1(x,N) = g_1(x,N+1) = \ldots$. This implies that $h_1(x) = \lim_{y\to\infty} g_1(x,y)$ is defined for all $x \in \mathbb{N}$. Similarly, we can prove that $h_2(x) = \lim_{y\to\infty} g_2(x,y)$ is defined for all $x \in \mathbb{N}$.

We defined that $g_1(i,k)$ is equal to

$$\max(g_1(i,k-1), g_1(i-1,k)+1, \ldots).$$

Hence, $g_1(x,y)$ is nondecreasing in y and increasing in y, i.e. monotonic. Similarly, $g_2(x,y)$ is monotonic.

We take an arbitrary x and denote by y_1 the last value of m_x. If $\phi_x(\langle x,y_1\rangle)$ is defined, the computation of $\phi_x(\langle x,y_1\rangle)$ terminates in N steps for some $N \in \mathbb{N}$. Then, $n_x \geq \phi_x(\langle x,y_1\rangle)$ after the N^{th} step of the algorithm.

Hence, $g_1(\langle x,y_1\rangle,k) \geq g_2(\phi_x(\langle x,y_1\rangle),k-1)+1$ for arbitrary $k > N$. Taking $k \to \infty$ we obtain that $h_1(\langle x,y_1\rangle) \geq h_2(\phi_x(\langle x,y_1\rangle))+1 > h_2(\phi_x(\langle x,y_1\rangle))$.

Similarly we can prove that for arbitrary x there exists an y_2 such that $h_2(\langle x,y_2\rangle) > h_1(\phi_x(\langle x,y_2\rangle)$ or $\phi_x(\langle x,y_2\rangle)$ is undefined. ∎

5.3.2 The system of notations S_g

Let $h(x)$ be lim-computable as witnessed by a monotonic $g(x,y)$. (We will need the monotonicity because Definition 15 requires that $\varphi_{q_S(x)}(0)$, $\varphi_{q_S(x)}(1)$, ... is increasing.)

Consider the system of notations S_g consisting of the following notations:

1. a for $a \in \mathbb{N}$;
2. α_i for $i = 1,2,\ldots$;
3. $\alpha_i - \omega\cdot a + b$ for $i = 1,2,\ldots$ and $a,b \in \mathbb{N}$;
4. ω^2.

The notations are natural numbers by the definition of system of notations. It is assumed that some effective encoding of mentioned expressions by natural numbers

is fixed. Further we shall call these expressions notations though in reality notations are their encodings by natural numbers.

The notations denote the following ordinals:

1. a denotes the ordinal a;

2. α_i denotes $\omega \cdot a_i$ where $a_i = h(i) + 1$.

3. $\alpha_i - \omega \cdot a$ denotes $\omega \cdot (a_i - a) + b$ if $a < a_i$ and $\omega + b$ otherwise;

4. ω^2 denotes the ordinal ω^2.

Next, we define $k_{S_g}, p_{S_g}, q_{S_g}$. It can be checked that they are defined so that the ordinals corresponding to notations are as we described.

The function $k_{S_g}(x)$ is defined to be 0 for the notation 0, 2(a limit ordinal) for notations $\alpha_i, \alpha_i - \omega \cdot a$ and ω^2, 1(a successor ordinal) for all other notations.

The function $p_{S_g}(x)$ is defined to be $b - 1$ if $x = b$ and $\alpha_i - \omega \cdot a + (b - 1)$ if $x = \alpha_i - \omega \cdot a + b$.

The function $q_{S_g}(x)$ is defined in the following way:

1. If $x = \alpha_i$ then $q_{S_g}(x)$ is a program such that $\varphi_{q_{S_g}(x)}(j) = \alpha_i - \omega + j$;

2. If $x = \alpha_i - \omega \cdot a$ then $q_{S_g}(x)$ is such that

$$\varphi_{q_{S_g}(x)}(j) = \begin{cases} j & \text{if } g(x,j) \leq a+1 \\ \alpha_i - \omega \cdot (a+1) + j & \text{if } g(x,j) > a+1 \end{cases}$$

3. If $x = \omega^2$ then $\varphi_{q_{S_g}(x)}(i) = \alpha_i$.

The system S_g can be defined for many functions g. The notations are the same only the ordinals denoted by these notations may be different. So, each IIM working in a system S_g for some g can work in system S_g for another g, too.

Let $M_1, M_2, \ldots$ be an enumeration of all IIM working in S_g and starting with ω^2 on the counter. We consider the sequence of partial recursive functions $\phi_1, \phi_2, \ldots$ such that $\phi_i(j)$ is computed by the following algorithm:

1. Simulate M_i on the input function f_1 such that $f_1(0) = j$ and $f_1(x) = 0$ if $x \neq 0$. If after reading $f_1(0), f_1(1), \ldots, f_1(k)$ M_i outputs the first conjecture, goto 2.

2. Simulate M_i on f_1 and the function $f_2(x)$ such that $f_2(k+1) = 1$ and $f_2(x) = f_1(x)$ if $x \neq k+1$. If on one of functions M_i diminishes the ordinal ω^2 to some α_l, define $\phi_i(j) = l$.

Proposition 4 *If $\phi_i(j)$ is undefined, M_i does not identify one of functions f_1 and f_2.*

PROOF. M_i has issued a conjecture after reading $f_1(0), f_1(1), \ldots, f_1(k)$. This conjecture is incorrect for at least one of the functions f_1 and f_2. If M_i identifies both f_1 and f_2, then M_i makes a mindchange on one of these functions.

$\phi_i(j)$ is undefined if and only if M_i does not decrease the ordinal on its counter on f_1 and f_2. Then, M_i does not make a mindchange on any of the functions f_1 and f_2. ∎

5.3.3 The main result

Next, we take the sequence $\phi_1, \phi_2, \ldots$ and construct the functions $g_1(x,y)$ and $g_2(x,y)$ from Lemma 13.

Lemma 14 *There exists a set of recursive functions U such that $U \in EX^{S_{g_1}}_{\omega^2}$ and $U \notin EX^{S_{g_2}}_{\omega^2}$.*

PROOF. We consider an $EX^{S_{g_1}}_{\omega^2}$-IIM M working as follows:

1. Read $f(0), \ldots, f(k)$. If $k = 0$ output a program computing
$$f_1(x) = \begin{cases} f(0) & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

2. If $k > 0$ and $f(k) = 0$, output the same conjecture as on $f(0), \ldots, f(k-1)$.

3. If $k > 0$, $f(1) = \ldots = f(k-1) = 0$ and $f(k) \neq 0$ diminish the notation on the counter from ω^2 to $\alpha_{f(0)}$ and output a program computing
$$f_1(x) = \begin{cases} f(x) & \text{if } x \leq k \\ 0 & \text{otherwise} \end{cases}$$

4. Otherwise, find i and j such that $f(0) = \langle i, j \rangle$ and simulate M_i on the input $f(0), \ldots, f(k)$ in the system of notations S_{g_2}. Then, each time when M_i modifies its ordinal:

(a) If M_i replaces ω^2 by α_l for some $l \in \mathbb{N}$, M replaces $\alpha_{f(0)}$ by $\alpha_{f(0)} - \omega + 1$.

(b) If M_i replaces $\alpha_l - \omega \cdot a_1 + b_1$ by $\alpha_l - \omega \cdot a_2 + b_2$, M replaces $\alpha_{f(0)} - \omega \cdot (a_1 + 1) + (b_1 + 1)$ by $\alpha_{f(0)} - \omega \cdot (a_2 + 1) + (b_2 + 1)$.

(c) If M_i replaces $\alpha_l - \omega \cdot a_1 + b_1$ by b_2, M replaces $\alpha_{f(0)} - \omega \cdot (a_1 + 1) + (b_1 + 1)$ by $(b_2 + 1)$.

Let m denote the largest number such that $m < k$ and the conjectures of M after reading $f(0)$, $f(1)$, ..., $f(m)$ and after reading $f(0)$, ..., $f(m+1)$ are different. If M_i makes a mindchange after reading $f(0)$, ..., $f(n)$ for some n such that $m < n < k$, M outputs a program computing

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

Otherwise, the conjecture remains the same as on $f(0), \ldots, f(k-1)$.

The set U consists of all total recursive functions identified by M.

Proposition 5 *If $i \in \mathbb{N}$ then M_i does not $EX^{S_{g_2}}_{\omega^2}$-identify U.*

PROOF. By Lemma 13, there is an y_1 such that $h_1(\langle i, y_1 \rangle) > h_2(\phi_i(\langle i, y_1 \rangle))$ or $\phi_i(\langle i, y_1 \rangle)$ is undefined.

Case 1. $\phi_i(\langle i, y_1 \rangle)$ is undefined.

Proposition 4 implies that M_i does not identify a function f such that $f(0) = \langle i, y_1 \rangle$ and $f(x) \neq 0$ for at most one $x > 0$. M always identifies such functions.

Case 2. $\phi_i(\langle i, y_1 \rangle)$ is defined and $h_1(\langle i, y_1 \rangle) > h_2(\phi_i(\langle i, y_1 \rangle))$.

We consider functions f such that

(a) $f(0) = \langle i, y_1 \rangle$,

(b) $f(0), f(1), \ldots, f(k)$ is an initial fragment of f_1 or f_2 (Proposition 4) after which M_i changes its ordinal from ω^2 to $\alpha_{\phi_i(\langle i, y_1 \rangle)}$, and

(c) M_i identifies f.

T denotes the set of all such functions.

The function f_1 always satisfies requirements (a) and (b). Hence, $f_1 \in T$ if and only if f_1 is identified by M_i.

If T is empty then f_1 is not identified by M_i. However, it is identified by M. Hence, U is not identified by M_i.

It remains to consider the case when T is nonempty.

Proposition 6 *If $f \in T$ and, after reading $f(0), \ldots, f(n)$, M_i makes its last mindchange on f, there exists a function f' such that*

1. *$f'(i) = f(i)$ for $i \in \{0, \ldots, n\}$;*
2. *$f(x) \neq f'(x)$ for some $x > n$;*
3. *M $EX^{S_{g_1}}_{\omega^2}$-identifies f'.*

PROOF. We consider M working on the input $f(0), \ldots, f(k)$. At the beginning it changes the ordinal notation from ω^2 to $\alpha_{\langle i,y_1\rangle}$.

When M_i puts the notation $\alpha_{\phi_i(\langle i,y_1\rangle)}$ in the system S_{g_2} on the counter, M puts the notation $\alpha_{\langle i,y_1\rangle} - \omega + 1$ in the system S_{g_1} on the counter. The notation $\alpha_{\phi_i(\langle i,y_1\rangle)}$ denotes the ordinal $\omega \cdot h_2(\phi_i(\langle i, y_1\rangle) + 1)$. The notation $\alpha_{\langle i,y_1\rangle} - \omega + 1$ denotes the ordinal $\omega \cdot h_1(\langle i, y_1\rangle) + 1$.

Lemma 13 implies $h_2(\phi_i(\langle i, y_1\rangle)) < h_1(\langle i, y_1\rangle)$. Hence, $h_2(\phi_i(\langle i, y_1\rangle)) + 1 \leq h_1(\langle i, y_1\rangle)$ and

$$\omega \cdot h_2(\phi_i(\langle i, y_1\rangle) + 1) < \omega \cdot h_1(\langle i, y_1\rangle) + 1$$

So, at the beginning, the ordinal on the counter of M is larger than the ordinal on the counter of M_i. The rules of modification for M guarantee that it always remains larger than the ordinal on the counter of M_i. So, each time when M_i modifies its notation, M can do it, too.

We consider

$$f'(x) = \begin{cases} f(x) & \text{if } x \leq n \\ f(n+1) + 1 & \text{if } x = n + 1 \\ 0 & \text{otherwise} \end{cases}$$

After reading $f'(0), \ldots, f'(n)$ IIM M_i makes a mindchange because $f(0) = f'(0), \ldots, f(n) = f'(n)$. We consider M computing the conjecture on the input $f'(0), \ldots, f'(n+1)$. M simulates M_i, finds that M_i has made a mindchange and changes its conjecture to the program computing $f'(x)$. So, M identifies f'. ∎

Proposition 7 *There exists a function $f_1 \in T$ and a number n such that M_i makes the last mindchange on f after reading $f_1(0), \ldots, f_1(n)$ and M_i does not identify any function f' such that $f' \neq f_1$ but $f_1(x) = f'(x)$ for $x \in \{0, \ldots, n\}$.*

PROOF. For each function $f \in T$ we take the ordinal which appears on the ordinal counter of M_i after this counter is modified for the last time. We select the smallest

among these ordinals and denote it α_0. Let f_1 be a function such that M_i puts α_0 on the ordinal counter on the input f_1. n is the number such that, after reading $f_1(0), \ldots, f_1(n)$, the ordinal is modified for last time.

T_1 denotes the set of all functions $f \in T$ such that $f_1(0) = f(0), \ldots, f_1(n) = f(n)$. If M_i identifies a function f' such that $f' \neq f_1$ but $f_1(x) = f'(x)$ for $x \in \{0, \ldots, n\}$, then $f' \in T_1$. To identify f', M_i has to make a mindchange and replace α_0 by a smaller ordinal f_1. A contradiction because we defined α_0 as the smallest ordinal which appears on the counter. ∎

Combining Propositions 6 and 7 we obtain Proposition 5. ∎

So, EX_{ω^2}-IIM can identify U in S_{g_2}. Lemma is proved. ∎

Similarly, we can construct U such that $U \in EX_{\omega^2}^{S_{g_2}}$ and $U \notin EX_{\omega^2}^{S_{g_1}}$. Hence, theorem holds with systems S_{g_1} and S_{g_2} as A and B. ∎

5.4 Two systems of notations: O and P

In this section, we consider the systems O and P (cf. section 2.5).

5.4.1 The system O

For inductive inference with procrastination, O is the strongest possible system of notations.

Theorem 12

$$EX_\alpha^A \subseteq EX_\alpha^O$$

for an arbitrary system of notations A.

Proof. By Lemma 2, if we have an EX_α^A-IIM, we can obtain $EX_\alpha^{S_1}$-IIM, replacing the ordinal notations x by $\varphi(x)$. ∎

Theorem 12 is based on the fact that an arbitrary system of notations can be embedded in O. If we consider only univalent systems of notations (systems containing at most one notation for each ordinal) then, for each system of notations we can construct a stronger system of notations. (This follows from Theorem 15.)

5.4.2 The system P: small ordinals

For small ordinals, every IIM working in P can be transformed into equivalent IIM in any other system of notations. Any procrastination behaviour which can be described with ordinals smaller than $\omega^2 + \omega \cdot 2$ in P can be described with the same ordinals in any other system of notations.

Theorem 13 *If A is a system of ordinal notations and α is an ordinal, $\alpha < \omega^2 + \omega \cdot 2$, then $EX^P_\alpha \subseteq EX^A_\alpha$.*

PROOF. We prove the theorem for $\alpha = \omega^2 + \omega + m$ case.

Lemma 15 *There exists a partial recursive function $t(x, y, z)$ such that, if*

1. *x and y are notations in the system P,*
2. *z is a notation in the system A, and*
3. *$v_P(y) < v_P(x) \leq v_A(z) < \omega^2 + \omega$,*

then $t(x, y, z)$ is a notation in A, $v_P(y) \leq v_A(t(x, y, z))$ and $v_A(t(x, y, z)) < v_A(z)$.

PROOF. We define an auxiliary function $SeekA(x, n)$. This function has two arguments: an ordinal notation x (in the system A) and a natural number n. It returns some natural number. The function $SeekA(x, n)$ is computed by the following algorithm:

1. $M_0 = \{L(x)\}, i = 0$;
2. If M_i contains only notations for the ordinal 0, return i as the value of $SeekA(x, n)$.
3. $M_{i+1} = \{L(\varphi_{q_A(y)}(k)) | y \in M_i \& k_A(y) \neq 0 \& k \in \{0, 1, \ldots, n\}\}, i = i + 1$, goto 2.

Proposition 8 *Let x be a notation in the system A for the ordinal $\omega \cdot a + b$. Then, the set M_i contains only notations for ordinals $0, \omega, \ldots, \omega \cdot (a - i)$.*

PROOF. By induction.
Base Case. $M_0 = \{L(x)\}$ and $L(x)$ denotes $\omega \cdot a$.
Inductive Case. Let $z \in M_{i+1}$. Then $z = L(\varphi_{q_A(y)}(k))$ for some $y \in M_i$.

The proposition holds for M_i. Hence, y denotes $\omega \cdot j$ for $j \leq (a - i)$. Then, $\varphi_{q_A(y)}(k)$ denotes an ordinal which is smaller than $\omega \cdot$, i.e. $\omega \cdot j_1 + j_2$ for $j_1 < j$ and $L(\varphi_{q_A(y)}(k))$ denotes $\omega \cdot j_1$.

We have $j \leq a - i$ and $j_1 \leq j - 1 \leq a - i - 1$. Hence, any $z \in M_{i+1}$ is a notation for one of $0, \ldots, \omega \cdot (a - i - 1)$. ∎

Proposition 9 *If x is a notation in system A for $\omega \cdot a + b$, then $SeekA(x, n) \leq a$ for arbitrary n.*

PROOF. By Proposition 8, M_a contains only notations for the ordinal 0. ∎

Proposition 10 *Let x be a notation for $\omega \cdot a$ (or greater ordinal) in the system A. There exists an n such that $SeekA(x, n) \geq a$.*

PROOF. Denote $x_1 = L(x)$. x_1 is a notation for the ordinal $\omega \cdot a$ (or greater ordinal). We consider the sequence of notations $\varphi_{q_A(x_1)}(0)$, $\varphi_{q_A(x_1)}(1)$, ... The sequence of ordinals denoted by these notations converges to the ordinal denoted by x_1. Hence, there exists a number m_1 such that $\varphi_{q_A(x_1)}(m_1)$ denotes an ordinal greater than or equal to $\omega \cdot (a - 1)$.

Let $x_2 = L(\varphi_{q_A(x_1)}(m_1))$. It is a notation for $\omega \cdot (a - 1)$ or greater ordinal. Then, similarly as m_1 and x_2 were obtained from x_1, we obtain m_2 and x_3 from x_2. We continue so until we obtain x_{a+1}.

Then, x_1 denotes ordinal $\omega \cdot a$ or greater ordinal, x_2 denotes $\omega \cdot (a - 1)$ or greater ordinal and so on, x_a denotes ω or greater ordinal, x_{a+1} denotes 0 or greater ordinal.

Let $m = \max(m_1, \ldots, m_a)$. When $SeekA(x, m)$ is computed, $x_1 \in M_0$, $x_2 \in M_1$, ..., $x_{a+1} \in M_a$. Hence, each of sets M_i for $i < a$ contains a notation for ordinal which is greater than 0 and $SeekA(x, m)$ is at least a. ∎

The function $t(x, y, z)$ is computed as follows:

1. If z denotes a successor ordinal, return $p_A(z)$ as the value of $t(x, y, z)$.
2. Otherwise, find a and b such that $y = \omega \cdot a + b$. (It is possible to do it because y is a notation in the system P.)
3. $n = 1$.
4. If $SeekA(\varphi_{q_A(z)}(i), n) \geq a$ and $N(\varphi_{q_A(z)}(i)) \geq b$ for some $i \in \{1, \ldots, n\}$, return $\varphi_{q_A(z)}(i)$ as the value of $t(x, y, z)$.

5. If $SeekA(\varphi_{q_A(z)}(i), n) \geq a+1$ for some $i \in \{1, \ldots, n\}$, return $\varphi_{q_A(z)}(i)$ as the value of $t(x, y, z)$.

6. $n = n+1$; goto 4.

We prove that the algorithm given above works correctly, i.e., if x, y, z satisfy the conditions of Lemma 15, then the algorithm terminates and returns the value $t(x, y, z)$ satisfying Lemma 15.

Case 1. z denotes a successor ordinal. Then $t(x, y, z) = p_A(z)$ is a notation for the ordinal preceding $v_A(z)$ i.e. for the largest ordinal which is less than $v_A(z)$.

The ordinal $v_P(y)$ is smaller than $v_A(z)$. Hence, $v_P(y) \leq v_A(t(x, y, z))$.

Case 2. z denotes a limit ordinal.

Then, $v_A(z)$ is less than $\omega^2 + \omega$, i.e. $v_A(z)$ is at most ω^2. $v_P(y) < v_A(z)$. Hence, $v_P(z)$ is less than ω^2, i.e. $v_P(z) = \omega \cdot a + b$ for some $a, b \in \mathbb{N}$.

The algorithm returns the value $t(x, y, z) = \varphi_{q_A(z)}(i)$ in two cases:

(a) $SeekA(\varphi_{q_A(z)}(i), n) \geq a+1$.

Then, $\varphi_{q_A(z)}(i)$ denotes $\omega \cdot (a+1)$ or greater ordinal (cf. Proposition 9).

(b) $SeekA(\varphi_{q_A(z)}(i), n) = a$ and $N(\varphi_{q_A(z)}(i)) \geq b$.

$\varphi_{q_A(z)}(i)$ denotes $\omega \cdot a$ or greater ordinal. $N(\varphi_{q_A(z)}(i)) \geq b$ implies that the ordinal denoted by $\varphi_{q_A(z)}(i)$ is at least $\omega \cdot a + b$.

In both cases the ordinal denoted by $\varphi_{q_A(z)}(i)$ is greater than or equal to $v_P(y)$.

It remains to prove that the algorithm always returns some notation.

z denotes the limit ordinal which is greater than $v_P(y) = \omega \cdot a + b$. Hence, z denotes $\omega \cdot (a+1)$ or greater ordinal. We consider two cases:

(a) z denotes an ordinal which is greater than $\omega \cdot (a+1)$.

Then, for some $i \in \mathbb{N}$, $\varphi_{q_A(z)}(i)$ is at least $\omega \cdot (a+1)$. Proposition 10 implies $SeekA(\varphi_{q_A(z)}(i), m) \geq a+1$ for some $m \in \mathbb{N}$. Hence, when n becomes greater than $\max(i, m)$, the algorithm returns a notation.

(b) z denotes $\omega \cdot (a+1)$.

For some $i \in \mathbb{N}$, $\varphi_{q_A(z)}(i)$ is at least $\omega \cdot a + b$. $\varphi_{q_A(z)}(i)$ denotes an ordinal which is less than $v_A(z) = \omega \cdot (a+1)$. Hence, it denotes $\omega \cdot a + c$ for $c \geq b$, and $N(\varphi_{q_A(z)}(i)) = c \geq b$. For some m, $SeekA(\varphi_{q_A(z)}(i), m) \geq a$ (Proposition 10). Hence, the algorithm returns a notation, when n reaches $\max(i, m)$. ∎

We assume that M_P is an EX^P_α-IIM. We show how to transform M_P to EX^A_α-IIM M_A.

Let x_0 be a notation for $\alpha = \omega^2 + \omega + m$ in the system A. x_1 denotes $L(x_0)$. n_0 is a number such that $\varphi_{q_A(x_1)}(n_0)$ denotes an ordinal which is greater than or equal to ω^2.

Conjectures of M_A are the same as conjectures of M_P. Ordinal notations on the counter are transformed as follows:

1. When M_P puts $\alpha = \omega^2 + \omega + m$ on the counter, M_A puts x_0. Further, when M_P replaces $\omega^2 + \omega + k$ by $\omega^2 + \omega + k - 1$, M_A replaces x by $p_A(x)$.

2. When M_P replaces $\omega^2 + \omega$ with $\omega^2 + k$ for some $k \in \mathbb{N}$, then M_A searches the sequence $\varphi_{q_A(x_1)}(n_0)$, $\varphi_{q_A(x_1)}(n_0+1)$, ... and finds an i such that $N(\varphi_{q_A(x_1)}(i)) \geq k$. M_A puts $\varphi_{q_A(x_1)}(i)$ on the counter.

3. Further, if M_P changes the ordinal from x to y, then M_A computes $t(x, y, z)$ where z is the notation on the counter of M_A and replaces z by $t(x, y, z)$.

Proposition 11 *The ordinal on the counter of M_A is always greater than or equal to the ordinal on the counter of M_P.*

PROOF. When the counter of M_P contains $\alpha = \omega^2 + \omega + m$ on the counter, the counter of M_A contains x_0, i.e. a notation for $\omega^2 + \omega + m$. Further, when M_P replaces $\omega^2 + \omega + k$ by $\omega^2 + \omega + k - 1$, M_A replaces x which denotes $\omega^2 + \omega + k$ by $p_A(x)$ which denotes preceding ordinal, i.e. $\omega^2 + \omega + k - 1$.

If M_A puts the notation $\varphi_{q_A(x_1)}(i)$ on the counter, then $\varphi_{q_A(x_1)}(i) \geq \omega^2$ and $N(\varphi_{q_A(x_1)}(i)) \geq k$. Hence, $\varphi_{q_A(x_1)}(i) \geq \omega^2 + k$, i.e. it is greater than or equal to the ordinal on the counter of M_P.

Further, consider the case, when M_A replaces z by $t(x, y, z)$. Assume that, before this modification, the ordinal on the counter of M_A is greater than or equal to the ordinal on the counter of M_P, i.e. $v_A(z) \geq v_P(x)$, Then Lemma 15 implies that $v_A(t(x, y, z)) \geq v_P(y)$, i.e., the ordinal on the M_A's counter after modification is greater than or equal to the ordinal on the M_P's counter after modification. ∎

Hence, each time when M_P modifies its counter, M_A can modify it, too. It means that always, when M_P outputs a conjecture, M_A can do it too, i.e. M_A simulates M_P successfully. ∎

5.4.3 The system P: large ordinals

However, for $\omega^2 + \omega \cdot 2$ and larger ordinals the situation is different.

Theorem 14 *There exists a system of ordinal notations A such that*

$$EX^P_{\omega^2+\omega\cdot 2} - EX^A_{\omega^2+\omega\cdot 2} \neq \emptyset.$$

PROOF. We consider an IIM M working in the system P according to the following instructions:

1. Read $f(0), \ldots, f(k)$. If $k = 0$ output a program computing the function f_0 such that $f_0(0) = f(0)$ and $f_0(x) = 0$ if $x \neq 0$.

2. If $k \neq 0$ and $f(k) = 0$ output the same conjecture as on $f(0), \ldots, f(k-1)$.

3. If $k \neq 0$ and $f(k) \neq 0$ then

   (a) If the counter contains 0, output the same conjecture as on $f(0)$, ..., $f(k-1)$.

   (b) If the counter contains a notation for a successor ordinal, replace it by the notation for preceding ordinal. Change conjecture to a program computing
   $$f_0(x) = \begin{cases} f(x) & \text{if } x \leq k \\ 0 & \text{otherwise} \end{cases}$$

   (c) If the counter contains a notation for a limit ordinal, replace it by the k^{th} notation from the sequence converging to that limit ordinal. (ω^2 is replaced by $\omega \cdot k$, $\omega \cdot a$, $\omega^2 + \omega$ and $\omega^2 + \omega \cdot 2$ are replaced by $\omega \cdot (a-1) + k$, $\omega^2 + k$ and $\omega^2 + \omega + k$, respectively.)

   Change conjecture similarly to the previous case.

Let U be the set of all functions identified by M. We construct a system of notations A such that $U \notin EX^A_{\omega^2+\omega\cdot 2}$.

Lemma 16 *Let A be a system of notations and M_1 be an IIM identifying U with an ordinal mindchange bound in the system A. Then, for any initial segment $f(0)$, $f(1)$, ..., $f(n)$, the ordinal on the counter of M_1 after reading $f(0)$, $f(1)$, ..., $f(n)$ is greater than or equal to the ordinal on the counter of M after reading the same initial segment.*

PROOF. By the way of contradiction. Let O be the set of all ordinals which appear on the counter of M_1 after reading segments $f(0), \ldots, f(n)$ such that the ordinal on the counter of M_1 is less than the ordinal on the counter of M. Let α be the smallest ordinal in O.

Consider the segment $f(0), \ldots, f(n)$ after reading which the counter of M_1 contains α and the counter of M contains greater ordinal. β denotes the ordinal on the counter of M after reading $f(0), \ldots, f(n)$.

Case 1. $\alpha = 0$.

Consider the functions

$$f_1(x) = \begin{cases} f(x) & \text{if } x \leq n, \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x) = \begin{cases} f(x) & \text{if } x \leq n, \\ 1 & \text{if } x = n+1, \\ 0 & \text{otherwise} \end{cases}$$

$f(0), \ldots, f(n)$ is the initial segment of both f_1 and f_2. After reading it M_1 has issued the same conjecture on both f_1 and f_2. It cannot change its conjecture on any of these two functions because, before mindchange, it needs to modify the counter. However, this is impossible because the counter contains the ordinal 0. Hence, M_1 does not identify at least one of functions f_1 and f_2.

$\beta > \alpha = 0$. Hence, M can modify its counter and make at least one more mindchange. After reading $f_2(n+1) = 1$, M makes a mindchange and outputs a correct program for f_2. On the function f_1, the correct program was issued earlier, after reading the last nonzero value among $f_1(0), \ldots, f_1(n)$. Hence, M identifies both f_1 and f_2.

We have proved that M_1 does not identify some function in U. A contradiction.

Case 2. $\alpha \neq 0$. Let m be

1. $n+1$ if β is a successor ordinal;

2. a number such that $\varphi_{q_A(\beta)}(m) > \alpha$ and $m > n$, if β is a limit ordinal.

Consider the functions

$$f_1(x) = \begin{cases} f(x) & \text{if } x \leq m, \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x) = \begin{cases} f(x) & \text{if } x \leq m, \\ 1 & \text{if } x = m, \\ 0 & \text{otherwise} \end{cases}$$

Similarly to the previous case, both f_1 and f_2 are identified by M. More, after reading the last nonzero value of f_1 or f_2 the counter of M contains a notation for an ordinal which is greater than or equal to α.

If M_1 identifies both f_1 and f_2, it makes a mindchange on one of these functions. Then, it also modifies the counter, i.e. replaces α by a smaller ordinal. We take the initial segment of f_1 or f_2 after reading which it happens. After reading it the ordinal on the counter of M_1 is smaller than α and α is less or equal than the ordinal on the counter of M.

Hence, the ordinal which is on the counter of M_1 belongs to O. However, we assumed that α is the smallest element of O. Contradiction. ∎

The system A. It is well-known that there exist lim-computable functions which grow faster than any recursive function. Let $h(x)$ be a function such that $h(x)$ is lim-computable as witnessed by a monotonic $g(x,y)$ and, for any recursive f, $h(x) > f(x)$ for all except finitely many x. We use $h(x)$ to define the system A.

The system A consists of notations:

1. α;
2. $\omega \cdot a + b$ for $a, b \in \mathbb{N}$;
3. $\alpha_i + j$ for $i, j \in \mathbb{N}$;
4. $\alpha_{i,j} + k$ for $i, j, k \in \mathbb{N}$.

$k_S(x)$ is defined to be 0, if $x = 0$, 1 if ($x = \omega \cdot a + b$ and $b > 0$) or ($x = \alpha_i + j$ and $j > 0$) or ($x = \alpha_{i,j} + k$ and $k > 0$) and 2 otherwise.

$p_S(x)$ is defined to be $\omega \cdot a + b - 1$, if $x = \omega \cdot a + b$, $\alpha_i + j - 1$, if $x = \alpha_i + j$, $\alpha_{i,j} + k - 1$, if $x = \alpha_{i,j} + k$.

$q_S(x)$ is defined as:

1. a program computing the sequence $\omega \cdot (a-1)$, $\omega \cdot (a-1) + 1$, ..., if $x = \omega \cdot a$.
2. a program computing α_0, $\alpha_1 + 1$, ..., if $x = \alpha$.
3. a program computing $\alpha_{i,0}$, $\alpha_{i,1} + 1$, ... if $x = \alpha_i$.
4. a program computing $t(0), t(1), \ldots$ where

$$t(y) = \begin{cases} \omega \cdot y & \text{if } g(i,y) < j, \\ t(y-1) + 1 & \text{if } g(i,x) \geq j \end{cases}$$

if $x = \alpha_{i,j}$.

It is easy to check that, if k_A, p_A and q_A are defined so, then $\omega \cdot a + b$ is a notation for the ordinal $\omega \cdot a + b$.

Proposition 12 *$\alpha_{i,j}$ denotes ω^2, if $h(i) < j$ and $\omega \cdot k$ for some $k \in \mathbb{N}$ otherwise.*

PROOF. Let $h(i) < j$. Then $g(i,x) < j$ for all $x \in \mathbb{N}$. Hence,

$$t(0) = 0, t(1) = \omega, t(2) = \omega \cdot 2, \ldots,$$

i.e. $0, \omega, \omega \cdot 2, \ldots$ is a sequence of ordinals converging to the ordinal with notation $\alpha_{i,j}$. Hence, $\alpha_{i,j}$ denotes ω^2.

Let $h(i) \geq j$. Monotonicity of $g(i,x)$ implies that there exists an N such that $g(i,x) < j$ if $j < N$ and $g(i,x) \geq j$ if $j \geq N$. Hence,

$$t(0) = 0, t(1) = \omega, \ldots, t(N-1) = \omega \cdot (N-1),$$

$$t(N) = \omega \cdot (N-1) + 1, t(N+1) = \omega \cdot (N-1) + 2, \ldots$$

This sequence converges to $\omega \cdot N$. Hence, $\alpha_{i,j}$ denotes $\omega \cdot N$ in this case. ∎

Hence, $\alpha_{i,j} + k$ denotes $\omega^2 + k$ if $h(i) < j$ and $\omega \cdot a + k$ for some $a \in \mathbb{N}$ otherwise.

Proposition 13 *For any $i \in \mathbb{N}$, α_i denotes $\omega^2 + \omega$.*

PROOF. α_i is the limit of the sequence

$$\alpha_{i,0}, \alpha_{i,1} + 1, \ldots, \alpha_{i,j} + j, \ldots$$

$\alpha_{i,j}$ is ω^2 for $j > h(i)$. Hence, α_i is the limit of the sequence

$$\omega^2 + h(i) + 1, \omega^2 + h(i) + 2, \ldots,$$

i.e. α_i denotes $\omega^2 + \omega$. ∎

Hence, $\alpha_i + j$ denotes $\omega^2 + \omega + j$ and α (the limit of $\alpha_0, \alpha_1 + 1, \ldots$) denotes $\omega^2 + \omega \cdot 2$.

By the way of contradiction, assume that an IIM M_1 $EX_{\omega^2+\omega\cdot 2}$-identifies U in the system A. We obtain the contradiction by constructing a recursive function h_1 such that $h_1(x) > h(x)$. $h_1(n)$ is computed by the algorithm below:

1. Simulate M_1 on everywhere zero function until it outputs a conjecture. (It certainly happens because everywhere zero function belongs to U and, hence is identified by M_1.)

   Let N be the number of the input values read by M_1 before issuing the first conjecture.

2. Let $s = 0$, $m_s = \max(n, N)$, and

$$f_i^s(x) = \begin{cases} 0 & \text{if } x \leq m_s, \\ i & \text{if } x = m_s + 1, \\ 0 & \text{otherwise .} \end{cases}$$

(a) Simulate M_1 on f_1^s and f_2^s until it changes its conjecture on one of these functions. Let i_s, j_s be such that M_1 makes a mindchange on the function $f_{i_s}^s$ after reading j_s values of $f_{i_s}^s$.

(b) If M_1 changes the ordinal from α_k to $\alpha_{k,l} + l$, goto step 3.

(c) Otherwise, M_1 changes the ordinal from $\alpha_k + p + 1$ to $\alpha_k + p$. Then, define $m_{s+1} = \max(j_s, p)$. Set

$$f_i^{s+1}(x) = \begin{cases} f_{i_s}^s(x) & \text{if } x \leq m_{s+1}, \\ i & \text{if } x = m_{s+1} + 1, \\ 0 & \text{otherwise .} \end{cases}$$

$s = s + 1$. Go to step 2a

3. Let $\alpha_{k,l} + l$ be the notation on the counter of M_1 after reading $f_{i_s}^s(0), \ldots, f_{i_s}^s(j_s)$. Set $h_1(n) = l$.

Lemma 17 *$h_1(n) \geq h(n)$ for all $n \in \mathbb{N}$.*

PROOF. We consider the segments $f_{i_s}^s(0), \ldots, f_{i_s}^s(j_s)$. Each next segment $f_{i_{s+1}}^{s+1}(0), \ldots, f_{i_{s+1}}^{s+1}(j_{s+1})$ is an extension of the previous segment $f_{i_s}^s(0), \ldots, f_{i_s}^s(j_s)$. Let $f(0), \ldots, f(j)$ be the last of these segments (i.e. the segment after reading which M_1 modifies the notation from α_k to $\alpha_{k,l} + l$). Consider M_1 working with the input $f(0), \ldots, f(j)$. It modifies the counter as follows:

First, M_1 replaces α with $\varphi_{q_A(\alpha)}(k) = \alpha_k + k$ for some $k \in \mathbb{N}$. Then, it replaces $\alpha_k + k$ with $\alpha_k + k - 1$ and so on, until α_k is on the counter. After that, M_1 replaces α_k by $\alpha_{k,l} + l$ for some $l \in \mathbb{N}$.

Proposition 14 *$k > n$.*

PROOF. M_1 makes the first modification on the segment $f_{i_1}^1(0), \ldots, f_{i_1}^1(j_1)$. This segment contains only one nonzero value: $f_{i_1}^1(m_1 + 1)$. So, M makes only one modification: it replaces $\omega^2 + \omega \cdot 2$ by $\omega^2 + \omega + m_1 + 1$ after reading $f_{i_1}^1(m_1 + 1)$.

Lemma 16 implies that the ordinal on the counter of M_1 must be greater than or equal to the ordinal on the counter of M. Hence, $\alpha_k + k$ denotes ordinal which is at

least $\omega^2+\omega+m_1+1$. We defined α_k+k as the notation for $\omega^2+\omega+k$. Hence, $k \geq m_1+1$. From $m_1=\max(n,N)$ it follows that $m_1+1>n$ and $k>n$. ∎

Proposition 15 $l \geq h(k)$.

PROOF. We use

Proposition 16 *After reading $f(0),\ldots,f(j)$, the counter of M contains an ordinal which is greater than or equal to ω^2.*

PROOF. If the counter of M contains $\omega^2+\omega$ or greater ordinal after reading $f(0)$, ..., $f(n)$, the proposition is evident. Otherwise, at some moment M replaces $\omega^2+\omega$ by a smaller ordinal.

M makes mindchanges and modifies the counter only after reading a nonzero value from the input. Nonzero values in the segment $f(0),\ldots,f(i)$ are $f(m_1+1)$, $f(m_2+1)$, ... We assume that M replaces $\omega^2+\omega$ after reading $f(m_r+1)$. Then, M replaces $\omega^2+\omega$ by ω^2+m_r+1.

Let α_k+p be the ordinal on the counter of M_1 after reading $f(0)$, ..., $f(j_r)$. By the definition of m_r, $m_r \geq p$.

Each segment $f^{s+1}_{i_{s+1}}(0)$, ..., $f^{s+1}_{i_{s+1}}(j_{s+1})$ is an extension of the previous segment $f^s_{i_s}(0)$, ..., $f^s_{i_s}(j_s)$. More, this extension contains exactly one nonzero value which does not belong to $f^s_{i_s}(0)$, ..., $f^s_{i_s}(j_s)$. Hence, while reading $f^{s+1}_{i_{s+1}}(j_s+1)$, ..., $f^{s+1}_{i_{s+1}}(j_{s+1})$, M modifies the counter only once. The machine M_1 modifies the counter on $f^{s+1}_{i_{s+1}}(j_s+1)$, ..., $f^{s+1}_{i_{s+1}}(j_{s+1})$ at least once because in step 2a we wait until M_1 makes a mindchange (and modifies the counter). Hence, the number of M_1's modifications is greater than or equal to the number of M's modifications.

Step 2a is repeated until M_1 replaces α_k by $\alpha_{k,l}+l$. After α_k+p appears on the counter of M_1, step 2a is executed at most $p+1$ times. (Each time M_1 modifies its counter at least once. After p modifications M_1 has α_k on the counter and after $p+1$ modifications M_1 has $\alpha_{k,l}+l$ for some $l \in \mathbb{N}$ on the counter. Then the algorithm goes to step 3.)

After $p+1$ modifications the ordinal on M's counter is at least $\omega^2+m_r+1-(p+1) \geq \omega^2$. ∎

After reading the segment $f(0),\ldots,f(j)$, the counter of M_1 contains $\alpha_{k,l}+l$ and the counter of M contains ω^2 or greater ordinal. By Lemma 16, $\alpha_{k,l}+l$ denotes ω^2 or greater ordinal.

Hence, $l \geq h(k)$ (cf. Proposition 12). ∎

The algorithm computing h_1 defines $h_1(n) = l$. Hence, $h_1(n) = l \geq h(k)$. From $k > n$ it follows that $h(k) \geq h(n)$ and $h_1(n) \geq h(n)$. Lemma is proved. ∎

So, there exists a recursive function h_1 which is greater than or equal to h. However, h grows faster than any recursive function. Contradiction, proving the theorem. ∎

So, for the ordinal $\omega^2 + \omega \cdot 2$, there are procrastination behaviours which can be defined using the system P but cannot be defined using some other systems of notations.

5.5 Better systems versus larger ordinals

There are two ways how to increase the power of an IIM with an ordinal mindchange bound:

- by using larger ordinals;
- by using more powerful system of notations.

Inductive inference using different ordinals and a fixed system of notations was investigated in [24]. In the previous sections of this paper we investigated the influence of system of notations for a fixed ordinal.

In this section, we consider possible tradeoffs between these two methods.

5.5.1 Larger ordinals instead of better systems

Theorem 15 shows that the use of greater ordinals cannot replace the use of more expressive system of notations.

Theorem 15 *If A is a univalent system of notations and α is an ordinal to which A assigns notation, there exists a system B such that*

$$EX^B_{\omega^2} \not\subseteq EX^A_\alpha$$

PROOF. We define

$$U_h = \{f | f \text{ is total recursive and there are at most } h(f(0))$$

$$x > 0 \text{ such that } f(x) \neq 0\}.$$

Lemma 18 *If h is* lim*-computable then*

$$U_h \in EX^B_{\omega^2}$$

for some system of notations B.

PROOF. Let $h(x)$ be lim-computable as witnessed by $g(x,y)$. Without loss of generality we assume that $g(x,y)$ is monotonous. (If it is not so, we can replace $g(x,y)$ by

$$g_1(x,y) = \max_{i \le x, j \le y} g(i,j).$$

$g_1(x,y)$ converges to $h_1(x)$ such that $h(x) \le h_1(x)$ for all x. Hence $U_h \subseteq U_{h_1}$. From $U_{h_1} \in EX^B_{\omega^2}$ it follows that $U_h \in EX^B_{\omega^2}$.)

We use the system of notations S_g defined in the proof of Theorem 11.

U_h is inferred by the IIM M described below:

1. M reads $f(0), \ldots, f(n)$. If $n = 0$ it outputs a program computing

$$f'(x) = \begin{cases} f(0) & x = 0 \\ 0 & \text{otherwise} \end{cases}$$

2. If $n > 0$ and $f(n) = 0$ it outputs the same conjecture as on $\langle f(0), \ldots, f(n-1) \rangle$.

3. If $n > 0$ and $f(n) \neq 0$ it outputs a program computing

$$f'(x) = \begin{cases} f(x) & x \le n \\ 0 & \text{otherwise} \end{cases}$$

as a conjecture. The counter is modified as follows:

(a) If the counter contains ω^2, it is replaced by $\alpha_{f(0)}$.

(b) If the counter contains $\alpha_{f(0)}$, it is replaced by $\alpha_{f(0)} - \omega$.

(c) If the counter contains $\alpha_{f(0)} - \omega \cdot k$, M searches the sequence of notations $\hat{\varphi}_{q_{S_g}(\alpha_{f(0)} - \omega \cdot k)}(0)$, $\hat{\varphi}_{q_{S_g}(\alpha_{f(0)} - \omega \cdot k)}(1)$, ... looking for a notation $\alpha_{f(0)} - \omega \cdot (k+1) + j$ for some j and puts it on the counter.

(d) If the counter contains $\alpha_{f(0)} - \omega \cdot k + j$, it is replaced by $\alpha_{f(0)} - \omega \cdot k + j - 1$.

After reading the last nonzero value of $f \in U$, M issues a correct conjecture. Hence, it suffices to prove that M is able to modify the counter (and make mind-change) always when it is necessary. First, we prove

Proposition 17 *If $i \leq h(f(0)) - 2$ there exists an m such that*

$$\varphi_{q_{S_g}(\alpha_{f(0)} - i\omega)}(m) = \alpha_{f(0)} - \omega \cdot (i+1) + k$$

for some k.

PROOF. From $\lim_{x \to \infty} g(f(0), x) = h(f(0))$ and $i \leq h(f(0)) - 2$ we have that $g(f(0).m) > i + 1$ for some m. Then, by the definition of q_{S_g},

$$\varphi_{q_{S_g}(\alpha_{f(0)} - \omega \cdot i)}(m) = \alpha_{f(0)} - \omega \cdot (i+1) + m.$$

∎

Hence, M can modify the ordinal from $\alpha_{f(0)} - \omega \cdot i$ to $\alpha_{f(0)} - \omega \cdot (i+1)$, if $i \leq h(f(0)) - 2$.

If $f \in U_h$ then f has at most $h(f(0))$ nonzero values. Hence, machine M modifies the counter at most $h(f(0))$ times. We must prove that all these modifications are possible. Before any modification at most $h(f(0)) - 1$ other modifications are made.

First, M replaces ω^2 by $\alpha_{f(0)}$, then $\alpha_{f(0)}$ by $\alpha_{f(0)} - \omega$, $\alpha_{f(0)} - \omega$ by $\alpha_{f(0)} - \omega \cdot 2 + j$, and so on. After $h(f(0)) - 1$ replacements the ordinal is at least $\alpha_{f(0)} - \omega \cdot (h(f(0) - 2)$. Proposition 17 implies that it can be replaced by a smaller ordinal. ∎

Lemma 19 *If A is a univalent system of notations and α is an ordinal to which A assigns notation, there exists a* lim*-computable function h such that*

$$U_h \notin EX_\alpha^B$$

PROOF. $M_1, M_2, \ldots$ is a numbering of all IIMs working in the system of notations A and putting the notation for α on their ordinal counter at the beginning. (The system A is univalent. Hence, α has only one notation and it can be checked whether IIM puts the notation for α on its ordinal counter.)

We construct a function $g(x, y)$ converging in the limit to $h(x)$. It will be constructed so that for each i the IIM M_i does not identify a 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 does not identify at least one of them.

1. Set $j = 0$, $m = 1$:

$$f_1 = f_2 = \begin{cases} i & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

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

3. Set k equal to the number of values read by M_i so far, t equal to the conjecture of M_i produced in previous step.

$$f_2(x) = \begin{cases} f_1(x) & \text{if } x < k \\ 1 & \text{if } x = k \\ 0 & \text{if } x > k \end{cases}$$

$m = m+1, g(i,j) = m, j = j+1$;

4. Simulate j steps of the computations of M_i on inputs from functions f_1 and f_2. If M_i does not change conjecture t on one of these functions, define $g(i,j) = m, j = j+1$ and repeat.

5. If M_i changes the conjecture t on a function $f_l (l \in \{1,2\})$, set t equal to the new conjecture of M_i on f_l, k equal to the number of values of f_l read by M_i, $f_1(x)$ equal to $f_l(x)$,

$$f_2(x) = \begin{cases} f_l(x) & \text{if } x < k \\ 1 & \text{if } x = k \\ 0 & \text{if } x > k \end{cases}$$

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

If Step 5 is executed infinitely many times, we can construct a function on which M_i makes infinitely many mindchanges. By Lemma 1, IIM with an ordinal mindchange counter cannot make infinitely many mindchanges. Hence, Step 5 is executed finitely many times. So, m is increased by 1 finite number of times and $h(i) = \lim_{x \to \infty} g(i,x)$ always exists.

We consider the moment when Step 5. is executed for last time. After this moment $f_1(x)$ and $f_2(x)$ are two different functions on which M_i outputs the same conjecture t and does not change it. Hence, it does not identify one of them.

Always, when a new nonzero value of f_1 or f_2 is defined, m is increased by 1. It implies that $g(i,j) = m$ remains greater than the number of x such that $f_1(x) \neq 0$ or $f_2(x) \neq 0$. Hence, the number of x such that $f_1(x) \neq 0$ or $f_2(x) \neq 0$ is at most $h(i) = \lim_{x \to \infty} g(i,x)$ and $f_1, f_2 \in U_h$.

We have proved that, for an arbitrary IIM M_i, there exists a function $f \in U_h$ which is not EX_α^B-identified by M_i. ∎

From Lemmas 18 and 19 the theorem follows. ∎

Theorem 16 *For an arbitrary constructive ordinal α there exist systems of notations A and B such that*

$$EX_{\omega^2}^A \not\subseteq EX_\alpha^B$$

PROOF. For each constructive ordinal α there exists a univalent system of notations B which assigns a notation to α[45]. We apply Theorem 15 to this α and B and obtain A. ∎

It can be noted that the system A can be constructed so that it is univalent, too. (Small modification in the definition of S_g suffices.)

Corollary 3 *Let S_1 be the Kleene's universal system of notations. For an arbitrary univalent system B and constructive ordinal α*

$$EX_{\omega^2}^{S_1} \not\subseteq EX_\alpha^B.$$

Proof. Follows from Theorem 15 and Theorem 12 ∎

So, we see that even the use of very large ordinals cannot replace the better system of notations.

5.5.2 Better systems instead of larger ordinals

On the other hand, the use of better system of notations cannot replace the use of larger ordinals.

Theorem 17 *For an arbitrary system of notations A and ordinal α there exists a set of recursive functions U such that*

1. $U \in EX_\alpha^A$;
2. *If $\beta < \alpha$, then $U \notin EX_\beta^B$ for any system of notations B.*

PROOF. Consider an IIM M working as follows:

1. Read $f(0), \ldots, f(k)$. If $k = 0$ output a program computing the function f_0 such that $f_0(0) = f(0)$ and $f_0(x) = 0$ if $x \neq 0$.

2. If $k \neq 0$ and $f(k) = 0$ output the same conjecture as on $f(0), \ldots, f(k-1)$.

3. If $k \neq 0$ and $f(k) \neq 0$ then

   (a) If the notation on the ordinal counter denotes 0, output the same conjecture as on $f(0), \ldots, f(k-1)$.

   (b) If the ordinal counter contains the notation γ which denotes a successor ordinal, replace it by $p_A(\gamma)$. Change the conjecture to a program computing

$$f_0(x) = \begin{cases} f(x) & \text{if } x \leq k \\ 0 & \text{otherwise} \end{cases}$$

   (c) If the ordinal counter contains the notation γ for a limit ordinal, replace γ by $\varphi_{q_A(\gamma)}(k)$, change the conjecture similarly to the previous case.

Let U be the set of all functions identified by M. Evidently $U \in EX^A_\alpha$. Similarly to Lemma 16 we can prove

Lemma 20 *Let B be a system of notations and M_1 be an IIM identifying U with an ordinal mindchange bound in the system B. Then, for any initial segment $f(0), f(1), \ldots, f(n)$, the ordinal on the counter of M_1 after reading $f(0), f(1), \ldots, f(n)$ is greater than or equal to the ordinal on the counter of M after reading the same initial segment.*

Hence, if M_1 identifies U, then its counter contains α or greater ordinal at the beginning. ∎

5.6 Summary

We have studied the dependence of the learning power on the used system of ordinal notations. We have proved that, for small ordinals (below ω^2) there is no such dependence. For ω^2 and greater ordinals, the dependence is rather strong.

The power of EX_α-identification is influenced by both ordinal α and used system of notations. Results of section 5.5 show that these two influences are, in general,

independent. In particular, even the use of very large ordinals cannot compensate the weakness of the system of notations (cf. Theorem 16). This shows the important role of the system of notations.

Chapter 6

Probability hierarchies

6.1 Overview

Within inductive inference, there has been much work on team learning. (cf. surveys in [28, 51]) It is well-known that a team of learning machines can learn more than a single machine. Various aspects of this phenomena have been investigated. Researchers have noted that the advantages of teams over single machines appear not only because there are more machines in team. The cooperation between learning machines and the diversity of their approaches are also important. (The last aspect, the diversity of approaches between learning machines in a team, was recently studied in [9].)

Probabilistic learning is closely related to team learning. Any team of machines can be simulated by a single probabilistic machine with the same success ratio. The simulation of a probabilistic machine by a team of deterministic machines is often possible, too.

In this paper, we consider **Fin**, finite learning of total recursive function2.2.3. **Fin** is supposed to be one of the simplest learning paradigms. However, if we consider probabilistic and team learning, the situation becomes very complex. By now, probabilistic **Fin**-type learning has been studied for 18 years. Still, we are far from the complete understanding of the situation.

The investigation of probabilistic FINite learning was started by Freivalds in [22]. He gave a complete description of learning capabilities for probabilistic machines with probabilities of success above $\frac{1}{2}$. These results were extended to team learning by Daley, Pitt, Valauthapillai and Will[20].

The further progress appeared to be very difficult. Daley, Kalyanasundaram and

Velauthapillai[18] determined learning capabilities for probabilistic learners with success probabilities in $[\frac{24}{49}, \frac{1}{2}]$. Later, Daley and Kalyanasundaram[17] extended that to $[\frac{12}{25}, \frac{1}{2}]$. Proofs became more and more complicated. (The full version of [17] is more than 100 pages long.)

PFin (Popperian **Fin**)-type learning is a simplified version of **Fin**-type learning (cf. section 2.2.3). In **PFin**-type learning, a learning machine is allowed to output only programs computing total recursive functions. Probabilistic and team **PFin**-type learning is simpler than **Fin**-type learning. However, it has many properties similar to **Fin**. Daley, Kalyanasundaram and Velauthapillai[19, 16] determined the capabilities of probabilistic **PFin**-type learners in interval $[\frac{3}{7}, \frac{1}{2}]$. However, even in **PFin**-type learning the situation becomes more and more complicated when the probability of success for learning machine decreases. [16] wrote "the prospects of determining all the learning capabilities and all the redundancy types for even the interval $]\frac{2}{5}, \frac{1}{2}]$ appear to be bleak indeed".

In this paper, we return to **PFin**-type learning. Instead of trying to determine exact points at which the learning capabilities are different (either single points or sequences of points generated by a formula) we propose an another approach. We investigate the probability structure on the whole and its properties.

We prove that the probability hierarchy for **PFin**-type learning (the set of success probabilities at which learning capabilities of probabilistic machines are different) is well-ordered in decreasing ordering. More precisely, it is order-isomorphic to ϵ_0, very large (and complicated) ordinal. (It is known that ϵ_0 expresses the set of all expressions possible in first-order arithmetic.)

This result shows that the probability hierarchy for **PFin** is very complex. The part of the hierarchy investigated before ($[\frac{3}{7}, 1]$) is order-isomorphic to the ordinal 3ω and is very simple compared to the entire probability hierarchy. Thus, we can conclude that finding an explicit description for the whole hierarchy is hardly possible. (The previous research shows that, even for segments like $[\frac{3}{7}, 1]$ with a simple topological structure, this task is difficult because of irregularities in the hierarchy[16].)

However, we construct a decision algorithm for the probability hierarchy of **PFin**. It receives two numbers $p_1, p_2 \in [0, 1]$ and answers whether the learning with probability p_1 is equivalent to the learning with probability p_2. Also, we construct a universal simulation algorithm receiving

- $p_1, p_2 \in [0, 1]$ such that **PFin**-learning with these probabilities is equivalent and

- **PFin**-learning machine M with the probability of success p_1

and transforming M into machine M' with the probability of success p_2.

We note that these decidability results (and most of other results in this paper as well) make heavy use of the fact that **PFin**-hierarchy is well-ordered. We suppose that this is the first application of well-ordered sets (and systems of notations for well-orderings) to a problem of such type.

Further, we consider relations between probabilistic and team learners. We prove that any probabilistic **PFin**-type learning machine can be simulated by a team of deterministic machines with the same success ratio. Thus, we prove that, for **PFin**-type learning, team learning is exactly of the same power as probabilistic learning.

6.2 Preliminaries

For results of this chapter, we need more definitions (in addition to those in chapter 2). We define probabilistic and team learning in section 6.2.1. Then, in section 6.2.2, we modify the definitions of a system of notations for the purposes of this chapter

6.2.1 Probabilistic and team learning

Scientific discoveries are rarely done by one person. Usually, a discovery is the result of collective effort. In the area of computational learning theory, this observation has inspired the research on team learning.

Let $M = \{M_1, \ldots, M_s\}$ be a team consisting of **Fin**-type learning machines M_1, $\ldots$, M_s. The team M $[r,s]$**Fin**-learns a function f if at least r of M_1, $\ldots$, M_s **Fin**-learn f. The collection of all $[r,s]$**Fin**-learnable sets is denoted $[r,s]$**Fin**.

Besides deterministic learning machines, we can consider probabilistic ones.

Let M be a probabilistic learning machine. M FIN$\langle p\rangle$-learns (**Fin**-learns with probability p) a set of functions U if, for any function $f \in U$, the probability that M FIN-learns f is at least p. **Fin**$\langle p\rangle$ denotes the collection of all **Fin**$\langle p\rangle$-learnable sets.

Probabilistic and team **PFin**-learning is defined similarly to probabilistic and team **Fin**-learning. The requirement that learners must output only programs computing total recursive functions is absolute, i.e.

1. All conjectures of all machines in a **PFin**-team must be programs computing total recursive functions.

2. A probabilistic **PFin**-learning machine is not allowed to output a program which does not compute total recursive function even with a very small probability.

Definition 30 The probability hierarchy for **Fin** is the set $A \subseteq \mathbb{R} \cap [0,1]$ such that

1. For any two different $p_1, p_2 \in A$,

$$\mathbf{Fin}\langle p_1 \rangle \neq \mathbf{Fin}\langle p_2 \rangle$$

i.e. learning with probability of success p_1 is not equivalent to learning with probability of success p_2.

2. If $x \in A$, $x \leq p$ and $[x, p[$ does not contain any points belonging to A, then

$$\mathbf{Fin}\langle x \rangle = \mathbf{Fin}\langle p \rangle.$$

Essentially, the probability hierarchy is the set of those probabilities at which the learning capabilities of probabilistic machines are different.

The probability hierarchy for **PFin** is defined similarly.

6.2.2 Systems of notations

In this chapter we use subsets of $\mathbb{Q} \cap [0,1]$ that are well-ordered in decreasing ordering. A subset of $\mathbb{Q}$ is well-ordered in decreasing ordering if it does not contain infinite monotonously increasing sequences. Below, we give our definition of a system of notations for well-ordered subsets of $\mathbb{Q}$. It is a modification of the definition of a system of notations for ordinals(section 2.5).

Let A be a subset of $\mathbb{Q}$ which is well-ordered in decreasing ordering. All elements of A can be classified as follows:

1. The greatest element of the set A. We call it the *maximal* element.

2. Elements x which have immediately preceding element in decreasing ordering (i.e. the element y such that $x < y$ and $[x, y]$ does not contain any points belonging to A). Such elements are called *successor* elements.

3. All other elements $x \in A$. They are called *limit* elements.

Definition 31 A system of notations for A is a tuple of functions $\langle k_S, p_S, q_S \rangle : \mathbb{Q} \to \mathbb{N}$ such that

1. $k_S(x)$ is equal to
   (a) 0, if x is the maximal element;
   (b) 1, if x is a successor element;
   (c) 2, if x is a limit element;
   (d) 3, if $x \notin A$;
2. If $k_S(x) = 1$, then $p_S(x)$ is defined and it is the element immediately preceding x in descending ordering.
3. If $k_S(x) = 2$, then $q_S(x)$ is defined and it is a program computing a decreasing sequence of elements of the set A converging to x.

Systems of notations are convenient for manipulating well-ordered sets in our proofs. Possibly, a system of notation is the most appropriate way of describing the probability hierarchy for **PFin**. The structure of this hierarchy is very complicated (cf. Section 6.9) and it seems unlikely that more explicit decriptions exist.

Below, we give a useful property of systems of notations.

Lemma 21 *Let $A \subseteq \mathbb{Q}$ be a set which is well-ordered in descending ordering and has a system of notations S. Let $f_1(p)$ be the largest number in A such that $f_1(p) \leq p$ and $f_2(p)$ be the smallest number in A such that $p \leq f_2(p)$. Then f_1 and f_2 are computable functions.*

PROOF. f_1 and f_2 are computed by the algorithm below:

1. Set x equal to an arbitrary number from A smaller than p.
2. (a) If $x = p$, output: $f_1(p) = f_2(p) = x$. Stop.
   (b) If x is a successor element and $p_S(x) \geq p$, then output: $f_1(p) = x$ and $f_2(p) = p_S(x)$. Stop.
   (c) If x is a successor element and $p_S(x) \leq p$, set $x = p_S(x)$.

(d) If x is a limit element and $x \neq p$, take a sequence

$$\varphi_{q_S(x)}(0), \varphi_{q_S(x)}(1), \ldots .$$

Search for the smallest i satisfying $\varphi_{q_S(x)}(i) \leq p$ and set $x = \varphi_{q_S(x)}(i)$. (Such i exists because this sequence is monotonously decreasing and converges to x and $x < p$.)

3. Repeat step 2.

While this algorithm works, x remains less or equal to p.

From the definition of the system of notations it follows that the values of f_1 and f_2 output by the algorithm are correct. It remains to prove that algorithm always outputs $f_1(p)$ and $f_2(p)$.

Assume, by way of contradiction, that the algorithm does not output $f_1(p)$ and $f_2(p)$ for some $p \in Q$. It can happen only if it goes into eternal loop, i.e. if Step 2 is executed infinitely many times.

During Step 2 the value of x increases. Let x_i be the value of x after the i^{th} repetition of Step 2.

$$x_1, x_2, x_3, \ldots$$

is infinite monotonously increasing sequence.

However, A is well-ordered. Hence, it does not contain infinite monotonously increasing sequences. A contradiction. ∎

6.3 Three examples

One can ask: what probabilistic and team inference has in common with well-ordered sets?

In Figure 6.3, we show the known parts of probability hierarchies for three learning criteria:

- **Ex** (learning in the limit, cf.Pitt and Smith[42, 43]),
- **Fin** (Freivalds[22], Daley, Kalyanasundaram and Velauthapillai[18]), and
- **PFin** (Daley, Kalyanasundaram and Velauthapillai[19, 16]).

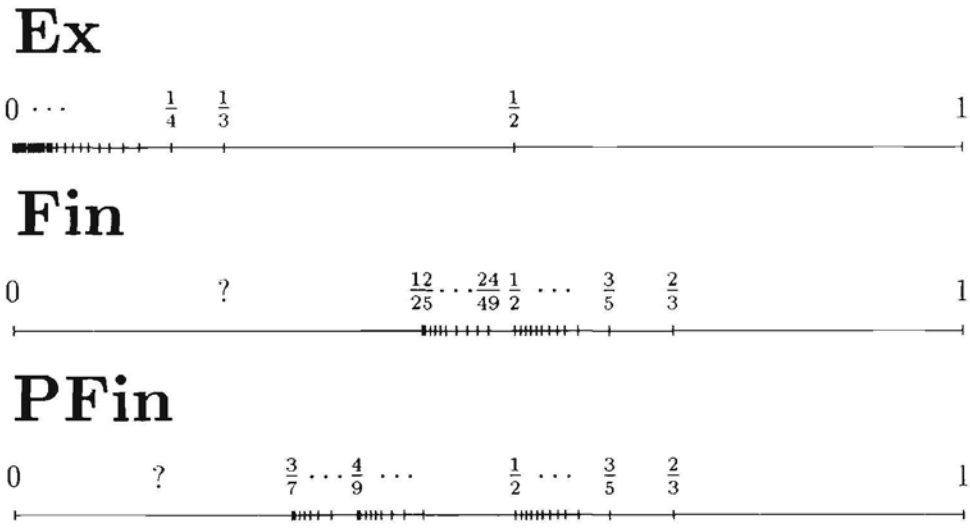

Figure 6.1: The probability hierarchies for **Ex**, **Fin** and **PFin**

We see that these probability hiearchies contain infinite decreasing sequences but none of them contains an infinite increasing sequence. Known parts of these hierarchies are well-ordered in decreasing ordering.

Below, we will show that, for **PFin**-type learning, the entire hierarchy is well-ordered and will use this property to study its properties.

Recently, a similar result was obtained for **Fin**[3] and counterparts of some other our results were derived (cf. Section 6.11 for more information).

6.4 Characterization of PFin-hieararchy

Below, we give a recursive description for the probability hierarchy of **PFin** and use this description to prove decidability.

We claim that the probability hierarchy is the set A defined by the following rules:

1. $1 \in A$;

2. If $p_1, p_2, \ldots, p_s \in A$ and $p \in [0,1]$ is a number such that there exist $q_1, \ldots, q_s \in [0,1]$ satisfying

(a) $q_1 + q_2 + \ldots + q_s = p$;

(b) $\frac{p}{q_i+1-p} = p_i$ for $i = 1, \ldots, s$,

then $p \in A$;

The outline for the proof of this result is as follows. We start with several technical lemmas in section 6.5. In section 6.6 we give the proof that all probabilities from A give different learning capabilities. Then, in section 6.7 we prove that A is well-ordered and has a system of notations. Finally, in section 6.8 we use technical results of section 6.7 to prove that all different learning capabilities are defined by probabilities in A. Together, these results imply that A is the probability hiearachy for **PFin**. Our diagonalization theorem uses methods from Kummer's paper on **PFin**-teams[34] but simulation part uses new techniques and is far more complicated.

6.5 Technical lemmas

In this subsection, we study the properties of the rule that generates the set A. The results of this subsection are used in various parts of section 6.4. First, we show that the rule 2 can be described without using variables q_i.

Lemma 22 *If there exist* $q_1, \ldots, q_s \in [0,1]$ *satisfying* $q_1 + q_2 + \ldots + q_s = p$ *and* $\frac{p}{q_i+1-p} = p_i$ *for* $i = 1, \ldots, s$, *then*

$$p = \frac{s}{(s-1) + \sum_{i=1}^{s} \frac{1}{p_i}}. \tag{6.1}$$

PROOF. $\frac{p}{q_i+1-p} = p_i$ is equivalent to $q_i = \frac{p}{p_i} + p - 1$. Hence,

$$p = \sum_{i=1}^{s} q_i = \sum_{i=1}^{s} (\frac{p}{p_i} + p - 1) = \left(\sum_{i=1}^{s} \frac{1}{p_i} \right) p + s \cdot p - s,$$

$$s = \left(\sum_{i=1}^{s} \frac{1}{p_i} \right) p + (s-1)p,$$

$$p = \frac{s}{(s-1) + \sum_{i=1}^{s} (\frac{1}{p_i})}.$$

∎

We shall use both forms of the rule 2. The rule with q_i is more natural in simulation and diagonalization arguments but is less convenient for algebraic manipulations. We also use a version of Lemma 23 where equality is replaced by inequality.

Lemma 23 *If there exist* $q_1, \ldots, q_s \in [0,1]$ *satisfying* $q_1 + q_2 + \ldots + q_s = p$ *and* $\frac{p}{q_i+1-p} \leq p_i$ *for* $i = 1, \ldots, s$, *then*

$$p \leq \frac{s}{(s-1) + \sum_{i=1}^{s} \frac{1}{p_i}}. \tag{6.2}$$

PROOF. Similar to the proof of Lemma 22, with $\leq$ or $\geq$ instead of $=$ where necessary. ∎

Lemma 22 suggests that the rule 2 can be considered as a function of $p_1, \ldots, p_s$. Next lemmas show that this function is monotonous and continous.

Lemma 24 *If*

1. $p \in A$ *follows from* $p_1 \in A, \ldots, p_s \in A$ *by rule 2;*
2. $p' \in A$ *follows from* $p'_1 \in A, \ldots, p'_s \in A$ *by rule 2;*
3. $p_1 \leq p'_1, \ldots, p_s \leq p'_s$.

then $p \leq p'$. *If* $p_i < p'_i$ *for at least one* i, *then* $p < p'$.

PROOF. By Lemma 22

$$p = \frac{s}{(s-1) + \sum_{i=1}^{s} \frac{1}{p_i}} \text{ and } p' = \frac{s}{(s-1) + \sum_{i=1}^{s} \frac{1}{p'_i}}.$$

From $p_i \leq p'_i$ it follows that $\frac{1}{p_i} \geq \frac{1}{p'_i}$ and

$$(s-1) + \sum_{i=1}^{s} \frac{1}{p_i} \geq (s-1) + \sum_{i=1}^{s} \frac{1}{p'_i}.$$

$$p = \frac{s}{(s-1) + \sum_{i=1}^{s} \frac{1}{p_i}} \leq \frac{s}{(s-1) + \sum_{i=1}^{s} \frac{1}{p'_i}} = p'.$$

If $p_i < p'_i$ for some i, then $1/p_i > 1/p'_i$ and all inequalities are strict. ∎

Lemma 25 *Let* $p_j = \lim_{i \to \infty} p_{j,i}$ *and* $r = \lim_{i \to \infty} r_i$. *If, for all* $i \in \mathbb{N}$, $r_i \in A$ *follows from* $p_{1,i} \in A, \ldots, p_{s,i} \in A$ *by rule 2, then* $r \in A$ *follows from* $p_1 \in A, \ldots, p_s \in A$ *by rule 2.*

PROOF.

$$r = \lim_{i\to\infty} r_i = \lim_{i\to\infty} \frac{s}{(s-1)+\sum_{j=1}^{s} \frac{1}{p_{j,i}}} =$$

$$\frac{s}{(s-1)+\sum_{j=1}^{s} \frac{1}{\lim_{j\to\infty} p_{j,i}}} = \frac{s}{(s-1)+\sum_{j=1}^{s} \frac{1}{p_j}}.$$

∎

The last result of this section relates the numbers generated by applications of the rule 2 to $p_1 \in A, \ldots, p_s \in A$ and $\frac{p_1}{1+p_1} \in A, \ldots, \frac{p_s}{1+p_s} \in A$.

Lemma 26 *An application of the rule 2 to $x_1 \in A, \ldots, x_s \in A$ generates $p \in A$ if and only if an application of the rule 2 to $\frac{x_1}{1+x_1} \in A, \ldots, \frac{x_s}{1+x_s} \in A$ generates $\frac{p}{1+p} \in A$.*

PROOF. Assume that 6.1 is true for $p_1 = x_1, \ldots, p_s = x_s$. Then,

$$\frac{x}{1+x} = \frac{\frac{s}{(s-1)+\sum_{i=1}^{s} \frac{1}{x_i}}}{1+\frac{s}{(s-1)+\sum_{i=1}^{s} \frac{1}{x_i}}} = \frac{s}{(s-1)+\sum_{i=1}^{s} \frac{1}{x_i}+s} =$$

$$\frac{s}{(s-1)+\sum_{i=1}^{s} 1+\frac{1}{x_i}} = \frac{s}{(s-1)+\sum_{i=1}^{s} \frac{1+x_i}{x_i}}.$$

This is precisely 6.1 for $p_1 = \frac{x_1}{1+x_1}, \ldots, p_s = \frac{x_s}{1+x_s}$.

The opposite direction (6.1 for $p_1 = \frac{x_1}{1+x_1}, \ldots, p_s = \frac{x_s}{1+x_s}$ implies 6.1 is true for $p_1 = x_1, \ldots, p_s = x_s$) is similar. ∎

6.6 Universal diagonalization

We consider sets of functions described by trees. Similarly to [34], we define trees as finite nonempty subsets of $\mathbb{N}^*$ which are closed under initial segments. The root of each tree is the empty string ϵ. Next, we define labelings of trees by positive reals.

Definition 32 Let $0 < p < q$. An (p, q)-labeling of a tree T is a pair of mappings[1] $\nu_1, \nu_2 : T \to \mathbb{R}^+$ such that

[1]This is a modification of the definition in [34] which had only one mapping ν (our ν_1). The definition in [34] uses quantities $c_1, \ldots, c_s$, a counterpart of ν_2. In our opinion, specifying ν_2 as a part of labeling makes notation easier. This modification is of technical character and does not change anything important.

1. $\nu_1(\epsilon) \geq p$ and $\nu_2(\epsilon) = 0$,

2. If $t_1, \ldots, t_s$ are all direct successors of t, then $\sum_{i=1}^{s} \nu_2(t_i) \leq \nu_1(t) + \nu_2(t)$[2] and $\nu_1(t_i) + \nu_2(t_i) \geq p$ for $i = 1, \ldots, s$,

3. For each branch the sum of the ν_1-labels of all of its nodes is at most q.

p_T denotes the largest number such that there is a $(p, 1)$ labeling of T.

This is an extension of definition in [34] that considered labelings by natural numbers only. We shall consider both labelings by arbitrary positive reals and labelings by natural numbers. Further, a "labeling" means a "labeling by positive reals". If we consider labelings by natural numbers, it is specified explicitly.

We start by showing that for a tree T and its subtrees T_i, p_T and p_{T_i} are related similarly to rule 2.

Lemma 27 *Let $r > 0$ and T be a tree with (p, q)-labeling. Then, there is a (pr, qr)-labeling for T.*

PROOF. We multiply all labels by r and obtain a (pr, qr)-labeling. ∎

Lemma 28 *Let $t_1, \ldots, t_s$ be all direct successors of the root in a tree T and T_1, T_2, ..., T_S be the subtrees with roots t_1, t_2, ..., t_s. Assume there are $q_1, \ldots, q_s$ such that $\sum_{i=1}^{s} q_i = p$ and*

$$p = p_{T_i}(q_i + 1 - p)$$

for $i \in \{1, \ldots, s\}$. Then $p_T = p$.

PROOF. First, we construct a $(p, 1)$-labeling. Let ν_1^i, ν_2^i be a $(p_{T_i}, 1)$-labeling for T_i. We define

$$\nu_1(t) = \begin{cases} p, & \text{if } t = \epsilon \\ p - q_i, & \text{if } t = t_i \\ (1 + q_i - p)\nu_1^i(x), & \text{if } t \text{ is a descandant of } t_i \end{cases}$$

$$\nu_2(t) = \begin{cases} 0, & \text{if } t = \epsilon \\ q_i, & \text{if } t = t_i \\ (1 + q_i - p)\nu_2^i(x), & \text{if } t \text{ is a descandant of } t_i \end{cases}$$

Properties 1 and 2 can be checked directly from the definitions of ν_1 and ν_2.

[2] Definition in [34] incorrectly uses $\nu_1(t)$ instead of $\nu_1(t) + \nu_2(t)$ here.

We prove Property 3. Let u be a direct successor of t_i. Then, the sum of ν_1^i-labels on any branch starting at u is at most $1 - p_{T_i}$. (By Property 3 of ν_1^i, it is at most 1 for any branch starting at t_i and $\nu_1^i(t_i) \geq p_{T_i}$.) Hence, the sum of ν-labels for such a branch is at most $(q_i + 1 - p)(1 - p_{T_i})$. A branch starting at ϵ consists of ϵ, t_i and a branch starting at a direct descendant of t_i. Hence, the sum of all its ν-labels is at most

$$p + (p - q_i) + (q_i + 1 - p)(1 - p_{T_i}) = p + 1 - (1 + q_i - p) + (q_i + 1 - p)(1 - p_{T_i}) =$$

$$p + 1 - (q_i + 1 - p)p_{T_i} = p + 1 - p = 1.$$

By way of contradiction, assume that there is $p' > p$ and a $(p', 1)$-labeling (ν_1', ν_2') for T. Let $q_i' = \nu_2'(t_i)$. If we add $\nu_2'(t_i)$ to $\nu_1'(t_i)$, we obtain a $(p', 1 - p' + q_i')$-labeling for T_i. By Lemma 27, there is a $(p'/(1 - p' + q_i'), 1)$ labeling for this subtree. Hence,

$$\frac{p'}{1 - p' + q_i'} \leq p_i = \frac{p}{1 - p + q_i} < \frac{p'}{1 - p + q_i},$$

$$(1 - p' + q_i') > (1 - p + q_i),$$

$$p' - q_i' < p - q_i.$$

We consider the sum of these expressions for all i.

$$(s - 1)p' \leq s \cdot p' - \sum_{i=1}^{s} q_i' = \sum_{i=1}^{s} (p' - q_i') < \sum_{i=1}^{s} (p - q_i) =$$

$$= s \cdot p - \sum_{i=1}^{s} q_i = (s - 1)p$$

and $p' \leq p$. Contradiction, proving the lemma. ∎

By Lemma 22, the relation between p_T and $p_{T_1}, \ldots, p_{T_s}$ is also expressed by the formula 6.1. Next, we show that the $(p_T, 1)$-labeling of Lemma 28 uses only rational numbers and, hence, can be transfomed into a labeling that uses only integers.

Lemma 29 *For any tree T, $p_T \in \mathbb{Q}$.*

PROOF. By induction over the depth of T. For a tree consisting of root only, $p = 1$.

Otherwise, let $t_1, \ldots, t_s$ be all direct successors of the root in T and $T_1, T_2, \ldots, T_s$ be the subtrees with roots $t_1, t_2, \ldots, t_s$. The depth of these subtrees is smaller than the depth of T. Hence, all p_{T_i} are rationals. Formula 6.1 implies that p_T is rational, too. ∎

Lemma 30 *$(p_T, 1)$-labeling constructed in the proof of Lemma 28 uses only rational numbers.*

PROOF. By induction over the depth of T. Again, the lemma is evident for the tree with the root only.

For other trees, notice that all q_i can expressed by p and p_{T_i}. Hence, $q_1, \ldots, q_s$ are rationals. Label of the root is the rational number p, labels of $t_1, \ldots, t_s$ are rationals $p - q_1, \ldots, p - q_s$ and labels of other nodes are $(1 - p + q_i)\nu_j^i(t)$. $(1 - p + q_i)$ is a rational number because p and q_i are rationals and $\nu_j^i(t)$ is a rational number because ν_j^i is a part of the $(p_{T_i}, 1)$-labeling for a tree of smaller depth. ∎

Corollary 4 *Let T be a tree. Then there is $n \in \mathbb{N}$ such that T has $(p_T n, n)$-labeling with labels from $\mathbb{N}$.*

PROOF. Let n be the least common denominator of all rational numbers in the $(p_T, 1)$-labeling ν_1, ν_2 of Lemma 28. Then, $n\nu_1(t), n\nu_2(t)$ is a $(p_T n, n)$ labeling and uses only natural numbers. ∎

Next, we define sets of functions S_T corresponding to trees T. K denotes the halting set. χ_K denotes the characteristic function of K. (K_s) denotes a recursive enumeration of K.

Definition 33 [34] Let T be a tree of depth d. S_T is the set of all recursive functions of the form

$$i_1 \ldots i_d 0^{t_1} a_1 0^{t_2} a_2 \ldots 0^{t_l} a_l 0^\omega$$

where each $t_h = \min\{t : |\{j : i_j \in K_t\}| \geq h\}$ is finite, $(a_1, \ldots, a_l) \in T$, and either $l = |\{j : i_j \in K_t\}|$ or $(a_1, \ldots, a_l)$ is a leaf of T.

Lemma 31 *[34] If T has an (m, n)-labeling by integers then*

$$S_T \in [m, n]\mathbf{PFin}.$$

The next lemma is an extension of Kummer's results to probabilistic learning. The proof is similar to Theorem 16 in [34]. We give it here for completeness.

Lemma 32 *If $S_T \in \langle p \rangle \mathbf{PFin}[A]$ and K is not Turing reducible to A, then T has a $(p - \epsilon, 1)$ labeling for any $\epsilon > 0$.*

PROOF. Let k be the depth of T. M denotes an IIM that identifies S_T with the A-oracle. For arbitrary $i_1, \ldots, i_k$ we enumerate a set $T_{i_1,\ldots,i_k}$.

Initialization. Let $t = 0, c' = -1, T_{i_1,\ldots,i_k} = \emptyset$.

Step l. Search for the smallest $s > t$ satisfying $P(c, s)$ for some $c > c'$. If the search terminates, enumerate $(\chi_{K_s}(i_1), \ldots, \chi_{K_s}(i_k))$ into $T_{i_1,\ldots,i_k}$, set $t = s$, $c' = c$ and go to Step $l + 1$.

$P(c, s)$ is true iff $c = |\{j : i_j \in K_s\}|$ and, for each $(a_1, \ldots, a_c) \in T$ and $\sigma_c = i_1 \ldots i_d 0^{t_1} a_1 0^{t_2} a_2 \ldots 0^{t_c} a_c$, the probability that M^A outputs a program computing a function with an initial segment σ_c while reading $\sigma_c 0^s$ is at least $p - \epsilon$.

Proposition 18 $(\chi_K(i_1), \ldots, \chi_K(i_k)) \in T_{i_1,\ldots,i_k}$.

PROOF. Let $c = |\{j : i_j \in K\}|$. $P(c, s)$ holds for all sufficiently large s because M^A infers all functions $\sigma_c 0^\infty$. After discovering it, $(\chi_K(i_1), \ldots, \chi_K(i_k)) = (\chi_{K_s}(i_1), \ldots, \chi_{K_s}(i_k))$ is enumerated into $K_{i_1,\ldots,i_k}$. ∎

Proposition 19 $|T_{i_1,\ldots,i_k}| = k + 1$ *for some* $i_1, \ldots, i_k$.

PROOF. If $(\chi_K(i_1), \ldots, \chi_K(i_k)) \in T_{i_1,\ldots,i_k}$ and $|T_{i_1,\ldots,i_k}| \leq k$ for all $i_1, \ldots, i_k$, then K is Turing-reducible to A (Fact 6 in [34]). ∎

Hence, there exist $s_1 < \ldots < s_{k+1}$ such that $P(l - 1, s_l)$ for $l = 1, \ldots, k + 1$. The label $\nu_1(\tau)$ of $\tau = (a_1, \ldots, a_{l-1})$ is the probability that:

1. M does not output a program while reading $\sigma_{l-2} 0^{s_{l-2}}$, and
2. M outputs a program computing a function with the initial segment σ_{l-1} while reading $\sigma_{l-1} 0^{s_{l-1}}$.

For $\tau = \epsilon$, there is no segment σ_{-1} and $\nu_1(\epsilon)$ is just the probability that M outputs a program computing a function with the initial segment σ_0 while reading $\sigma_0 0^{s_0}$.

The label $\nu_2(\tau)$ is 0 for $\tau = \epsilon$ and the probability that M outputs a program computing a function with the initial segment σ_{l-1} while reading $\sigma_{l-2} 0^{s_{l-2}}$ for $\tau = (a_1, \ldots, a_{l-1})$.

Next, we verify that all conditions of Definition 32 are satisfied. Property 1 follows from the definitions of $\nu_1(\epsilon)$, $\nu_2(\epsilon)$ and $P(0, s)$.

For property 2, notice that $\nu_1(t) + \nu_2(t)$ is the total probability that M outputs a function consistent with σ_{l-1} while reading $\sigma_{l-1} 0^{s_{l-1}}$. $\nu_1(t_i)$ are the probabilities that

a particular continuation of σ_{l-1} is an initial segment of the function. These events are mutually exclusive. Hence, $\sum_{i=1}^{s} \nu_1(t_i) \leq \nu_1(t) + \nu_2(t)$. $\nu_1(t_1) + \nu_2(t_1) \geq p - \epsilon$ is true because M outputs a program consistent with $\sigma_l 0^{s_l}$ with a probability at least $p - \epsilon$ (cf. definition of $P(c, s)$).

Property 3 is true because the sum of all ν_1-labels on any branch is equal to the probability that M^A outputs a conjecture while reading $\sigma_k 0^{s_k}$ and, hence, is at most 1. ∎

If there is no oracle A, we get

Corollary 5 *If $S_T \in \langle p \rangle$**PFin**, then T has a $(p - \epsilon, 1)$ labeling for any $\epsilon > 0$.*

Corollary 6 *For a tree T, $S_T \in \langle p_T \rangle$**PFin** and $S_T \notin \langle p_T + \epsilon \rangle$**PFin** for any $\epsilon > 0$.*

PROOF. Corollary 4 and Lemma 31 imply that $S_T \in [p_T n, n]$**PFin** for appropriate n. A $[p_T n, n]$**PFin** team can be simulated by a $\langle p_T \rangle$**PFin** probabilistic machine that chooses one of n machines in the team equiprobably.

If $S_T \in \langle p_T + \epsilon \rangle$**PFin**, then, there is a $(p_T + \epsilon/2, 1)$ labeling of T (Corollary 5). This is impossible because p_T is the largest number such that there is a $(p_T, 1)$ labeling of T. ∎

Lemma 33 *$A = \{p_T : T$ is a tree $\}$.*

PROOF. By induction. If $p \in A$ follows from $p_1, \ldots, p_s \in A$ by rule 2 and $p_{T_i} = p_i$ for trees T_i, we construct a tree T consisting of the root, $T_1, \ldots, T_s$ and make the roots of $T_1, T_2, \ldots, T_s$ children of T's root. Then, $p_T = p$ (cf. Lemma 28). Hence, there is a tree T with $p_T = p$ for any $p \in A$.

Similarly, we can show that $p_T \in A$ for any tree T. ∎

Corollary 7 *$A \subseteq \mathbb{Q}$.*

PROOF. Follows from Lemmas 29 and 33. ∎

Theorem 18 *If $p, q \in A$ and $p \neq q$, then $\langle p \rangle$**PFin** $\neq \langle q \rangle$**PFin**.*

PROOF. Follows from Corollary 6 and Lemma 33. ∎

6.7 Well-ordering and system of notations

It remains to prove that, for any probability p $\mathbf{PFin}\langle p\rangle$-type learning is equivalent to $PFIN$-type learning with some probability belonging to A. Our diagonalization technique was similar to [34]. The simulation part is more complicated. Simulation techniques in [34] rely on fact that each team issues finitely many conjectures and, hence, there are finitely many possible behaviours of these conjectures. A probabilistic machine can issue infinitely many conjectures and these conjectures have infinitely many possible behaviours. This makes simulation far more complicated.

We need an algorithmic structure for manipulating an infinite number of possibilities. We establish it by proving that A is well-ordered and has a system of notations.

Theorem 19 *The set A is well-ordered in decreasing ordering and has a system of notations.*

PROOF. We construct a system of notations for the set A inductively. First, we construct a system of notations for $A \cap [\frac{1}{2}, 1]$. Then we extend it, obtaining system of notations for $A \cap [\frac{1}{3}, 1]$, $A \cap [\frac{1}{4}, 1]$ and so on.

Freivalds[22] proved

$$A \cap \left[\frac{1}{2}, 1\right] = \left\{\frac{1}{2}\right\} \cup \left\{\frac{n}{2n-1} | n \in \mathbb{N} \& n \geq 1\right\}.$$

A system of notations for $A \cap [\frac{1}{2}, 1]$ can be constructed easily from this description. Below, we show how to construct a system of notations for $A \cap [\frac{1}{n+1}, 1]$ using a system of notations for $A \cap [\frac{1}{n}, 1]$.

An outline of our construction is as follows:

1. Split the segment $[\frac{1}{n+1}, \frac{1}{n}]$ into smaller segments $[r_{i+1}, r_i]$ so that, if $p \in [r_{i+1}, r_i]$ and $p \in A$ follows from the rule 2, then $p_1 \geq r_i, \ldots, p_s \geq r_i$. (This property allows us to obtain a system of notations for $A \cap [r_{i+1}, r_i]$ from a given system of notations for $A \cap [r_i, 1]$ without using any knowledge about $A \cap [r_{i+1}, r_i]$.) We give the splitting and prove its properties in subsection 6.7.1.

2. Using transfinite induction over the segments $[r_{i+1}, r_i]$, extend the system of notations for $A \cap [\frac{1}{n}, 1]$ to larger and larger segments $A \cap [r_{i+1}, 1]$, finally obtaining a system of notations for $A \cap [\frac{1}{n+1}, 1]$. This part is described in subsections 6.7.2, 6.7.3, 6.7.4 and 6.7.5.

6.7.1 Splitting the segment $[\frac{1}{n+1}, \frac{1}{n}]$

The splitting consists of two steps.

1. First, we take $\frac{p}{1+p}$ for $p \in A \cap [\frac{1}{n}, \frac{1}{n-1}]$. These points split $[\frac{1}{n+1}, \frac{1}{n}]$ into segments $[\frac{p}{1+p}, \frac{r}{1+r}]$.

2. Each segment $[\frac{p}{1+p}, \frac{r}{1+r}]$ is split further by the sequence

$$r_0 = \frac{r}{1+r}, r_{i+1} = \frac{2}{1 + \frac{1}{p} + \frac{1}{r_i}}.$$

   $r_0, r_1, r_2, \ldots$ is a monotonously decreasing sequence converging to $\frac{p}{1+p}$. It splits $[\frac{p}{1+p}, \frac{r}{1+r}]$ into segments $[r_1, r_0]$, $[r_2, r_1]$, ...

A_n denotes the set consisting of all $\frac{p}{1+p}$ and $r_0, r_1, \ldots$ for all segments $[\frac{p}{1+p}, \frac{r}{1+r}]$. Next, we prove several properties of the segments $[r_{i+1}, r_i]$ that will be used further.

Lemma 34 *Let $[\frac{p}{1+p}, \frac{r}{1+r}]$ be a segment obtained in the first step of the splitting. If $x \in A$ follows from $p_1, \ldots, p_s \in A$ by the rule 2 and $x \in [\frac{p}{1+p}, \frac{r}{1+r}]$ then*

$$p_1 \le p, p_2 \le p, \ldots, p_s \le p.$$

PROOF. We have

$$p_j = \frac{x}{1 - x + q_j} < \frac{x}{1 - x + 0} = \frac{x}{1-x},$$

$$p_j(1-x) \le x,$$

$$p_j \le x(1 + p_j),$$

$$\frac{p_j}{1+p_j} \le x.$$

By definition of p, $\frac{p_j}{1+p_j} \le \frac{p}{1+p}$ and $p_j \le p$. ∎

Lemma 35 *Let $x \in A \cap [r_{i+1}, r_i]$. If $x \in A$ follows from $p_1, \ldots, p_s \in A$ by the rule 2, then*

$$p_1 \ge r_i, p_2 \ge r_i, \ldots, p_s \ge r_i.$$

PROOF. We prove $p_1 \ge r_i$ only. ($p_2 \ge r_i, \ldots$ are proved similarly.)

Assume that $[r_{i+1}, r_i]$ was obtained by splitting $[\frac{p}{1+p}, \frac{r}{1+r}]$. Then, $p_1 \le p, p_2 \le p, \ldots, p_s \le p$ (Lemma 34).

From

$$\frac{x}{1-x+q_j} = p_j$$

it follows that

$$q_j = \frac{x}{p_j} - 1 + x.$$

We have $p_2 \leq p$. Hence,

$$q_2 \geq \frac{x}{p} - 1 + x,$$

$$q_1 \leq x - q_2 \leq 1 - \frac{x}{p},$$

$$p_1 = \frac{x}{1-x+q_1} \geq \frac{x}{2-x-\frac{x}{p}} = \frac{1}{\frac{2}{x} - 1 - \frac{1}{p}}.$$

From $x \in [r_{i+1}, r_i]$ we have that $x \geq r_{i+1}$ and

$$p_1 \geq \frac{1}{\frac{2}{x} - 1 - \frac{1}{p}} \geq \frac{1}{\frac{2}{r_{i+1}} - 1 - \frac{1}{p}} \geq \frac{1}{(1 + \frac{1}{r_i} + \frac{1}{p}) - 1 - \frac{1}{p}} = r_i.$$

∎

We have proved that all $x \in A \cap [r_{i+1}, r_i]$ are generated by applications of the rule 2 to $p_1, \ldots, p_s \in A \cap [r_i, 1]$. The next lemma bounds the number s.

Lemma 36 *Let $x \in A \cap [r_{i+1}, r_i]$. If $x \in A$ follows from $p_1, \ldots, p_s \in A$ by the rule 2, then*

$$s \leq \frac{x}{\frac{x}{p} + x - 1}.$$

PROOF. From Lemma 34 we have

$$q_j = \frac{x}{p_j} + x - 1 \geq \frac{x}{p} + x - 1.$$

Hence,

$$x = \sum_{j=1}^{s} q_j \geq s\left(\frac{x}{p} + x - 1\right).$$

$$s \leq \frac{x}{\frac{x}{p} + x - 1}.$$

∎

6.7.2 Well-ordering

Lemma 37 *A_n is well-ordered.*

PROOF. $A \cap [\frac{1}{n}, 1]$ is well-ordered by inductive assumption. Hence, $A \cap [\frac{1}{n}, \frac{1}{n-1}]$ is well-ordered, too. The set $\{\frac{p}{1+p} | p \in A \cap [\frac{1}{n}, \frac{1}{n-1}]\}$ is order-isomorphic to $A \cap [\frac{1}{n}, \frac{1}{n-1}]$. Hence, it is well-ordered and the set of segments $[\frac{p}{1+p}, \frac{q}{1+q}]$ into which it splits $[\frac{1}{n+1}, \frac{1}{n}]$ is well-ordered, too.

A_n is obtained by replacing each segment $[\frac{p}{1+p}, \frac{r}{1+r}]$ with the sequence $r_0, r_1, \ldots$. Each sequence is well-ordered. Hence, the entire set A_n is well-ordered. ∎

Hence, we can use transfinite induction over this set.

Lemma 38 *$A \cap [\frac{1}{n+1}, \frac{1}{n}]$ is well-ordered in decreasing ordering.*

PROOF. By transfinite induction over A_n.
Base case. The set $A \cap [\frac{1}{n}, 1]$ is well-ordered.
Inductive case. Let $x \in A_n$. We assume that $A \cap [x', 1]$ is well-ordered for $x' > x$ and prove that $A \cap [x, 1]$ is well-ordered, too. There are three cases:

1. $x = \frac{p}{1+p}$ for $p \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$ and p is a limit element.

   Let p be the limit of $p_1, p_2, \ldots$. Then, $\frac{p}{1+p}$ is the limit of $\frac{p_1}{1+p_1}, \frac{p_2}{1+p_2}, \ldots$ because the function $\frac{x}{1+x}$ is continous. By inductive assumption, each $[\frac{p_i}{1+p_i}, 1]$ is well-ordered. Hence, their union $[\frac{p}{1+p}, 1]$ is well-ordered.

2. $x = \frac{p}{1+p}$ for $p \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$ and p is not a limit element.

   We take the segment $[\frac{p}{1+p}, \frac{r}{1+r}]$ obtained in the first step of the splitting and the corresponding sequence $r_0, r_1, \ldots$. $\frac{p}{1+p}$ is the limit of $r_0, r_1, \ldots$. $[\frac{p}{1+p}, 1]$ is well-ordered because each $[r_i, 1]$ is well-ordered.

3. $x \neq \frac{p}{1+p}$ for any $p \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$. Then, $x \neq r_0$ because $r_0 = \frac{r}{1+r}$ for $r \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$. Hence, $x = r_{i+1}$ for some $i \geq 0$.

   $A \cap [r_i, 1]$ is well-ordered because $r_{i+1} < r_i$. Hence, it is enough to prove that $A \cap [r_{i+1}, r_i]$ is well-ordered.

   By way of contradiction, assume that $A \cap [r_{i+1}, r_i]$ contains an infinite increasing sequence $x_1, x_2, \ldots$. We use

   Lemma 39 *Let $x_1 \in A \cap [r_{i+1}, r_i]$, $x_2 \in A \cap [r_{i+1}, r_i]$, $\ldots$. There is an $s \in \mathbb{N}$ and sequences $x'_1, x'_2, \ldots$ and $p_{j,1}, p_{j,2}, \ldots$ for $j \in \{1, \ldots, s\}$ such that*

(a) $x'_1, x'_2, \ldots$ *is a subsequence of* $x_1, x_2, \ldots$,

(b) $x'_k \in A$ *follows from* $p_{1,k}, \ldots, p_{s,k}$ *and the rule 2, and*

(c) $p_{j,1} = p_{j,2} = \ldots$ *or* $p_{j,1} > p_{j,2} > \ldots$ *for all* $j \in \{1, \ldots, s\}$.

PROOF. Denote

$$s_1 = \left\lceil \frac{x}{\frac{x}{r_{i+1}} + x - 1} \right\rceil .$$

Consider the applications of the rule 2 that prove $r_1 \in A, r_2 \in A, \ldots$. By Lemma 36,

$$s \leq \frac{x}{\frac{x}{p} + x - 1} \leq \frac{x}{\frac{x}{r_{i+1}} + x - 1} \leq s_1$$

in each of these applications. Hence, there exists an $s_0 \in \{1, \ldots, s_1\}$ such that infinitely many of $x_1, \ldots$ are generated by applications of the rule 2 with $s = s_0$. We denote this subsequence $x_1^{(0)}, x_2^{(0)}, \ldots$.

Next, we select $x_1^{(1)}$, $x_2^{(1)}$, ..., a subsequence of $x_1^{(0)}$, $x_2^{(0)}$, ... Then, we select $x_1^{(2)}$, $x_2^{(2)}$, ..., a subsequence of $x_1^{(1)}$, $x_2^{(1)}$, ... We continue so until we obtain $x_1^{(s_0)}$, $x_2^{(s_0)}$, ...

The subsequence $x_1^{(k)}, x_2^{(k)}, \ldots$ is generated from $x_1^{(k-1)}, x_2^{(k-1)}, \ldots$ as follows: Let $p_{1,j}^{(k-1)}, \ldots, p_{s_0,j}^{(k-1)}$ be the values of $p_1, \ldots, p_{s_0}$ in the application of the rule 2 that proves $x_j^{(k-1)} \in A$. We use a following modification of well-known theorem.

Theorem 20 *Let* $y_1, y_2, \ldots$ *be a sequence of real numbers. Then* y_1, y_2, ... *contains*

- *a subsequence* $y_{n_1}, y_{n_2}, \ldots$ *such that* $y_{n_1} = y_{n_2} = \ldots$, *or*
- *an infinite monotonously increasing subsequence, or*
- *an infinite monotonously decreasing subsequence.*

The sequence $p_{k,1}^{(k-1)}, p_{k,2}^{(k-1)}, \ldots$ does not contain an infinite monotonously increasing subsequence because all elements of this sequence belong to $A \cap [r_i, 1]$ and $A \cap [r_i, 1]$ is well-ordered in decreasing ordering. Hence, this sequence contains an infinite subsequence consisting of equal elements or an infinite monotonously decreasing subsequence.

Let this subsequence be $p_{k,n_1}^{(k-1)}$, $p_{k,n_2}^{(k-1)}$, ... We choose $r_{n_1}^{(k-1)}, r_{n_2}^{(k-1)}, \ldots$ as the sequence $x_1^{(k)}, x_2^{(k)}, \ldots$.

$x_1^{(s_0)}, x_2^{(s_0)}, \ldots$ is the needed sequence $x'_1, x'_2, \ldots$. We have

$$p_{1,k} = p_{2,k} = \ldots \text{ or } p_{1,k} > p_{2,k} > \ldots$$

because such property holds for the sequence $x_1^{(k)}, x_2^{(k)}, \ldots$ and $x_1^{(s_0)}, x_2^{(s_0)}, \ldots$ is a subsequence of $x_1^{(k)}, x_2^{(k)}, \ldots$. ∎

We have

$$p_{1,1} \geq p_{2,1} \geq \ldots$$

$$\ldots$$

$$p_{1,s} \geq p_{2,s} \geq \ldots.$$

By Lemma 24,

$$x'_1 \geq x'_2 \geq x'_3 \ldots.$$

Hence, $x_1, x_2, \ldots$ contains an infinite monotonously decreasing subsequence. A contradiction with the assumption that $x_1, x_2, \ldots$ is monotonously increasing. ∎

Next, we construct a system of notations S for $A \cap [\frac{1}{n+1}, \frac{1}{n}]$. We start with technical results necessary for our construction. In section 6.7.3, we show how to distinguish limit elements from successor elements. In section 6.7.4, we define (x, d)-minimal sets and show that such sets can be computed algorithmically. Finally, in section 6.7.5, we use these results to construct a system of notations.

6.7.3 Distinguishing elements of different types

The maximal element of the set A is 1. It does not belong to $A \cap [r_{i+1}, r_i]$. Hence, $A \cap [r_{i+1}, r_i]$ does not contain the maximal element and, constructing a system of notations, we should distinguish numbers p of three types:

1. $p \in A \cap [r_{i+1}, r_i]$ and p is a successor. Then $k_S(p) = 1$.
2. $p \in A \cap [r_{i+1}, r_i]$ and p is a limit element. Then $k_S(p) = 2$.
3. $p \notin A \cap [r_{i+1}, r_i]$. Then $k_S(p) = 3$.

Two lemmas below shows how to distinguish between limit and successor elements.

Lemma 40 *Let $x \in A \cap [r_{i+1}, r_i]$. Then x is a limit element if and only if it can be generated by rule 2 so that at least one of $p_1, \ldots, p_s$ is limit element.*

PROOF.

"if" part. Assume that p_j is a limit element. Let $p_{j,1}, p_{j,2}, \ldots$ be a monotonously decreasing sequence converging to p_j and x_k be the number generated by the application of the rule 2 to p_1, ..., p_{j-1}, $p_{j,k}$, p_{j+1}, ..., p_s. Then, $x_1, x_2, \ldots$ is a monotonously decreasing sequence converging to x. Hence, x is a limit element.

"only if" part. Let x be a limit element and $x_1, x_2, \ldots$ be a monotonously decreasing sequence converging to x. We apply Lemma 39 to $x_1, x_2, \ldots$ and obtain a subsequence $x'_1, x'_2, \ldots$.

We consider the sequences $p_{j,1}, p_{j,2}, \ldots$. Let

$$p'_j = \lim_{k \to \infty} p_{j,k}.$$

x can be generated from $p'_1, p'_2, \ldots, p'_s$ by application of rule 2 (cf. Lemma 25). We have

$$p_{j,1} = p_{j,2} = \ldots \text{ or } p_{j,1} > p_{j,2} > \ldots.$$

for any $j \in \{1, \ldots, m\}$. If $p_{j,1} = p_{j,2} = \ldots$ for all j, then, $x'_1 = x'_2 = \ldots$. A contradiction with the assumption that $x_1, x_2, \ldots$ is monotonously decreasing.

Hence,

$$p_{j,1} > p_{j,2} > \ldots$$

for at least one j and $p'_j = \lim_{k \to \infty} p_{j,k}$ is a limit element. ∎

Lemma 41 *Let $x \in A_n$. Then x is a limit element.*

PROOF. We have three cases.

1. $x = \frac{p}{1+p}$ for $p \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$.

   Let p be the limit of $p_1, p_2, \ldots$. Then, $\frac{p}{1+p}$ is the limit of $\frac{p_1}{1+p_1}, \frac{p_2}{1+p_2}, \ldots$ because the function $\frac{x}{1+x}$ is continous.

2. $x = \frac{p}{1+p}$ for $p \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$ and p is not a limit element.

   We take the segment $[\frac{p}{1+p}, \frac{r}{1+r}]$ obtained in the first step of the splitting and the corresponding sequence $r_0, r_1, \ldots$. $\frac{p}{1+p}$ is the limit of $r_0, r_1, \ldots$.

3. $x \neq \frac{p}{1+p}$ for any $p \in A \cap [\frac{1}{n+1}, \frac{1}{n}]$.

Then, $x = r_i$. We prove the lemma by induction over i.

Base Case. If $i = 0$, then $r_i = \frac{r}{1+r}$ and we already know that $\frac{r}{1+r}$ is a limit element.

Inductive Case. Lemma 22 and the definition of r_{i+1} imply that $r_{i+1} \in A$ follows from $r_i \in A$ and $p \in A$ by the rule 2. If r_i is a limit element, then, by Lemma 40, r_{i+1} is a limit element, too. ∎

6.7.4 (x, d)-minimal sets

In the algorithms of subsection 6.7.5, we will often need to compute the largest element of $A \cap [r_{i+1}, r_i]$ which is less than some given x. This will be done by checking $p_1 \in A \cap [r_i, 1]$, $p_2 \in A \cap [r_i, 1]$, ..., $p_s \in A \cap [r_i, 1]$ that can generate $x \in A$ by rule 2. There are infinitely many possible combinations of $p_1, \ldots, p_s$. Hence, we need

- to prove that it is enough to check finitely many combinations $p_1 \in A \cap [r_i, 1]$, $p_2 \in A \cap [r_i, 1]$, ..., $p_s \in A \cap [r_i, 1]$, and
- to construct an algorithm finding the list of combinations $p_1 \in A \cap [r_i, 1]$, $p_2 \in A \cap [r_i, 1]$, ..., $p_s \in A \cap [r_i, 1]$ which must be checked when the functions k_S, p_S, q_S are computed.

We do it below. First, we give formal definitions.

Definition 34 A tuple $\langle p_1, \ldots, p_s \rangle$ is said to be (x, d)-allowed if $p_1 \in A \cap [r_i, 1]$, ..., $p_s \in A \cap [r_i, 1]$ and $\sum_{j=1}^{s} (\frac{x}{p_j} + x - 1) \leq d$.

Definition 35 A tuple $\langle p_1, \ldots, p_s \rangle$ is said to be less than or equal to $\langle p'_1, \ldots, p'_s \rangle$ if $p_1 \leq p'_1, \ldots, p_s \leq p'_s$.

Definition 36 A set of tuples P is said to be (x, d)-minimal if,

1. It contains only (x, d)-allowed tuples;
2. For each (x, d)-allowed tuple $\langle p_1, \ldots, p_s \rangle$ there is a tuple belonging to P which is less than or equal to $\langle p_1, \ldots, p_s \rangle$

Next three lemmas show why (x,d)-allowed tuples and (x,d)-minimal sets are important for our construction.

Lemma 42 *$\langle p_1,\ldots,p_s\rangle$ is (x,x)-allowed if and only if the application of the rule 2 to $p_1,\ldots,p_s$ generates a number p satisfying $p \geq x$.*

PROOF. Let $d = \sum_{j=1}^{s}(x+\frac{x}{p_j}-1)$. $\langle p_1,\ldots,p_s\rangle$ is (x,x)-allowed if and only if $d \leq x$. Hence, it is enough to prove that $d \leq x$ if and only if $x \leq p$.

$$d = \sum_{j=1}^{s}\left(x+\frac{x}{p_j}-1\right) = \sum_{j=1}^{s}\left(x+\frac{x}{p_j}-1\right) - p + p$$

$$= \sum_{j=1}^{s}\left(x+\frac{x}{p_j}-1\right) - \sum_{j=1}^{s}\left(p+\frac{p}{p_j}-1\right) + p = \left(\sum_{j=1}^{s}\left(1+\frac{1}{p_j}\right)\right)(x-p)+p.$$

We have

$$\sum_{j=1}^{s}\left(1+\frac{1}{p_j}\right) \geq 1+\frac{1}{p_j} > 1.$$

Hence, if $x > p$,then $(x-p) > 0$ and $d > (x-p)+p = x$. If $x \leq p$, then $(x-p) \leq 0$ and $d \leq (x-p)+p = x$. ∎

Lemma 43 *Let P be a (x,x)-minimal set. Then, for any $p_1,\ldots,p_s$ that generates $p \geq x$ by an application of the rule 2, there exists a tuple $\langle p'_1,\ldots,p'_s\rangle \in S$ such that $p'_1 \leq p_1, \ldots, p'_s \leq p_s$.*

PROOF. $\langle p_1,\ldots,p_s\rangle$ is (x,x)-allowed (cf. Lemma 42). By the definition of (x,x)-minimal set, P contains a tuple $\langle p'_1,\ldots,p'_s\rangle$ such that $p'_1 \leq p_1, \ldots, p'_s \leq p_s$. ∎

Lemma 44 *Let P be a (x,x)-minimal set. $p_1 \in A \cap [r_i,1],\ldots,p_s \in A \cap [r_i,1]$. If $x \in A$ follows from $p_1,\ldots,p_s \in A$ and the rule 2, then $\langle p_1,\ldots,p_s\rangle \in P$.*

PROOF. By Lemma 42, $\langle p_1,\ldots,p_s\rangle$ is (x,x)-allowed. Hence, by Lemma 43, there exists (x,x)-allowed $\langle p'_1,\ldots,p'_s\rangle \in P$ such that $p'_1 \leq p_1,\ldots,p'_s \leq p_s$.

Let x' be the number generated by an application of the rule 2 to $p'_1 \in A, \ldots, p'_s \in A$. If $p'_j < p_j$ for some i, then $x' < x$ (Lemma 24) and $\langle p'_1,\ldots,p'_s\rangle$ is not (x,x)-allowed (Lemma 42).

However, (x,x)-allowed set contains only (x,x)-allowed tuples. Hence, $p_1 = p'_1,\ldots,p_s = p'_s$, i.e. $\langle p_1,\ldots,p_s\rangle \in P$. ∎

Next lemma shows that (x,d)-minimal sets can be computed algorithmically. Its proof also shows that a finite (x,d)-minimal set always exists.

Lemma 45 *Assume that a system of notations for $A \cap [r_i, 1]$ is given. There is an algorithm $xdminimal(x, d)$ which receives $x \in A \cap [r_{i+1}, r_i]$ and $d \in [0, x]$ and returns a (x, d)-minimal set.*

PROOF. We use an auxilary procedure $findsmallest(P, x, d)$. It receives numbers x, d and an (x, d)-minimal set P and returns the smallest d' such that $d' > d$ and $\sum_{i=1}^{s}(\frac{x}{p_i} + x - 1) = d'$ for some $p_1, \ldots, p_s \in A$.

Both *findsmallest* and *xdminimal* use a constant p_0. p_0 is defined as the smallest number in $A \cap [r_i, 1]$ such that $x + \frac{x}{p_0} - 1 > 0$. Equivalently, p_0 is the number in $A \cap [r_i, 1]$ with the smallest $x + \frac{x}{p_0} - 1$ such that $x + \frac{x}{p_0} - 1 > 0$. Δ denotes $\frac{x}{p_0} + x - 1$.

Algorithm $findsmallest(P, x, d)$:

1. Let $d' = 0$;
2. For each $\langle p_1, \ldots, p_s \rangle \in P$ do:
   (a) For each $j \in \{1, \ldots, s\}$:
      i. $p'_j = \max\{p | p \in A \cap [r_i, 1] \text{ and } p < p_j\}$;
      ii. $d_1 = \sum_{k=1}^{j-1}(\frac{x}{p_k} + x - 1) + (\frac{x}{p'_j} + x - 1) + \sum_{k=j+1}^{s}(\frac{x}{p_k} + x - 1)$. If $d_1 > d$, then $d' = \min(d', d_1)$.
   (b) $d_2 = \sum_{j=1}^{s}(\frac{x}{p_j} + x - 1) + (\frac{x}{p_0} + x - 1)$; If $d_2 > d$, then $d' = \min(d', d_2)$.
3. Return d' as the result;

Algorithm $xdminimal(x, d)$

1. Let $P = \emptyset$;
2. If $d < \Delta$, return the empty set as the result;
3. Let $y = p_0$.
4. while $(\frac{x}{y} + x - 1 > 0)$ do:
   (a) $d' = d - (\frac{x}{y} + x - 1)$;
   (b) $P_1 = xdminimal(x, d')$;
   (c) If $P_1 = \emptyset$, add $\langle y \rangle$ to P. Otherwise, for each $\langle p_1, \ldots, p_s \rangle \in P_1$, add $\langle y, p_1, \ldots, p_s \rangle$ to P;
   (d) Replace y by a greater element of $A \cap [r_i, 1]$:

i. If y is a successor element, replace y by $p_{S_1}(y)$;

ii. If y is a limit element, replace y by y' where y' is the smallest element of $A \cap [r_i, 1]$ such that

$$\frac{x}{y'} + x - 1 \leq d - findsmallest(P_1, x, d').$$

5. Return P.

Proof of correctness for $xdminimal(x, d)$. We prove the correctness by induction over $\lfloor \frac{d}{\Delta} \rfloor$.

Base Case. $d \in [0, \Delta[$.

Then, $\frac{x}{y} + x - 1 \geq \Delta$ for any y. Hence, $\sum_{j=1}^{s} (\frac{x}{p_j} + x - 1) \geq \Delta$ for any $\langle p_1, \ldots, p_s \rangle$ and there are no (x, d)-allowed tuples. In this case, the algorithm returns the empty set. Hence, it works correctly.

Inductive Case. We assume that the lemma holds for $d \in [0, k\Delta[$ and prove it for $d \in [k\Delta, (k+1)\Delta[$. We use

Lemma 46 *If* $xdminimal(x, d)$ *calls* $xdminimal(x, d')$, *then* $d' \leq d - \Delta$

PROOF. From the description of $xdminimal$ we have $d' = d - (\frac{x}{y} + x - 1)$. By definition of p_0 and Δ, $\frac{x}{y} + x - 1 \geq \Delta$ and $d' \leq d - \Delta$. ∎

Hence, $xdminimal(x, d)$ calls only $xdminimal(x, d')$ with $d' < (k+1)\Delta - \Delta = k\Delta$. The correctness of such $xdminimal(x, d')$ follows from the inductive assumption.

First, we prove that the computation of $xdminimal(x, d)$ always terminates. Each $xdminimal(x, d')$ called by $xdminimal(x, d)$ terminates because $xdminimal(x, d')$ is correct. Hence, each while loop terminates and, if $xdminimal(x, d)$ does not stop then while loop is executed infinitely many times.

Let y_j be the value of y during the j^{th}-th execution of while loop. y is increased at the end of each while loop. Hence, $y_1 < y_2 < \ldots$.

$y_1 \in A \cap [r_i, 1], y_2 \in A \cap [r_i, 1], \ldots$. If while loop is executed infinitely many times, then $y_1, y_2 \ldots$ is an infinite monotonously increasing sequence. However, $A \cap [r_i, 1]$ does not contain such sequences because it is well-ordered.

Hence, while loop is executed finitely many times and $xdminimal(x, d)$ terminates. Let $P = xdminimal(x, d)$. Next, we prove that P is a (x, d)-minimal set.

Assume, by way of contradiction, that it is not. Then, there exists an (x, d)-allowed tuple $\langle p_1, \ldots, p_s \rangle$ such that P does not contain any tuple that is less than or equal to $\langle p_1, \ldots, p_s \rangle$.

We assume that $\langle p'_1, p_2, \ldots, p_s \rangle$ is not (x,d)-allowed for any $p'_1 \in A \cap [r_i, 1]$ satisfying $p'_1 < p_1$. (Otherwise, we can replace p_1 by the smallest $p'_1 \in A \cap [r_i, 1]$ such that $\langle p'_1, p_2, \ldots, p_s \rangle$ is (x,d)-allowed.)

Consider two cases:

1. In $xdminimal(x,d)$, while loop is executed with $y = p_1$.

   Denote $d' = d - (\frac{x}{p_1} + x - 1)$. The tuple $\langle p_2, \ldots, p_s \rangle$ is (x,d')-allowed.

   $xdminimal(x,d)$ calls $xdminimal(x,d')$. $xdminimal(x,d')$ works correctly, i.e. returns an (x,d')-minimal set P_1. Hence, P_1 contains a tuple $\langle p'_2, \ldots, p'_s \rangle$ that is less than or equal to $\langle p_2, \ldots, p_s \rangle$.

   $xdminimal(x,d)$ adds $\langle p_1, p'_2, \ldots, p'_s \rangle$ to P because $\langle p'_2, \ldots, p'_s \rangle$ belongs to the set returned by $xdminimal(x,d')$. Hence, P contains the tuple $\langle p_1, p'_2, \ldots, p'_s \rangle$ that is less than or equal to $\langle p_1, p_2, \ldots, p_s \rangle$. A contradiction.

2. While loop is not executed with $y = p_1$.

   Let y_1 be the greatest number such that $y_1 < p_1$ and while loop is executed with $y = y_1$. Let y_2 be the number by which y_2 is replaced in the end of while loop.

   $y_1 < y_2$ because y is always replaced by a greater number. By definition of y_1, $y_2 > p_1$. (Otherwise y_2 would have been instead of y_1.)

   (a) y_1 is a successor element.

   Then, y_1, y_2, p_1 all belong to A and $y_1 < p_1 < y_2$. When $xdminimal(x,d)$ replaces y_1 by a greater element of A, it chooses the smallest element of A that is greater than y_1. It can be p_1 or some number between y_1 and p_1 but not y_2. A contradiction.

   (b) r_0 is a limit element.

   We assumed that $\langle p'_1, p_2, \ldots, p_s \rangle$ is not (x,d)-allowed for any $p'_1 \in A \cap [r_i, 1]$ satisfying $p'_1 < p_1$. Hence, $\langle y_1, p_2, \ldots, p_s \rangle$ is not (x,d)-allowed i.e.

$$\left(\frac{x}{y_1} + x - 1\right) + \sum_{j=2}^{s} \left(\frac{x}{p_j} + x - 1\right) > d.$$

$$\sum_{j=2}^{s} \left(\frac{x}{p_j} + x - 1\right) > d - \left(\frac{x}{y_1} + x - 1\right) = d'$$

Hence,

$$\sum_{j=2}^{s}\left(\frac{x}{p_j}+x-1\right)\geq findsmallest(x,d',P_1)$$

where P_1 is the (x,d')-minimal set obtained by $xdminimal(x,d')$. However,

$$\sum_{j=1}^{s}\left(\frac{x}{p_j}+x-1\right)\leq d$$

because $\langle p_1,p_2,\ldots,p_s\rangle$ is (x,d)-allowed. Hence,

$$\frac{x}{p_1}+x-1\leq d-\sum_{j=2}^{s}\left(\frac{x}{p_j}+x-1\right)\leq d-findsmallest(x,d',P_1).$$

By the definition, y_2 is the smallest number such that

$$\frac{x}{y_2}+x-1\leq d-findsmallest(x,d',P_1).$$

This implies $y_2\leq p_1$. A contradiction with $y_2>p_1$. ∎

6.7.5 System of notations

We extend the system of notations S from $A\cap[\frac{1}{n},1]$ to $A\cap[\frac{1}{n+1},1]$. Below, we give the algorithms computing $k_S(x)$, $p_S(x)$ and $q_S(x)$ for $x\in[\frac{1}{n+1},\frac{1}{n}]$. These algorithms use the procedure $xdminimal(x,d)$ defined in the previous subsection. They also use the system S for $A\cap[\frac{1}{n},1]$.

Function $k_S(x)$.

1. Use the system for $A\cap[\frac{1}{n},\frac{1}{n-1}]$ to find whether $x=\frac{p}{1+p}$ for some $p\in A\cap[\frac{1}{n},\frac{1}{n-1}]$. If yes, then $k_S(x)=2$.

2. Otherwise, find the segments $[\frac{p}{1+p},\frac{r}{1+r}]$ and $[r_{i+1},r_i]$ containing x. If $x=r_{i+1}$ or $x=r_i$, then $k_S(x)=2$.

3. Otherwise, find a (x,x)-minimal set P using $xdminimal(x,x)$.

4. If there exists $\langle p_1,\ldots,p_s\rangle\in P$ such that x is generated by an application of the rule 2 to $p_1,\ldots,p_s$ and at least one of $p_1,\ldots,p_s$ is a limit element, then $k_S(x)=2$.

5. Otherwise, if there exists $\langle p_1,\ldots,p_s\rangle\in P$ such that x is generated by an application of the rule 2 to $p_1,\ldots,p_s$, then $k_S(x)=1$.

6. Otherwise, $k_S(x) = 3$.

Function $p_S(x)$.

1. Find the interval $[r_{i+1}, r_i]$ containing x. Execute $xdminimal(x,x)$ and find a (x,x)-minimal set.

2. Let P_1 be the set consisting of all tuples $\langle p_1, \ldots, p_s \rangle$ such that

   (a) $\langle p_1, \ldots, p_s \rangle \in P$ or

   (b) $\langle p_1, \ldots, p_{j-1}, p'_j, p_{j+1}, \ldots, p_s \rangle \in P$ and $p_j = p_S(p'_j)$ for some $j \in \{1, \ldots, s\}$ or

   (c) $\langle p_1, \ldots, p_{j-1}, p', p_j, \ldots, p_s \rangle \in P$ for some $j \in \{1, \ldots, s\}$ and $p' \in A \cap [r_i, 1]$.

3. For each tuple $\langle p_1, \ldots, p_s \rangle \in P_1$ find the number $p \in A$ generated by an application of the rule 2 to $p_1, \ldots, p_s$.

   $p_S(x)$ is the smallest of those p which are greater than x.

Function $q_S(x)$.

1. If $x = \frac{p}{1+p}$, $p \in A \cap [\frac{1}{n}, \frac{1}{n-1}]$ and p is a limit element, $q_S(x)$ is a program computing $\frac{\varphi_{q_S(p)}(0)}{1+\varphi_{q_S(p)}(0)}$, $\frac{\varphi_{q_S(p)}(1)}{1+\varphi_{q_S(p)}(1)}$, ...

2. If $x = \frac{p}{1+p}$, $p \in A \cap [\frac{1}{n}, \frac{1}{n-1}]$ and p is a successor element, find $r = p_S(p)$. $q_S(x)$ is a program computing the sequence r_0, r_1, ... corresponding to $[\frac{p}{1+p}, \frac{r}{1+r}]$.

3. Otherwise, search the set P returned by $xdminimal(x,x)$ and find $p_1 \in A \cap [r_i, 1]$, ..., $p_s \in A \cap [r_i, 1]$ such that x is generated by an application of the rule 2 to $p_1, \ldots, p_s$ and p_j is a limit element.

   $q_S(x)$ is a program computing the sequence x_1, x_2, ... where x_k is generated by an application of the rule 2 to p_1, ..., p_{j-1}, $\varphi_{q_{S_1}(p_j)}(k)$, p_{j+1}, ..., p_s.

Lemma 47 *S is a system of notations for $A \cap [\frac{1}{n+1}, 1]$.*

PROOF. By transfinite induction over A_n.
Base Case. It is clear that S is a correct system of notations for $A \cap [\frac{1}{n}, 1]$.
Inductive Case. Let $y \in A_n$. We assume that S is correct for all $A \cap [y', 1]$ with $y' > y$ and prove that it is correct for $A \cap [y, 1]$. We consider two cases:

1. $y = \frac{p}{1+p}$ and $p \in A \cap [\frac{1}{n}, \frac{1}{n-1}]$.

   y is a limit of a sequence consisting of elements of A_n (cf. proof of Lemma 41). Hence, if $x > y$, then $x > y'$ where y' is some element of this sequence. The functions $k_S(x)$, $p_S(x)$, $q_S(x)$ are correct because S is correct for $A \cap [y', 1]$ (by inductive assumption). It remains to prove the correctness of $k_S(x)$, $p_S(x)$, $q_S(x)$ for $x = y$.

   $k_S(y) = 2$. This is correct because, by Lemma 41, y is a limit element. The function $p_S(x)$ is defined only for successor elements. Hence, we do not need to check its correctness for the limit element y. The correctness of the sequence computed by $q_S(y)$ is proved in the proof of Lemma 41.

2. $y = r_{i+1}$ for $i \geq 0$. In this case, we assume that S is correct for $A \cap [r_i, 1]$ and prove the correctness for $A \cap [r_{i+1}, r_i]$.

   By Lemma 45, $xdminimal(x, d)$ returns an (x, d)-minimal set if it has access to a system of notations for $A \cap [r_i, 1]$. We know that S is correct for $A \cap [r_i, 1]$. Hence, the set P returned by $xdminimal(x, x)$ is (x, x)-minimal.

   2.1. Proof of correctness for k_S.

   If $x \in A \cap [r_{i+1}, r_i]$, then $x \in A$ follows from $p_1 \in A, \ldots, p_s \in A$ and the rule 2, for some $p_1, \ldots, p_s$. By Lemma 35, $p_1 \in A \cap [r_i, 1], \ldots, p_s \in A \cap [r_i, 1]$. By Lemma 44, $\langle p_1, \ldots, p_s \rangle$ belongs to P.

   Hence, if $x \in A$, the algorithm computing k_S finds $p_1, \ldots, p_s$ such that $p \in A$ follows from $p_1 \in A, \ldots, p_s \in A$ and the rule 2.

   Hence, it distinguishes $x \in A$ and $x \notin A$ correctly. By Lemma 40, it distinguishes limit and successor elements correctly.

   2.2. Proof of correctness for p_S.

   We prove that $p_S(x)$ returns the element of $A \cap [r_{i+1}, r_i]$ immediately preceding x i.e. $(\forall z \in A \cap [r_{i+1}, r_i])(x < z \Rightarrow p_S(x) \leq z)$.

   Let $z \in A \cap [r_{i+1}, r_i]$ and $x < z$. Consider $p_1, \ldots, p_s$ that generate $z \in A$ by rule 2.

   P contains a tuple $\langle p'_1, \ldots, p'_s \rangle$ such that $p'_1 \leq p_1, \ldots, p'_s \leq p_s$ (Lemma 43). An application of the rule 2 to $p'_1, \ldots, p'_s$ generates $p \in A$ with $p \geq x$ (Lemma 42). Consider two cases:

(a) $p > x$.

The algorithm computing p_S adds $\langle p'_1, \ldots, p'_s \rangle$ to the set P_1. Later, it sets $p_S(x)$ equal to a number that is less or equal to p. (It is so because $\langle p'_1, \ldots, p'_s \rangle \in P_1$ and $p'_1, \ldots, p'_s$ generates $p > x$. The algorithm selects $p_S(x)$ as the smallest of all p satisfying these conditions.)

By Lemma 24, $p \leq z$. Hence, $p_S(x) \leq p \leq z$.

(b) $p = x$

If $p_1 = p'_1, \ldots, p_s = p'_s$ then $p = z$. However, $p < z$. Hence, $p_j < p'_j$ for some i. Let $p''_j = p_{S_1}(p'_j)$. We have $p''_j \leq p_j$ because $p_{S_1}(p'_j)$ is the smallest element of A that is greater than p'_j. Let p denote the number generated by the rule 2 from $p'_1, \ldots, p'_{j-1}, p''_j, p'_{j+1}, \ldots, p'_s$.

By Lemma 24, $x < p$. Hence, the algorithm for $p_S(x)$ adds the tuple

$$\langle p'_1, \ldots, p'_{j-1}, p''_j, p'_{j+1}, \ldots, p'_s \rangle$$

to the set P_1 and, then, checking tuples in P_1, sets $p_S(x)$ equal to a number which is greater than or equal to p. This implies $p_S(x) \leq p$. From $p'_1 \leq p_1, \ldots, p'_{j-1} \leq p_{j-1}, p''_j \leq p_j, p'_{j+1} \leq p_{j+1}, \ldots, p'_s \leq p_s$ it follows that $p \leq z$ (Lemma 24). Hence, $p_S(x) \leq p \leq z$.

So, in both cases $p_S(x)$ is less than or equal to any $z \in A$ satisfying $x < z$. On the other hand, $p_S(x) \in A$ and $x < p_S(x)$. (It can be seen from the algorithm computing p_S.)

Hence, $p_S(x)$ is the smallest element of A satisfying $x < p_S(x)$, i.e. the algorithm computes p_S correctly.

2.3. Proof of correctness for q_S.

We already proved that, if there exist $p_1, \ldots, p_s$ such that $x \in A$ follows from $p_1 \in A, \ldots, p_s \in A$, then such combination is found by $xdminimal(x, x)$ (cf. proof of correctness for k_S). If there exists such a combination with one of $p_1, \ldots, p_s$ being limit element, it is found. The algorithm computing q_S generates a program computing required sequence from such combination correctly.

The correctness of S for $A \cap [\frac{1}{n}, 1]$ follows by transfinite induction. ∎

By Lemmas 38 and 47, $A \cap [\frac{1}{n}, 1]$ is well-ordered and has a system of notations for any n. Hence, A is well-ordered and has a system of notations. This completes the proof of Theorem 19. ∎

6.8 Universal simulation

Theorem 21 *For any $p \in A$ there exists k such that* $\mathbf{PFin}\langle x\rangle \subseteq [pk, k]\mathbf{PFin}$ *for all x which are greater than any $p' \in A \cap [0, p[$. There exists an algorithm which receives a probabilistic machine M and a probability x and outputs a team $L_1, \ldots, L_k$ which identifies the same set of functions.*

PROOF. By transfinite induction.

Base Case. For $p > \frac{1}{2}$, the theorem follows from the results of [22].

Inductive Case. We assume that the theorem is true for all $p \in A$ such that $p > p_0$ and prove it for $p = p_0$.

p_0' is the smallest element of A which is greater than or equal to $\frac{p_0}{1-p_0}$. p_0' can be computed effectively when x is known(cf. Section 6.2.2). $\Delta = \frac{x}{p_0'} + x - 1$. P is a (x, x)-minimal set (cf. Section 6.7.4). P can be computed from x, too (Lemma 45).

Next, we give a technical lemma which is very important part of our simulation technique. It expresses relevant results of section 6.7.4 in a form appropriate for the proof of this theorem.

Lemma 48 *Let $p_1, \ldots, p_s$, $q_1, \ldots, q_s$ be such that*

1. $q_1 \geq \Delta, \ldots, q_s \geq \Delta$;
2. $q_1 + \ldots + q_s \leq x$;
3. $p_1 = \frac{x}{1-x+q_1}, \ldots, p_s = \frac{x}{1-x+q_s}$.

There exist $\langle p_1', \ldots, p_s'\rangle \in P$ and $q_1', \ldots, q_s'$ such that

1. *For any $i \in \{1, \ldots, s\}$, $p_i \geq p_i'$ or ($p_i < p_i'$ and $A \cap]p_i, p_i'[= \emptyset$).*
2. $q_1' + \ldots + q_s' \leq p_0$;
3. $p_1' = \frac{p_0}{1-p_0+q_1'}, \ldots, p_s' = \frac{p_0}{1-p_0+q_s'}$.

PROOF. Let p_i'' be the smallest element of A such that $p_i \leq p_i''$ and q_i'' be such that $p_i'' = \frac{x}{1-x+q_i''}$. $p_i \leq p_i''$ implies $1 - x + q_i'' \geq q - x + q_i$ and $q_i \geq q_i''$. We replace all p_i and q_i by p_i'' and q_i''. Then, $q_1'' + \ldots + q_s'' \leq x$ because $q_i'' \leq q_i$.

Let x'' be the number generated by an application of the rule 2 to $p_1'' \in A, \ldots, p_s'' \in A$. Then, $x'' \geq x$ (Lemma 24). P contains a tuple $\langle p_1', \ldots, p_s'\rangle$ such that $p_1' \leq p_1'', \ldots, p_s' \leq p_s''$ (Lemma 43). By definition of (x, x)-minimal set, this tuple is (x, x)-allowed

and $p'_1 \in A, \ldots, p'_s \in A$. Hence, an application of the rule 2 to it generates $x' \geq x$. More, $x' \geq p_0$ because p_0 is the smallest element of A such that $x \leq p_0$.

Let $q''_1, \ldots, q''_s$ be the values of $q_1, \ldots, q_s$ in the application of the rule 2 to $p'_1, \ldots, p'_s$. We have $q''_i = x' + \frac{x'}{p'_i} - 1$ (cf. proof of Lemma 22). Let $q'_i = p_0 + \frac{p_0}{p'_i} - 1$. Then,

$$q'_1 + \ldots + q'_s = \left(p_0 + \frac{p_0}{p'_1} - 1\right) + \ldots + \left(p_0 + \frac{p_0}{p'_s} - 1\right) \leq$$

$$\left(x + \frac{x}{p'_1} - 1\right) + (p_0 - x) + \ldots + \left(x + \frac{x}{p'_s} - 1\right) + (p_0 - x) \leq$$

$$q''_1 + \ldots + q''_s + s(p_0 - x) = x + s(p_0 - x) \leq x + (p_0 - x) = p_0,$$

proving 2. 3 follows by substituting the expression for q'_i.

If $p'_i < p''_i$, then $p'_i < p_i$ because p''_i is the smallest element of A such that $p_i \leq p''_i$. If $p'_i = p''_i$ and $p_i = p''_i$, then $p_i = p'_i$. If $p'_i = p''_i$ and $p_i < p''_i$, then $p_i < p'_i$, $]p'_i, p_i[=]p'_i, p''_i[$ and $A \cap]p'_i, p''_i[= \emptyset$ by the definition of p''_i. In all three cases, 1 is true. ∎

In the simulation algorithm for $p = p_0$, we use several simulation algorithms for $p > p_0$ as subroutines. Namely, we use:

1. A simulation algorithm for $p = p'_0$.

2. Simulation algorithms for $p = p'_i$, for $\langle p_1, \ldots p'_s \rangle \in P$ and $i \in \{1, \ldots, s\}$.

The existence of these simulation algorithms is implied by the assumption that Theorem 21 holds for $p > p_0$.

A $[pk, k]$**PFin**-team $L = \{L_1, \ldots, L_k\}$ simulates a probabilistic **PFin**$\langle x \rangle$-machine M as follows:

1. $L_1, \ldots, L_k$ read $f(0)$, $f(1)$, ..., simulate M and wait until the probability that M has issued a conjecture reaches x. Then pk machines $(L_1, \ldots, L_{pk})$ issue conjectures $h_1, \ldots, h_{pk}$.

2. The first values of the functions computed by h_1, ..., h_{pk} are identical to the values of f, i.e.

   $$\varphi_{h_1}(i) = \ldots = \varphi_{h_{pk}}(i) = f(i)$$

   for $i \leq m$ where $f(m)$ is the last value of f read by L before issuing conjectures. The next values of these functions are computed as follows:

(a) Simulate the conjectures of M issued before L outputs $h_1, \ldots, h_{pk}$. If all these programs output the same $f(n)$, $h_1, \ldots, h_{pk}$ output this value, too. Otherwise, for each possible value of $f(n)$ compute the probability that M has issued conjecture with this value.

If there is only one value with the probability at least Δ, all programs $h_1, \ldots, h_{pk}$ output this value.

(b) Otherwise, $d_1, \ldots, d_s$ denote these values and $q_1, \ldots, q_s$ denote the probabilities that $f(n) = d_1, \ldots, f(n) = d_s$, respectively. Let

$$p_1 = \frac{x}{1 - x + q_1}, \ldots, p_s = \frac{x}{1 - x + q_s}.$$

The programs $h_1, \ldots, h_{pk}$ compute $p_1, \ldots, p_s$, search the set P and find the tuple $\langle p'_1, \ldots, p'_s \rangle \in P$ (cf. Lemma 48). Then, they compute

$$q'_1 = p_0 + \frac{p_0}{p'_1} - 1, \ldots, q'_s = p + \frac{p_0}{p'_s} - 1.$$

$q'_1 k$ of programs $h_1, \ldots, h_{pk}$ output $f(m) = d_1$, $q'_2 k$ programs output $f(m) = d_2$ and so on.

Further, $q'_i k$ programs together with $(1 - p_0)k$ machines $L_{p_0 k+1}, \ldots, L_k$ simulate M on functions with $f(m) = d_i$ according to the algorithm for $p = p'_i$.

3. After $L_1, \ldots, L_{p_0 k}$ have issued conjectures, all remaining machines in the team L read the next values of the input function and simulate the conjectures issued by the probabilistic machine M before conjectures of $L_1, \ldots, L_{p_0 k}$. They wait until the splitting of conjectures in step 2b happens or the probability of conjectures consistent with the input segment becomes less than Δ.

(a) If the splitting in 2b happens, $L_{p_0 k+1}, \ldots, L_k$ (i.e. all machines which have not issued conjectures yet) read $f(m)$. If it is equal to d_i, they participiate in the simulation according to the algorithm for $p = p'_i$.

(b) If the probability of conjectures consistent with an input segment becomes less than Δ (i.e. almost all conjectures of M have different initial segments). $L_{p_0 k+1}, \ldots, L_k$ start a simulation according to the algorithm for $p = p'_0$. (We recall that p'_0 is the smallest element of A which is greater than or equal to $\frac{p_0}{1-p_0}$.)

Proof of correctness. In our simulation algorithm, we use simulation algorithms for the ratios of successful machines greater than p_0. We must prove that

- these algorithms can do required simulations,
- these simulations give us at least $p_0 k$ correct programs.

1. Step 2b. Here, we use $q_i' k$ programs and $(1 - p_0)k$ machines $L_{p_0 k+1}, \ldots, L_k$ to simulate M on functions with $f(m) = d_i$ according to the algorithm for $p = p_i'$.

   (a) The simulation is possible.

   M can be transformed into machine M' with the probability of success p_i. (We take a machine M' which, before outputing a conjecture, checks whether its conjecture f has $f(m) = d_i$. If $f(m) = d_i$, then it outputs the conjecture. If $f(m) \neq d_i$, then M' does not output the conjecture. Instead, it begins the learning on the same input data once again. The probability of success of M' is equal to the conditional probability of success of M, if it is known that M issues the conjecture with $f(m) = d_i$. It is $\frac{x}{1-x+q_i} = p_i$.)

   p_i' is selected so that p_i is greater than any $p \in A \cap [0, p_i'[$. Theorem 21 holds for $p = p_i'$ because $p_i' > p_0$. Hence, a probabilistic machine with success probability p_i can be simulated by a team with the ratio of successful programs p_i'.

   (b) The simulation gives $p_0 k$ correct programs.

   We have $q_i' k$ programs and $(1 - p_0)k$ machines which have not issued conjectures yet. So, together we have $(1 - p_0 + q_i')k$ programs. If they work as a team with success ratio p_i', at least $p_i'(1 - p_0 + q_i')k$ programs are correct. By the definition of p_i',

$$p_i' = \frac{p_0}{1 - p_0 + q_i'} \text{ and } p_i'(1 - p_0 + q_i')k = p_0 k.$$

2. Step 3b. The probability of conjectures consistent with input segment becomes less than Δ and we use the simulation algorithm for $p = p_0'$.

   (a) The simulation is possible.

Similarly to the previous case, M can be transformed into machine M' which identifies only functions consistent with input read so far. The probability of success of M' is equal to the success ratio of M.

The probability of issued conjectures is at most Δ. Hence, the success ratio is at least $\frac{x}{1-x+\Delta}$. By the definition of Δ, it is greater or equal than any $p \in A \cap [0, p_0'[$. Hence, M' can be simulated by a team with success ratio p_0'.

(b) The simulation gives $p_0 k$ correct programs.

$(1-p_0)k$ machines $(L_{p_0 k+1}, \ldots, L_k)$ participate in this simulation. $p_0'(1-p_0)k$ of them are successful. By the definition of p_0', we have $p_0' \geq \frac{p_0}{1-p_0}$. Hence,

$$p_0'(1-p_0)k \geq \frac{p_0}{1-p_0}(1-p_0)k = p_0 k.$$

The size of L. We show how to select the size of the team L so that be it will able to perform all described simulations. Two conditions must be satisfied:

1. When the machines of the team split, the amount of machines saying that $f(m) = d_i$ must be integer for any d_i i.e., $q_i' k$ must be integer in all cases.

2. When the simulation algorithm for the success ratio p_0 uses another simulation algorithm (with the ratio of successful machines $p > p_0$). certain team size k' is required for simulation with $[pk', k']$**PFin**-team. The amount of machines participiating in this simulation (when it is used as the subroutine of the simulation for the ratio p_0) must be multiple of k'.

   Formally. it is equivalent to:

   (a) For all $\langle p_1', \ldots, p_s' \rangle \in P$ and $i \in \{1, \ldots, s\}$, $(1 - p_0 + q_i')k$ must be a multiple of k_i where k_i is the size of the team with the success ratio p_i'.

   (b) $(1-p_0)k$ must be a multiple of k_0. the size of the simulation team with the success ratio p_0'.

The set P is finite. Hence, we have only finitely many requirements about the size of simulating team for $p = p_0$.

All p_i' belong to A and A is the subset of rational numbers(cf. Section 6.6). It implies that all q_i' are rationals, too. Hence, all requirements about the size of the team are equivalent to requiring that the team size is a multiple of some finite

number of integers $k_1, \ldots, k_m$. If we select the size k so, the simulation algorithm will be able to perform all required simulations. ∎

Theorem 21 implies

Corollary 8 *Let $x, y \in [0,1]$ and $x < y$. If there is no $p \in A$ satisfying $x \leq p < y$, then*

$$\mathbf{PFin}\langle x \rangle = \mathbf{PFin}\langle y \rangle.$$

PROOF. Any machine which succeeds with probability y, succeds with probability $x < y$, too. Hence, it suffices to prove that any machine with the probability of success x can be simulated by a machine with the probability of success y, i.e.

$$\mathbf{PFin}\langle x \rangle \subseteq \mathbf{PFin}\langle y \rangle.$$

Let p be the smallest element of A which is greater than x. Theorem 21 implies

$$\mathbf{PFin}\langle x \rangle \subseteq [pk, k]\mathbf{PFin} \subseteq \mathbf{PFin}\langle p \rangle.$$

We have $y \leq p$ and, hence,

$$\mathbf{PFin}\langle p \rangle \subseteq \mathbf{PFin}\langle y \rangle$$

$$\mathbf{PFin}\langle x \rangle \subseteq \mathbf{PFin}\langle y \rangle$$

∎

So, if Theorem 18 does not prove that the power of learning machines with probabilities x and y is different, then these probabilities are equivalent. Hence,

Theorem 22 *A is the probability hierarchy for probabilistic* **PFin***-type learning in the range* $[0,1]$.

PROOF. Follows from Theorem 18 and Corollary 8. ∎

Theorem 22 has a following important corollary.

Theorem 23 *Probabilistic* **PFin***-type learning probability structure is decidable i.e. there is an algorithm that receives as input two probabilities p_1 and p_2 and computes whether* $\mathbf{PFin}\langle p_1 \rangle = \mathbf{PFin}\langle p_2 \rangle$.

PROOF. Use the algorithm of Lemma 21 to find the intervals $[f_1(p_1), f_2(p_1)]$ and $[f_1(p_2), f_2(p_2)]$. These two intervals are equal if and only if $\mathbf{PFin}\langle p_1 \rangle = \mathbf{PFin}\langle p_2 \rangle$. ∎

6.9 Relative complexity

From Theorem 19 we know that **PFin**-type probability hierarchy is well-ordered. A question appears: what is the ordering type of this hierarchy? To what particular ordinal is it order-isomorphic? We analyse the proof of Theorem 19 step by step.

$\alpha(x)$ denotes the ordering type of $A \cap]x, 1]$ for $x \leq \frac{1}{2}$ and the ordering type of $A \cap [x, 1]$ for $x > \frac{1}{2}$. If $x \geq y$, then $\alpha(x) \leq \alpha(y)$ because $A \cap]x, 1] \subseteq A \cap]y, 1]$. We will often use this inequality.

Lemma 49 $\alpha(\frac{1}{2}) = \omega$.

PROOF. $A \cap]\frac{1}{2}, 1]$ consists of a single sequence 1, 2/3, 3/5, ...[22]. ∎

First, we prove lower bounds on the ordering type of A. $l(p)$ is the largest ordinal α such that there is an ω^α-sequence in $A \cap]p, 1]$ which converges to p. We define $l(p) = 0$ if there is no such sequence for any α.

It is easy to see that $\alpha(p) \geq \omega^{l(p)}$. However, there may be a large gap between these two ordinals. For example, if $A \cap]p, 1]$ has the ordering type $\omega^\omega + 1$, there is no infinite monotonous sequence converging to p and $l(p) = 0$. We use the function l to prove lower bounds.

Lemma 50

$$l\left(\frac{p}{1+p}\right) \geq \alpha(p).$$

PROOF. Transfinite induction over $p \in A$.
Base Case. Let $p = 1$. The ordering type of $A \cap [1, 1] = \{1\}$ is 1. The ordering type of $A \cap]1/2, 1]$ is ω and $l(1/2) = 1$.
Inductive Case. Consider two cases:

1. p is a successor element.

   Let $p \in [\frac{1}{n}, \frac{1}{n-1}]$. r denotes the element immediately preceding p. We have $\alpha(p) = \alpha(r) + 1$ because p is the only element of $A \cap [p, 1]$ which does not belong to $A \cap [r, 1]$. By inductive assumption, $l(\frac{r}{1+r}) \geq \alpha(p)$.

   Consider the splitting of $[\frac{1}{n+1}, \frac{1}{n}]$ in the proof of Theorem 19 (subsection 6.7.1). In the first step, one of segments is $[\frac{p}{1+p}, \frac{r}{1+r}]$ because $[p, r]$ does not contain other elements of A. We consider the sequence r_0, r_1, ... corresponding to $[\frac{p}{1+p}, \frac{r}{1+r}]$.

Lemma 51 $l(r_i) \geq \alpha(r)$.

PROOF. By induction.

Base Case. Let $i = 0$. Then, $r_0 = \frac{r}{1+r}$ and $l(r_0) = l(\frac{r}{1+r}) \geq \alpha(r)$.

Inductive Case. We prove $l(r_{i+1}) \geq l(r_i)$. Then $l(r_{i+1}) \geq \alpha(r)$ follows from $l(r_i) \geq \alpha(r)$. We use

Lemma 52 *If a set is obtained from ω^α by removing a proper initial segment, it still has ordering type ω^α.*

PROOF. If we remove a segment with ordering type β, we obtain the set with ordering type $\omega^\alpha - \beta$ (Definition 11). We have $\omega^\alpha - \beta = \omega^\alpha$ for all $\beta < \omega^\alpha$. ∎

Let

$$f(x) = \frac{2}{1 + \frac{1}{x} + \frac{1}{p}}.$$

$f(x)$ maps $x \in A$ to the number generated from x and p by rule 2(Lemma 22). Let x_0 be the number such that $f(x) = r_i$. The function f maps (x_0, r_i) to (r_i, r_{i+1}). ($r_{i+1} = f(r_i)$ by the definition of r_{i+1}.)

We take an $\omega^{l(r)}$ sequence converging to r_i and remove all $x < x_0$ from it. The remaining sequence is still $\omega^{l(r)}$ (Lemma 52). f maps it to a sequence converging to r_{i+1} and preserves the ordering. Hence, $l(r_{i+1}) \geq l(r)$. ∎

We take the union of α^β sequences converging to $r_0, r_1, \ldots$ and obtain a $\alpha^{\beta+1}$ sequence converging to $\lim_{i\to\infty} r_i = \frac{p}{1+p}$. Hence, $l(\frac{p}{1+p}) \geq \alpha(r) + 1 = \alpha(p)$.

2. p is a limit element.

Let $p_0, p_1, \ldots$ be a decreasing sequence converging to p. Then,

$$\alpha(p) = \lim_{i\to\infty} \alpha(p_i).$$

We take the union of $\omega^{\alpha(p_i)}$ sequences converging to $\frac{p_i}{1+p_i}$. It has the ordering type

$$\lim_{i\to\infty} \omega^{\alpha(p_i)} = \omega^{\lim_{i\to\infty} \alpha(p_i)} = \omega^{\alpha(p)}$$

and converges to $\frac{p}{1+p}$. Hence, $l(\frac{p}{1+p}) \geq \alpha(p)$. ∎

Lemma 53 $\alpha\left(\frac{p}{1+p}\right) \geq \omega^{\alpha(p)}$.

PROOF. Follows from Lemma 50 and $\alpha(\frac{p}{1+p}) \geq \omega^{l(\frac{p}{1+p})}$. ∎

The upper bound proof is more complicated. We prove a counterpart of Lemma 53.

Lemma 54 $\alpha\left(\frac{p}{1+p}\right) \leq \omega^{\alpha(p)}$.

PROOF. Transfinite induction over $p \in A$.
Base Case. Let $p = 1$. The ordering type of $A \cap [1,1]$ is 1 and the ordering type of $A \cap [1/2,1]$ is ω.
Inductive Case. Consider two cases:

1. p is a successor element.

   Let r be the element immediately preceding p. Similarly to the proof of Lemma 50, r_0, r_1, ... is the sequence in the splitting of $[\frac{1}{n+1}, \frac{1}{n}]$ corresponding to $[\frac{p}{1+p}, \frac{r}{1+r}]$.

   Lemma 55 $\alpha(r_i) \leq w^{\alpha(r)} \cdot c_i$ *for some* $c_i \in \mathbb{N}$.

   PROOF. By induction.

   Base Case. If $i = 0$, $r_0 = \frac{r}{1+r}$ and $\alpha(\frac{r}{1+r}) \leq \omega^{\alpha(r)}$ by inductive assumption.

   Inductive Case. Let P be a (r_{i+1}, r_{i+1})-minimal set (cf. section 6.7.4). Let $A(p_1, \ldots, p_s)$ denote the set of all $x \in A \cap]r_{i+1}, r_i]$ generated by applications of the rule 2 to $p'_1 \in A$, ..., $p'_s \in A$ such that $p_1 \leq p'_1$, ..., $p_s \leq p'_s$. $\alpha'(p_1, \ldots, p_s)$ denotes the ordering type of $A(p_1, \ldots, p_s)$.

   Lemma 56

$$\alpha(r_{i+1}) \leq \alpha(r_i)(+) \sum_{\langle p_1,\ldots,p_s\rangle \in P} \alpha'(p_1, \ldots, p_s).$$

   PROOF. We have

$$A \cap]r_{i+1}, 1] = (A \cap]r_i, 1]) \cup_{\langle p_1,\ldots,p_s\rangle \in P} A(p_1, \ldots, p_s).$$

   By Lemma 1, the ordering type of $A \cap]r_{i+1}, 1]$ is less than or equal to the natural sum of the ordering types of $A \cap]r_i, 1]$ and all $A(p_1, \ldots, p_s)$. ∎

Next, we bound each $\alpha'(p_1, \ldots, p_s)$. We start with an auxilary lemma.

Lemma 57 *If $p \in A$ follows from an application of the rule 2 to $p_1 \in A, \ldots,$ $p_s \in A$, then*

$$\alpha(p) \geq \alpha(p_1)(+)\ldots(+)\alpha(p_s).$$

PROOF. Without the loss of generality, we assume that $p_1 \leq p_2 \leq \ldots \leq p_s$. Then, $\alpha(p_1) \geq \alpha(p_2) \geq \ldots \geq \alpha(p_s)$. We prove the lemma by transfinite induction over p_1.

Base Case. p_1 is the maximum element, i.e. $p_1 = 1$.

Then, $p_1 = \ldots = p_s = 1$. An application of the rule 2 to $p_1, \ldots, p_s$ generates $p = s/(2s-1)$.

$$A \cap \left[\frac{s}{2s-1}, 1\right] = \left\{\frac{s}{2s-1}, \frac{s-1}{2s-3}, \ldots, \frac{2}{3}, 1\right\}.$$

The ordering type of this set is s, i.e. $\alpha(p) = s$. On the other hand, $\alpha(p_1) = \ldots = \alpha(p_s) = 1$ and

$$\alpha(p_1)(+)\ldots(+)\alpha(p_s) = s.$$

Inductive Case. We have two possibilities:

(a) p_1 is a successor element.

j denotes the maximum number such that $p_1 = \ldots = p_j$. Let

$$p'_i = \begin{cases} \text{predecessor of } p_1, & \text{if } i \leq j \\ p_i, & \text{if } i > j \end{cases}.$$

We have $\alpha(p_i) = \alpha(p'_i) + 1$ for $i \leq j$ and $\alpha(p_i) = \alpha(p'_i)$ for $i > j$. Hence,

$$\alpha(p_1)(+)\ldots(+)\alpha(p_s) =$$

$$(\alpha(p'_1) + 1)(+)\ldots(+)(\alpha(p'_j) + 1)(+)\alpha(p'_{j+1})(+)\ldots(+)\alpha(p'_s) =$$

$$\alpha(p'_1)(+)\ldots(+)\alpha(p'_s) + j.$$

Let x_0 be the number generated by an application of the rule 2 to $p'_1, \ldots, p'_s$ and x_i, for $i \in \{1, \ldots, j\}$, be the number generated by an application of the rule 2 to $p_1, \ldots, p_i, p'_{i+1}, \ldots, p'_s$. By inductive assumption,

$$\alpha(x_0) \geq \alpha(p'_1)(+)\ldots(+)\alpha(p'_s).$$

We have $p_i < p'_i$ for $i \leq j$. By Lemma 24, $x_i < x_{i-1}$. Hence,

$$\alpha(x_i) \geq \alpha(x_{i-1}) + 1.$$

We have $p_i = p'_i$ for $i > j$. Hence, $x_j = p$ and

$$\alpha(p) = \alpha(x_j) \geq \alpha(x_0) + j \geq \alpha(p'_1)(+)\ldots(+)\alpha(p'_s) + j =$$

$$\alpha(p_1)(+)\ldots(+)\alpha(p_s).$$

(b) p_1 is a limit element.

Again, j is the maximum number such that $p_1 = \ldots = p_j$. Let $p'_1, p'_2, \ldots$ be a monotonous sequence converging to p_1 and x_i be the number generated by an application of the rule 2 to p'_i, ..., p'_i, p_{j+1}, ..., p_s. By inductive assumption,

$$\alpha(x_i) \geq \underbrace{\alpha(p'_i)(+)\ldots(+)\alpha(p'_i)}_{j \text{ times}}(+)\alpha(p_{j+1})(+)\ldots(+)\alpha(p_s). \qquad (6.3)$$

We have $p_1 = \ldots = p_j = \lim_{i\to\infty} p'_i$ By Lemma 25, $p = \lim_{i\to\infty} x_i$. Hence, if we take $i \to \infty$ in (6.3) and apply the fact that $(+)$ is continous, we get

$$\alpha(p) \geq \alpha(p_1)(+)\ldots(+)\alpha(p_s).$$

∎

Lemma 58 *Let* $\langle p_1, \ldots, p_s\rangle \in P$. *Then*

$$\alpha'(p_1, \ldots, p_s) \leq \omega^{\alpha(r)} \cdot \text{const}\,.$$

PROOF. Lemma 2 implies that $\alpha'(p_1, \ldots, p_s)$ is at most the natural product of $\alpha(p_1)$, ..., $\alpha(p_s)$. Let α_j be the largest ordinal such that $\alpha(p_j) \geq \omega^{\alpha_j}$. Then, $\alpha(p_j) \leq c_j\omega^{\alpha_j}$. (If there is no such c_j, then $\alpha(p_j) \leq \lim \omega^{\alpha_j} \cdot c = \omega^{\alpha_j+1}$ and α_j is not the largest ordinal with such a property.) Hence,

$$\alpha'(p_1, \ldots, p_s) \leq \omega^{\alpha_1} \cdot c_1(\cdot)\omega^{\alpha_2} \cdot c_2 \ldots \omega^{\alpha_s} \cdot c_s = \omega^{\alpha_1(+)\alpha_2(+)\ldots(+)\alpha_s} \cdot (c_1c_2\ldots c_s).$$

Let p'_j be such that $p'_j \in A$ and $\alpha(p'_j) = \alpha_j$. We have $\alpha_j \leq \alpha(r)$ because $p_j \geq r_i$, $\alpha(p_j) \leq \alpha(r_i)$, $\alpha(p_j) \geq \omega^{\alpha_j}$ and $\alpha(r_i) \leq \omega^{\alpha(r)} \cdot \text{const} < \omega^{\alpha(r)+1}$. Hence, $p'_j \geq r$ and both Lemma 50 and Lemma 54 are true for $p = p'_j$. This means that $\alpha(\frac{p'_j}{1+p'_j}) = \omega^{\alpha_j}$. Hence, $\frac{p'_j}{1+p'_j} \geq p_j$ because $\alpha(p_j) \geq \omega^{\alpha_j}$.

Let p' be the number generated by an application of the rule 2 to $p'_1, \ldots, p'_s$. By Lemma 26, $\frac{p'}{1+p'}$ is generated by an application of the rule 2 to $\frac{p'_1}{1+p'_1}, \ldots, \frac{p'_s}{1+p'_s}$.

$\frac{p'}{1+p'}$ is greater than or equal to the number generated by an application of rule 2 to $p_1, \ldots, p_s$ because $\frac{p'_j}{1+p'_j} \geq p_j$. This number is at least r_{i+1} because the tuple $\langle p_1, \ldots, p_s \rangle$ belongs to the (r_{i+1}, r_{i+1})-allowed set P. Hence, $\frac{p'}{1+p'} \geq r_{i+1}$. We have $\frac{p'}{1+p'} \geq \frac{r}{1+r}$ because $[\frac{p}{1+p}, \frac{r}{1+r}]$ does not contain any points of type $\frac{p'}{1+p'}$ with $p' \in A$. This implies $p' \geq r$.

By Lemma 57,

$$\alpha(p') \geq \alpha(p_1)(+)\ldots(+)\alpha(p_s).$$

This implies

$$\alpha'(p_1, \ldots, p_s) \leq \omega^{\alpha(p_1)(+)\ldots(+)\alpha(p_s)} \cdot (c_1 \ldots c_s) \leq$$

$$\omega^{\alpha(p')} \cdot (c_1 \ldots c_s) \leq \omega^{\alpha(r)} \cdot (c_1 \ldots c_s).$$

∎

Now, we are ready to finish the proof of Lemma 55. By Lemma 56, $\alpha(r_{i+1})$ is less than or equal to the natural sum of $\alpha(r_i)$ and $\alpha'(p_1, \ldots, p_s)$. We have $\alpha(r_i) \leq \omega^{\alpha(r)} \cdot \text{const}$ by inductive assumption and

$$\alpha'(p_1, \ldots, p_s) \leq \omega^{\alpha(r)} \cdot \text{const}$$

by Lemma 58. Hence, the natural sum of these ordinals is at most $\omega^{\alpha(r)} \cdot \text{const}$, too. ∎

$$\alpha\left(\frac{p}{1+p}\right) = \lim_{i\to\infty} \alpha(r_i) \leq \lim_{i\to\infty} \omega^{\alpha(r)} \cdot c_i \leq \lim_{i\to\infty} \omega \cdot \omega^{\alpha(r)} = \omega^{\alpha(r)+1} = \omega^{\alpha(p)}.$$

2. p is a limit element.

Let $p_0, p_1, \ldots$ be a decreasing sequence converging to p. By inductive assumption, $\alpha(\frac{p_i}{1+p_i}) \leq \omega^{\alpha(p_i)}$. We have

$$\alpha\left(\frac{p}{1+p}\right) = \lim_{i\to\infty} \alpha\left(\frac{p_i}{1+p_i}\right) \leq \lim_{i\to\infty} \omega^{\alpha(p_i)} = \omega^{\lim_{i\to\infty} \alpha(p_i)} = \omega^{\alpha(p)}.$$

∎

Lemma 59 $\alpha(\frac{p}{1+p}) = \omega^{\alpha(p)}$.

PROOF. Follows from Lemmas 53 and 54. ∎

Theorem 24 *The ordering type of A is at least* ϵ_0.

PROOF. The ordering type of $A \cap]\frac{1}{2}, 1]$ is ω (Lemma 49). The ordering type of $A \cap]\frac{1}{3}, 1]$ is ω^ω (Lemma 59 with $p = 1/2$), the ordering type of $A \cap]\frac{1}{4}, 1]$ is ω^{ω^ω} and so on. The ordering type of A is the limit of this sequence, i.e.

$$\epsilon_0 = \lim(\omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \ldots).$$

∎

It is known that the ordinal ϵ_0 expresses the set of all expressions possible in first-order arithmetic. It is also claimed that the ordinal ϵ_0 is so large that it is very difficult to find any intuitive description for ϵ_0.

So, we see that **PFin**, a very simple learning criterion, generates a very complex probability hierarchy.

Table below shows how the complexity of the hierarchy increases. All results in this table can be obtained using Lemma 59.

Interval	Ordering type of the probability hiearchy
$[\frac{1}{2}, 1]$	ω
$[\frac{4}{9}, 1]$	$\omega \cdot 2$
$[\frac{3}{7}, 1]$	3ω
$[\frac{2}{5}, 1]$	ω^2
$[\frac{3}{8}, 1]$	ω^3
$[\frac{1}{3}, 1]$	ω^ω
$[\frac{1}{4}, 1]$	ω^{ω^ω}
$[0, 1]$	ϵ_0

It shows that the known part of hiearchy ($[\frac{3}{7}, 1]$) is very simple compared to the entire hierarchy.

Notes. The points of the probability hierarchy in the intervals $[\frac{1}{2}, 1]$, $[\frac{4}{9}, 1]$ and $[\frac{3}{7}, 1]$ were explicitly decribed in [22], [19] and [16], respectively.

In [16], an ω^2 sequence of points converging to $\frac{2}{5}$ was presented and it was conjectured that this sequence forms the backbone of the learning capabilities in the interval $[\frac{2}{5}, 1]$.

6.10 Probabilistic versus team learning

For **Ex**-identification, there is a precise correspondence between probabilistic and team learners (Pitt's connection[42]). Any probabilistic learner can be simulated by any team with the ratio of successful machines equal to the probability of success for the probabilistic learner.

However, the situation is more complicated for finite learning (**Fin** and **PFin**). Here, the learning power of a team depends not only on the ratio of succesful machines. Team size is also important.

Theorem 25 *[53, 30]* $[1,2]\mathbf{PFin} \subset [2,4]\mathbf{PFin}$.

So, a team of 4 learning machines where 2 machines are required to be successful has more learning power than team of 2 learning machines where 1 must succeed. However, in both teams the ratio of successful machines to all machines is the same($\frac{1}{2}$).

This phenomena is called *redundancy*. Various redundancy types have been discovered for various ratios of successful machines(cf. [20, 16, 30]). The theorem below is the example of infinite redundancy[16, 20].

Theorem 26 *[16] If* $k \bmod 3 \neq 0$, *then*

$$[2k,5k]\mathbf{PFin} \subset [8k,20k]\mathbf{PFin}.$$

In particular,

$$[2,5]\mathbf{PFin} \subset [8,20]\mathbf{PFin} \subset [32,80]\mathbf{PFin} \subset \ldots.$$

So, for the ratio of successful machines 2/5 there are infinitely many different team sizes with different learning power.

However, even for **PFin**, any probabilistic machine can be simulated by a team with the same ratio of success, if we choose the team size carefully. A simple corollary of Theorem 21 is

Corollary 9 *If* $p, q \in \mathbb{N}^+$, *then there exists* k *such that*

$$\mathbf{PFin}\langle\frac{p}{q}\rangle = [pk,qk]\mathbf{PFin}.$$

This shows that probabilistic **PFin**-learning and team **PFin**-learning are of the same power.

Corollary 10 *If $p, q \in \mathbb{N}^+$, then there exists k such that*

$$[pl, ql]\mathbf{PFin} \subseteq [pk, qk]\mathbf{PFin}$$

for any $l \in \mathbb{N}^+$.

PROOF. The team of ql machines can be simulated by single probabilistic machine which equiprobably chooses one of machines in team and simulates it. Hence, Theorem 9 implies that

$$[pl, ql]\mathbf{PFin} \subseteq PFIN(\frac{p}{q}) = [pk, qk]\mathbf{PFin}.$$

∎

So, we see that redundancy structures can be very complicated but always there is the "best" team size such that team of this size can simulate any other team with the same ratio of successful machines. It exists even if there are infinitely many team sizes with different learning power (like for ratio 2/5, Theorem 26).

6.11 Summary

We have investigated the structure of probability hierarchy for **PFin**-type learning. Instead of trying to determine the exact points at which the learning capabilities change, we focused on the structural properties of the hieararchy.

We have developed a universal diagonalization algorithm (Theorem 18) and a universal simulation algorithm (Theorem 21). These algorithms are very general forms of diagonalization and simulation arguments used for probabilistic **PFin** [16, 19].

Universal diagonalization theorem gives the method that can be used to obtain any possible diagonalization for probabilistic **PFin**. Universal simulation algorithm can be used for any possible simulation.

These two results together give us a recursive description of the set of points A at which the learning capabilities are different.

This set is well-ordered in decreasing ordering. (This property is essential to the proof of Theorem 21.) Its structure is very complicated. Namely, its ordering type is ϵ_0, the ordering-type of the set of all expressions possible in first-order arithmetic.

It shows the huge complexity of the probabilistic **PFin**-hiearchy and explains why it is so difficult to find the points at which the learning capabilities are different.

A simple corollary of our results is that the probabilistic and team **PFin**-type learning is of the same power, i.e. any probabilistic learning machine can be simulated by a team with the same success ratio.

Several open problems remain:

1. Unrestricted finite learning(**Fin**).

   The major open problem is the generalization of our results for other learning paradigms such as (non-Popperian) **Fin**-type learning and language learning in the limit.

   Theorem 18 can be proved for (nonPopperian) **Fin**-type learning, too. Hence, if

   $$\mathbf{PFin}\langle p_1\rangle \neq \mathbf{PFin}\langle p_2\rangle,$$

   then

   $$\mathbf{Fin}\langle p_1\rangle \neq \mathbf{Fin}\langle p_2\rangle.$$

   So, the probability hierarchy of **Fin** is at least as complicated as the probability hierarchy of **Fin**. It is even more complicated because it is known[18, 19] that

   $$\mathbf{Fin}\langle 24/49\rangle \subset \mathbf{Fin}\langle 1/2\rangle$$

   but

   $$\mathbf{PFin}\langle 24/49\rangle = \mathbf{PFin}\langle 1/2\rangle.$$

   The simulation techniques for **Fin** are much more complicated than simulation techniques for **PFin**. However, we hope that some combination of our methods and other ideas (cf. [18, 17]) can help to identify the set of all possible diagonalization methods for **Fin** and to prove that no other diagonalization methods exists (i.e. to construct universal simulation for **Fin**).

   A step in that direction was made in [3] by proving that **Fin**-hiearchy is well-ordered and recursively enumerable. It still remains open whether it is decidable. The proof technique in [3] is different from ours and uses capability trees[17].

2. Probabilistic language learning.

The probability hierarchy of language learning in the limit[26] has some similarities to **Fin** and **PFin**-hiearhies.

It is an interesting open problem whether some analogues of our results can be obtained for language learning in the limit.

3. What is the computational complexity of decision algorithms for the **PFin**-hierarchy?

4. How dense is the probability hierarchy?

   Can we prove the result of the following type:

   If $p_1, p_2 \in [\frac{1}{n+1}, \frac{1}{n}]$ and $|p_1 - p_2| < (1/2)^n$, then

   $$\mathbf{PFin}(p_1) \neq \mathbf{PFin}(p_2)?$$

Other properties of the whole hierarchy can be studied, too.

Chapter 7

Conclusion

We have shown two applications of constructive ordinals in the theory of inductive inference.

The first was counting mindchanges (cf. Freivalds and Smith[24]). Here, we examined the ordinal mindchange complexity of natural language classes and derived various sufficient conditions for the existence of such bounds. These conditions showed that ordinal bounds are strongly related to other notions of inductive inference. We also examined the role of the system of notations. Our results showed that various systems of notation form very complicated structure. Together, all these results show that the ordinal bounds on the number of mindchanges is rich and interesting research area with both concrete and more abstract results.

The second was the probability hierarchy of **PFin**. Here, ordinals and well-orderings were applied to obtain a global information about probability structure. In future, we should find other applications for powerful techniques of chapter 6. An extension of our results to criteria of success other than **PFin** is one possible application.

Bibliography

[1] A. Ambainis, *The power of procrastination in inductive inference: how it depends on used ordinal notations.* Proceedings of the 2nd European Conference on Computational Learning Theory, Lecture Notes in Computer Science, 904:99-111, 1995.

[2] A. Ambainis, *Probabilistic and team* **PFin***-type learning: general properties.* Proceedings of the 9th Conference on Computational Learning Theory, pp. 157-168, ACM, 1996.

[3] A. Ambainis, K. Apsītis, R. Freivalds, C.H. Smith, *Matrix games and team learning*, accepted for ALT'97, 1997.

[4] A. Ambainis, J. Case, S. Jain, and M. Suraj. *Not-so-nearly-minimal-size program inference.* In preparation, extends [12].

[5] A. Ambainis, S. Jain, A. Sharma, The ordinal mind change complexity of language identification, *in* "Proceedings of EuroCOLT'97". Lecture Notes in Computer Science, 1208:301-315, 1997.

[6] D. Angluin, Finding patterns commmon to a set of strings, *Journal of Computer and System Sciences*, 21(1980), 46-62.

[7] D. Angluin, C.H. Smith, *Inductive inference: theory and methods.* Computing Surveys, 15:237-269, 1983.

[8] K. Apsītis, Derived sets and inductive inference, *in* "Proceedings of AII'94". Lecture Notes in Computer Science, 872(1994), pp. 26-39, Springer-Verlag.

[9] K. Apsītis, R. Freivalds, C.H. Smith, On duality in learning and the selection of learning teams. *Information and Computation*, 129:53-62, 1996.

[10] S. Arikawa, S. Miyano, A. Shinohara, T. Shinohara, and A. Yamamoto. Algorithmic learning theory with elementary formal systems. *IEICE Trans. Inf. and Syst.*, E75-D No. 4:405-414, 1992.

[11] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference, *Theoret. Comput. Sci.*, 25(1983), 193-220.

[12] J. Case, S. Jain, and M. Suraj. Not-so-nearly-minimal-size program inference. In Klaus P. Jantke and Steffen Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 77-96. Springer-Verlag, 1995.

[13] M. Changizi, Self-monitoring machines and an ω^ω hierarchy of loops. *Inform. and Comput.*, 128(1996), 127-138.

[14] A. Church, *The constructive second number class.* Bulletin American Mathematical Society, 44:224-232, 1938

[15] A. Church, S. Kleene, *Formal definitions in the theory of ordinal numbers.* Fund. Math., 28:11-21, 1937

[16] R. Daley, B. Kalyanasundaram, *Use of reduction arguments in determining Popperian FIN-type learning capabilities.* Proceedings of the 4th International Workshop on Algorithmic Learning Theory, Lecture Notes in Computer Science, 744:173-186, 1993

[17] R. Daley, B. Kalyanasundaram, *FINite learning capabilities and their limits.* To appear at COLT'97. Full version available at http://www.cs.pitt.edu/ daley/fin/fin.html.

[18] R. Daley, B. Kalyanasundaram, M. Velauthapillai, *Breaking the probability $\frac{1}{2}$ barrier in FIN-type learning.* Journal of Computer and System Sciences, 25:574-599, 1995.

[19] R. Daley, B. Kalyanasundaram, M. Velauthapillai, *The power of probabilism in Popperian FINite learning.* Proceedings of the 3rd International Workshop on Analogical and Inductive Inference, Lecture Notes in Computer Science, 642:151-169, 1992.

[20] R. Daley, L. Pitt, M. Velauthapillai, T. Will *Relations between probabilistic and team one-shot learners.* Proceedings of the 4th Conference on Computational Learning Theory, pp. 228-239, Morgan-Kaufmann, 1991.

[21] S. Feferman, Classification of recursive functions by means of hierarchies, *Trans. Amer. Math. Soc.*, 104(1962), 101-122.

[22] R. Freivalds, *Finite identification of general recursive functions by probabilistic strategies.* Proceedings of the Conference on Algebraic, Arithmetic and Categorical Methods in Computation Theory, pp. 138-145. Akademie-Verlag, Berlin, 1979

[23] R. Freivalds, J. Bārzdiņš, K. Podnieks, Inductive inference of recursive functions: complexity bounds, *in* "Baltic Computer Science", Lecture Notes in Computer Science, 502(1991), pp.111-155, Springer-Verlag.

[24] R. Freivalds, C. H. Smith, The role of procrastination in machine learning. *Information and Computation*, 107:237-271, 1993.

[25] E. M. Gold, Language identification in the limit. *Information and Control*, 10:447-477, 1967.

[26] S. Jain, A.Sharma, Computational limits on team identification of languages. *Information and Computation*, 130:19-60, 1996.

[27] S. Jain and A. Sharma. On the intrinsic complexity of language identification. In *Proceedings of the Seventh Annual Conference on Computational Learning Theory, New Brunswick, New Jersey*, pages 278–286. ACM-Press, July 1994.

[28] S. Jain, A.Sharma, On identification by teams and probabilistic machines. K. P. Jantke, S. Lange, (eds.) *Algorithmic Learning for Knowledge-Based Systems* Lecture Notes in Computer Science, 961:108-145, 1995.

[29] S. Jain, A. Sharma, Elementary formal systems, intrinsic complexity, and procrastination, *in* "Proceedings of 9th Annual Conference on Computational Learning Theory", pp. 181-192, ACM, 1996.

[30] S. Jain, A. Sharma, M. Velauthapillai, Finite identification of functions by teams with success ratio $\frac{1}{2}$ and above, *Information and Computation*, 121:201-213, 1995.

[31] K. P. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.

[32] S. Kapur. Monotonic language learning. In *Proceedings of the Third Workshop on Algorithmic Learning Theory*. JSAI Press, 1992. Proceedings reprinted as Lecture Notes in Artificial Intelligence, Springer-Verlag.

[33] P. Kilpeläinen, H. Mannila, and E. Ukkonen. MDL learning of unions of simple pattern languages from positive examples. In *Proceedings of the Second European Conference on Computational Learning Theory, Lecture Notes in Artficial Intelligence 904*. Springer-Verlag, 1995.

[34] M. Kummer, *The strength of noninclusions for teams of finite learners*, Proceedings of the 7th Conference on Computational Learning Theory, pp. 268-277, ACM, 1994.

[35] S. Kleene, On notation for ordinal numbers. *Journal of Symbolic Logic*, 3:150-155, 1938.

[36] K. Kuratowski, A. Mostowski, *Set Theory*. North-Holland Publishing Company, Amsterdam, 1967.

[37] S. Lange and T. Zeugmann. Monotonic versus non-monotonic language learning. In *Proceedings of the Second International Workshop on Nonmonotonic and Inductive Logic*, pages 254–269. Springer-Verlag, 1993. Lecture Notes in Artificial Intelligence 659.

[38] M. Machtey, P. Young, *An Introduction to the General Theory of Algorithms*. North-Holland, 1978

[39] T. Motoki, T. Shinohara, and K. Wright. The correct definition of finite elasticity: Corrigendum to identification of unions. In L. Valiant and M. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory, Santa Cruz, California*, page 375. Morgan Kaufman, 1991.

[40] Y. Mukouchi. Inductive inference of an approximate concept from positive data. In S. Arikawa and K. P. Jantke, editors, *Algorithmic Learning Theory. 4th International Workshop on Analogical and Inductive Inference, AII'94 and 5th International Workshop on Algorithm Learning Theory. ALT'94*, Lecture Notes in Artificial Intelligence, 872, pages 484–499. Springer-Verlag, 1994.

[41] D. Osherson, M. Stob, S. Weinstein, *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press, Cambridge, MA, 1986

[42] L. Pitt, Probabilistic inductive inference, *Journal of the ACM*, 36:383-433, 1989.

[43] L. Pitt, C. H. Smith, Probability and plurality for aggregations of learning machines. *Information and Computation*, 77:77-92, 1988.

[44] H. Rogers Jr., Godel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331-341, 1958

[45] H. Rogers Jr., *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New-York, 1967. Reprinted, MIT Press, 1987.

[46] G. E. Sacks. *Higher Recursion Theory.* Springer-Verlag, 1990.

[47] M. Sato and T. Moriyama. Inductive inference of length bounded EFS's from positive data. Technical Report DMSIS-RR-94-2, Department of Mathematical Sciences and Information Sciences, University of Osaka Prefecture, Japan, 1994.

[48] W. Sierpinski, *Cardinal and ordinal numbers.* PWN - Polish Scientific Publishers, 1965

[49] T. Shinohara. *Studies on Inductive Inference from Positive Data.* PhD thesis, Kyushu University, Kyushu, Japan, 1986.

[50] T. Shinohara. Rich classes inferable from positive data: Length-bounded elementary formal systems. *Information and Computation*, 108:175–186, 1994.

[51] C.H. Smith, *Three decades of team learning.* Proceedings of the 5th International Workshop on Algorithmic Learning Theory, Lecture Notes in Computer Science, 872:211-228, 1994

[52] R. Soare, *Recursively Enumerable Sets and Degrees.* Springer-Verlag, 1987.

[53] M. Velauthapillai, *Inductive inference with bounded number of mind changes.* Proceedings of COLT'89, pp. 200-213, 1989

[54] K. Wright. Identification of unions of languages drawn from an identifiable class. In R. Rivest, D. Haussler, and M. K. Warmuth, editors, *Proceedings of*

the Second Annual Workshop on Computational Learning Theory, Santa Cruz, California, pages 328–333. Morgan Kaufmann Publishers, Inc., 1989.

[55] T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In K.P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, pages 190–258. Lecture Notes in Artificial Intelligence No. 961, Springer-Verlag, 1995.