# Parsimony Hierarchies For Inductive Inference *

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

John Case
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
USA

Sanjay Jain
School of Computing
National University of Singapore
Singapore 119260
Republic of Singapore

Mandayam Suraj
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
USA

**Abstract**

Freivalds defined an acceptable programming system independent criterion for learning programs for functions in which the final programs were required to be both correct and "nearly" minimal size, i.e, within a computable function of being purely minimal size. Kinber showed that this parsimony requirement on final programs limits learning power. However, in scientific inference, parsimony is considered highly desirable. A *lim-computable function* is (by definition) one calculable by a total procedure allowed to change its mind finitely many times about its output. Investigated is the possibility of assuaging somewhat the limitation on learning power resulting from requiring parsimonious final programs by use of criteria which require the final, correct programs to be "not-so-nearly" minimal size, e.g., to be within a lim-computable function of actual minimal size. It is shown that some parsimony in the final program is thereby retained, yet learning power strictly increases. Considered, then, are lim-computable functions as above but for which *notations for* constructive ordinals are used to bound the number of mind changes allowed regarding the output. This is a variant of an idea introduced by Freivalds and Smith. For this ordinal notation complexity bounded version of lim-computability, the power of the resultant learning criteria form finely graded, infinitely ramifying, infinite hierarchies intermediate between the computable and the lim-computable cases. Some of these hierarchies, for the natural notations determining them, are shown to be optimally tight.

**Keywords:** Computational Learning Theory, Minimal Size Program, Constructive Ordinal Notations, Limiting Computable Function.

**AMS Subject Code Classification:** 68Q32.

## 1 Introduction

This work is in the context of computability-theoretic inductive inference or learning theory [Put63, Gol67, JORS99, Odi99].

*Email addresses of the authors are: ambainis@lanet.lv, case@cis.udel.edu, sanjay@comp.nus.edu.sg, and suraj@cis.udel.edu, respectively. Grant support was received by A. Ambainis from Latvian Science Council Grant 01.0354, J. Case from NSF grant CCR-0208616, and by S. Jain from NUS grant R252-000-127-112.

In Section 1.1 are informally presented what we need of the basic concepts and motivations from inductive inference including parsimony constraints on inferred (learned) programs.

As we will see, degrees of parsimony will be measured with *notations* from Kleene's $O$ for the constructive ordinals [Rog67]. Section 1.2 presents a corresponding informal introduction with motivations and examples.

Section 1.3 summarizes our principal results. These feature infinite hierarchies of success criteria re inferring parsimonious programs, and they are based on $O$-measured degrees of parsimony.

Section 2 contains the needed basic terminology and definitions presented rigorously.

Our main results are presented with proof in Section 3, and Section 4 presents further and preliminary results and open questions.

## 1.1 Inductive Inference Machines and Parsimony

$N$ denotes the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$. A basic paradigm from machine inductive inference pertains to the *algorithmic*, trial and error "learning" of programs for (computable) functions : $N \to N$ — given, as input, enumerations of the graphs of the functions. We can think of a function $f : N \to N$ as encoding the set of all possible deterministic outcomes of the experiments on some corresponding phenomenon (e.g., from chemistry): an input $x$ to $f$ represents the code of a possible experiment, e.g., mixing particular volumes of particular chemicals in a test tube, and $f(x)$ represents the outcome, e.g., the mixture turned blue and fizzed (see, for example, [BB75, CS83]).[1] A program $p$ for such a function corresponds, then, to a *predictive explanation* for the associated phenomenon. Possession of $p$ enables one to predict the outcomes of any experiment (with associated code) $x$ regarding the phenomenon: use $p$ to compute $f(x)$ (and decode). The trial and error aspect of algorithmically learning such explanatory programs $p$ models that the scientific community changes its "mind" over time as to explanations for a phenomenon.

Informally, *acceptable programming systems* (synonym: *acceptable numberings*) [Rog58, Rog67, MY78, Roy87] are those programming systems for the partial computable functions which are intercompilable with naturally occurring general purpose programming formalisms such as Turing machine formalisms and the LISP programming language.[2] The programs output by inductive inference algorithms (or machines) will be from some fixed acceptable system. When an inductive inference machine $\mathbf{M}$ is given the graph of a function $f$ as input and, after some resultant succession of (trial and error) output programs, it converges to a single program $p$, we write $\mathbf{M}(f) = p$. If this $p$ is a correct program (explanation) for $f$, we say that $\mathbf{M}$ $\mathbf{Ex}$-*identifies* $f$. If some $\mathbf{M}$ $\mathbf{Ex}$-identifies every function in a class of functions $\mathcal{C}$, we write $\mathcal{C} \in \mathbf{Ex}$. For example, the class of primitive recursive functions is in $\mathbf{Ex}$, and the class of computable functions is not [Gol67, BB75].

In scientific inference, parsimony of explanations is considered highly desirable. Program size is one of many ways to measure parsimony of programs [JORS99], and one can think of parsimony as a special case or variant of Occam's Razor. It is known, for computability-theoretic inductive inference, that requiring the final and correct programs to be minimal size [Sch98] is highly restrictive on inferring power [Fre75, Fre90] (and that the resultant inferring or learning power is dependent on which acceptable programming system is employed). Known too is the adverse effect on learning power of requiring the final and correct programs to be merely within a computable factor of

[1] Fulk [Ful85] argues that the set of possible, distinguishable experiments *one can actually do* on a phenomenon is countable.

[2] Typically, for theoretical work, one works with *numerical* names for the programs in these systems — whence the term 'numbering.'

minimal size [Kin74, Che82] (but that the resulting inferring power is independent of the underlying acceptable programming system [Fre75]). Hence, we see that, while parsimony is desirable, parsimony restrictions of even the weaker kind described just above limit inferring power.[3]

In the present study we will be exclusively concerned with criteria of success for inductive inference (parsimony restricted or not), and these criteria will be independent of the fixed underlying acceptable programming system from which the output hypotheses are drawn. For computable $f$, let $\mathrm{MinProg}(f) \stackrel{\mathrm{def}}{=}$ the numerically least program for $f$. For $\mathcal{C} \in \mathbf{Ex}$ as witnessed by $\mathbf{M}$, if there is a computable function $g$ such that, for every $f \in \mathcal{C}$, $\mathbf{M}(f) \leq g(\mathrm{MinProg}(f))$, then we say $\mathcal{C} \in \mathbf{Mex}$ (as witnessed by $\mathbf{M}$ and $g$). In this setting we call $g$ a *parsimony factor*. From [Kin74] we have, then, that $\mathbf{Mex} \subset \mathbf{Ex}$.[4] For example, the set of functions which have value zero on almost all arguments $\in (\mathbf{Ex} - \mathbf{Mex})$ [Kin74]. However, the class $\mathcal{S}_0$ of functions $f$ which, on input zero, output a program for $f$ is in $\mathbf{Mex}$ [Che82], and $\mathcal{S}_0$ is very large since, by the Kleene recursion theorem [Rog67], it contains a 1-variant of each computable function [BB75].

### 1.2 Using Notations for Constructive Ordinals As Parsimony Measures

To begin explaining the results of the present paper: a *lim-computable function* is one calculable by a total procedure allowed to change its mind finitely many times about its output.[5]

Let $\mathbf{LimMex}$ be the variant of $\mathbf{Mex}$ in which the parsimony factors $g$ can be lim-computable. It is noted in [Che82] that $\mathbf{LimMex} \subset \mathbf{Ex}$. Hence, even the parsimony restriction for which the parsimony factors are allowed to be lim-computable lowers inferring or learning power compared with no parsimony restriction. However, we see, by Corollary 33 in Section 3 below, that $\mathbf{Mex} \subset \mathbf{LimMex}$. Hence, while use of lim-computable parsimony factors lowers learning power compared with no parsimony restriction, use of lim-computable parsimony factors does not lower learning power as much as the use of computable parsimony factors. Therefore, the use of lim-computable parsimony factors partly assuages the limitation on learning power of parsimony restrictions compared with computable parsimony factors. With the use of lim-computable parsimony factors some desirable parsimony in the final programs is retained, yet learning power strictly increases over the use of computable parsimony factors.

As we described above, a (total) procedure that computes a lim-computable function is allowed to change its mind finitely many times about its output. One may, of course, consider the effects of restricting the number of mind changes of such procedures. In a different context, Putnam [Put65] studied mind change bounds $k \in N$ on the way to convergence of lim-computable characteristic functions and Ershov [Ers68a] showed these form a strict hierarchy in $k$.

Intuitively *ordinals* [Sie65, KM67] are representations of well-orderings. 0 represents the empty ordering, 1 represents the ordering of 0 by itself, 2 the ordering $0 < 1$, 3 the ordering $0 < 1 < 2$, $\ldots$ . The ordinal $\omega$ represents the standard ordering of all of $N$. $\omega + 1$ represents the ordering of $N$ consisting of the positive integers in standard order *followed by* 0. $\omega + \omega$ represents the ordering of $N$ consisting of the even numbers in standard order followed by the odd numbers in standard order. The *constructive ordinals* are just those that have a program (called a *notation*) in some system which specifies how to build them (lay them out end to end, so to speak). We will employ, as our

[3]Case, Jain and Sharma [CJS96] study the effects on learning power of placing restrictions on the size differences between successive hypotheses output by an inductive inference machine. This topic is not pursued herein.

[4]'$\subseteq$' denotes subset; $\subset$' denotes proper subset.

[5]Post [Sha71] first noticed that such functions characterize the functions computable from an oracle for the halting problem.

system of notations, Kleene's general system $O$ [Kle38, Kle44, Kle55, Rog67, Sac90]. This system has at least one notation for each constructive ordinal and comes with Kleene's standard, useful order relation $<_o$ on the notations in $O$. $<_o$ naturally embeds into the ordering of the corresponding constructive ordinals.

Everyone knows how to use finite ordinals (the natural numbers) for counting, including for counting *down*. Freivalds and Smith [FS93] employed *notations for* constructive ordinals as devices for algorithmic counting down.[6] They used such notations, for example, for algorithmic counting down the mind changes of inductive inference procedures. This allowed for handling and studying constructive, "transfinite" bounds on mind changes of inductive inference machines.

Herein we use notations $u$ from $O$ to bound the mind changes on the way to convergence of (total) procedures for lim-computable functions. The resultant functions we call $\lim_u$-computable. In Section 2.4 below we will formally define the associated $O$-countdown functions and use them to define formally the $\lim_u$-computable functions. Further below, in the current section, we present some informal examples. Of course we want to use $\lim_u$-computable functions as parsimony factors. Actually, for $u \in O$ for a positive constructive ordinal, to make the corresponding parsimony restricted inference criterion $\mathbf{Lim}_u\mathbf{Mex}$ acceptable system independent, we need to employ monotonically non-decreasing $\lim_u$-computable parsimony factors. Of course it is natural anyhow to employ non-decreasing parsimony factors. Our main results present interesting infinite hierarchies for the criteria $\mathbf{Lim}_u\mathbf{Mex}$ associated with particular transfinite subsequences of $u$'s *along certain natural* $<_o$*-paths*. Thus, with the (non-decreasing) $\lim_u$-computable functions providing degrees of parsimony, we get infinite gradations in inferring or learning power.

In the present paper, we do not study the interesting connection between the $\lim_u$-computable functions and (transfinite) Ershov hierarchies [Add65, Ers68b, Ers70, EHK81, Sel84].[7] However, in [CS03], we show how to characterize the $\lim_u$-computable functions *and variants* in terms of concepts from [EHK81, Sel84].[8]

The finite ordinals have unique notations in $O$ (see Section 2.3 below). For $n \in N$, we write $\underline{n}$ for the corresponding notation in $O$. The $\lim_{\underline{n}}$-computable functions are just those for which an associated (total) mind changing procedure makes not more than $n$ mind changes.

Let $K$ be the diagonal halting problem set [Rog67]. Then, since $K$ is c.e. [Soa96][9], the characteristic function of $K$ is $\lim_{\underline{1}}$-computable.

The first limit ordinal, $\omega$, and each constructive ordinal larger than $\omega$, have infinitely many distinct notations in $O$. Suppose $w$ is any notation in $O$ for $\omega$. The $\lim_w$-computable functions are just those for which some mind changing procedure declares, if and when it makes its first mind change, the number of *further* mind changes that it may make about its output. Visually, think of

[6] The results of [FS93] are also informally surveyed in [GS95].

Case, Jain and Suraj [CJS95] announced preliminary versions of some of the results of the present work, and herein are presented considerable improvements (and an emendation).

Subsequently to [CJS95], natural $O$-notations for constructive ordinals polynomial in $\omega$ have been extensively employed to provide natural classifications of standard problems within the context of learning grammars for formal languages [JS97, AJS99, JS99].

[7] The Ershov hierarchies are based on effective iteration of the Boolean operations including up into the constructive transfinite.

[8] While notations in $O$ as employed in [Ers68b, Ers70, EHK81, Sel84] are not directly treated as counters, in Jockusch's review [Joc82] of [EHK81] he provides the explicit intuition that a notation, as employed in [EHK81], serves as a *kind* of counter. This intuition is also explicit in [GD01].

[9] Where appropriate, we use "computably enumerable" (c.e.) and "computable" instead of "recursively enumerable" (r.e.) and "recursive", respectively, following recommendations in [Soa96].

$\omega$ as

$$0\ 1\ 2\ 3\ \dots\ n\ \dots\ )$$

where the right paren can be thought of as an "open" right hand end marker. The corresponding counter starts at the right paren (or at some number to the left of the right paren); if and when there is a first mind change, the counter algorithmically leaps to some $n$ to the left; this leaves room for at most $n$ more mind changes with the counter moving to the left.

Let $f_K$ be the function which enumerates $K$ in increasing order. Again, let $w$ be any notation in $O$ for $\omega$. Then $f_K$ is $\lim_w$-computable, but, for any $n \in N$, $f_K$ is not $\lim_{\underline{n}}$-computable. A proof is in the Appendix.

For the next examples it is useful to discuss the very basic computable operations (for $O$-notations), $+_o$, $\times_o$, and $\{\cdot\}_o$. These operations on $O$ naturally embed into the corresponding (constructive) ordinal operations of addition, multiplication, and exponentiation, respectively.[10] Just as Kleene essentially changed his definition of $+_o$ between [Kle38] and [Kle55] to obtain some auxiliary, useful properties, *we have added an extra base case to his latter definition of* $+_o$ to get useful properties we need. Our definition of $\times_o$ *also features an extra base case over what would be needed merely to get the embedding into the corresponding ordinal operation.* We did this too for technical usefulness. We guarantee thereby, for example, the technically helpful properties that, for all $y$, $\underline{0} +_o y = y$, $\underline{0} \times_o y = \underline{0}$, and $\underline{1} \times_o y = y$. In Section 2.5 below we present our "definitions" of $+_o$, $\times_o$, and $\{\cdot\}_o$ (as theorems) and present the resultant properties we need.

Suppose $w \in O$ is for $\omega$ and $k \in N$. Of course $w +_o \underline{k}$ is a notation for the ordinal $\omega + k$, which ordinal looks like a copy of the ordinal $\omega$ followed (on the right) by a copy of the ordinal $k$. The $\lim_{w+_o\underline{k}}$-computable functions are those computed by (total) procedures which may make up to $k$ mind changes before they behave like a $\lim_w$-procedure. For $w +_o w = w \times_o \underline{2}$, the corresponding ordinal, $\omega + \omega = \omega \times 2$, looks like a copy of $\omega$ followed by a copy of $\omega$, i.e, like two copies of $\omega$ laid end to end. A $\lim_{w\times_o\underline{2}}$-procedure is one which declares, if and when it makes its first mind change, the number of further mind changes that it may make about its output, *but* can later revise this number once (possibly making a mind change with the revision too). We leave to the reader to work out the equivalence with visualizing a counter leaping and walking down from the right end — in a picture of $\omega \times 2$. For $n > 2$, a $\lim_{w\times_o\underline{n}}$-procedure is similar to a $\lim_{w\times_o\underline{2}}$-procedure, except that it can revise the number of further mind changes it may make $(n-1)$ times. $w \times_o w = \{w^{\underline{2}}\}_o$, and the corresponding ordinal $\omega \times_o \omega = \omega^2$, can be visualized as $\omega$ copies of $\omega$ laid out end to end. A $\lim_{\{w^{\underline{2}}\}_o}$-procedure is one that can revise the number of further mind changes it may make "$\omega$" times; that is, it can computably pick any number as the number of such revisions it can make.[11]

Suppose $w \in O$ is for $\omega$. For each $n > 0$, we give a fairly natural example below of a function $f_n$ that is $\lim_{w^{\underline{n}}}$-computable, but not $\lim_{w^{\underline{n-1}}}$-computable.

The *pattern languages* were formally introduced by Angluin [Ang80]. A *pattern language* is (by definition) one generated by all the positive length substitution instances in a pattern, such as, *for example*,

`abXYcbbZXa`

— where the variables (for substitutions) are depicted in upper case and the constants/terminals in lower case. Since then, much work has been done on pattern languages [Sal94a, Sal94b, CJK+01]

[10]Intuitively, for $u, v \in O$, $u +_o v$ is a *program* for laying out the ordinal $|u|_o$ followed by the ordinal $|v|_o$ to form the ordinal $|u +_o v|_o$. $\times_o$, and $\{\cdot\}_o$ are for standardly iterating this action of $+_o$.

[11]This characterization of counting down from $\{w^{\underline{2}}\}_o$ very much depends on the use of *this particular notation* for $\omega^2$. See below for more on such dependence on notation for ordinals $\omega^2$ and higher.

and finite unions thereof [Shi83, Wri89, KMU95, BUV96, CJLZ99].[12]

Let $D_j$ denote the finite set with canonical index $j$ and let $W_i$ be the c.e. set which is the domain of program $i$ in our fixed acceptable system [Rog67]. Suppose $n > 0$. By careful modification of the proofs of [JS97, Theorem 1 and Corollary 1, pages 74–80], we have that, for each $n > 0$, there exists a *total* function $f_n$ such that,

(a) for each $i, j$, if $W_i$ happens to be the union of 1 to $n$ pattern languages and $f_n(i) = j$, then $D_j$ is a set of (code numbers of) 1 to $n$ patterns, the union of whose corresponding pattern languages is $W_i$, *and*,

(b) for each $w \in O$ for $\omega$, $f_n$ is $\lim_{w^{\underline{n}}}$-computable, but not $\lim_{w^{\underline{n-1}}}$-computable.

N.B. The notations featured above are *natural* ones for the corresponding ordinals — natural for those of us accustomed to thinking in terms of building ordinals from, for example, finite ordinals and $\omega$ by $+, \times$, etc. From Theorem 53 below in Section 4.2, we have, for example, a not so natural notation $v$ for $\omega^2$ such that $f_3$ from just above *is* $\lim_v$-computable! This illustrates why we emphasize that our counters for counting down are *notations for* ordinals (and not ordinals themselves).[13] On the other hand, as we note in the last paragraph of Section 4.2 below, by an obvious embedding, our Strong Hierarchy Theorem (Theorem 38 below) *mutatis mutandis* also holds for some weaker systems of notations than $O$, including non-maximal systems based on exponential polynomials in finite ordinals, $\omega$, ... . Such latter systems may have no "problematic" notations, but the corresponding hierarchies are not as extensive as those based on $O$. By Theorem 57 (also in Section 4.2) the interestingly "problematic" $O$-notations don't appear for ordinals below $\omega^2$.

### 1.3 Informal Summary of Results

By Theorems 19 and 22 in Section 3, we have that, for all $n \in N$, $\mathbf{Lim}_{\underline{n}}\mathbf{Mex} = \mathbf{Mex}$ and, for all $u, u' \in O$ and notations $w$ for $\omega$, such that $u \leq_o u' <_o u \times_o w$, $\mathbf{Lim}_{u'}\mathbf{Mex} = \mathbf{Lim}_u\mathbf{Mex}$. Hence, this suggests, for $\mathbf{Lim}_u\mathbf{Mex}$ strict hierarchies, we should examine leaps of notation from $u$ to $u \times_o w$, where $w$ is a notation for $\omega$.

One of our main results, the Strong Hierarchy Theorem (Theorem 38 in Section 3 below) implies that the collapses of Theorems 19 and 22 *are* optimal *for $u$'s along certain natural $<_o$-paths.* In particular it yields, for $v, w \in O$ where $w$ is for $\omega$,
(i) $\mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^v\}_o \times_o w}\mathbf{Mex}$, and
(ii) [$v$ is a notation for a limit ordinal $\Rightarrow (\forall u <_o v)[\mathbf{Lim}_{\{w^u\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}]$].

Hence, for example, for $w \in O$ for $\omega$, Theorem 38 implies the hierarchy shown in Figure 1.[14]

The hierarchy of Figure 1 extends, for example, to certain notations for the $\epsilon$ numbers and beyond too (see the discussion after Theorem 38 in Section 3 below).

[12] Nix [Nix83] as well as Shinohara and Arikawa [SA95] outline interesting applications of pattern inference algorithms. For example, pattern language learning algorithms have been successfully applied toward some problems in molecular biology (see [SSS+94, SA95]). Pattern languages and finite unions of pattern languages turn out to be subclasses of Smullyan's [Smu61] Elementary Formal Systems (EFSs), and Arikawa, Shinohara and Yamamoto [ASY92] show that the EFSs can also be treated as a logic programming language over strings. The investigations of the learnability of subclasses of EFSs are interesting because they yield corresponding results about the learnability of subclasses of logic programs. Hence, these results have relevance for Inductive Logic Programming (ILP) [MR94, LD94, BM95, Mit97].

[13] Of course *algorithmic* counting down would seem to require working with a finite object/program such as a notation instead of a possibly infinite object such as an ordinal.

[14] We note that by our carefully chosen definitions of $+_o$, $\times_o$, and $\{\cdot^{\cdot}\}_o$, the following example equivalences, used in this hierarchy, hold: $w +_o w = w \times_o \underline{2}$ and $\{w^w\}_o \times_o w = \{w^{w +_o \underline{1}}\}_o$.

$$
\begin{array}{lll}
\mathbf{Mex} & \subset & \mathbf{Lim}_{w}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{\underline{2}}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{\underline{n}}\}_o}\mathbf{Mex} \subset \ldots \\
& \subset & \mathbf{Lim}_{\{w^{w}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{(w+_o\underline{1})}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{(w+_o\underline{n})}\}_o}\mathbf{Mex} \subset \ldots \\
& \subset & \mathbf{Lim}_{\{w^{(w\times_o\underline{2})}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{(w\times_o\underline{2})+_o\underline{1}}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{(w\times_o\underline{2})+_o\underline{n}}\}_o}\mathbf{Mex} \subset \ldots \\
& \subset & \mathbf{Lim}_{\{w^{(w\times_o\underline{3})}\}_o}\mathbf{Mex} \subset \ldots \\
& \vdots & \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o+_o\underline{n}}\}_o}\mathbf{Mex} \subset \ldots \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o+_o w}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o+_o(w+_o\underline{n})}\}_o}\mathbf{Mex} \subset \ldots \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o+_o(w\times_o\underline{2})}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o+_o((w\times_o\underline{2})+_o\underline{n})}\}_o}\mathbf{Mex} \subset \ldots \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o+_o(w\times_o\underline{3})}\}_o}\mathbf{Mex} \subset \ldots \\
& \vdots & \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o\times_o\underline{2}}\}_o}\mathbf{Mex} \subset \ldots \\
& \vdots & \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{\underline{3}}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
& \vdots & \\
& \subset & \mathbf{Lim}_{\{w^{\{w^{w}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
& \vdots &
\end{array}
$$

Figure 1: Example optimal hierarchy along a natural $<_o$-path for ordinals $< \epsilon_0$.

Theorem 38 features notations in $O$ of the form $\{w^v\}_o$, where $w$ is for $\omega$. Our other main result, the General Hierarchy Theorem (Theorem 31 in Section 3 below) holds for *any* notations $u \in O$, but the hierarchy it specifies is not as tight as that of Theorem 38. Theorem 31 yields, for all $u \in O$, for all $w$ for $\omega$, $\mathbf{Lim}_u\mathbf{Mex} \subset \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$.

Theorem 31 implies, for example, there are infinite, infinitely ramifying hierarchies of our parsimony restricted learning criteria which lie along $<_o$-paths, which paths have notations for each constructive ordinal (see the discussion following the proof of Theorem 31).

Section 4, which presents further and preliminary results and open questions, is divided into three sections.

Section 4.1 presents some preliminary results involving parsimony factors obtained by iterating the limit taking process. Theorems 45 and 47 essentially characterize all the criteria $\mathbf{Lim}^2_{u_2,u_1}\mathbf{Mex}$, where, for the associated parsimony factors, the first limit taken is restricted to $\leq_o u_1$ mind changes, and the second limit taken is restricted to $\leq_o u_2$ mind changes. For example, by Theorem 45, for notations $w_2, w_1$ for $\omega$, $\mathbf{Lim}^2_{w_2,w_1}\mathbf{Mex} = \mathbf{Ex}$. Theorem 51 yields $\mathbf{Lim}^3_{\underline{1},\underline{2},\underline{2}}\mathbf{Mex} = \mathbf{Ex}$.

In Section 4.2, by Theorem 58, for $u, v \in O$ for the same ordinal $< \omega^2$, $\mathbf{Lim}_u\mathbf{Mex} = \mathbf{Lim}_v\mathbf{Mex}$, but, by Corollary 56, for each constructive ordinal $\alpha \geq \omega^2$, there exist notations $u, v$ for $\alpha$ such that $\mathbf{Lim}_u\mathbf{Mex} \neq \mathbf{Lim}_v\mathbf{Mex}$. It is also seen, in this same section, that if we define, for constructive ordinals $\alpha$, $\mathbf{Lim}_\alpha\mathbf{Mex} \stackrel{\text{def}}{=} \bigcup_{u \text{ for } \alpha} \mathbf{Lim}_u\mathbf{Mex}$, then there are only three distinct criteria of the form $\mathbf{Lim}_\alpha\mathbf{Mex}$, and they are in $\subset$-order: $\mathbf{Mex} \subset \mathbf{Lim}_\omega\mathbf{Mex} \subset \mathbf{Lim}_{\omega^2}\mathbf{Mex}$. This contrasts with the subtle and finely graded and ramified infinite $\mathbf{Lim}_u\mathbf{Mex}$-hierarchies based instead on $u$'s $\in O$.

In Section 4.3 we recall the definition of the criteria $\mathbf{Ex}_u$, for $u \in O$, essentially from [FS93]. This is a variant of $\mathbf{Ex}$ where the learning or inductive inference machines themselves make $\leq_o u$ mind changes. From Theorems 60 and 65, we have that $\mathbf{Mex} \not\subseteq \bigcup_{u\in O} \mathbf{Ex}_u$, and, for all $u, w \in O$ such that $w$ is for $\omega$, $\mathbf{Ex}_{w\times_o(u+_o\underline{1})} \not\subseteq \mathbf{Lim}_u\mathbf{Mex}$. However, by Theorem 62, for all $u \in O$, for all $w$ for $\omega$, $\mathbf{Ex}_u \subset \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$. Also, by Theorem 63, for $u, v, w \in O$ such that $u <_o \{w^v\}_o$ and $w$ is for $\omega$, we have, $\mathbf{Ex}_u \subset \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$.

# 2 Preliminaries

This section includes formal definitions from inductive inference, a description of notations for constructive ordinals, formal definitions for $O$-countdown functions and the $\mathbf{Lim}_u\mathbf{Mex}$ criteria mentioned above, and other necessary terminology.

## 2.1 Notation

As noted earlier, $N$ denotes the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$. $N^+$ denotes the set of positive natural numbers, $\{1, 2, 3, \ldots\}$. $i, j, k, m, n, p, q, s, t, w, x, y, z$ (with or without subscripts, superscripts, ...) range over $N$. $*$ denotes a non-member of $N$ such that $(\forall n \in N)[n < * < \infty]$ ($*$ represents 'unbounded but finite'). $u, v$, range over $N \cup \{*\}$. $X, Y, Z$, (with or without superscripts, subscripts, ...) range over subsets of $N$.

$\emptyset$ denotes the empty set. $\in, \notin, \subseteq, \subset, \supseteq, \supset$ respectively denote 'is a member of', 'is not a member of', 'is a subset of', 'is a proper subset of', 'is a superset of' and 'is a proper superset of'. $\uparrow$ denotes 'is undefined'. $\downarrow$ denotes 'is defined'.

For $S$, a subset of $N$, $\mathrm{card}(S)$ denotes the cardinality of $S$. So then, '$\mathrm{card}(S) \leq *$' means that $\mathrm{card}(S)$ is finite. $\max(S)$ and $\min(S)$ denote, respectively, the maximum and minimum of the set $S$, where $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$. $f, g, h$ and $F, G, H$ (with or without subscripts, superscripts, ...) range over *total* functions with arguments and values from $N$. We say that $g$ *dominates* $f \Leftrightarrow (\forall x)[g(x) \geq f(x)]$. $\psi$ ranges over *partial* functions with arguments and values from $N$. $\mathrm{range}(\psi)$ denotes the range of $\psi$. The set of all total computable functions of one variable is denoted by $\mathcal{R}$. $\mathcal{C}, \mathcal{S}$ (with or without subscripts, superscripts, ...) range over subsets of $\mathcal{R}$. $P$ (with or without subscripts, superscripts, ...), ranges over predicates with arguments from $N$.

$\langle \cdot, \cdot \rangle$ stands for an arbitrary, computable, one-to-one encoding of all pairs of natural numbers onto $N$ that is strictly monotonically increasing in both arguments [Rog67]. Similarly, $\langle \cdot, \ldots, \cdot \rangle$ can be used for encoding tuples of $n$ numbers onto $N$. $\pi_i^n$ denote the corresponding inverses: $\pi_i^n(\langle x_1, \ldots, x_n \rangle) = x_i$.

As in Section 1.1 above $\varphi$ denotes a fixed acceptable programming system. $W_p$ denotes the domain of $\varphi_p$. $\Phi$ denotes an arbitrary fixed Blum complexity measure for the $\varphi$-system [Blu67].[15]

We let

$$\varphi_{p,s}(x) = \begin{cases} \varphi_p(x), & \text{if } x \leq s \text{ and } \Phi_p(x) \leq s; \\ \uparrow, & \text{otherwise.} \end{cases}$$

[15] For example, if $\varphi$ is based on Turing machines, we could take $\Phi_p(x)$ to be the number of Turing machine steps Turing machine $p$ executes on input $x$ (undefined if infinite).

As defined earlier in Section 1.1, for a computable function $f$, $\mathrm{MinProg}(f) \stackrel{\text{def}}{=} \min(\{p \mid \varphi_p = f\})$.

The quantifier '$\forall^\infty$' means 'for all but finitely many'; '$\exists^\infty$' means 'there exists infinitely many'; and, '$\exists!$' means 'there exists a unique'.

## 2.2 Inductive Inference Criteria and Identification

In this section, we present relevant definitions and results from computability-theoretic learning theory.

Let $f[n]$ denote the finite initial segment $((0, f(0)), (1, f(1)), \ldots, (n-1, f(n-1)))$. Let $\mathrm{SEG} = \{f[n] \mid f \in \mathcal{R} \wedge n \in N\}$. We let $\sigma$ and $\tau$, with or without decorations, range over SEG.

A *learning machine* [Gol67, BB75, CS83] is a computable mapping from the set of all such finite initial segments of total functions: $N \to N$, into $N \cup \{?\}$.[16] Natural number outputs are interpreted as programs in the $\varphi$-system. Initially, a learning machine is allowed to output ?'s to indicate that it has not decided on its first program output yet, but once it outputs some program, it is not allowed to output ?'s again. We say that $\mathbf{M}(f)$ *converges to* $p$ (written $\mathbf{M}(f)\downarrow = p$) iff $(\forall^\infty n)[\mathbf{M}(f[n]) = p]$; $\mathbf{M}(f)$ is undefined if no such $p$ exists (written $\mathbf{M}(f)\uparrow$).

**Definition 1 (Gold [Gol67], Blum–Blum [BB75], Case–Smith [CS83])**

*(a)* $\mathbf{M}$ $\mathbf{Ex}$–identifies $f$ *(written:* $f \in \mathbf{Ex}(\mathbf{M})$*)* $\stackrel{\text{def}}{\Leftrightarrow}$ $(\exists i \mid \varphi_i = f)[\mathbf{M}(f)\downarrow = i]$.

*(b)* $\mathbf{Ex} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Ex}(\mathbf{M})]\}$.

A machine $\mathbf{M}$ that $\mathbf{Ex}$–identifies $f$ eventually converges to a program for $f$ in the $\varphi$ system. The criterion $\mathbf{Ex}$ is, however, *independent* of the acceptable programming system from which programs for functions are learned.[17] In other words, if $\psi$ is *any* acceptable programming system, and $\mathbf{Ex}_\psi$ is defined just as $\mathbf{Ex}$ is defined in Definition 1, except $\varphi$ is replaced by $\psi$ (therefore the outputs of inductive inference machines are interpreted as programs in the $\psi$ system), then $\mathbf{Ex}_\psi = \mathbf{Ex}$.

The criterion $\mathbf{Mex}$ (Definition 2 just below) is also independent of the underlying acceptable programming system [Fre75].

**Definition 2 (Freivalds [Fre75], Chen [Che82])**

*(a) Suppose $g$ is a total (not necessarily computable) monotonically non-decreasing function. Then, a learning machine* $\mathbf{M}$ $g$-$\mathbf{Mex}$–identifies $f$ *(written:* $f \in g\text{-}\mathbf{Mex}(\mathbf{M})$*)* $\stackrel{\text{def}}{\Leftrightarrow}$ $\mathbf{M}$ $\mathbf{Ex}$*–identifies $f$ and* $\mathbf{M}(f) \leq g(\mathrm{MinProg}(f))$.

*(b) A learning machine* $\mathbf{M}$ $\mathbf{Mex}$-identifies $\mathcal{S}$ $\stackrel{\text{def}}{\Leftrightarrow}$ *there exists a* computable *monotonically non-decreasing $g$ such that* $\mathcal{S} \subseteq g\text{-}\mathbf{Mex}(\mathbf{M})$.

*(c)* $\mathbf{Mex} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Mex}\text{-}\textit{identifies}\ \mathcal{S}]\}$.

In the definition just above, the $g$'s represent *parsimony factors* by which the parsimony constraints are loosened. The final programs are, in a sense, *nearly-minimal-size*. Kinber was first [Kin74, Fre75] to show that $\mathbf{Mex} \subset \mathbf{Ex}$.

Nearly minimal size program inference, as defined by $\mathbf{Mex}$, requires that the final program size be within a *computable* parsimony factor of the actual minimum program. In the present paper, we relax the computable parsimony factor constraint imposed by $\mathbf{Mex}$ and investigate whether

[16] Since an arbitrary enumeration of a *total* function $f$ can be algorithmically converted into the enumeration $(0, f(0)), (1, f(1)), \ldots$, the restriction in this definition to increasing domain order is without loss of generality.

[17] This result easily follows from the definition of $\mathbf{Ex}$ and the fact that any two acceptable programming systems are computably isomorphic [Rog58, Rog67].

learning power is thereby enhanced. One way we do so is by allowing lim-computable parsimony factors. It will be convenient to refer sometimes to lim-computable functions as $\lim_*$-computable functions — and their formal definition is handled with the next two definitions.

**Definition 3** *Suppose $h : (N \times N) \to N$. Then,*

$$\lim_{t\to\infty} h(x,t) \stackrel{\text{def}}{=} \begin{cases} y, & \textit{if } (\forall^\infty t)[h(x,t) = y]; \\ \uparrow, & \textit{otherwise.} \end{cases}$$

We write $h(x,\infty)$ for $\lim_{t\to\infty} h(x,t)$.

**Definition 4**
*(a) $g : N \to N$ is* **lim$_*$-computable** $\stackrel{\text{def}}{\Leftrightarrow}$ $(\exists$ *computable* $h : (N \times N) \to N)(\forall x)[g(x) = h(x,\infty)]$.
*(b) $g : N \to N$ is* **lim$_*$-computable as witnessed by** $h \stackrel{\text{def}}{\Leftrightarrow}$ $h$ *computable:* $(N \times N) \to N$ *and* $(\forall x)[g(x) = h(x,\infty)]$.

Intuitively, in Definition 4, $h(x,t)$ is the output at discrete time $t$ of a mind changing algorithm for $g$ (acting on input $x$). $(\forall x)[g(x) = h(x,\infty)]$, for $h$ computable, means, then, that, for all $x$, at all but finitely many times $t$, the output of the mind changing algorithm on input $x$ is $g(x)$.

It is easy to show that

$$(\exists \text{ lim-computable } g)(\forall \text{ computable } f)(\forall^\infty x)[g(x) > f(x)].$$

We can extend Definition 4 to define $\lim_*$-computable functions from $N^n \to N$, for $n > 1$. We omit the details.

**Definition 5**
*(a) A learning machine* **M** **Lim$_*$Mex**–identifies $\mathcal{S} \stackrel{\text{def}}{\Leftrightarrow}$ *there is a monotonically non-decreasing lim-computable function $g$ such that,* $\mathcal{S} \subseteq g$-**Mex**(**M**).
*(b)* **Lim$_*$Mex** $= \{\mathcal{S} \mid (\exists \mathbf{M})[\mathbf{M}$ **Lim$_*$Mex**–*identifies* $\mathcal{S}]\}$.

Hence, our definition requires that the machines converge to a program that is *not-so-nearly-minimal-size.* We mostly write **LimMex** instead of **Lim$_*$Mex**. Chen [Che82] showed that **LimMex** $\subset$ **Ex**. From Corollary 33 in Section 3, **Mex** $\subset$ **LimMex**.

In Definition 4, the underlying mind-changing algorithm is allowed to change its mind a finite, yet unbounded, number of times. One may, of course, consider the effects of restricting the number of mind changes of such algorithms.

As noted earlier, in the context of computability-theoretic learning theory, Freivalds and Smith [FS93] first introduced the use of notations for constructive ordinals to bound mind changes.

In this paper, we use notations for constructive ordinals to bound the mind changes in procedures for lim-computable functions. We will formally define, for each notation $u \in O$, the concept of a $\lim_u$-computable function below (Definition 10). We will then use (non-decreasing) $\lim_u$-computable functions as parsimony factors.

But first, we present an introduction to $O$, constructive ordinals, and needed associated properties.

### 2.3 $O$ and Constructive Ordinals

We proceed very informally. Some familiarity with a treatment of constructive ordinals such as the ones in [Rog67, Sac90] may be useful to readers of this section.

In Kleene's system $O$ [Kle38, Kle44, Kle55, Rog67, Sac90], $2^0$ is (by definition) the notation for 0. *Successor* ordinals are those with an immediate predecessor; for example, $1, 2, 3, \omega + 1, \ldots$ are successor ordinals with respective immediate predecessors $0, 1, 2, \omega, \ldots$ . If $u$ is a notation for the immediate predecessor of a successor ordinal, then a notation for that successor ordinal is (by definition) $2^u$. All other ordinals are *limit* ordinals; for example, $\omega$, $\omega + \omega$, ... are limit ordinals. Kleene [Kle38, Kle44, Kle55, Rog67, Sac90] defined a natural partial ordering of notations, $<_o$, so that two notations so ordered represent respective ordinals with the second larger than the first. We omit details. Suppose $\varphi_p(0), \varphi_p(1), \varphi_p(2), \ldots$ are each notations in $<_o$ order. Suppose that the corresponding ordinals are longer and longer initial segments of some limit ordinal which is their sup. For example, some such $p$ generates the respective notations for $0, 1, 2, \ldots$ in $<_o$ order, and $\omega$ is the sup of this sequence. In general, then, $p$ essentially describes how to build the limit ordinal which is the sup of the ordinals with notations $\varphi_p(0), \varphi_p(1), \varphi_p(2), \ldots$ . A notation for this limit ordinal is (by definition) $3 \cdot 5^p$. Clearly such limit ordinals have infinitely many such notations, different ones for different generating $p$'s. Nothing else is a notation. We define '$x =_o y$' to mean '$x, y \in O$ and $x = y$'. As in the literature on constructive ordinals, we use '$x \leq_o y$' for '$x <_o y \vee x =_o y$', '$x \geq_o y$' to mean '$y \leq_o x$ and '$x >_o y$' to mean '$y <_o x$'. We also recall the function $|\cdot|_o : O \rightarrow$ the set of ordinals, defined as follows [Kle38, Kle55, Rog67, Sac90]

$$\begin{aligned} |1|_o &= 0; \\ |2^u|_o &= |u|_o + 1; \\ |3 \cdot 5^p|_o &= \lim_{n \to \infty} |\varphi_p(n)|_o. \end{aligned}$$

For all $x, y \in O$, it is true that, if $x <_o y$ then $|x|_o < |y|_o$. It is also true that, for all $y \in O$, if $|y|_o = \beta$, then for every $\alpha < \beta$, there is an $x$ such that $x <_o y$ and $|x|_o = \beta$. If $u \in O$ and $|u|_o = \alpha$, then we say that $u$ is *for* $\alpha$. It is useful, while reading this paper, to remember that $|1|_o = 0$, $|2|_o = 1$, and $|4|_o = 2$.

**Convention 6** *If $e$ is an expression evaluating to a number $n$ in $N$, then $\underline{e}$ is the unique notation for $n$ in the $O$ notation system.*

So, for example, $\underline{0} = 1$ and $|\underline{0}|_o = 0$; $\underline{2} = 2^{2^1} = 4$ and $|\underline{2}|_o = 2$. We shall use the following properties of $<_o$ in later proofs.

**Fact 7 (Kleene [Kle55])** *For all $x, y \in N$,*
*(a) $x <_o y \Rightarrow x \in O \wedge y \in O$.*
*(b) $x \in O \Rightarrow 1 \leq_o x$.*
*(c) $x <_o y \Rightarrow y \neq 1$.*
*(d) $x <_o 2^y \Rightarrow x \leq_o y$.*
*(e) $x <_o 3 \cdot 5^p \Rightarrow (\exists n)[x <_o \varphi_p(n)]$.*
*(f) $x \leq_o z \wedge y \leq_o z \Rightarrow x <_o y \vee x =_o y \vee x >_o y$.*

The following fact about notations will also be used.

**Fact 8 (Rogers [Rog67], Sacks [Sac90])** *There exist computable functions $h_1$ and $h_2$ such that, for all $v \in O$,*
*(a) $W_{h_1(v)} = \{u \mid u <_o v\}$;*
*(b) $W_{h_2(v)} = \{\langle u_1, u_2 \rangle \mid u_1 <_o u_2 <_o v\}$ is a well-ordering isomorphic to $|v|_o$.*

### 2.4 $O$-Countdown Functions and Learning Criteria

**Definition 9** *A computable mapping* $\mathbf{F} : (N \times N) \to O$ *is an* $O$**-countdown function** $\stackrel{\text{def}}{\Leftrightarrow}$ *for all* $x$, $t$, $\mathbf{F}(x, t+1) \leq_o \mathbf{F}(x, t)$.

Intuitively, $h$ in Definition 10 just below plays a similar role to $h$ in Definition 4, and the function $\mathbf{F}$ in Definition 10 serves as a "transfinite" counter that counts down from a preset value. As before, $t$ can be thought of a discrete time paramenter. Further explanation is given just after Definition 10.

**Definition 10** *Suppose* $u \in O$. $g : N \to N$ *is* $\mathbf{lim}_u$**-computable** $\stackrel{\text{def}}{\Leftrightarrow}$ *there exists a computable function* $h : (N \times N) \to N$ *and an* $O$*-countdown function* $\mathbf{F}$*, such that, for all* $x$ *and* $t$,
*(a)* $g(x) = h(x, \infty)$,
*(b)* $\mathbf{F}(x, 0) \leq_o u$, *and*
*(c)* $h(x, t+1) \neq h(x, t) \Rightarrow \mathbf{F}(x, t+1) <_o \mathbf{F}(x, t)$.

We define *lim*$_u$*-computable as witnessed by* $\mathbf{F}$ and $h$ along the lines of part (b) of Definition 4; we omit the details.

Part (b) of Definition 10 initializes the transfinite counter at $\leq_o u$. From Fact 7(a), it follows that part (c) restricts the counter values to be notations $\in O$. By Fact 8, the notations $<_o u$ are well-ordered; hence, part (c) also implies that the counter cannot descend infinitely. Part (c) also guarantees that, when $h$ has a mind change, then the counter *must* decrement. This part does *not* restrict how much it decrements. It also allows the transfinite counter to decrement without an accompanying mind change in $h$. Note that parts (b) and (c) imply that $h(y, \infty)$ is defined and part (a) defines $g(y)$ to be the value $h(y, \infty)$. Thus, $\mathbf{F}$ acts as a countdown function that starts with a notation in $O$ and then counts down.

In Definition 11 below, we use (non-decreasing) lim$_u$-computable functions as parsimony factors. This is similar to what we did in Definition 5. When lim$_u$-computable functions are so used as parsimony factors, we shall ofter refer to them as *parsimony factors of order* $u$.

**Definition 11**
*Suppose* $u \in O$. *Then,*
*(a) A learning machine* $\mathbf{M}$ $\mathbf{Lim}_u\mathbf{Mex}$*–identifies* $\mathcal{S}$ $\stackrel{\text{def}}{\Leftrightarrow}$ *there is a monotonically non-decreasing lim*$_u$*-computable function* $g$ *such that,* $\mathcal{S} \subseteq g$-$\mathbf{Mex}(\mathbf{M})$.
*(b)* $\mathbf{Lim}_u\mathbf{Mex} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Lim}_u\mathbf{Mex}$*–identifies* $\mathcal{S}]\}$.

In Definition 11 just above, we restrict the parsimony factor, $g$, to be monotonically non-decreasing. The reason is that then (but not otherwise) the $\mathbf{Lim}_u\mathbf{Mex}$ criteria are easily shown to be independent of the underlying acceptable system. It even turns out (and is easy to show) that there is a lim$_{\underline{1}}$-computable function $g$ such that, for no $n \in N$ and for *no* lim$_{\underline{n}}$-computable, *monotonically non-decreasing* function $g'$ do we have that $g'$ dominates $g$.

### 2.5 Properties of $O$-Notations

We need the results of this section regarding addition, multiplication and exponentiation of ordinal notations; as noted above, these operations are denoted $+_o, \times_o$ and $\{\cdot\}_o$ respectively. Recall from Section 1.2 above that our $+_o$ and $\times_o$ feature extra base cases over what would be needed merely to get the embedding into the corresponding ordinal operations and that these extra base cases are to ensure some additional technically useful properties.

In unparenthesized expressions involving these operations, $\{\cdot\}_o$ has higher priority than $\times_o$ which in turn has higher priority than $+_o$.[18]

For these theorems, it is useful to recall that 1 is a notation for 0 and that 2 is a notation for 1 (i.e., $|1|_o = 0$, $|2|_o = 1$) and that 0 and 7 are not notations (for any ordinal) in the $O$ system of notations.

**Theorem 12** *There exists a computable function $+_o$ such that, for all $x, y \in N$,*

$$x +_o y = \begin{cases} y, & \textit{if } x = 1; \text{ (clause i)} \\ x, & \textit{if } y = 1 \ \wedge \ x \neq 0 \ \wedge \ x \neq 1; \text{ (clause ii)} \\ 2^{(x +_o m)}, & \textit{if } y = 2^m \ \wedge \ m \neq 0 \ \wedge \ x \neq 1; \text{ (clause iii)} \\ 3 \cdot 5^q, & \textit{if } y = 3 \cdot 5^p \ \wedge \ x \neq 1, \text{ (clause iv)} \\ & \quad (\textit{ where } (\forall n)[\varphi_q(n) = x +_o \varphi_p(n)] \ ); \\ 7, & \textit{otherwise.} \text{ (clause v)} \end{cases}$$

*Furthermore, $q$ in Clause (iv) just above is a computable, 1–1 function of $x$ and $p$.*

Proof.

The proof is an easy modification of that of [Rog67, Theorem XVII, Chapter 11, pages 209-210] and uses the 1–1 Parametric Recursion Theorem.[19] □

The 1–1-ness of $q$ in Theorem 12 is crucial for proving the right-to-left implication of part (e) of Theorem 13, which in turn is necessary for proving many results that follow.

The just above theorem is a modification of a theorem in [Kle55]. The first Clause (clause (i)) has been added in order to make $+_o$ have the useful property: for all $y$, $1 +_o y = y$ (i.e., for all $y$, $\underline{0} +_o y = y$), which is not true without the special handling of the $x = 1$ case. This property is used, for example, in the proof of Lemma 35. As in [Kle55], we treat the $x = 0$ case specially (doing so helps to ensure that $0 +_o 2 = 2^7$ (and not 1), which is essential in proving the right-to-left side of Theorem 13(a).)

**Theorem 13** *For all $x, y$ and $z \in N$,*
*(a) $x, y \in O \Leftrightarrow x +_o y \in O$.*
*(b) $x, y \in O \Rightarrow |x +_o y|_o = |x|_o + |y|_o$.*
*(c) $x, y \in O \wedge y \neq 1 \Rightarrow x <_o x +_o y$.*
*(d) $x, y \in O \wedge z <_o y \Leftrightarrow x +_o z <_o x +_o y$.*
*(e) $x \in O \wedge z =_o y \Leftrightarrow x +_o z =_o x +_o y$.*
*(f) $y \in O \ \wedge \ x \leq_o z <_o x +_o y \Leftrightarrow (\exists y')[y' <_o y \wedge x +_o y' =_o z]$.*
*Furthermore, in the left-to-right implication of (f), $y'$ is unique.*

Proof.

The proof is straightforward, but tedious, and involves careful application of transfinite induction on notations.[20] It can be found in the Appendix. □

[18] Also, we assume that each of these operators has higher priority than $<_o, \leq_o, >_o, \ldots$, which have higher priority than $\in, =, \neq$, which have higher priority than $\wedge$ and $\vee$, which in turn have higher priority than $\Rightarrow$ and $\Leftrightarrow$.

[19] Our extra base case causes no subtleties with modifying Rogers' construction.

[20] An example of a proof by transfinite induction is part of the proof of Theorem 34. See [Kle44, Kle55, Sac90] for other examples of such inductions.

We note that, similar to the case of $+$ for ordinals, $+_o$ for notations is non-commutative; $+_o$ is, however, also non-associative unlike $+$ for ordinals. We adopt the convention that $x +_o y +_o z$ means $(x +_o y) +_o z$. The non-associativity of $+_o$ leads to some subtleties while working with notations, that are otherwise absent when working with ordinals.

We next define $\times_o$.

**Theorem 14** *There is a computable function $\times_o$ such that, for all $x, y \in N$,*

$$x \times_o y = \begin{cases} 1, & \text{if } y = 1 \ \vee \ x = 1; \text{ (clause i)} \\ y, & \text{if } x = 2 \ \wedge \ y \neq 1; \text{ (clause ii)} \\ (x \times_o m) +_o x, & \text{if } y = 2^m \ \wedge \ m \neq 0 \ \wedge \\ & \quad x \neq 1 \ \wedge \ x \neq 2; \text{ (clause iii)} \\ 3 \cdot 5^q, & \text{if } y = 3 \cdot 5^p \ \wedge \ x \neq 1 \ \wedge \ x \neq 2, \text{ (clause iv)} \\ & \quad (\text{ where } (\forall n)[\varphi_q(n) = x \times_o \varphi_p(n)] \text{ }); \\ 7, & \text{otherwise. (clause v)} \end{cases}$$

*Furthermore, $q$ as in Clause (iv) is a computable, 1–1 function of $x$ and $p$.*

PROOF.

The proof is also similar to that of [Rog67, Theorem XVII, Chapter 11, pages 209-210] (and uses the 1–1 Parametric Recursion Theorem). □

The 1–1-ness of $q$ in Theorem 14 is crucial for proving the right-to-left implication of part (e) of Theorem 15, which in turn is necessary for proving many results that follow.

The first two Clauses (clauses (i) and (ii)) in the just above Theorem ensure that useful properties of $\times_o$ hold. For example: (a) for all $y$, $1 \times_o y = 1$ (i.e., for all $y$, $\underline{0} \times_o y = \underline{0}$); (b) for all $y$, $2 \times_o y = y$ (i.e., for all $y$, $\underline{1} \times_o y = y$). Property (b) just listed, is essential, for example, in proving Lemma 24 presented further below.

Similar to the case of $\times$ (usually written $\cdot$) for ordinals, $\times_o$ for notations is not commutative; however, $\times$ for ordinals is associative, but $\times_o$ for notations is not. As for $+_o$, in any unparenthesized expressions involving $\times_o$, we associate to the left.

Analogous to Theorem 13, we have the following theorem for $\times_o$.

**Theorem 15** *For all $x, y$ and $z \in N$,*
*(a)* $x, y \in O \ \vee \ x = 1 \ \vee \ y = 1 \Leftrightarrow x \times_o y \in O$.
*(b)* $x, y \in O \Rightarrow |x \times_o y|_o = |x|_o \times |y|_o$.
*(c)* $x >_o 1 \ \wedge \ y >_o 2 \Rightarrow x <_o x \times_o y$.
*(d)* $x >_o 1 \ \wedge \ y \in O \ \wedge \ z <_o y \Leftrightarrow x \times_o z <_o x \times_o y$.
*(e)* $x = 1 \ \vee \ (x \in O \ \wedge \ z =_o y) \Leftrightarrow x \times_o z =_o x \times_o y$.
*(f)* $y \in O \ \wedge \ x \leq_o z <_o x \times_o y \Leftrightarrow (\exists x' <_o x)(\exists y' \mid 1 <_o y' <_o y)[z =_o (x \times_o y') +_o x']$.
*Furthermore, in (f)$\Rightarrow$, $x'$ and $y'$ are both unique.*

PROOF.

As for $+_o$, the proof is straightforward, but tedious, and involves careful application of transfinite induction on notations. It can be found in the Appendix. □

Finally, we define exponentiation for notations in $O$ as follows.

**Theorem 16** *There is a computable function $\{\cdot^{\cdot}\}_o$ such that, for all $x, y \in N$,*

$$\{x^y\}_o = \begin{cases} 1, & \textit{if } x = 1 \ \wedge \ y \neq 1; \text{ (clause i)} \\ 2, & \textit{if } x = 2 \ \vee \\ & \quad (y = 1 \ \wedge \ x \neq 1 \ \wedge \ x \neq 2); \text{ (clause ii)} \\ \{x^m\}_o \times_o x, & \textit{if } y = 2^m \ \wedge \ m \neq 0 \ \wedge \\ & \quad x \neq 1 \ \wedge \ x \neq 2; \text{ (clause iii)} \\ 3 \cdot 5^q, & \textit{if } y = 3 \cdot 5^p \ \wedge \ x \neq 1 \ \wedge \ x \neq 2, \text{ (clause iv)} \\ & \quad (\ \textit{where } (\forall n)[\varphi_q(n) = \{x^{\varphi_p(n)}\}_o]\ ); \\ 7, & \textit{otherwise.} \text{ (clause v)} \end{cases}$$

*Furthermore, $q$ as in Clause (iv), is a computable, 1–1 function of $x$ and $p$.*

Proof.

The proof is again similar to that of [Rog67, Theorem XVII, Chapter 11, pages 209-210] (and uses the 1–1 Parametric Recursion Theorem). □

The 1–1-ness of $q$ in Theorem 16 is crucial for proving the right-to-left implication of part (e) of Theorem 17.

Similar to exponentiation for ordinals, $\{\cdot^{\cdot}\}_o$ for notations is neither commutative nor associative. We note that, in general, for $x, y \in N$, $\{x^y\}_o$ is quite different from $x^y$.

The following are some useful properties of $\{\cdot^{\cdot}\}_o$.

**Theorem 17** *For all $x, y$ and $z \in N$,*

*(a) $x, y \in O \ \wedge \ (x \neq 1 \ \vee \ y \neq 1) \Rightarrow \{x^y\}_o \in O$.*
*(b) $x, y \in O \ \wedge \ (x \neq 1 \ \vee \ y \neq 1) \Rightarrow |\{x^y\}_o|_o = |x|_o^{|y|_o}$.*
*(c) $x >_o 2 \ \wedge \ y >_o 2 \Rightarrow x <_o \{x^y\}_o$.*
*(d) $x >_o 2 \ \wedge \ z <_o y \Rightarrow \{x^z\}_o <_o \{x^y\}_o$.*
*(e) $x >_o 2 \ \wedge \ z = y \in O \Rightarrow \{x^z\}_o = \{x^y\}_o \in O$.*
*(f) $x >_o 1 \ \wedge \ x' <_o x \ \wedge \ y \in O \Rightarrow \{x^y\}_o \times_o x' <_o \{x^{(y +_o 2)}\}_o$.*

Proof.

As for $+_o$ and $\times_o$, the proof is straightforward, but tedious, and involves careful application of transfinite induction on notations. It can be found in the Appendix. □

We note that, with modifications for special cases, the right-to-left implications of parts (a), (d), (e) and (f) of Theorem 17 are true, but not necessary for this paper. So also is a property similar to Theorem 15(f).

# 3 Main Results

We recall that by Convention 6, for $n \in N$, $\underline{n}$ is the unique notation in $O$ for $n$. Clearly, we have

**Proposition 18** *For all $u, v \in O$ such that $u \leq_o v$,*
$\mathbf{Mex} = \mathbf{Lim}_{\underline{0}}\mathbf{Mex} \subseteq \mathbf{Lim}_u\mathbf{Mex} \subseteq \mathbf{Lim}_v\mathbf{Mex} \subseteq \mathbf{LimMex} \subseteq \mathbf{Ex}$.

The next Theorem shows that one does not always get increased learning power by using parsimony factors of greater orders. It shows more specifically that, when boosting the order $u$ to any order *strictly* $<_o$ the order obtained by multiplying (on the right, with $\times_o$) $u$ by some $w$ for $\omega$, we get collapse of the corresponding parsimony restricted criteria.

**Theorem 19** *For all $u, u' \in O$ and notations $w$ for $\omega$, such that $u \leq_o u' <_o u \times_o w$,* $\mathbf{Lim}_{u'}\mathbf{Mex} = \mathbf{Lim}_u\mathbf{Mex}$.

PROOF OF THEOREM 19. Follows from

**Lemma 20** *For all $u, u' \in O$ and all notations $w$ for $\omega$ such that $u' <_o u \times_o w$, every monotonically non-decreasing $\lim_{u'}$-computable function is dominated by a monotonically non-decreasing $\lim_u$-computable function.*

PROOF OF LEMMA 20. Let $u, u', w \in O$, where $|w|_o = \omega$, be such that $u' <_o u \times_o w$. Using the definition of $\times_o$ and Fact 7(e), there exists $k \in N$ such that $u' <_o u \times_o \underline{k}$. Clearly, such a $k$ must be greater than 0. Hence, it suffices to prove that for each $k > 0$, every monotonically non-decreasing $\lim_{u \times_o \underline{k}}$-computable function is dominated by a monotonically non-decreasing $\lim_u$-computable function.

Therefore, suppose $k > 0$ and $g$ is a monotonically non-decreasing $\lim_{u \times_o \underline{k}}$-computable function as witnessed by $h$ and $\mathbf{F}$. Therefore, for all $x$, $\mathbf{F}(x, 0) \leq_o u \times_o \underline{k}$. Let $k'$ be the largest integer, $0 \leq k' \leq k$, such that, for all but finitely many $x$, $\mathbf{F}(x, \infty) \geq_o u \times_o \underline{k'}$. Let $x_m$ be such that, for all $x \geq x_m$, $\mathbf{F}(x, \infty) \geq_o u \times_o \underline{k'}$.

Clearly, by our choice of $k'$ and $x_m$, there exist infinitely many $x \geq x_m$ such that $u \times_o \underline{k'} \leq_o \mathbf{F}(x, \infty) <_o u \times_o \underline{k' + 1}$. Also, using Fact 8(a), $X = \{x \geq x_m \mid u \times_o \underline{k'} \leq_o \mathbf{F}(x, \infty) <_o u \times_o \underline{k' + 1}\}$ is computably enumerable. Let $Y = \{y_0 < y_1 < \ldots\}$ be an infinite computable subset of $X$.

We now define $h'$, $\mathbf{F}'$ as follows.

For all $x$, let $h'(x, 0) = h(y_x, t')$, where $t'$ is the least number such that $\mathbf{F}(y_x, t') <_o u \times_o \underline{k' + 1}$; for $t > 0$, $h'(x, t) = h(y_x, t' + t)$.

For all $x$, let $\mathbf{F}'(x, t) = u'$, where $\mathbf{F}(y_x, t' + t) = u \times_o \underline{k'} +_o u'$ ($u'$ can be effectively found as follows: since $u \times_o \underline{k'} \leq_o \mathbf{F}(y_x, t' + t) <_o u \times_o \underline{k'} +_o u$, by Theorem 13(f), $(\exists!\, u' <_o u)[\mathbf{F}(y_x, t' + t) = u \times_o \underline{k'} +_o u']$; then, using Fact 8(a), we can effectively find this $u'$).

Let $g'(x) = h'(x, \infty)$. It is easy to verify that for all $x$, $g'(x) \geq g(x)$, and $g'$ is monotonically non-decreasing, and that $h'$ and $\mathbf{F}'$ witness that $g'$ is $\lim_u$-computable. □ (LEMMA 20)

Clearly, the theorem follows from Lemma 20. □ (THEOREM 19)

When $u$ is a notation for a finite ordinal, we can get the following result.

**Lemma 21** *For all $n \in N$, every monotonically non-decreasing $\lim_{\underline{n}}$-computable function is dominated by a monotonically non-decreasing computable function.*

PROOF.

From Lemma 20, we get that for every $n > 0$, every monotonically non-decreasing $\lim_{\underline{n}}$-computable function is dominated by a monotonically non-decreasing $\lim_{\underline{1}}$-computable function. Hence, it is sufficient to prove that every monotonically non-decreasing $\lim_{\underline{1}}$-computable function is dominated by a monotonically non-decreasing computable function.

Therefore, suppose $g$ is a monotonically non-decreasing $\lim_{\underline{1}}$-computable function as witnessed by $h$ and $\mathbf{F}$. Hence, for each $x$, there exists at most one $t$ such that $h(x, t) \neq h(x, t + 1)$.

*Case 1:* $(\forall^\infty x)(\forall t)[h(x, t) = h(x, t + 1)]$.

In this case, let $x_0$ be such that $(\forall x \geq x_0)(\forall t)[h(x, t) = h(x, t + 1)]$. Define $g'$ as follows: for all $x$, $g'(x) = h(x_0 + x, 0)$.

*Case 2:*. Not Case 1.

Let $X = \{x \mid (\exists t)[h(x,t) \neq h(x,t+1)]\}$. Let $Y = \{y_0 < y_1 < \ldots\}$ be an infinite computable subset of $X$. Define $g'$ as follows: for all $x$, $g'(x) = h(y_x, t+1)$, where $t$ is such that $h(y_x, t) \neq h(y_x, t+1)$; clearly, by the properties of $h$ and $Y$, there is exactly one such $t$ for each $y \in Y$.

Also, in both the above cases, $g'$ is computable, monotonically non-decreasing and dominates $g$. □

Hence, we get

**Theorem 22** *For all $n \in N$, $\mathbf{Lim}_{\underline{n}}\mathbf{Mex} = \mathbf{Mex}$.*

One of our main results, our Strong Hierarchy Theorem (Theorem 38 below), shows that the collapses of the previous two theorems are optimal for parsimony orders along naturally associated $<_o$-paths. Before we present our Strong Hierarchy Theorem, we present another of our main results, our General Hierarchy Theorem, as the next theorem (Theorem 31 below). To do this, we require some preliminary results.

As noted earlier, parsimony factors are always monotonically non-decreasing ($\lim_u$-computable) functions. In constructions below, we sometimes first define $\lim_u$-computable functions that are not necessarily monotonically non-decreasing, before we define appropriate parsimony factors that dominate them. Lemma 23 below gives us useful sufficient conditions to determine the orders of so obtained dominating parsimony factors.

**Lemma 23** *Suppose $u \in O$. Suppose $\{\psi_n \mid n \in N\}$ is a family of (partial) computable functions, where each $\psi_n$ is a function from $N^{n+1}$ to $N$ (which are defined when all their arguments $\leq_o u$), with the following three properties:*

*(1) For $u_n, \ldots, u_0 \leq_o u$, $\psi_n(u_n, \ldots, u_0)$ can be obtained effectively from $n$ and $u_n, \ldots, u_0$.*

*(2) For each $n$, $\psi_n$ is strictly monotonically increasing (with respect to $\leq_o$) on each of its arguments (as long as its arguments are $\leq_o u$).*

*(3) For each $n$, $\psi_n(u, \ldots, u) <_o \psi_{n+1}(u, \ldots, u)$.*

*Let $v$ be a notation for $\lim_{n\to\infty} |\psi_n(u, \ldots, u)|_o$, obtained using any computable procedure that generates, in increasing order w.r.t $<_o$, an infinite subset of $\{z \leq_o \psi_n(u, \ldots, u) \mid n \in N\}$. Then, every $\lim_u$-computable function $f$ is dominated by a monotonically non-decreasing $\lim_v$-computable function $f'$ (i.e., for all $x$, $f(x) \leq f'(x)$).*

Proof. Suppose the hypotheses. Suppose $f$ is $\lim_u$-computable as witnessed by $h$ and $\mathbf{F}$. We will define a function $f'$ bounding $f$, which is $\lim_v$-computable as witnessed by $h'$ and $\mathbf{F}'$ defined below.

Let

$$\begin{aligned} h'(x,t) &= \max(\{h(x',t) \mid x' \leq x\}) \\ \mathbf{F}'(x,t) &= \psi_{x+1}(\mathbf{F}(0,t), \mathbf{F}(1,t), \ldots, \mathbf{F}(x,t)) \\ f'(x) &= h'(x, \infty). \end{aligned}$$

It is easy to verify that $f'$, $h'$, $\mathbf{F}'$ satisfy the properties as claimed. □

**Lemma 24** *Suppose $u, w \in O$ and $w$ is for $\omega$. Then, for each $\lim_u$-computable $g$, there exists a monotonically non-decreasing $\lim_{\{(u +_o \underline{1})^w\}_o}$-computable $g'$, such that $g'$ dominates $g$.*

Proof of Lemma 24.

If $g$ is a $\lim_{\underline{0}}$-computable function, then it is clearly computable. Define $g'(x) = \max(\{g(i) \mid i \leq x\})$. Clearly, $g'$ is computable, monotonically non-decreasing, and dominates $g$. Also, for all $w$ for $\omega$, $\{(\underline{0} +_o \underline{1})^w\}_o = \underline{1}$. Since every computable function is also $\lim_{\underline{1}}$-computable, the lemma holds for $u = \underline{0}$.

For the $u > \underline{0}$ cases, we will use Lemma 23. Let $\psi_n$ be defined recursively as follows (Note: we are only interested in defining $\psi_n$ on arguments $\leq_o u$; by Fact 8(a), doing so is algorithmically possible.)

Firstly, for all $i \in N$, let

$$v_i = \{(u +_o \underline{1})^{\underline{i}}\}_o.$$

For all $u_0 \leq_o u$, let

$$\psi_0(u_0) \stackrel{\text{def}}{=} u_0.$$

For all $n > 0$, for all $u_0, \ldots, u_n \leq_o u$, let

$$\psi_n(u_n, \ldots, u_0) \stackrel{\text{def}}{=} (v_n \times_o u_n) +_o \psi_{n-1}(u_{n-1}, \ldots, u_0)$$

We note that $\psi_n(u_n, \ldots, u_0)$ is just the summation of $v_n \times_o u_n, \ldots, v_1 \times_o u_1, u_0$ in a *right associative* manner. Clearly, $\{\psi_n \mid n \in N\}$ satisfies property (1) of Lemma 23.

**Claim 25** *For all $n$, for all $u_0, \ldots, u_n \leq_o u$, we have that $\psi_n(u_n, \ldots, u_0) <_o v_{n+1}$.*

Proof of Claim 25.

We prove this Claim by induction on $n$.

*Base Case:* $n = 0$.

$\psi_0(u_0) = u_0$. Also, $v_1 = \{(u +_o \underline{1})^{\underline{1}}\}_o = u +_o \underline{1}$. Since $u_0 \leq_o u$, we get $\psi_0(u_0) <_o v_1$.

*Inductive Case:* Suppose the claim is true for $n = k$ (Inductive Hypothesis (IH)).

We will show that it holds for $n = k + 1$.

$\psi_{k+1}(u_{k+1}, \ldots, u_0) = v_{k+1} \times_o u_{k+1} +_o \psi_k(u_k, \ldots, u_0) <_o$ (by IH) $v_{k+1} \times_o u_{k+1} +_o v_{k+1} =$ (by Theorems 12 and 14) $v_{k+1} \times_o (u_{k+1} +_o \underline{1}) = \{(u +_o \underline{1})^{\underline{k+1}}\}_o \times_o (u_{k+1} +_o \underline{1})$; also, since $u_{k+1} \leq_o u$, we get[21] that $u_{k+1} +_o \underline{1} \leq_o u +_o \underline{1}$, and then, by parts (d) and (e) of Theorem 15, we get $\{(u +_o \underline{1})^{\underline{k+1}}\}_o \times_o (u_{k+1} +_o \underline{1}) \leq_o \{(u +_o \underline{1})^{\underline{k+1}}\}_o \times_o (u +_o \underline{1}) =$ (by Theorem 16) $\{(u +_o \underline{1})^{\underline{k+2}}\}_o = v_{k+2}$.

□ (Claim 25)

From Claim 25 and the definitions of $\psi_n$'s, we get for all $n$, $\psi_n(u, \ldots, u) <_o v_{n+1} \leq_o$ (since $u >_o \underline{0}$) $\psi_{n+1}(u, \ldots, u)$. Hence, $\{\psi_n \mid n \in N\}$ satisfies property (3) of Lemma 23.

**Claim 26** *Suppose $n, u_n, \ldots, u_0$ and $u'_n, \ldots, u'_0$ are given such that for $i \leq n$, each $u_i, u'_i \leq_o u$, and, there exists a $j \leq n$, such that*

*(a) for all $j'$, $j < j' \leq n$, $u_{j'} = u'_{j'}$.*

*(b) $u_j <_o u'_j$.*

*Then $\psi_n(u_n, \ldots, u_0) <_o \psi_n(u'_n, \ldots, u'_0)$.*

[21] Proof: If $u_{k+1} = u$, then $u_{k+1} +_o \underline{1} = 2^{u_{k+1}} = 2^u = u +_o \underline{1}$; otherwise, if $u_{k+1} <_o u$, then using the contrapositive of Fact 7(d), we get $u_{k+1} +_o \underline{1} \leq_o u$, which, by Theorem 13(c), is $<_o u +_o \underline{1}$. Therefore, $u_{k+1} +_o \underline{1} \leq_o u +_o \underline{1}$.

PROOF OF CLAIM 26. Suppose the hypotheses.

To show the claim, it suffices to show that for all $j \leq n$, $\psi_j(u_j, \ldots, u_0) <_o (v_j \times_o u'_j)$. For then, since $u_j = u'_j$ for all $j'$ such that $j < j' \leq n$, we get

$$\begin{aligned}
\psi_n(u'_n, \ldots, u'_0) &=_o ((v_n \times_o u'_n) +_o (\ldots +_o ((v_{j+1} \times_o u'_{j+1}) +_o (v_j \times_o u'_j)) \ldots)) \\
&=_o ((v_n \times_o u_n) +_o (\ldots +_o ((v_{j+1} \times_o u_{j+1}) +_o (v_j \times_o u'_j)) \ldots)) \\
&>_o ((v_n \times_o u_n) +_o (\ldots +_o ((v_{j+1} \times_o u_{j+1}) +_o \psi_j(u_j, \ldots, u_0)) \ldots)) \\
&=_o \psi_n(u_n, \ldots, u_0).
\end{aligned}$$

We proceed with the proof of Claim 26. If $j = 0$, then $\psi_0(u_0) = u_0 <_o u'_0$. Also, by Theorem 16 we have that $v_0 = \underline{1}$. Furthermore, since we ensured that for all $y$, $\underline{1} \times_o y = y$ (as noted a few paragraphs after Theorem 14), we get $u'_0 = (v_0 \times_o u'_0)$. Hence, $\psi_0(u_0) <_o (v_0 \times_o u'_0)$.

If $j > 0$, then we have

$$\begin{aligned}
\psi_j(u_j, \ldots, u_0) &=_o (v_j \times_o u_j) +_o \psi_{j-1}(u_{j-1}, \ldots, u_0) \\
&<_o (v_j \times_o u_j) +_o v_j && \text{(by Claim 25)} \\
&=_o (v_j \times_o (u_j +_o \underline{1})) && \text{(by Theorem 14)} \\
&\leq_o (v_j \times_o u'_j). && \text{(by Fact 7(d), Theorem 15)}
\end{aligned}$$

□ (CLAIM 26)

Thus $\psi_n$ is monotonically increasing in each of its arguments (as long as they are $\leq_o u$). Hence, property (2) of Lemma 23 is satisfied by $\{\psi_n \mid n \in N\}$. We note that, for all $n$, $\psi_n(u, \ldots, u) <_o \{(u +_o \underline{1})^{\underline{n+1}}\}_o <_o$ (by Theorem 17(d)) $\{(u +_o \underline{1})^w\}_o$, for any $w$ for $\omega$. Also, for all $n$, $\lim_{n\to\infty} |\psi_n(u, \ldots, u)|_o = (|u +_o \underline{1}|_o)^\omega$. Hence, taking $v = \{(u +_o \underline{1})^w\}_o$ in Lemma 23, Lemma 24 follows. □ (LEMMA 24)

We next define, for each $u \in O$, a useful, self-referential class of computable functions, $\mathcal{C}_u$, that helps to establish our Hierarchy Theorems: $\mathcal{C}_u$ *is* learnable with parsimony factors of order $\{(u +_o \underline{2})^w\}_o$, for any $w$ for $\omega$ (Lemma 29); however, $\mathcal{C}_u$ is *not* learnable with parsimony factors of order $u$ (Lemma 30).

We recall that $\langle \cdot, \cdot, \cdot, \cdot \rangle$ denotes a computable bijection from $N \times N \times N \times N$ onto $N$ and that $\pi_i^4$, $1 \leq i \leq 4$, denote the corresponding projection functions (i.e $\pi_i^4(\langle x_1, x_2, x_3, x_4 \rangle) = x_i$).

**Definition 27** *Suppose $u \in O$. Then,*

$\mathcal{C}_u = \{f \mid$ *for some* $p$

*(a)* $\lim_{x\to\infty} \pi_1^4(f(x))\downarrow = p \wedge \varphi_p = f$, *and*

*(b)* $\lim_{x\to\infty} \pi_2^4(f(x))\downarrow \geq p$, *and*

*(c)* $\pi_3^4(f(0)) \leq_o u$, *and*

*(d)* $(\forall x)[\pi_3^4(f(x+1)) \leq_o \pi_3^4(f(x))]$, *and*

*(e)* $(\forall x)[\pi_2^4(f(x)) \neq \pi_2^4(f(x+1)) \Rightarrow \pi_3^4(f(x+1)) <_o \pi_3^4(f(x))]$.

$\}$

It is helpful to note the following facts about the just above definition. Values in the range of each $f \in \mathcal{C}_u$ are interpreted as quadruples; on inputs $0, 1, 2, \ldots$ to $f$, the first elements of these quadruples may vary but they eventually settle down at some number, say $p$; the second elements settle down at a number $\geq p$; the third elements form a non-increasing (w.r.t. $\leq_o$) chain of notations in $O$ constrained to decrease whenever there is a mind change in the corresponding second elements; finally, the fourth elements of the quadruple are completely unrestricted (which freedom we use in the diagonalization in step 4 of the proof of Lemma 30).

**Lemma 28** *For every $u \in O$, there exists a $lim_{u+_o\underline{1}}$-computable $g$ (which is not necessarily monotonically non-decreasing) and a machine $\mathbf{M}$ such that,*
*(a) $\mathbf{M}$ $\mathbf{Ex}$-identifies $\mathcal{C}_u$ and*
*(b) for all $\varphi_i \in \mathcal{C}_u$, $\mathbf{M}(\varphi_i) \leq g(i)$.*

Proof. Let $\mathbf{M}$ be defined as follows. $\mathbf{M}(f[0]) = 0$; for $n > 0$, $\mathbf{M}(f[n]) = \pi_1^4(f(n-1))$.
Let $h$ and $\mathbf{F}$ be defined as follows.
For all $i$, let $h(i, 0) = 0$, $\mathbf{F}(i, 0) = u +_o \underline{1}$.
For all $i, t$, we define $h(i, t+1), \mathbf{F}(i, t+1)$ as follows.

**begin** computation of $h(i, t+1), \mathbf{F}(i, t+1)$.
  **if** $\Phi_i(0) > t$
    **then**
      Let $h(i, t+1) = h(i, t)$ and $\mathbf{F}(i, t+1) = \mathbf{F}(i, t)$;
    **else**
      Let $x$ be the largest value $\leq t$, such that $(\forall x' \leq x)[\Phi_i(x) \leq t]$.
      Enumerate $S = \{u' \mid u' <_o \mathbf{F}(i, t)\}$ for $t$ steps using a fixed enumerator.
        (This is possible since $S$ is uniformly c.e. in $\mathbf{F}(i, t)$, by Fact 8(a).)
      **if** $\pi_3^4(\varphi_i(x)) \in S$ within $t$ steps of the fixed enumeration above
        **then**
          Let $h(i, t+1) = \pi_2^4(\varphi_i(x))$ and $\mathbf{F}(i, t+1) = \pi_3^4(\varphi_i(x))$;
        **else**
          Let $h(i, t+1) = h(i, t)$ and $\mathbf{F}(i, t) = \mathbf{F}(i, t)$.
      **endif**
  **endif**
**end** computation of $h(i, t+1), \mathbf{F}(i, t+1)$.

Let $g(i) = h(i, \infty)$. It is easy to verify that $g$ and $\mathbf{M}$ satisfy the requirements of the theorem. □

Now, given any $u \in O$, by Lemma 28, there exists a $\lim_{u+_o\underline{1}}$-computable function $g$ and a machine $\mathbf{M}$ which $\mathbf{Ex}$-identifies $\mathcal{C}_u$ such that, for all $i$, $\varphi_i \in \mathcal{C}_u$ implies that $\mathbf{M}(\varphi_i) \leq g(\mathrm{MinProg}(\varphi_i))$. This $g$ is not necessarily monotonically non-decreasing. However, by noting that the Definition of $+_o$ gives $(u +_o \underline{1}) +_o \underline{1} = u +_o \underline{2}$, and then using Lemma 24, we get that there exists a monotonically non-decreasing $\lim_{\{(u+_o\underline{2})^w\}_o}$-computable function $g'$ such that $g'$ dominates $g$. Hence, we get the following.

**Lemma 29** *For all $u \in O$, for all $w$ for $\omega$, $\mathcal{C}_u \in \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$.*

**Lemma 30** $(\forall u \in O)[\mathcal{C}_u \notin \mathbf{Lim}_u\mathbf{Mex}]$.

PROOF. Suppose by way of contradiction that $\mathcal{C}_u \subseteq g$-$\mathbf{Mex}(\mathbf{M})$, where $g$ is a monotonically non-decreasing $\lim_u$-computable function as witnessed by $h$ and $\mathbf{F}$.

Then by the Operator Recursion Theorem [Cas74, Cas94], there exists a 1–1 increasing function $p$ such that the (partial) functions $\varphi_{p(\cdot)}$ may be defined as follows. Intuitively, while defining these functions, we attempt to define $\varphi_{p(0)}$ such that $\varphi_{p(0)} \in \mathcal{C}_u$, and, either (a) $\mathbf{M}(\varphi_{p(0)})\downarrow$ and $\mathrm{MinProg}(\varphi_{p(0)})$ is such that $g(\mathrm{MinProg}(\varphi_{p(0)})) < \mathbf{M}(\varphi_{p(0)})$, or (b) $\mathbf{M}(\varphi_{p(0)})\uparrow$; failing this, we will ensure that for some $i > 0$, $\varphi_{p(i)} \in \mathcal{C}_u$ and either (a) $\mathbf{M}(\varphi_{p(i)})\downarrow$ and $\varphi_{\mathbf{M}(\varphi_{p(i)})} \neq \varphi_{p(i)}$, or $\mathbf{M}(\varphi_{p(i)})\uparrow$.

The procedure uses global variables cur_bound, cur_notation, cur_maxindex, cur_index, Cancel, and last_mindchange. Then $\mathrm{cur\_bound}_s$, $\mathrm{cur\_notation}_s$, $\mathrm{cur\_maxindex}_s$, $\mathrm{cur\_index}_s$, $\mathrm{Cancel}_s$, and $\mathrm{last\_mindchange}_s$ denote the values of these variables at the start of stage $s$.

Initially, we let $\mathrm{cur\_bound}_0 = h(p(0), 0)$, $\mathrm{cur\_notation}_0 = u$, $\mathrm{cur\_maxindex}_0 = \mathrm{cur\_bound}_0 + 2$. $\mathrm{cur\_index}_0 = 1$. $\mathrm{Cancel}_0 = \emptyset$, $\mathrm{last\_mindchange}_0 = 0$.

$x_s$ denotes the least $x$ such that $\varphi_{p(0)}(x)$ has not been defined at the start of stage $s$. Thus $x_0 = 0$. Go to stage 0.

Begin stage $s$.

1. For $x < x_s$, let $\varphi_{p(\mathrm{cur\_index}_s)}(x) = \varphi_{p(0)}(x)$.
2. Let $y = x_s$.

   repeat

   2.1. Let $\varphi_{p(\mathrm{cur\_index}_s)}(y) = \langle p(\mathrm{cur\_index}_s), p(\mathrm{cur\_maxindex}_s), \mathrm{cur\_notation}_s, 0\rangle$.

   2.2. If $h(p(0), \mathrm{last\_mindchange}_s) \neq h(p(0), y)$, then go to step 3.

   2.3. If there exists $i \leq \mathrm{cur\_bound}_s$, $i \notin \mathrm{Cancel}_s$, and $z$, such that $x_s < z \leq y$ and $\Phi_i(z) \leq y$, then go to step 4.

   2.4. If $\mathbf{M}(\varphi_{p(\mathrm{cur\_index}_s)}[y+1]) > \mathrm{cur\_bound}_s$, then go to step 5.

   2.5. Let $y = y + 1$.

   forever
3. For $x \leq y$, let $\varphi_{p(0)}(x) = \varphi_{p(\mathrm{cur\_index}_s)}(x)$.

   Decrement notation: $\mathrm{cur\_notation}_{s+1} = \mathbf{F}(p(0), y)$.

   Change current bound: $\mathrm{cur\_bound}_{s+1} = h(p(0), y)$.

   Change current maximum index: $\mathrm{cur\_maxindex}_{s+1} = \mathrm{cur\_maxindex}_s + \mathrm{cur\_bound}_{s+1} + 2$.

   Record where the last mind change happened: $\mathrm{last\_mindchange}_{s+1} = y$.

   Carry forward unchanged all other global variables (cur_index and Cancel).

   Go to stage $s + 1$. (Therefore $x_{s+1} = y + 1$.)
4. Let $k \in \{0, 1\}$ such that

   $\varphi_i(z) \neq \langle p(\mathrm{cur\_index}_s), p(\mathrm{cur\_maxindex}_s), \mathrm{cur\_notation}_s, k\rangle$. For $x_s \leq x \leq y$, let $\varphi_{p(0)}(x) = \langle p(\mathrm{cur\_index}_s), p(\mathrm{cur\_maxindex}_s), \mathrm{cur\_notation}_s, k\rangle$.

   Change to next unused index: $\mathrm{cur\_index}_{s+1} = \mathrm{cur\_index}_s + 1$.

   Record diagonalized program: $\mathrm{Cancel}_{s+1} = \mathrm{Cancel}_s \cup \{i\}$.

   Carry forward unchanged all other global variables (cur_notation, cur_bound, cur_maxindex, and last_mindchange).

   Go to stage $s + 1$. (Therefore $x_{s+1} = y + 1$.)

5. For $x_s \leq x \leq y$, let $\varphi_{p(0)}(x) = \varphi_{p(\text{cur\_index}_s)}(x)$.
   Carry forward unchanged all global variables (cur_notation, cur_bound, cur_maxindex, last_mindchange, cur_index, and Cancel).
   Go to stage $s+1$. (Therefore $x_{s+1} = y+1$.)

End stage $s$.

We now consider the following cases.
*Case 1*: Each stage is entered and terminates.

Since ordinals are well ordered, step 2.2 can succeed only finitely often. Thus, (as $s$ goes to infinity) cur_notation$_s$, cur_bound$_s$, cur_maxindex$_s$, last_mindchange$_s$ all stabilize to, say, cur_notation, cur_bound, cur_maxindex, last_mindchange. Now after the last success of step 2.2, step 2.3 can succeed only finitely often ($\leq$ cur_bound$+1$ times, since each success of step 2.3 cancels a new program $\leq$ cur_bound). Thus for all but finitely many stages, step 2.4 must succeed. It follows, from steps 1 and 5, that $\varphi_{p(\text{cur\_index})} = \varphi_{p(0)}$ and $\mathbf{M}(\varphi_{p(\text{cur\_index})})$ outputs a program $> h(p(0), \infty) = g(p(0))$ infinitely often.

Clearly, $\varphi_{p(0)} \in \mathcal{C}_u$. Also, since $g$ is monotonically non-decreasing, $g(p(0)) \geq g(\text{MinProg}(\varphi_{p(0)})) = g(\text{MinProg}(\varphi_{p(\text{cur\_index})}))$. Therefore, either $\mathbf{M}(\varphi_{p(0)})\uparrow$ or $\mathbf{M}(\varphi_{p(0)})\downarrow \not\leq g(\text{MinProg}(\varphi_{p(0)}))$.

Thus $\varphi_{p(0)} \in \mathcal{C}_u - g\text{-}\mathbf{Mex}(\mathbf{M})$.
*Case 2*: Stage $s$ is entered but does not terminate.

In this case $\varphi_{p(\text{cur\_index}_s)} \in \mathcal{C}_u$, but $\mathbf{M}(\varphi_{p(\text{cur\_index}_s)})$ either does not converge, or converges to a program $\leq$ cur_bound$_s$. However each of the programs $\leq$ cur_bound$_s$ either convergently differs from $\varphi_{p(\text{cur\_index}_s)}$ (i.e., is in Cancel$_s$) or is undefined on all $x > x_s$ (since step 2.3 does not succeed). Thus, $\mathbf{M}$ does not $g$-$\mathbf{Mex}$-identify $\varphi_{p(\text{cur\_index}_s)}$ (In fact, in this case, $\mathbf{M}$ does not even $\mathbf{Ex}$-identify $\varphi_{p(\text{cur\_index}_s)}$). □

**Theorem 31 (General Hierarchy Theorem)** *For all $u \in O$, for all $w$ for $\omega$,* $\mathbf{Lim}_u\mathbf{Mex} \subset \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$.

Proof. By Theorem 13(c), we get that for all $u$, $u <_o (u +_o \underline{2})$, which, by Theorem 17(c), is $<_o \{(u +_o \underline{2})^w\}_o$ (we recall that $\underline{2} = 2^{2^1} = 4 >_o 2$). Hence, the theorem follows from Proposition 18 and Lemmas 30 and 29. □

Theorem 31 implies that there are infinite, infinitely ramifying hierarchies of parsimony restricted learning criteria. Furthermore, there are infinite hierarchies of parsimony restricted learning criteria which lie along $<_o$-paths, which paths have notations for each constructive ordinal. To see this, first fix $w \in O$ for $\omega$. Let $O = \{x_0, x_1, x_2, \ldots\}$. Let $y_0 = z_0 = x_0$. Let $y_{n+1} = \{(z_n +_o \underline{2})^w\}_o$. Let $z_{n+1} = y_{n+1} +_o x_{n+1}$. Then, by Theorem 13 parts (c) and (e), for all $n$, $y_n <_o (y_n +_o x_n) +_o \underline{2}$. This latter, by Theorem 17(c), is $<_o \{((y_n +_o x_n) +_o \underline{2})^w\}_o$, which $= y_{n+1}$. Hence, the downward closure under $<_o$ of the set of $O$-notations, $\{y_0 <_o y_1 <_o y_2 <_o \cdots\}$, is, for terminology from [Rog67], a maximal, univalent system of notations, and, from Proposition 18 and Theorem 31, $(\forall n \in N)[\mathbf{Lim}_{y_n}\mathbf{Mex} \subset \mathbf{Lim}_{y_{n+1}}\mathbf{Mex}]$.

As a Corollary to Theorem 31 and Proposition 18, we get

**Corollary 32** *For all $u \in O$,* $\mathbf{Lim}_u\mathbf{Mex} \subset \mathbf{LimMex}$.

Since $\mathbf{Lim}_{\underline{0}}\mathbf{Mex} = \mathbf{Mex}$, we get

**Corollary 33** $\mathbf{Mex} \subset \mathbf{LimMex}$.

Our next few results (Theorem 34, Lemma 35 and Lemma 37) lead up to another of our main results, our Strong Hierarchy Theorem (Theorem 38), which essentially states that for notations of the form $\{w^v\}_o$, where $v \in O$ and $w$ is for $\omega$, Theorem 19 gives as much collapsing as possible.

The following Theorem is *a* notational analog of Cantor's Normal Form (CNF) Theorem[22] for ordinals ([Sie65, Theorem 2, Chapter XIV.19, page 323] and [KM67, Theorems 2 and 5, Chapter VII, Section 7]).[23] In Theorem 34 below, as stated, the parenthesization of the $+_o$ terms to the right is essential, since $+_o$ is not associative.

**Theorem 34 (Notational CNF Theorem)** *For all $v \in O$, for all $w$ for $\omega$, for all $x >_o \underline{0}$,*

$$x <_o \{w^v\}_o \Leftrightarrow \begin{cases} \text{there exists a unique } k \in N, \text{ unique } n_0, n_1, \ldots, n_k \in N^+, \\ \text{and unique } v_0, v_1, \ldots, v_k \text{ where } v >_o v_0 > v_1 \ldots >_o v_k, \text{ such that} \\ x = \{w^{v_0}\}_o \times_o \underline{n_0} +_o (\{w^{v_1}\}_o \times_o \underline{n_1} +_o (\ldots +_o (\{w^{v_k}\}_o \times_o \underline{n_k}) \ldots)). \end{cases}$$

*Furthermore, for the left to right direction of the $\Leftrightarrow$ statement just above, the values of $k, n_0, \ldots, n_k$, and $v_0, \ldots, v_k$ can be algorithmically obtained from $v, w$, and $x$.*

We call the above unique representation of $x <_o \{w^v\}_o$, for $v$, $w$, and $x$ as in the above theorem, the *notational CNF of $x$ with respect to $\{w^v\}_o$*. Where it is clear from the context, we will drop the phrase "with respect to $\{w^v\}_o$" while referring to the notational CNF of such an $x$. Also, Theorem 34 above has several other variants that are also true; for example, we could get similar theorems for notations of the form $w_1 \times_o w_2 \times_o w_3$, or $\{w_1^{w_2}\}_o \times_o w_3$, where $w_1, w_2$ and $w_3$ are notations for $\omega$. We omit the details.

Furthermore, we note that for every ordinal $\alpha > 0$, there exists a (unique) ordinal $\beta$ such that $\omega^\beta \leq \alpha < \omega^{\beta+1}$ [Sie65, Theorem 2, Chapter XIV.18, page 321]. Therefore, the above Notational CNF Theorem is sufficiently general to apply to notations for as large a constructive ordinal as we may choose.

Proof of Notational CNF.

We first prove the '$\Leftrightarrow$' part of the theorem.

($\Leftarrow$) This direction of the proof uses reasoning and facts very similar to those used in the proofs of Claims 25 and 26 of Lemma 24. It uses the fact that, for all $w$ for $\omega$, and for all $n \in N$, for all $v', v'' \in O$, $v' <_o v'' \Rightarrow \{w^{v'}\}_o \times_o \underline{n} <_o \{w^{v''}\}_o$.

($\Rightarrow$) The proof is by transfinite induction on $v$. Suppose $v, w \in O$ such that $w$ is for $\omega$. Suppose $x \in O$ such that $\underline{0} <_o x <_o \{w^v\}_o$.

*Base Case:* $v = \underline{0}$. Since $\{w^{\underline{0}}\}_o = \underline{1}$, this case is vacuously true.

*Inductive Case:* $v >_o \underline{0}$.

*Inductive Hypothesis (IH):* The ($\Rightarrow$) direction holds for all notations $<_o v$.

We will show that the ($\Rightarrow$) direction holds for $v$.

Subcase 1: $v = u +_o \underline{1}$.

[22] The CNF Theorem states: for any ordinal $\beta > 0$, there exists a unique $k$, unique $n_0, n_1, \ldots, n_k \in N^+$, and unique ordinals $\alpha_0, \alpha_1, \ldots, \alpha_k$, where $\alpha_0 > \alpha_1 > \ldots > \alpha_k$, such that $\beta = \omega^{\alpha_0} \times n_0 + \omega^{\alpha_1} \times n_1 + \ldots + \omega^{\alpha_k} \times n_k$.

[23] For a web page describing a programmatic implementation of an algorithm that may be used to perform operations such as addition, subtraction, etc., on ordinals represented in Cantor Normal Form, see [Beh97].

Clearly, either $x <_o \{w^u\}_o$ or $\{w^u\}_o \leq_o x <_o \{w^{u+_o\underline{1}}\}_o$. If the former is true, the ($\Rightarrow$) direction follows from IH. Therefore, suppose the latter. Then, by Theorem 15(f), there exists a unique $n \in N^+$ and a unique $x' <_o \{w^u\}_o$ such that $x = \{w^u\}_o \times_o \underline{n} +_o x'$. If $x' = \underline{0}$, then clearly the ($\Rightarrow$) direction holds with $k = 0$, $v_0 = u$, and $n_0 = n$; otherwise, if $x >_o \underline{0}$, the ($\Rightarrow$) direction follows by applying IH to $x'$ and noting that $v >_o u$.

Subcase 2: $v = 3 \cdot 5^p$.

Let $\{w^v\}_o = 3 \cdot 5^q$. Therefore, by Fact 7(e), $(\exists n \in N)[x <_o \varphi_q(n)]$. From Theorem 16, $\varphi_q(n) = \{w^{\varphi_p(n)}\}_o$, and since $\varphi_p(n) <_o v$, the ($\Rightarrow$) direction follows using IH. □ ('$\Leftrightarrow$' PART)

Finally, we prove the 'furthermore' part of the theorem. Suppose $v, w \in O$ such that $w$ is for $\omega$. Suppose $x \in O$ such that $\underline{0} <_o x <_o \{w^v\}_o$. By using Fact 8(a) we can enumerate, uniformly effectively in $v$ and $w$, the set $S = \{(k, n_0, n_1, \ldots, n_k, v_0, v_1, \ldots, v_k) \mid k \in N, n_0, n_1, \ldots, n_k \in N^+, v >_o v_0 >_o v_1 \ldots >_o v_k$ and $\{w^{v_0}\}_o \times_o \underline{n_0} +_o (\{w^{v_1}\}_o \times_o \underline{n_1} +_o (\ldots +_o (\{w^{v_k}\}_o \times_o \underline{n_k}) \ldots)) <_o \{w^v\}_o\}$. Clearly, then, there is a procedure that is effective in $v, w$, and $x$ to find $(k, n_0, n_1, \ldots, n_k, v_0, v_1, \ldots, v_k) \in S$ such that $x = \{w^{v_0}\}_o \times_o \underline{n_0} +_o (\{w^{v_1}\}_o \times_o \underline{n_1} +_o (\ldots +_o (\{w^{v_k}\}_o \times_o \underline{n_k}) \ldots))$.

□ (NOTATIONAL CNF THEOREM)

We next present a lemma (Lemma 35) which is a strengthening of Lemma 24 for notations in certain forms. In Lemma 24, if we let $u = \{w^v\}_o$, for some notation $v \in O$ and some $w$ for $O$, then we get that every $\lim_u$-computable function is dominated by a parsimony factor of order $\{(\{w^v\}_o +_o \underline{1})^w\}_o$. Lemma 35, however, implies that we can in fact do much better and use parsimony factors of order just $\{w^{v+_o\underline{1}}\}_o$.

**Lemma 35** *Suppose $u <_o \{w^v\}_o$, where $w$ is for $\omega$. Then, every $\lim_u$-computable function is dominated by a monotonically non-decreasing $\lim_{\{w^v\}_o}$-computable function.*

PROOF OF LEMMA 35.

Suppose $v, w \in O$ such that $w$ is for $\omega$. If $u = \underline{0}$, then the lemma follows using reasoning similar to that in the proof of Lemma 24. For the $u >_o \underline{0}$ cases, we will use Lemma 23.

Firstly, for all notations $x, y <_o \{w^v\}_o$, we define the *natural sum* of $x$ and $y$ and denote the operation by $(+)_o$; this operation is a notational analog of the *natural sum of ordinals* from [KM67, Chapter VII, Section 7, pages 259-260].[24]

Suppose $x, y >_o \underline{0}$. Let $x$'s and $y$'s (unique) notational CNFs, obtained via the Notational CNF Theorem 34, be

$$\begin{aligned} x &= \{w^{v'_0}\}_o \times_o \underline{n'_0} +_o (\{w^{v'_1}\}_o \times_o \underline{n'_1} +_o (\ldots +_o (\{w^{v'_{k'}}\}_o \times_o \underline{n'_{k'}}) \ldots)), \\ y &= \{w^{v''_0}\}_o \times_o \underline{n''_0} +_o (\{w^{v''_1}\}_o \times_o \underline{n''_1} +_o (\ldots +_o (\{w^{v''_{k''}}\}_o \times_o \underline{n''_{k''}}) \ldots)), \end{aligned}$$

where $k', k'' \in N$, $n'_0, \ldots, n'_{k'}, n''_0, \ldots, n''_{k''} \in N^+$, $v >_o v'_0 \ldots >_o v'_{k'}$, and $v >_o v''_0 \ldots >_o v''_{k''}$. (We note that in general, $k'$ may be different from $k''$.)

By Fact 7(f), we get that for all $v' \in V' = \{v'_i \mid i \leq k'\}$, for all $v'' \in V'' = \{v''_i \mid i \leq k''\}$, either $v' <_o v''$ or $v' =_o v''$ or $v' >_o v''$. Let $v_0 >_o v_1 \ldots >_o v_k$ be the notations in $V' \cup V''$.

By Theorem 14 (clause i), we have that for all $x$, $x \times_o \underline{0} = \underline{0}$. Also, as we noted a few paragraphs after Theorem 12, clause (i) of Theorem 12 is a special base case that we added to Kleene's definition

[24] N.B. our notational $(+)_o$ depends on the choice of the pre-given $v$ and $w$.

of $+_o$ from [Kle55], in order to get that for all $y$, $\underline{0} +_o y = y$. Thus, we have that for any $x, y, \in O$, $(x \times_o \underline{0}) +_o y = y$.

Using the facts from the just above paragraph, then, there is exactly one way to write $x$ and $y$ as

$$\begin{aligned} x &= \{w^{v_0}\}_o \times_o \underline{m'_0} +_o (\{w^{v_1}\}_o \times_o \underline{m'_1} +_o (\ldots +_o (\{w^{v_k}\}_o \times_o \underline{m'_k}) \ldots)), \\ y &= \{w^{v_0}\}_o \times_o \underline{m''_0} +_o (\{w^{v_1}\}_o \times_o \underline{m''_1} +_o (\ldots +_o (\{w^{v_k}\}_o \times_o \underline{m''_k}) \ldots)), \end{aligned}$$

where $m'_0, \ldots, m'_k$ and $m''_0, \ldots, m''_k$ are possibly 0, and $v >_o v_0$. Furthermore, we clearly can still effectively determine all notations involved in the above representations of $x$ and $y$. We call the immediately above forms of $x$ and $y$ the *mutually completed notational CNFs of $x$ and $y$*. It is similarly possible to extend the definition of mutual completion of notational CNFs to sequences of more than two notations. We omit the details.

Now, the *natural sum* of $x$ and $y$: If $x = \underline{0}$ or $y = \underline{0}$,

$$x \,(+)_o\, y \stackrel{\text{def}}{=} x +_o y;$$

Otherwise, $x \,(+)_o\, y \stackrel{\text{def}}{=}$

$$\{w^{v_0}\}_o \times_o \underline{(m'_0 + m''_0)} +_o (\{w^{v_1}\}_o \times_o \underline{(m'_1 + m''_1)} +_o (\ldots +_o (\{w^{v_k}\}_o \times_o \underline{(m'_k + m''_k)}) \ldots)).$$

Clearly, $(+)_o$ is an effective operation. It is also commutative and associative, *unlike* $+_o$. Next, we prove the following properties of $(+)_o$ .

**Claim 36** *Suppose $v, x, y, w \in O$ such that $w$ is for $\omega$. Suppose $x, y <_o \{w^v\}_o$. Then,*
(a) $x \,(+)_o\, y <_o \{w^v\}_o$,
(b) $(\forall y')[y' <_o y \Rightarrow (x \,(+)_o\, y' <_o x \,(+)_o\, y)]$,
(c) $(\forall x')[x' <_o x \Rightarrow (x' \,(+)_o\, y <_o x \,(+)_o\, y)]$.

PROOF OF CLAIM 36.

The proof is fairly straightforward, and, hence, we only sketch the details. Part (a) may be proved along the lines of Claim 26 of Lemma 24. Part (c) follows from Part (b) and the commutativity of $(+)_o$ .

Part (b) may be proved by transfinite induction. In the induction, we use the fact that if the leftmost term of (the notational CNF of) $y$ that is different from the corresponding term of (the notational CNF of) $y'$ is $\{w^{v_0}\}_o \times_o \underline{n_0}$ and the corresponding term of $y'$ is $\{w^{v'_0}\}_o \times_o \underline{n'_0}$, then either $v'_0 <_o v_0$ or $(v'_0 = v_0 \ \wedge \ n'_0 < n_0)$. Then, we consider the mutually completed notational CNFs of $x \,(+)_o\, y$ and $x \,(+)_o\, y'$ and use a proof that is similar to that of the proofs of Claims 25 and 26 of Lemma 24, to show that $x \,(+)_o\, y' <_o x \,(+)_o\, y$. $\square$ (CLAIM 36)

Next, we define a family of partial computable functions $\{\psi_n \mid n \in N\}$ that will enable us to use Lemma 23 to get the desired result. For all $n$, for all $u_0, \ldots, u_n \leq_o u <_o \{w^v\}_o$, let

$$\psi_n(u_n, \ldots, u_0) = u_0 \,(+)_o\, \ldots \,(+)_o\, u_n.$$

Clearly, using Claim 36, the definition of $(+)_o$ and the fact that $u >_o \underline{0}$, we see that $u$ and $\{\psi_n \mid n \in N\}$ satisfy the hypotheses of Lemma 23. Also, using Claim 36, it is easy to show that $\alpha = \lim_{n \to \infty} |\psi_n(u, \ldots, u)|_o \leq |\{w^v\}_o|_o$. Let $v^*$ be the unique notation for $\alpha$ such that $v^* \leq_o \{w^v\}_o$. Therefore, we have $u <_o v^* \leq_o \{w^v\}_o$.

Lastly, applying Lemma 23 to $u$, $\{\psi_n \mid n \in N\}$ and $v^*$, Lemma 35 follows. $\square$ (Lemma 35)

As an application of Lemma 35, we get the following. It is important to note that, for any notations $v, w$, where $w$ is for $\omega$, the notations $(\{w^v\}_o \times_o w)$ and $\{w^{v +_o \underline{1}}\}_o$ are not only for the same ordinal, but also *are exactly the same number, i.e., notation* (this follows from our carefully chosen "definitions" of $\times_o$ and $\{\cdot\}_o$).

**Lemma 37** *For all $v, w \in O$, where $w$ is for $\omega$, $\mathcal{C}_{\{w^v\}_o} \in \mathbf{Lim}_{\{w^v\}_o \times_o w}\mathbf{Mex}$.*

Proof. Follows from Lemmas 28 and 35, using reasoning similar to that presented just before Lemma 29. $\square$

As indicated earlier, we get, for notations in special form, the following strengthening of the General Hierarchy Theorem (Theorem 31).

**Theorem 38 (Strong Hierarchy Theorem)** *Suppose $u, v, w \in O$ where $w$ is for $\omega$. Then,*
*(a) $[\{w^v\}_o \leq_o u <_o \{w^v\}_o \times_o w] \Rightarrow \mathbf{Lim}_u\mathbf{Mex} = \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$,*
*(b) $\mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^v\}_o \times_o w}\mathbf{Mex}$,*
*(c) $v$ is a notation for a limit ordinal $\Rightarrow$ [ for all $u <_o v, \mathbf{Lim}_{\{w^u\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$].*

Proof. (a) Follows from Theorem 19.

(b) Follows from Lemmas 30 and 37, and Proposition 18.

(c) Suppose $v \in O$ is for a limit ordinal and $v' <_o v$. By part (b), we have $\mathbf{Lim}_{\{w^{v'}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{(v' +_o \underline{1})}\}_o}\mathbf{Mex}$. Also, since $v$ is for a limit ordinal, $(v' +_o \underline{1}) <_o v$.[25] Hence, by Theorem 17(d), $\{w^{(v' +_o \underline{1})}\}_o <_o \{w^v\}_o$. Therefore, by Proposition 18, $\mathbf{Lim}_{\{w^{(v' +_o \underline{1})}\}_o}\mathbf{Mex} \subseteq \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$. Thus $\mathbf{Lim}_{\{w^{v'}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$. $\square$

As a corollary to Theorem 38, we get

**Corollary 39** *Suppose $u, v, w \in O$, where $w$ is for $\omega$. Then,*
*$u <_o \{w^v\}_o \Rightarrow \mathbf{Lim}_u\mathbf{Mex} \subset \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$.*

Proof.

Follows using an easy induction on $v$, along with Proposition 18, Theorems 34 and 38, and Fact 7(e). $\square$

Next, suppose $n \in N$ and $w$ is for $\omega$. Then, Theorem 38 gives us, for example, (i) $\mathbf{Lim}_{\{w^{\underline{n}}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{\underline{n+1}}\}_o}\mathbf{Mex}$; (ii) $\mathbf{Lim}_{\{w^{\underline{n}}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^w\}_o}\mathbf{Mex}$, and (iii) $\mathbf{Lim}_{\{w^{(w \times_o \underline{n})}\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{\{w^{\underline{2}}\}_o}\}_o}\mathbf{Mex}$. We may repeatedly apply the Strong Hierarchy Theorem to get a hierarchy of the form shown in Figure 1, Section 1.3.

Ordinal $\epsilon_0$ is the limit of the sequence $\omega, \omega^\omega, \omega^{\omega^\omega}, \ldots$ as well as the least fixed point of $\lambda\beta[\omega^\beta]$. Furthermore, it is the least ordinal not expressible as a polynomial in $0, 1, \ldots, \omega$ using finitely many applications of addition, multiplication and exponentiation.[26] The example of Figure 1 may be extended to notations for $\epsilon$ numbers and beyond by a strategy we illustrate for $\epsilon_0$. Let

$$S = \{u \mid u \leq_o \text{ some notation in } w, \{w^w\}_o, \{w^{\{w^w\}_o}\}_o, \ldots\}$$

[25] Proof: Since $v' <_o v$, by Fact 7(d), we get $v' +_o \underline{1} \leq_o v$. Since $v$ is a limit ordinal, $v' +_o \underline{1} \neq v$. Hence $v' +_o \underline{1} <_o v$.

[26] See [Rog67, Exercises 11-51 to 11-53, Page 221].

Let $e'$ be such that for all $n$, (a) $\varphi_{e'}(n) \in S$, (b) $\varphi_{e'}(n) <_o \varphi_{e'}(n+1)$, and (c) for all $u \in S$, there exists an $n$ such that $u \leq_o \varphi_{e'}(n)$. Then, clearly, $3 \cdot 5^{e'} \in O$ and $e = 3 \cdot 5^{e'}$ is a notation for $\epsilon_0$. Choosing any such $e$ for $\epsilon_0$, we can employ $\{w^e\}_o$ as another notation for $\epsilon_0$ and continue the hierarchy of Figure 1, by that shown in Figure 2 below, for example.

$$
\begin{array}{ll}
\vdots & \\
\subset & \mathbf{Lim}_{\{w^e\}_o}\mathbf{Mex} \subset \mathbf{Lim}_{\{w^{e+_o\underline{1}}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{e+_o\underline{n}}\}_o}\mathbf{Mex} \subset \ldots \\
\subset & \mathbf{Lim}_{\{w^{e+_o w}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{e+_o\{w^w\}_o}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{e+_o\{w^{\{w^w\}_o}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\subset & \mathbf{Lim}_{\{w^{e\times_o\underline{2}}\}_o}\mathbf{Mex} \subset \ldots \\
\vdots & \\
\subset & \mathbf{Lim}_{\{w^{e\times_o w}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{e\times_o\{w^w\}_o}\}_o}\mathbf{Mex} \subset \ldots \subset \mathbf{Lim}_{\{w^{e\times_o\{w^{\{w^w\}_o}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\subset & \mathbf{Lim}_{\{w^{\{e^{\underline{2}}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\subset & \mathbf{Lim}_{\{w^{\{e^{\underline{3}}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\vdots & \\
\subset & \mathbf{Lim}_{\{w^{\{e^w\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\vdots & \\
\subset & \mathbf{Lim}_{\{w^{\{e^{\{w^w\}_o}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\vdots & \\
\subset & \mathbf{Lim}_{\{w^{\{e^e\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\vdots & \\
\subset & \mathbf{Lim}_{\{w^{\{e^{\{e^e\}_o}\}_o}\}_o}\mathbf{Mex} \subset \ldots \\
\vdots &
\end{array}
$$

Figure 2: Example optimal continuation of hierarchy of Figure 1 along a $<_o$-path for ordinals $< \epsilon_1$.

We have not attempted to obtain notational analogs of ordinal normal form theorems other than for Cantor's Normal Form Theorem but expect there are such results which would yield interesting Strong Hierarchies for the $\mathbf{Lim}_u\mathbf{Mex}$ criteria and which are based on notations, e.g., for the $\epsilon$ ordinals, which do not have to be of the form $\{w^v\}_o$ (for example, $e$ just above).

## 4 Further Results and Future Work

As stated in Theorem 22, for all $n \in O$, $\mathbf{Lim}_{\underline{n}}\mathbf{Mex} = \mathbf{Mex}$. We had originally hoped that this result would not be true and that there would be a fine hierarchy between **Mex** and **LimMex** based on $\lim_{\underline{n}}$-computable parsimony factors. In the next section we successfully explore some different sources of restricted–parsimony fine structure.

## 4.1 Iterated-Limits Parsimony Factors

Just as we defined $\lim_u$-computable in Definition 10, we may define, for $n > 1$, $\lim^n_{u_n,\ldots,u_1}$-computable functions. These $\lim^n_{u_n,\ldots,u_1}$-computable functions are used in the definitions of corresponding $\mathbf{Lim}^n_{u_n,\ldots,u_1}\mathbf{Mex}$ learning criteria.

We can extend Definition 3 to

**Definition 40** *Suppose $n > 0$. Suppose $h : N^{n+1} \to N$. Then,*

*(a) for all $x$, and any sequence of $n-1$ numbers $t_1, \ldots, t_{n-1}$,*

$$\lim_{t\to\infty} h(x, t_{n-1}, \ldots, t_1, t) \stackrel{\text{def}}{=} \begin{cases} y, & \text{if } (\forall^\infty t)[h(x, t_{n-1}, \ldots, t_1, t) = y]; \\ \uparrow, & \text{otherwise.} \end{cases}$$

*(b) for each $m$ such that $0 < m \leq n$, for any sequence of $n-m$ numbers $t_m, \ldots, t_{n-1}$,*

$$h(x, t_{n-1}, \ldots, t_m, \infty, \ldots, \infty) \stackrel{\text{def}}{=} \lim_{t\to\infty} h(x, t_{n-1}, \ldots, t_m, t, \infty, \ldots, \infty).$$

**Definition 41** *Suppose $m, n \in N$ such that $0 < m \leq n$. Then,*
$\mathbf{F} : N^{n+1} \to O$ *is an* $(m,n)O$-countdown function $\stackrel{\text{def}}{\Leftrightarrow}$ *for all $x$, for all sequences of $n-m+1$ numbers $t_m, \ldots, t_n$, $\mathbf{F}(x, t_n, \ldots, t_m + 1, \infty, \ldots, \infty) \leq_o \mathbf{F}(x, t_n, \ldots, t_m, \infty, \ldots, \infty)$.*

We next illustrate our general definition for, $n > 0$, of $\lim^n_{u_n,\ldots,u_1}$-computable functions, via the $n = 2$ case.

**Definition 42** *Suppose $u_2, u_1 \in (O \cup \{*\})$. Then,*
$g : N \to N$ *is* $\lim^2_{u_2,u_1}$-computable $\stackrel{\text{def}}{\Leftrightarrow}$ *there exists computable $h$ such that, for all $x, t_1, t_2$*
*(a) $g(x) = h(x, \infty, \infty)$,*
*(b) for all $t$, $\mathrm{card}(\{t \mid h(x, t_2, t) \neq h(x, t_2, t+1)\})$ is finite,*
*(c) for all $t$, $\mathrm{card}(\{t \mid h(x, t, \infty) \neq h(x, t+1, \infty)\})$ is finite,*
*(d) $(u_1 \neq *) \Rightarrow$ there exists a $(1,2)O$-countdown function $\mathbf{F}_1$ such that*
$\mathbf{F}_1(x, t_2, 0) \leq_o u_1 \wedge$
$h(x, t_2, t_1) \neq h(x, t_2, t_1 + 1) \Rightarrow \mathbf{F}_1(x, t_2, t_1 + 1) <_o \mathbf{F}_1(x, t_2, t_1)$,
*and*
*(e) $(u_2 \neq *) \Rightarrow$ there exists a $(2,2)O$-countdown function $\mathbf{F}_2$ such that*
$\mathbf{F}_2(x, t_2, \infty) \leq_o u_2 \wedge$
$h(x, t_2, \infty) \neq h(x, t_2 + 1, \infty) \Rightarrow \mathbf{F}_2(x, t_2 + 1, \infty) <_o \mathbf{F}_2(x, t_2, \infty)$.

We define *$\lim^2_{u_2,u_1}$-computable as witnessed by $h$*, $\mathbf{F}_1$, and $\mathbf{F}_2$ along the lines of part (b) of Definition 4; we omit the details.

Let $\underline{N} \stackrel{\text{def}}{=} \{\underline{n} \mid n \in N\}$. It turns out that $\lim^n_{u_n,\ldots,u_1}$-computable functions, where each $u_i \in (\underline{N} \cup \{*\})$, have useful characterizations as stated in Proposition 43 below. When we are proving results for the $\mathbf{Lim}^n_{u_n,\ldots,u_1}\mathbf{Mex}$ criteria where each $u_i \in (\underline{N} \cup \{*\})$, we shall use these characterizations in our proofs involving iterated limits below, since, then, we do not have to explicitly consider the more complicated corresponding $(m,n)O$-countdown functions.

**Proposition 43** *Suppose $n > 0$ and $u_1, \ldots, u_n \in (\underline{N} \cup \{*\})$. Then, $g : N \to N$ is* $\lim^n_{u_n,\ldots,u_1}$-computable $\Leftrightarrow$ *there exists a computable $h : N^{n+1} \to N$ such that for all $x$,*
*(a) $g(x) = h(x, \infty, \ldots, \infty)$,*
*(b) for all $i$ such that $1 \leq i < n$ and $u_i \neq *$, for any sequence of $n-i$ numbers $t_n, \ldots, t_{i+1}$, $\mathrm{card}(\{t \mid h(x, t_n, \ldots, t_{i+1}, t, \infty, \ldots, \infty) \neq h(x, t_n, \ldots, t_{i+1}, t+1, \infty, \ldots, \infty)\}) \leq |u_i|_o$, and*
*(c) for all $i$ such that $1 \leq i < n$ and $u_i = *$, for any sequence of $n-i$ numbers $t_n, \ldots, t_{i+1}$, $\mathrm{card}(\{t \mid h(x, t_n, \ldots, t_{i+1}, t, \infty, \ldots, \infty) \neq h(x, t_n, \ldots, t_{i+1}, t+1, \infty, \ldots, \infty)\})$ is finite.*

We write *lim$^n$-computable* for $\lim^n_{*,\ldots,*}$-computable. For $n > 0$, we define $\mathbf{Lim}^n_{u_n,\ldots,u_1}\mathbf{Mex}$ just as in Definition 11, except that we use $\lim^n_{u_n,\ldots,u_1}$-computable parsimony factors instead of $\lim_u$-computable ones. We mostly write $\mathbf{Lim}^n\mathbf{Mex}$ instead of $\mathbf{Lim}^n_{*,\ldots,*}\mathbf{Mex}$.

For parsimony factors computed by two levels of iterated limits, the resulting learning criteria turn out to be, in some cases, identical to $\mathbf{Mex}$, and in other cases, to $\mathbf{LimMex}$. This is illustrated by the next two results. Proposition 44 just below is easy to prove.

**Proposition 44**
*(a) For $u \in (O \cup \{*\})$, $\mathbf{Lim}^2_{u,\underline{0}}\mathbf{Mex} = \mathbf{Lim}^2_{\underline{0},u}\mathbf{Mex} = \mathbf{Lim}_u\mathbf{Mex}$.*
*(b) For $u_1, u_2, v_1, v_2 \in O$, $u_1 \leq_o v_1 \ \wedge \ u_2 \leq_o v_2 \Rightarrow \mathbf{Lim}^2_{\underline{0},\underline{0}}\mathbf{Mex} \subseteq \mathbf{Lim}^2_{u_2,u_1}\mathbf{Mex} \subseteq \mathbf{Lim}^2_{v_2,v_1}\mathbf{Mex} \subseteq \mathbf{Lim}^2\mathbf{Mex}$.*

**Theorem 45** *For any notations $w_1$ and $w_2$ for $\omega$, $\mathbf{Lim}^2_{w_2,w_1}\mathbf{Mex} = \mathbf{Ex}$.*

PROOF. ($\subseteq$) Follows from the definition of $\mathbf{Lim}^2_{w_2,w_1}\mathbf{Mex}$.
($\supseteq$) Suppose $\mathcal{S} \in \mathbf{Ex}$ as witnessed by $\mathbf{M}$.

We first define the following two (computable) predicates. For each $i$, $t_1$ and $t_2$, let $P(i,t_2,t_1) \equiv (\forall x < t_2)[\Phi_i(x) \leq t_1]$ and $Q(i,t_2,t_1) \equiv (\forall y \mid t_2 \leq y \leq t_1)[(\forall x < y)[\Phi_i(x) \leq t_1] \Rightarrow [\mathbf{M}(\varphi_i[t_2]) = \mathbf{M}(\varphi_i[y])]]$.

Also, for all $x, t_2$, let $P(i,t_2,\infty) = \lim_{t\to\infty} P(i,t_2,t)$ and $Q(i,t_2,\infty) = \lim_{t\to\infty} Q(i,t_2,t)$; then $P(i,t_2,\infty) \equiv (\forall x < t_2)[\varphi_i(x)\downarrow]$ and $Q(i,t_2,\infty) \equiv (\forall y \geq t_2)[(\forall x < y)[\varphi_i(x)\downarrow] \Rightarrow \mathbf{M}(\varphi_i[t_2]) = \mathbf{M}(\varphi_i[y])]$.

We next define $h'$ as follows. It is to be understood, that for values of $h'$, any ?'s are changed to 0's.

$$h'(i,t_2,t_1) = \begin{cases} \mathbf{M}(\varphi_i[t_2]), & \text{if } P(i,t_2,t_1) \ \wedge \ Q(i,t_2,t_1); \\ 0, & \text{otherwise.} \end{cases}$$

The following can be easily verified.

(1) $(\forall i)(\forall t_1,t_2)[P(i,t_2,t_1) \Rightarrow P(i,t_2,t_1+1)]$.

(2) $(\forall i)(\forall t_1,t_2)[\neg Q(i,t_2,t_1) \Rightarrow \neg Q(i,t_2,t_1+1)]$.

(3) $(\forall i)(\forall t_1,t_2)[\neg P(i,t_2,\infty) \Rightarrow \neg P(i,t_2+1,\infty)]$.

(4) $(\forall i)(\forall t_1,t_2)[Q(i,t_2,\infty) \Rightarrow Q(i,t_2+1,\infty)]$.

Item (1) implies that for all $i$, $t_2$, $\mathrm{card}(\{t \mid P(i,t_2,t) \neq P(i,t_2,t+1)\}) \leq 1$. Item (2) implies that for all $i$, $t_2$, $\mathrm{card}(\{t \mid Q(i,t_2,t) \neq Q(i,t_2,t+1)\}) \leq 1$. Hence, for all $i$, $t_2$, $\mathrm{card}(\{t \mid (P(i,t_2,t) \ \wedge \ Q(i,t_2,t)) \neq (P(i,t_2,t+1) \ \wedge \ Q(i,t_2,t+1))\}) \leq 2$. Therefore, from the definition of $h'$, we get that for all $i, t_2$, $\mathrm{card}(\{t \mid h'(i,t_2,t) \neq h'(i,t_2,t+1)\}) \leq 2$. Similarly, from items (3) and (4), we can conclude that $\mathrm{card}(\{t \mid h'(i,t,\infty) \neq h'(i,t+1,\infty)\}) \leq 2$.

Let $g'(i) = h'(i,\infty,\infty)$. Hence, $g'$ is a not necessarily monotonically non-decreasing, $\lim^2_{2,2}$-computable function. It is easy to verify that, for all $i$, if $\varphi_i \in \mathbf{Ex}(\mathbf{M})$, then $g'(i) = \mathbf{M}(\varphi_i)$. Let $p_o$ be the predecessor function for $O$ (i.e., for all $x$, if $x = 2^y$, then $p_o(x) = y$; $p_o(x)\uparrow$, otherwise.) Let $h$ be defined as follows. $h(i,t_2,t_1) = \max(\{h'(x,t_2,t_1) \mid x \leq i\})$. Let $g(i) = h(i,\infty,\infty)$.

We get for all $i$, $t_2$, $\mathrm{card}(\{t \mid h(i,t_2,t) \neq h(i,t_2,t+1)\}) \leq 2(i+1)$; and for all $i$, $\mathrm{card}(\{t \mid h(i,t,\infty) \neq h(i,t+1,\infty)\}) \leq 2(i+1)$. For all $i$, $t_1$ and $t_2$, let

$$\mathbf{F}_1(i,t_2,t_1) = \begin{cases} \underline{2(i+1)}, & \text{if } t_1 = 0; \\ p_o(\mathbf{F}_1(i,t_2,t_1-1)), & \text{if } t_1 > 0 \ \wedge \ h(i,t_2,t_1) \neq h(i,t_2,t_1-1); \\ \mathbf{F}_1(i,t_2,t_1), & \text{otherwise.} \end{cases}$$

$$\mathbf{F}_2(i,t_2,t_1) = \begin{cases} \underline{2(i+1)}, & \text{if } t_2 = 0; \\ p_o(\mathbf{F}_2(i,t_2-1,t_1)), & \text{if } t_2 > 0 \ \wedge \ h(i,t_2,t_1) \neq h(i,t_2-1,t_1); \\ \mathbf{F}_2(i,t_2-1,t_1), & \text{otherwise.} \end{cases}$$

Let $w_1$ and $w_2$ be any notations for $\omega$. Clearly, $h$, $\mathbf{F}_1$ and $\mathbf{F}_2$ witness that $g$ is a monotonically non-decreasing $\lim^2_{w_2,w_1}$-computable function. Also, $g$ dominates $g'$.

Therefore, $\mathcal{S} \in \mathbf{Lim}^2_{w_2,w_1}\mathbf{Mex}$. □

**Corollary 46**
*For any notations $w_1$ and $w_2$ for $\omega$,* $\mathbf{Mex} \subset \mathbf{LimMex} \subset \mathbf{Lim}^2_{w_2,w_1}\mathbf{Mex} = \mathbf{Ex}$.

The following theorem resolves the remaining relationships between $\mathbf{Lim}^2_{u_2,u_1}\mathbf{Mex}$ classes.

**Theorem 47** *For all $n \geq 0$:* $\mathbf{LimMex} = \mathbf{Lim}^2_{\underline{n},*}\mathbf{Mex} = \mathbf{Lim}^2_{*,\underline{n}}\mathbf{Mex} = \mathbf{Lim}^2_{\underline{1},\underline{1}}\mathbf{Mex}$.

Proof of Theorem 47. We first prove Lemmas 48, 49 and 50. The theorem then follows easily from these lemmas.

**Lemma 48** *Every lim-computable function $g$ is a $\lim^2_{\underline{1},\underline{1}}$-computable function.*

Proof of Lemma 48. Suppose $g$ is lim-computable as witnessed by computable $h$. We define

$$h_1(x,y) = \begin{cases} h(x,y), & \text{if } (\forall t \geq y)[h(x,t) = h(x,y)]; \\ 0, & \text{otherwise;} \end{cases}$$

and

$$h_2(x,y,z) = \begin{cases} h(x,y), & \text{if } (\forall t \mid y \leq t \leq y+z)[h(x,y) = h(x,t)]; \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, $h_2$ is a computable function. It is easy to verify the following:
(a) For each $x, y$, $\mathrm{card}(\{z \mid h_2(x,y,z) \neq h_2(x,y,z+1)\}) \leq 1$;
(b) For each $x$, $\mathrm{card}(\{y \mid h_1(x,y) \neq h_1(x,y+1)\}) \leq 1$;
(c) For all $x, y$, $h_2(x,y,\infty) = h_1(x,y)$;
(d) For all $x$, $h_1(x,\infty) = h(x,\infty) = g(x)$.
Clearly, $g$ is $\lim^2_{\underline{1},\underline{1}}$-computable as witnessed by $h_2$. □ (Lemma 48)

**Lemma 49** *Suppose $n \in N$. Then, for each monotonically non-decreasing $\lim^2_{\underline{n},*}$-computable function $g$, there exists a lim-computable function $g'$ such that $g'$ dominates $g$.*

PROOF OF LEMMA 49. By induction on $n$.
The lemma clearly holds for $n = 0$. Suppose by induction that the lemma holds for $n = m$. We will show that the lemma holds for $n = m+1$. Therefore, suppose $g$ is a $\lim^2_{\underline{m+1},*}$-computable function. Hence, there exist $h_1$ and computable $h_2$ such that the following three conditions are true.

(a) For all $x, y$, $h_1(x,y) = h_2(x,y,\infty)$.

(b) For all $x$, $g(x) = h_1(x,\infty)$.

(c) For all $x$, $\mathrm{card}(\{y \mid h_1(x,y) \neq h_1(x,y+1)\}) \leq m+1$.

*Case 1*: For all but finitely many $x$, $\mathrm{card}(\{y \mid h_1(x,y) \neq h_1(x,y+1)\}) \leq m$.

In this case, it is easy to show that $g(x)$ is $\lim^2_{\underline{m},*}$-computable. To see this, let $y_0$ be such that,

$$(\forall x \mid \mathrm{card}(\{y \mid h_1(x,y) \neq h_1(x,y+1)\}) = m+1)(\forall y \geq y_0)[h_1(x,y) = h_1(x,y_0)]$$

Let $h_2'$ be computable, such that for all $x$, $y$ and $y$, $h_2'(x,y,z) = h_2(x,y+y_0,z)$. It is easy to verify that $g$ is $\lim^2_{\underline{m},*}$-computable as witnessed by $h_2'$. Hence, by the inductive hypothesis, there is a lim-computable $g'$ such that $g'$ dominates $g$.

*Case 2*: For infinitely many $x$, $\mathrm{card}(\{y \mid h_1(x,y) \neq h_1(x,y+1)\}) = m+1$.

We define a function $H$ as follows.

$$H(x) = \min(\{\langle z, y_0, y_1, y_2, \ldots, y_{m+1}\rangle \mid z \geq x \ \wedge \ (\forall j \leq m)[y_j < y_{j+1} \ \wedge \ h_1(z,y_j) \neq h_1(z,y_{j+1})]\})$$

$H$ is lim-computable since, for each $x$, the characteristic function of the set, $\{\langle z, y_0, y_1, y_2, \ldots, y_{m+1}\rangle \mid z \geq x \ \wedge \ (\forall j \leq m)[y_j < y_{j+1} \ \wedge \ h_1(z,y_j) \neq h_1(z,y_{j+1})]\}$ is lim-computable.

We next use $H$ to define $g'$ satisfying the requirements of the lemma. First define $H_1, H_2$ as follows: suppose $H(x) = \langle z, y_0, y_1, \ldots, y_{m+1}\rangle$; then let $H_1(x) = z$ and $H_2(x) = y_{m+1}$. Clearly, $H_1, H_2$ are lim-computable functions. Now it is easy to verify that, if $x \leq x'$, then $H_1(x) \leq H_1(x')$. Let $g'(x) = h_1(H_1(x), H_2(x)) = g(H_1(x))$. Since $h_1, H_1, H_2$ are lim-computable functions, it follows that $g'$ is also lim-computable and dominates $g$. □ (LEMMA 49)

**Lemma 50** *Every $\lim^2_{*,\underline{n}}$-computable function $g$ is $\lim^2_{\underline{n},*}$-computable.*

PROOF OF LEMMA 50. Let $h_1, h_2$ be such that

(a) for all $x$, $g(x) = h_1(x,\infty)$;

(b) for all $x, y$, $h_1(x,y) = h_2(x,y,\infty)$;

(c) for all $x, y$, $\mathrm{card}(\{t \mid h_2(x,y,t) \neq h_2(x,y,t+1)\}) \leq n$.

For all $i, x, y$, we define

$$G_2^i(x,y) = \begin{cases} h_2(x,y,z), & \text{if } \mathrm{card}(\{t \mid h_2(x,y,t) \neq h_2(x,y,t+1)\}) \geq i \text{ and } z = 1+ \\ & \min(\{z' \mid \mathrm{card}(\{t \leq z' \mid h_2(x,y,t) \neq h_2(x,y,t+1)\}) \\ & \qquad \geq i\}); \\ \uparrow, & \text{otherwise.} \end{cases}$$

Intuitively, $G_2^i$, for each $x, y$ assumes that $h_2(x,y,\cdot)$ makes exactly $i$ mind changes. Based on this assumption, it tries to compute $\lim_{t\to\infty} h_2(x,y,t)$.

We note that $\max(\{i \leq n \mid \mathrm{card}(\{z \mid G_2^i(x,z){\downarrow}\}) \geq y\})$, is a lim-computable function in $x, y$. Let $X$ be a computable function such that $X(x,y,\infty) = \max(\{i \leq n \mid \mathrm{card}(\{z \mid G_2^i(x,z){\downarrow}\}) \geq y\})$. $X(x,y,\infty)$ is monotonically non-increasing in $y$ and is bounded by $n$. Moreover, for all $x$, for all $i > X(x,\infty,\infty)$, $\mathrm{card}(\{z \mid G_2^i(x,z){\downarrow}\})$ is finite.

We recall from Section 2.1 that $\max(\emptyset) = 0$. Let $H(x,y,z) = G_2^{X(x,y,z)}(x,w)$, where $w = \max(\{w' \leq z \mid G_2^{X(x,y,z)}(x,w')\downarrow \text{ in } \leq z \text{ steps}\})$. Then, for all $x$, $\mathrm{card}(\{y \mid H(x,y,\infty) \neq H(x,y+1,\infty)\}) \leq n$. Moreover, using the definition of $X$, $G_2^{X(x,\infty,\infty)}$ and $H$, it is easy to verify that $H(x,\infty,\infty) = h_2(x,\infty,\infty) = g(x)$. Lemma follows. □ (LEMMA 50)

From Lemma 48, we get $\mathbf{LimMex} \subseteq \mathbf{Lim}^2_{\underline{1},\underline{1}}\mathbf{Mex}$. Clearly, for any lim-computable $g'$, there exists a monotonically non-decreasing lim-computable $g''$ such that $g''$ dominates $g'$. Hence, by Lemma 49, we get that for all $n$, $\mathbf{Lim}^2_{\underline{n},*}\mathbf{Mex} \subseteq \mathbf{LimMex}$. Finally, by Lemma 50, for all $n$, $\mathbf{Lim}^2_{*,\underline{n}}\mathbf{Mex} \subseteq \mathbf{Lim}^2_{\underline{n},*}\mathbf{Mex}$. Thus, the theorem follows from the above three lemmas.

□ (THEOREM 47)

Thus, if both $u_1$ and $u_2$ are $\underline{0}$, $\mathbf{Lim}^2_{u_2,u_1}\mathbf{Mex} = \mathbf{Mex}$. Else, when both of $u_1, u_2$ are not equal to $\underline{0}$, $\mathbf{Lim}^2_{u_2,u_1}\mathbf{Mex}$ is equal to either $\mathbf{LimMex}$ or $\mathbf{Ex}$; otherwise, if $u_i$ is the member of $\{u_1,u_2\}$ that is not $\underline{0}$, then $\mathbf{Lim}^2_{u_2,u_1}\mathbf{Mex}$ is identical to $\mathbf{Lim}_{u_i}\mathbf{Mex}$.

We next briefly consider iterating limits to three levels. A proposition similar to Proposition 44 holds, but we omit stating it here. The following theorem shows that for parsimony factors that use three levels of iterated limits, allowing even a small number of mind changes in each limit essentially retains no parsimony in the final programs learned.

**Theorem 51** $\mathbf{Ex} = \mathbf{Lim}^3_{\underline{1},\underline{2},\underline{2}}\mathbf{Mex}$.

PROOF OF THEOREM 51. We first prove Lemma 52 below.

**Lemma 52** *Suppose $f$ is a computable function from $N^2 \to N$. Let $F(i) = \lim_{t\to\infty} f(i,t)$. Then, there exists a $lim^3_{\underline{1},\underline{2},\underline{2}}$ function $g$ such that, for all $i$, $g(i) = \max(\{F(j) \mid j \leq i \ \wedge \ F(j)\downarrow\})$.*

PROOF OF LEMMA 52. Suppose $f, F$ are given as in the lemma. Below, we define $g(.), h_1(.,.), h_2(.,.,.)$ and $h_3(.,.,.,.)$ (where $h_3$ is computable), such that each of the following seven clauses is true.

(a) For all $i$, $g(i) = h_1(i,\infty)$;
(b) For all $i,n$, $h_1(i,n) = h_2(i,n,\infty)$.
(c) For all $i,n,m$, $h_2(i,n,m) = h_3(i,n,m,\infty)$.
(d) For all $i$, $\mathrm{card}(\{n \mid h_1(i,n) \neq h_1(i,n+1)\}) \leq 1$.
(e) For all $i,n$, $\mathrm{card}(\{m \mid h_2(i,n,m) \neq h_2(i,n,m+1)\}) \leq 2$.
(f) For all $i,n,m$, $\mathrm{card}(\{t \mid h_3(i,n,m,t) \neq h_3(i,n,m,t+1)\}) \leq 2$.
(g) $g(i) = \max(\{F(j) \mid j \leq i \ \wedge \ F(j)\downarrow\})$.

The lemma then easily follows from the above clauses. We now define $h_3$.

Let $X_3^{i,n,m,t} = \{j \leq i \mid (\forall w \leq m)[f(j,n) = f(j,n+w)]\}$.
Let $Y_3^{i,n,m,t} = \{j \leq i \mid (\forall w \leq m+t)[f(j,n) = f(j,n+w)]\}$.
Let $Z_3^{i,n,m,t} = \{j \leq i \mid (\exists w \leq t)[f(j,n+m) \neq f(j,n+m+w)]\}$.
Let $P_3(i,n,m,t) \equiv [(X_3^{i,n,m,t} = Y_3^{i,n,m,t})$ and $(X_3^{i,n,m,t} \cup Z_3^{i,n,m,t} = \{x \mid x \leq i\})]$.

$$h_3(i,n,m,t) = \begin{cases} \max(\{f(j,n) \mid j \in X_3^{i,n,m,t}\}), & \text{if } P_3(i,n,m,t); \\ 0, & \text{otherwise.} \end{cases}$$

If $i$, $n$ and $m$ are fixed, $Y_3^{i,n,m,t}$ is monotonically non-increasing and $Z_3^{i,n,m,t}$ is monotonically non-decreasing in $t$ (in the set containment sense). $X_3^{i,n,m,t}$ is, clearly, fixed once $i,m$ and $n$ are.

Thus, if $t < t' < t''$, then $P_3(i,n,m,t)$ and $\neg P_3(i,n,m,t')$ implies $\neg P_3(i,n,m,t'')$. So (f) is true. Let

$X_2^{i,n,m} = X_3^{i,n,m,\infty} = X_3^{i,n,m,0} = \{j \le i \mid (\forall w \le m)[f(j,n) = f(j,n+w)]\}$.
$Y_2^{i,n,m} = Y_3^{i,n,m,\infty} = \{j \le i \mid (\forall w)[f(j,n) = f(j,n+w)]\}$.
$Z_2^{i,n,m} = Z_3^{i,n,m,\infty} = \{j \le i \mid (\exists w)[f(j,n+m) \ne f(j,n+m+w)]\}$.
$P_2(i,m,n) = P_3(i,n,m,\infty) \equiv [(X_2^{i,n,m} = Y_2^{i,n,m})$ and $(X_2^{i,n,m} \cup Z_2^{i,n,m} = \{x \mid x \le i\})]$.
Let $h_2(i,n,m) = h_3(i,n,m,\infty)$. Thus (c) holds. It is easy to verify that

$$h_2(i,n,m) = \begin{cases} \max(\{f(j,n) \mid j \in X_2^{i,n,m}\}), & \text{if } P_2(i,n,m); \\ 0, & \text{otherwise.} \end{cases}$$

Now $X_2^{i,n,m}$ is a monotonically non-increasing function of $m$, $Y_2^{i,n,m}$ is independent of $m$, and $Z_2^{i,n,m}$ is a monotonically non-increasing function of $m$. Thus, if $m < m' < m''$, then $P_2(i,n,m)$ and $\neg P_2(i,n,m')$ implies $\neg P_2(i,n,m'')$. It immediately follows that (e) holds. Let

$X_1^{i,n} = X_2^{i,n,\infty} = \{j \le i \mid (\forall w)[f(j,n) = f(j,n+w)]\}$.
$Y_1^{i,n} = Y_2^{i,n,\infty} = Y_2^{i,n,0} = \{j \le i \mid (\forall w)[f(j,n) = f(j,n+w)]\}$,
$Z_1^{i,n} = Z_2^{i,n,\infty} = \{j \le i \mid F(j)\uparrow\}$.
$P_1(i,n) = P_2(i,n,\infty) \equiv [X_1^{i,n} = Y_1^{i,n}$ and $X_1^{i,n} \cup Z_1^{i,n} = \{x \mid x \le i\}]$
Let $h_1(i,n) = h_2(i,n,\infty)$. Thus (b) holds.
It is easy to verify that

$$h_1(i,n) = \begin{cases} \max(\{f(j,n) \mid j \in X_1^{i,n}\}), & \text{if } P_1(i,n); \\ 0, & \text{otherwise.} \end{cases}$$

Note that $X_1^{i,n} = Y_1^{i,n}$ always holds. Thus $P_1(i,n)$ is equivalent to $X_1^{i,n} = \{j \le i \mid F(j)\downarrow\}$. Hence,

$$h_1(i,n) = \begin{cases} \max(\{f(j,n) \mid j \le i \ \wedge \ F(j)\downarrow\}), & \text{if } P_1(i,n); \\ 0, & \text{otherwise.} \end{cases}$$

Note that for all $i, n, n'$ such that $n < n'$, if $P_1(i,n)$ then $P_1(i,n')$. Thus (d) holds. Let $g(i) = \lim_{n\to\infty} h_1(i,n)$. Thus (a) holds. Now, $g(i) = h_1(i,\infty) = \max(\{F(j) \mid j \le i \ \wedge \ F(j)\downarrow\})$. Thus (g) holds. □ (LEMMA 52)

Since, for every machine $\mathbf{M}$ there exists a computable function $f$ such that, $(\forall i \mid \varphi_i \in \mathbf{Ex}(\mathbf{M}))[f(i,\infty) = \mathbf{M}(\varphi_i)]$ the theorem follows. □ (THEOREM 51)

There are many mostly uninvestigated questions still open. For $u \in O$, do the $\mathbf{Lim}_{\underline{u}}\mathbf{Mex}$ criteria have "limiting-standardizability" style characterizations similar to those first obtained for $\mathbf{Mex}$ in [Fre75]. Generally, except for the cases noted above and their easy consequences, for $u_1, \ldots, u_n \in (O \cup \{*\})$, how do the learning classes $\mathbf{Lim}^n_{u_n,\ldots,u_1}\mathbf{Mex}$ compare to one another?

### 4.2 Notation Dependence Results

As mentioned in Section 1.2, Case and Suraj [CS03] show how to characterize the $\lim_u$-computable functions and variants in terms of concepts from [EHK81, Sel84]. One of these characterizations is exploited to prove

**Theorem 53 (Case–Suraj [CS03])** *Every lim-computable function is also $\lim_v$-computable, for some notation $v$ for $\omega^2$.*

Using Theorem 53, we can get the following.

**Theorem 54** *For all $\mathcal{C}$, if $\mathcal{C} \in$ **LimMex** then there exists a notation $v$ for $\omega^2$ such that $\mathcal{C} \in$ $\mathbf{Lim}_v\mathbf{Mex}$.*

PROOF. Suppose $\mathcal{C} \in$ **LimMex** as witnessed by **M** and a monotonically non-decreasing lim-computable $g$. By Theorem 53, there exists a notation $v$ for $\omega^2$, such that $g$ is $\lim_v$-computable. Hence, $\mathcal{C} \in \mathbf{Lim}_v\mathbf{Mex}$ as witnessed by the same **M** and $g$. $\square$

Theorem 54 essentially says that for every class of functions that is **LimMex**-identifiable, there is some notation $v$ for an ordinal as small as $\omega^2$, such that this class of functions is $\mathbf{Lim}_v\mathbf{Mex}$-identifiable. However, we note that by General Hierarchy Theorem (Theorem 31), that one may $\mathbf{Lim}_{v'}\mathbf{Mex}$-identify strictly more classes of functions by using a suitable notation $v' >_o v$.

Also, by Corollary 32, for all $u \in O$, there exist a class of functions $\mathcal{C}$ such that $\mathcal{C} \in (\mathbf{LimMex} - \mathbf{Lim}_u\mathbf{Mex})$. Hence, by Theorem 54, we have

**Corollary 55** *For each $u \in O$, there exists a notation $v$ for $\omega^2$, such that $\mathbf{Lim}_v\mathbf{Mex} \not\subseteq \mathbf{Lim}_u\mathbf{Mex}$.*

Using Corollary 55, we get

**Corollary 56** *For each constructive ordinal $\alpha$ such that $\alpha \geq \omega^2$, there exist notations $u$, $v$ for $\alpha$ such that $\mathbf{Lim}_u\mathbf{Mex} \neq \mathbf{Lim}_v\mathbf{Mex}$.*

PROOF. Suppose $\alpha$ is a constructive ordinal $\geq \omega^2$ and $u$ is for $\alpha$. By Corollary 55, we get that there exists a notation $u'$ for $\omega^2$, such that $\mathbf{Lim}_{u'}\mathbf{Mex} \not\subseteq \mathbf{Lim}_u\mathbf{Mex}$.

Let $v$ be a notation for $\alpha$ such that $u' \leq_o v \leq_o (u' +_o u)$; as noted in Section 2.3, such a notation exists, and by Fact 7, it is unique. By Proposition 18, $\mathbf{Lim}_{u'}\mathbf{Mex} \subseteq \mathbf{Lim}_v\mathbf{Mex}$; therefore, $\mathbf{Lim}_v\mathbf{Mex} \not\subseteq \mathbf{Lim}_u\mathbf{Mex}$. Hence the corollary follows. $\square$

We do not know if there are $u, v \in O$ such that $\mathbf{Lim}_u\mathbf{Mex} \subset \mathbf{Lim}_v\mathbf{Mex}$ and yet $|u|_o > |v|_o$.

Also from [CS03], we have

**Theorem 57 (Case–Suraj [CS03])** *For all $u, v \in O$, such that $u$ and $v$ are for the same ordinal $< \omega^2$, every $\lim_u$-computable function is also $\lim_v$-computable.*

From Theorem 57, we get

**Theorem 58** *For all $u, v \in O$, such that $u$ and $v$ are for the same ordinal $< \omega^2$, $\mathbf{Lim}_u\mathbf{Mex} = \mathbf{Lim}_v\mathbf{Mex}$.*

It is interesting to ask what happens if we base parsimony restricted criteria on constructive ordinals instead of on notations in $O$ for them. One way to do so is to let, for any constructive ordinal $\alpha$,

$$\mathbf{Lim}_\alpha\mathbf{Mex} \stackrel{\text{def}}{=} \bigcup_{u \text{ for } \alpha} \mathbf{Lim}_u\mathbf{Mex}.$$

While we have seen that the criteria $\mathbf{Lim}_u\mathbf{Mex}$, for $u \in O$, form subtle and finely graded and ramified infinite hierarchies, there are but three distinct criteria of the form $\mathbf{Lim}_\alpha\mathbf{Mex}$, for constructive ordinals $\alpha$, and they form a linear, finite hierarchy thus.

**Theorem 59**

*(a) For all constructive ordinals $\alpha$ such that $\alpha \geq \omega^2$, $\mathbf{Lim}_\alpha\mathbf{Mex} = \mathbf{LimMex}$.*

*(b) $\mathbf{Mex} \subset \mathbf{Lim}_\omega\mathbf{Mex} \subset \mathbf{Lim}_{\omega^2}\mathbf{Mex}$.*

PROOF.

From Theorem 54, we get **LimMex** $\subseteq$ $\mathbf{Lim}_{\omega^2}\mathbf{Mex}$. From Proposition 18, we get $\mathbf{Lim}_{\omega^2}\mathbf{Mex} \subseteq$ **LimMex**. Hence Part (a) follows.

(b) Follows from Theorems 58, 38 and 22. $\square$

We have not *fully* considered the possible dependencies of the $\lim_u$-computable functions or of the classes $\mathbf{Lim}_u\mathbf{Mex}$ on the many notation *systems* alternative to $O$([Amb95]). Of course, for *example*, by an obvious embedding, our Strong Hierarchy Theorem (Theorem 38 just above) *mutatis mutandis* clearly follows for computably related, non maximal ([Rog67]) notation systems based on exponential polynomials in finite ordinals, $\omega$, and a suitably specified collection of $\epsilon$ numbers. Also, we have not considered parsimony restricted criteria based on notations or programs for non-well orderings (with, for example, no computable infinite descending chains) [Ers68b, SSV97, AFS99].

### 4.3 Comparison with $O$-Bounded Mind Change Ex-Identification

We recall that SEG is the set of all finite initial segments of total function: $N \to N$. We next consider, for $u, v \in O$, connections between the $\mathbf{Lim}_v\mathbf{Mex}$ criteria and the $\mathbf{Ex}_u$ criteria that were introduced by Freivalds and Smith [FS93]. To define $\mathbf{Ex}_u$, we first define *$O$-mind change counters*, similar to those defined in [JS97, AJS99]: an *$O$-mind change counter* is a computable mapping, $\mathbf{F}$, from SEG into $O$, such that for all $n$ and $f$, $\mathbf{F}(f[n]) \geq_o \mathbf{F}(f[n+1])$. Now, for each $u \in O$, a machine $\mathbf{M}$ *$\mathbf{Ex}_u$-identifies* $f$ if and only if $\mathbf{M}$ (i) **Ex**-identifies $f$, and (ii) there exists an $O$-mind change counter $\mathbf{F}$ such that for every $n$, $[? \neq \mathbf{M}(f[n]) \neq \mathbf{M}(f[n+1])] \Rightarrow \mathbf{F}(f[n+1]) <_o \mathbf{F}(f[n]) \leq_o u$. As can be seen, the $\mathbf{Ex}_u$ criteria impose restrictions on the mind changes of the *learning machines* themselves.

Besides [FS93], other works that study various aspects of imposing similar restrictions on the mind changes of learning machines include those of Apsītis [Aps94], Ambainis [Amb95], Jain and Sharma [JS97, JS99, JS01], Sharma, Stephan and Ventsov [SSV97], Ambainis, Freivalds and Smith [AFS99] and, Ambainis, Jain, and Sharma [AJS99].

**Theorem 60** $\mathbf{Mex} \not\subseteq \bigcup_{u \in O} \mathbf{Ex}_u$.

PROOF OF THEOREM 60.

Let **EvntlyConst** denote the set of eventually constant functions, i.e., $\{f \mid (\exists k, n)(\forall x > n)[f(x) = k]\}$. Also, let $\mathrm{MinProg}_\psi(f) = \min(\{p \mid \psi_p = f\})$.

We show that there exists a class of functions $\mathcal{C} \subseteq$ **EvntlyConst** such that $\mathcal{C} \in$ **Mex**, but for all $u \in O$, $\mathcal{C} \notin \mathbf{Ex}_u$. Actually, we first construct an acceptable programming system $\psi$ and show that such a $\mathcal{C} \in \mathbf{Mex}_\psi$, where $\mathbf{Mex}_\psi$ is defined just like **Mex** is defined in Definition 2, except that **Ex** is changed to $\mathbf{Ex}_\psi$ and $\mathrm{MinProg}(f)$ is changed to $\mathrm{MinProg}_\psi(f)$. Then, using the fact that **Mex** is acceptable programming system independent, we get $\mathcal{C} \in$ **Mex**.

Fix any $i$. For $\tau \in$ SEG, $\mathrm{len}(\tau)$ denotes the number of elements in $\tau$. Thus $\mathrm{len}(f[n]) = n$. $\mathrm{content}(\tau)$ denotes the set of pairs in the range of $\tau$. Thus $\mathrm{content}(f[n]) = \{(i, f(i)) \mid i < n\}$. For any $\sigma, \tau \in$ SEG, we say that *$\tau$ is an extension of $\sigma$* iff $\mathrm{content}(\sigma) \subseteq \mathrm{content}(\tau)$. Similarly, *$f$ is an extension of $\sigma$* iff $\mathrm{content}(\sigma) \subset$ the graph of $f$.

Let $\sigma_0, \sigma_1, \ldots$, be an enumeration of SEG, such that each element of SEG appears exactly once in this enumeration. Let $f_{i,j}$ be *the $j$-extension of $\sigma_i$*; i.e., the unique total function that is an extension of $\sigma_i$ and has value $j$ on all numbers $\geq \mathrm{len}(\sigma)$.

We next define an acceptable programming system, $\psi$ as follows. Firstly, let $h$ be the following computable function. For all $n$,

$$h(p) = \begin{cases} 0, & \text{if } p = 0; \\ h(p-1) \times 2 + 3, & \text{if } p > 1. \end{cases}$$

Next, we partition $N - \text{range}(h)$ into computable sets, $S_i$, as follows. For all $i, j$, let

$$\begin{aligned} S_{i,j} &= \{p \mid h(\langle i,j\rangle) < p < h(\langle i,j\rangle + 1)\}. \\ S_i &= \cup_{j \in N} S_{i,j}. \\ S &= \cup_{i \in N} S_i. \end{aligned}$$

Clearly, from the definition of $h$, for each $i$, $\text{card}(S_{i,j}) = h(\langle i,j\rangle) + 2$. Therefore, for each $i$, $\text{card}(S_i)$ is infinite. Also, clearly, for each $i$, $S_i$ is computable. $S_i = \{s_{i,0} < s_{i,1} \ldots\}$.

Finally, we define $\psi$ as follows. For all $p$, $\psi_{h(p)} = \varphi_p$. For all $i, j$, $\psi_{s_{i,j}} = f_{i,j}$. It follows easily from the above definition of $\psi$ that $\psi$ is an acceptable programming system.

**Lemma 61** *For each $i$, there are infinitely many $j$ such that* $\text{MinProg}_\psi(f_{i,j}) = s_{i,j}$.

PROOF OF LEMMA 61. For any $j$, $S_{i,j} = \{p \mid h(\langle i,j\rangle) < p < h(\langle i,j\rangle + 1)\}$. As noted earlier, $\text{card}(S_{i,j}) = h(\langle i,j\rangle) + 2$. Since $\text{card}(\{n \mid n \leq h(\langle i,j\rangle)\}) = h(\langle i,j\rangle) + 1$, we get that there exists $p \in S_{i,j}$ such that $p = \text{MinProg}_\psi(\psi_p)$.

Hence, the lemma follows. $\square$ (LEMMA 61)

Clearly, for all $p \in S$, $\psi_p \in$ **EvntlyConst**. Let **M** be a learning machine, defined as follows.

**begin** procedure for defining **M** (on input $f[n]$).
  **if** $n \leq 1$
    **then**
      Let $\mathbf{M}(f[n]) =\ ?$
    **else**
      Let $m$ be the least number $< n$, such that, $(\forall m' \mid m \leq m' < n)[f(m') = f(n-1)]$.
      Let $j = f(m)$ and let $i$ be such that $\sigma_i = f[m]$.
      Let $\mathbf{M}(f[n]) = s_{i,j}$.
  **endif**
**end** procedure for defining **M**.

Interpreting the outputs of **M** above as programs in the $\psi$ system, we get, clearly, that **M** $\mathbf{Ex}_\psi$-identifies **EvntlyConst**.

Also, let $g$ be a computable function such that for all $x$, $g(x) = x$. Let $\mathcal{C} = \{f \mid f \in g\text{-}\mathbf{Mex}_\psi(\mathbf{M})\}$. Using Lemma 61, it is easy to show that, for each $i$, there are infinitely many $j$ such that $\mathbf{M}(f_{i,j})\!\downarrow = s_{i,j}$. Therefore, we get that $\mathcal{C}$ is *dense*; i.e., for every $i$, there exists $f \in \mathcal{C}$ such that $f$ is an extension of $\sigma_i$. Also, clearly, $\mathcal{C} \in \mathbf{Mex}_\psi$; hence $\mathcal{C} \in \mathbf{Mex}$.

We next show that, for all $u \in O$, $\mathcal{C} \notin \mathbf{Ex}_u$. Suppose by way of contradiction that, for some $u \in O$, $\mathcal{C} \in \mathbf{Ex}_u$ as witnessed by some machine $\mathbf{M}'$ and $O$-mind change counter $\mathbf{F}$. Since for every $n$, $[? \neq \mathbf{M}'(f[n]) \neq \mathbf{M}'(f[n+1])] \Rightarrow \mathbf{F}(f[n+1]) <_o \mathbf{F}(f[n]) \leq_o u$, and there are no infinite descending chains of notations in $O$, we get that $\mathbf{M}'$ makes only finitely many mind changes on any $f$. Therefore, there is a $\sigma$ such that $\mathbf{M}'$ does not make a mind change on any $\tau$ that is an extension of $\sigma$. Clearly, since $\mathcal{C}$ is dense, there are infinitely many $f$'s in $\mathcal{C}$ that are each extensions of $\sigma$; yet, $\mathbf{M}'$ $\mathbf{Ex}_u$-identifies at most one of them. Therefore, $\mathbf{M}'$ does not $\mathbf{Ex}_u$-identify $\mathcal{C}$.

Hence, the theorem follows. □ (Theorem 60)

**Theorem 62** *For all $u \in O$, for all $w$ for $\omega$, $\mathbf{Ex}_u \subset \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$.*

Proof.

We show that for all $u$, for all $w$ for $\omega$, $\mathbf{Ex}_u \subseteq \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$. Then, the theorem follows using Theorem 60.

Suppose $\mathcal{C} \in \mathbf{Ex}_u$ as witnessed by machine $\mathbf{M}$ and $O$-mind change counter $\mathbf{F}'$. We first show how to construct a (not necessarily monotonically non-decreasing) $\lim_{u+_o\underline{1}}$-computable function $g$, such that, $(\forall f \in \mathcal{C})[g(\mathrm{MinProg}(f) \geq \mathbf{M}(f)]$. In this interest, we present a simple modification of the construction from the proof of Lemma 28 to get $h$ and $\mathbf{F}$ that witness that $g$ is $\lim_{u+_o\underline{1}}$-computable.

Let $h$ and $\mathbf{F}$ be defined as follows. For all $i$, let $h(i,0) = 0$, $\mathbf{F}(i,0) = u +_o \underline{1}$. For all $i,t$, we define $h(i,t+1), \mathbf{F}(i,t+1)$ as follows.

**begin** computation of $h(i,t+1), \mathbf{F}(i,t+1)$.

**if** $\Phi_i(0) > t$

**then**

Let $h(i,t+1) = h(i,t)$ and $\mathbf{F}(i,t+1) = \mathbf{F}(i,t)$;

**else**

Let $x$ be the largest value $\leq t$, such that $(\forall x' \leq x)[\Phi_i(x') \leq t]$.
Let $h(i,t+1) = \mathbf{M}(\varphi_i[x+1])$ and $\mathbf{F}(i,t+1) = \mathbf{F}'(\varphi_i[x+1])$.

**endif**

**end** computation of $h(i,t+1), \mathbf{F}(i,t+1)$.

Let $g(i) = h(i,\infty)$. Clearly, for all $i$ such that $\varphi_i \in \mathcal{C}$, we get $h(i,\infty) = \mathbf{M}(f)$; in particular, for all $f \in \mathcal{C}$, $h(\mathrm{MinProg}(f),\infty) = \mathbf{M}(f)$. Clearly, also, $h$ and $\mathbf{F}$ witness that $g$ is $\lim_{u+_o\underline{1}}$-computable. Suppose $w$ is a notation for $\omega$. Then, by Lemma 24, there exists a monotonically non-decreasing $\lim_{\{(u+_o\underline{2})^w\}_o}$-computable function $g'$ which dominates $g$. It is easy to verify that $\mathcal{C} \in \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$ as witnessed by $\mathbf{M}$ and $g'$. Hence, $\mathbf{Ex}_u \subset \mathbf{Lim}_{\{(u+_o\underline{2})^w\}_o}\mathbf{Mex}$. □

**Theorem 63** *Suppose $u,v,w \in O$ such that $u <_o \{w^v\}_o$ and $w$ is for $\omega$. Then, $\mathbf{Ex}_u \subset \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$.*

Proof.

We show that for all $u,v,w \in O$ such that $u <_o \{w^v\}_o$ and $w$ is for $\omega$, $\mathbf{Ex}_u \subseteq \mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex}$. Then, the theorem follows using Theorem 60.

If $v = \underline{0}$, then $\{w^v\}_o = \underline{1}$. Therefore, $\mathbf{Lim}_{\{w^v\}_o}\mathbf{Mex} = \mathbf{Lim}_{\underline{1}}\mathbf{Mex}$. From Corollary 22, $\mathbf{Lim}_{\underline{1}}\mathbf{Mex} = \mathbf{Mex}$. Also, $\mathbf{Ex}_{\underline{0}} \subseteq \mathbf{Mex}$ [Che82, Theorem 6.1, page 80]. Therefore, the $v = \underline{0}$ case follows.

The $v >_o \underline{0}$ case is similar to the proof of Theorem 62, except that it makes use of Lemma 35 instead of Lemma 24 and the fact that $u <_o \{w^v\}_o \Rightarrow u +_o \underline{1} <_o \{w^v\}_o$. We omit the details. $\square$

As a corollary to the theorem just above, we get

**Corollary 64** *Suppose $v, w \in O$ such that $w$ is for $\omega$. Then,* $\mathbf{Ex}_{\{w^v\}_o} \subset \mathbf{Lim}_{\{w^v\}_o \times_o w}\mathbf{Mex}$.

For $n \in N^+$, Theorem 62 implies for all notations $w$ for $\omega$, $\mathbf{Ex}_{\underline{n}} \subset \mathbf{Lim}_{\{(\underline{n}+_o\underline{2})^w\}_o}\mathbf{Mex}$; the latter, by Theorem 58 and the fact that $|\{(\underline{n} +_o \underline{2})^w\}_o|_o = \omega$, is $= \mathbf{Lim}_\omega\mathbf{Mex}$. Chen [Che82, Corollary 6.2, pages 80-81] shows that, in fact, $\mathbf{Ex}_{\underline{n}} \subset \mathbf{Mex}$; hence $\mathbf{Ex}_{\underline{n}} \subset \mathbf{Lim}_{\underline{0}}\mathbf{Mex}$. However, $\mathbf{Ex}_w \not\subseteq \mathbf{Mex}$ follows from Theorem 65 below.

**Theorem 65** *For all $u, w \in O$ such that $w$ is for $\omega$,* $\mathbf{Ex}_{w\times_o(u+_o\underline{1})} - \mathbf{Lim}_u\mathbf{Mex} \neq \emptyset$.

Proof.

Let $u, w \in O$ be such that $w$ is for $\omega$. We use $\mathcal{C}_u$ from Definition 27 to show that $\mathcal{C}_u \in \mathbf{Ex}_{w\times_o(u+_o\underline{1})} - \mathbf{Lim}_u\mathbf{Mex}$. The negative part of this follows from Lemma 30.

The positive part follows, in part, from a careful analysis of a construction from [FW79] as presented in [JORS99]. So that we do not need to present this construction again, in the remainder of this proof we proceed informally to show that $\mathcal{C}_u \in \mathbf{Ex}_{w\times_o(u+_o\underline{1})}$.

For $f \in \mathcal{C}_u$, from Parts (a) through (e) of Definition 27, we see that $\lim_{x\to\infty} \pi_2^4(f(x))\downarrow \geq$ a program for $f$, and that this convergence involves $\leq_o u$ mind changes. Hence, informally, the number of different values of $\pi_2^4(f(x))$ on the way to convergence is $\leq_o (u +_o \underline{1})$.

Proposition 10.7 of [JORS99, Page 227] generalizes a result from [FW79], and careful examination of the proof of this Proposition (for the special case from [FW79]) provides the existence of a machine we'll call $\mathbf{M}_{FW}$ with the following useful property. If $\mathbf{M}_{FW}$ receives as input *both* an upper bound on a program for a function $f$ *and* successively longer initial segments of $f$, then it converges to a program for $f$. Furthermore, in the process, $\mathbf{M}_{FW}$ makes $\leq_o w$ mind changes.

We present informally, then, a machine $\mathbf{M}$ which witnesses that $\mathcal{C}_u \in \mathbf{Ex}_{w\times_o(u+_o\underline{1})}$. For any $x \in N$ and any function $f$, $\mathbf{M}$, on $f[x+1]$, employs $\pi_2^4(f(x))$ as a *candidate* upper bound on a program for $f$ to feed to $\mathbf{M}_{FW}$ together with $f[x+1]$; then $\mathbf{M}$ outputs $\mathbf{M}_{FW}$'s resultant output. Each time $\pi_2^4(f(x))$ makes a mind change, the $\mathbf{M}_{FW}$ component of $\mathbf{M}$ starts over. From above, $\mathbf{M}$ need look at $\leq_o (u+_o\underline{1})$ different values of $\pi_2^4(f(x))$, and, for each of these, the $\mathbf{M}_{FW}$ component of $\mathbf{M}$ can be restricted to make $\leq_o w$ mind changes. Hence, $\mathbf{M}$ makes $\leq_o (w\times_o(u+_o\underline{1}))$ mind changes. To see this, think of $|w \times_o (u +_o \underline{1})|_o$ informally as $(|u|_o + 1)$ copies of $\omega$ laid out end to end; and the counting down for $\mathbf{M}$ as starting at the right end of the rightmost copy of $\omega$, jumping to right end of the next copy of $\omega$ to the left whenever $\pi_2^4(f(x))$ makes a mind change, and counting down inside its currently employed copy of $\omega$ when operating $\mathbf{M}_{FW}$ between $\pi_2^4(f(x))$'s mind changes. The *notation* $(w \times_o (u +_o \underline{1}))$ makes it possible to find ones way *algorithmically* from right to left across the corresponding ordinal conceptualized as $(|u|_o + 1)$ copies of $\omega$ laid out end to end. $\square$

Since for all $u \in O$, $\mathbf{Mex} \subseteq \mathbf{Lim}_u\mathbf{Mex}$, using Theorem 60, we get the following corollary to Theorem 65 above.

**Corollary 66** *For all $u, w \in O$ such that $w$ is for $\omega$,* $\mathbf{Lim}_u\mathbf{Mex} \not\subseteq \mathbf{Ex}_{w\times_o(u+_o\underline{1})}$.

We also note that the proof of Theorem 65 may be easily adapted to show that for all $u, v \in O$, such that $|v|_o \geq \omega$, we get $\mathbf{Ex}_{v\times_o(u+_o\underline{1})} - \mathbf{Lim}_u\mathbf{Mex} \neq \emptyset$.

It is open whether the comparisons of this section can be substantially improved.

# A Appendix

In this appendix, we prove some of the theorems we stated without proof in various sections of this paper. Included here are proofs of an example from the Section 1 and theorems about $+_o$, $\times_o$ and $\{\cdot\}_o$. The proofs of these latter theorems are not complicated. However, they require careful attention to (often tedious) detail. Transfinite induction on notations for the constructive ordinals forms the basis of these latter proofs (see [Kle44, Kle55, Sac90] for other examples of such inductions).

The following theorem (Theorem 67) is (informally) stated in Section 1.2 as an example of a $\lim_\omega$-computable function that is not $\lim_n$-computable, for any $n \in N$. Here, we state the correct formal version of this theorem next and also its proof.

**Theorem 67** *Let $K \stackrel{\text{def}}{=} \{i \mid \varphi_i(i)\downarrow\}$. Suppose $K = \{k_0 < k_1 < \ldots\}$. Let $k$ be the one argument total function such that for all $x$, $k(x) \stackrel{\text{def}}{=} k_x$. Then,*
*(a) For every notation $w$ for $\omega$, $k$ is $\lim_w$-computable.*
*(b) For every $n \in N$, $k$ is not $\lim_{\underline{n}}$-computable.*

Proof. Let $w$ be a notation for $\omega$. We shall construct a computable $h : (N \times N) \to N$, and a computable $O$-countdown function $\mathbf{F}$ that witness that $k$ is $\lim_w$-computable.

Firstly, let $g$ be a 1–1-computable function such that range$(g) = K$. Clearly, since $K$ is r.e. and infinite, such a $g$ exists. For all $j$, let $G_j \stackrel{\text{def}}{=} \{g(i) \mid i \leq j\}$. Let $G_j = \{g_j^0 < g_j^1 < \ldots < g_j^j\}$.

Then, for all $x, t$, let $h(x,t) \stackrel{\text{def}}{=} g_{x+t}^x$. For all $x, t$, let $\mathbf{F}(x,t) = \underline{h(x,t)}$.

From the definition of $g$ and $h$, we get that for all $x$, $h(x,\infty) = k_x$. Clearly, for all $x, t$, $\mathbf{F}(x,t) \leq_o \mathbf{F}(x,0) <_o w$. Also, for all $x$, for all $t > 0$, $\mathbf{F}(x,t) \leq_o \mathbf{F}(x,t-1)$. Furthermore, for all $x, t$, if $h(x,t) \neq h(x,t+1)$, then $\mathbf{F}(x,t+1) <_o \mathbf{F}(x,t)$.

Hence $k$ is $\lim_w$-computable as witnessed by $h$ and $\mathbf{F}$. Therefore, part (a) follows.

To show part (b), suppose by way of contradiction that $n$ is the least number in $N$ such that $k$ is $\lim_{\underline{n}}$-computable as witnessed by computable $h$. Therefore, for all $x$, $h(x,\infty) = k_x$. Clearly, $n \neq 0$, for otherwise, $K$ would be computable, which it is not. Also, we can conclude that

$$(\exists^\infty x)[\text{card}(\{t \mid h(x,t) \neq h(x,t+1)\}) = n], \tag{1}$$

since $k$ would be $\lim_{\underline{n-1}}$-computable otherwise, which would be a contradiction to our assumption that $n$ is the least value such that $k$ is $\lim_{\underline{n}}$-computable. We note that for each $x$ that satisfies equation (1), $h(x,\infty)$ can be algorithmically determined.

By the Kleene recursion theorem [Rog67], there exists a program $e$ that computes a partial computable function and behaves as follows (even though we define program $e$'s behavior on all inputs, we are really only interested in the behavior of program $e$ on input $e$).

Begin program $e$.

1. On any input $i$, search, by dovetailing, for an $x \geq 0$ such that $\text{card}(\{t \mid h(x,t) \neq h(x,t+1)\}) = n$ and $h(x,\infty) = k_x > e$. (Note: By equation (1) and the fact that $K$ is infinite, such an $x$ will eventually be found.)
2. If and when Step 1 succeeds, for the first such $x$ found, search for $x$ values in $K$ (using a fixed enumerator for $K$) each less than $k_x$ and not equal to $e$.
3. If and when Step 2 succeeds, output 0.

End program e.

*Case 1:* On input $e$, Step 2 of program $e$ eventually succeeds.

Then $e \in K$. Therefore, by Step 2, there are now at least $x+1$ numbers (including $e$) in $K$ that are less than $k_x$, a contradiction to the fact that there are exactly $x$ such numbers.

*Case 2:* On input $e$, Step 2 of program $e$ never succeeds

Therefore $e \notin K$. In this case, there are less than $x$ numbers in $K$ that are less than $k_x$, which is again a contradiction.

$\square$

We next restate Facts 7 and 8 here, as they are referred to quite heavily in the proofs below.

**Fact 7 (Kleene [Kle55])** *For all $x, y \in N$,*
*(a) $x <_o y \Rightarrow x \in O \wedge y \in O$.*
*(b) $x \in O \Rightarrow 1 \leq_o x$.*
*(c) $x <_o y \Rightarrow y \neq 1$.*
*(d) $x <_o 2^y \Rightarrow x \leq_o y$.*
*(e) $x <_o 3 \cdot 5^p \Rightarrow (\exists n)[x <_o \varphi_p(n)]$.*
*(f) $x \leq_o z \wedge y \leq_o z \Rightarrow x <_o y \vee x =_o y \vee x >_o y$.*

The following fact on notations [Rog67, Sac90] is also important to us.

**Fact 8 (Rogers [Rog67], Sacks [Sac90])** *There exist computable functions $h_1$ and $h_2$ such that, for all $v \in O$,*
*(a) $W_{h_1(v)} = \{u \mid u <_o v\}$;*
*(b) $W_{h_2(v)} = \{\langle u_1, u_2\rangle \mid u_1 <_o u_2 <_o v\}$ is a well-ordering isomorphic to $|v|_o$.*

We restate Theorem 12 from Section 2.5. We assume priorities on the various operators such as $+_o, \times_o, \in, =, \wedge, \Rightarrow, \ldots$, as in Section 2.5. Also, we use the following.

**Remark 68** *It is possible to define $+_o$, $\times_o$, and $\{\cdot\}_o$ such that the range of the $q$'s in Theorems 12, 14, and 16 are mutually disjoint. We assume that this is indeed the case. This assumption helps to simplify the proofs of Theorems 15 and 17; however, Theorems 15 and 17 are unaffected by this assumption. Clearly, Theorem 13 holds even with this assumption.*

**Theorem 12** *There exists a computable function $+_o$ such that, for all $x, y \in N$,*

$$x +_o y = \begin{cases} y, & \text{if } x = 1; \text{ (clause i)} \\ x, & \text{if } y = 1 \ \wedge \ x \neq 0 \ \wedge \ x \neq 1; \text{ (clause ii)} \\ 2^{(x +_o m)}, & \text{if } y = 2^m \ \wedge \ m \neq 0 \ \wedge \ x \neq 1; \text{ (clause iii)} \\ 3 \cdot 5^q, & \text{if } y = 3 \cdot 5^p \ \wedge \ x \neq 1, \text{ (clause iv)} \\ & (\text{where } (\forall n)[\varphi_q(n) = x +_o \varphi_p(n)] \ ); \\ 7, & \text{otherwise. (clause v)} \end{cases}$$

*Furthermore, $q$ in Clause (v) above is a computable, 1–1 function of $x$ and $p$.*

As we noted earlier, the 1–1-ness of $q$ just mentioned is crucial for proving the right-to-left implication of part (e) of Theorem 13, which in turn is necessary for proving many results that follow.

Theorem 13 is restated and proved next.

**Theorem 13** *For all $x, y$ and $z \in N$,*
*(a) $x, y \in O \Leftrightarrow x +_o y \in O$.*
*(b) $x, y \in O \Rightarrow |x +_o y|_o = |x|_o + |y|_o$.*
*(c) $x, y \in O \wedge y \neq 1 \Rightarrow x <_o x +_o y$.*
*(d) $x, y \in O \wedge z <_o y \Leftrightarrow x +_o z <_o x +_o y$.*
*(e) $x \in O \wedge z =_o y \Leftrightarrow x +_o z =_o x +_o y$.*
*(f) $y \in O \ \wedge \ x \leq_o z <_o x +_o y \Leftrightarrow (\exists y')[y' <_o y \wedge x +_o y' =_o z]$.*
*Furthermore, in the left-to-right implication of (f), $y'$ is unique.*

Proof of Theorem 13.

If $x = 1$, then $|x|_o = 0$. Therefore, for all $y$, $x +_o y = y$, and all the left-to-right (abbreviated $\Rightarrow$) and right-to-left (abbreviated $\Leftarrow$) parts of the theorem can be easily seen to be true. So, we only need to prove the theorem for $x \neq 1$.

We will prove, for $x \neq 1$, the left-to-right implications of Theorem 13 (abbreviated (a)$\Rightarrow$, (b)$\Rightarrow$, (c)$\Rightarrow$, (d)$\Rightarrow$, (e)$\Rightarrow$ and (f)$\Rightarrow$) first and then prove the right-to-left implications (abbreviated (a)$\Leftarrow$, (d)$\Leftarrow$, (e)$\Leftarrow$ and (f)$\Leftarrow$) next.

Proof of '$\Rightarrow$' implications of Theorem 13.

If any of $x$, $y$ or $z \notin O$, all the '$\Rightarrow$' implications are true; so we only need to prove the '$\Rightarrow$' implications when $x$, $y$ and $z \in O$. We use proof by transfinite induction on $y$ w.r.t. $<_o$-order.

Base Case: $y =_o 1$

Since $x \neq 1$ and $x \neq 0$, $x +_o y = x$. Hence, (a)$\Rightarrow$, (b)$\Rightarrow$ and (e)$\Rightarrow$ hold. (c)$\Rightarrow$ is trivial since $y = 1$. (d)$\Rightarrow$ is vacuously true by Fact 7(c). (f)$\Rightarrow$ is true since $x <_o x$ is false.

Inductive Case: $y =_o v \neq 1$, where $v \in O$.

Inductive Hypothesis (IH): For all $x, y$ and $z \in O$ such that $y <_o v$, (a)$\Rightarrow$, (b)$\Rightarrow$, (c)$\Rightarrow$, (d)$\Rightarrow$, (e)$\Rightarrow$ and (f)$\Rightarrow$ hold.

We will show that for all $x, z$, for $y =_o v$, all the '$\Rightarrow$ implications listed in IH above hold. To do this, we consider the following subcases based on the form of $y$.

Subcase 1: $y =_o 2^m$.

Therefore, $m \in O$ and $m <_o y$. Suppose $x \in O$. Then, $x +_o y = 2^{x +_o m}$. By IH(a)$\Rightarrow$, $x +_o m \in O$. Therefore, $x +_o y \in O$. Hence, (a)$\Rightarrow$ and (e)$\Rightarrow$ hold. Also, by IH(b)$\Rightarrow$, $|x +_o m|_o = |x|_o + |m|_o$; so, $|x +_o y|_o = |x|_o + |m|_o + 1 = |x|_o + |y|_o$. Therefore, (b)$\Rightarrow$ holds. By IH(c)$\Rightarrow$, $x \leq_o x +_o m$; so (c)$\Rightarrow$ holds.

Suppose $z <_o y$. Then, $z \leq_o m$. By IH(d)$\Rightarrow$ and IH(e)$\Rightarrow$, $x +_o z \leq_o x +_o m$; so (d)$\Rightarrow$ holds.

Suppose $y \in O$ and $x \leq_o z <_o x +_o y$. Then, $x \leq_o z \leq_o x +_o m <_o x +_o y$. If $z <_o x +_o m$, by IH(f)$\Rightarrow$, (f)$\Rightarrow$ holds; if $z = x +_o m$, then clearly, $(\exists y' = m <_o y)[z =_o x +_o y']$ and (f)$\Rightarrow$ holds.

Subcase 2: $y =_o 3 \cdot 5^p$.

Then, $x +_o y = 3 \cdot 5^q$, where for all $n$, $\varphi_q(n) = x +_o \varphi_p(n)$. Also, for all $n$, $\varphi_p(n) \in O$ and $\varphi_p(n) <_o \varphi_p(n+1) <_o 3 \cdot 5^p$. Therefore, by IH(a)$\Rightarrow$ and IH(d)$\Rightarrow$, for all $n$, $\varphi_q(n) \in O$ and $\varphi_q(n) <_o \varphi_q(n+1)$. Therefore, by the definition of $O$, $3 \cdot 5^q \in O$. Hence, (a)$\Rightarrow$ and (e)$\Rightarrow$ hold, and, for all $n$, $\varphi_q(n) <_o 3 \cdot 5^q = x +_o y$. Also, by IH(b)$\Rightarrow$, for all $n$, $|x + \varphi_p(n)|_o = |x|_o + |\varphi_p(n)|_o$. Therefore, $|x +_o y|_o = \lim_{n \to \infty} |x +_o \varphi_p(n)|_o$, = (by IH(b)$\Rightarrow$) $\lim_{n \to \infty}(|x|_o + |\varphi_p(n)|_o) = |x|_o + |y|_o$. Hence, (b)$\Rightarrow$ holds. By IH(c)$\Rightarrow$, $x \leq_o x +_o \varphi_p(0)$, which is $<_o x +_o y$. So, (c)$\Rightarrow$ holds.

Suppose $z <_o y$. Then, by Fact 7(e), let $n$ be such that $z <_o \varphi_p(n)$. By IH(d)$\Rightarrow$, $x +_o z <_o x +_o \varphi_p(n)$, which is $<_o x +_o y$; so (d)$\Rightarrow$ holds.

Suppose $y \in O$ and $x \leq_o z <_o x +_o y$. Then, by Fact 7(e), let $n$ be such that $x \leq_o z <_o x +_o \varphi_p(n)$. By IH(f)$\Rightarrow$, (f)$\Rightarrow$ holds. $\square$ ('$\Rightarrow$' IMPLICATIONS OF THEOREM 13)

We next prove the '$\Leftarrow$' implications.

PROOF OF '$\Leftarrow$' IMPLICATIONS OF THEOREM 13.

We note that (f)$\Leftarrow$ follows trivially from (c)$\Rightarrow$, (d)$\Rightarrow$ and (a)$\Leftarrow$. Also, if $x +_o y \notin O$, then (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ are trivially true. Therefore, we need to only consider the cases when $x +_o y \in O$. Hence, we next prove (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ by transfinite induction on $w = x +_o y$ w.r.t. $<_o$-order.

We will prove the '$\Leftarrow$' implications by induction on $w =_o x +_o y$ when $w$ is in the range of $+_o$. For each such $w$, for each $x, y$ such that $w =_o x +_o y$, we will prove that the '$\Leftarrow$' implications hold.

Since 1 is the least notation in $O$ (w.r.t. $<_o$ order) in the range of $+_o$, we begin the induction with $w =_o 1$.

Base Case: $w =_o 1$.

Clearly, Clauses (iv) and (v) in the definition of $+_o$ cannot apply to give $w = 1$. Clause (iii) cannot apply for otherwise, $x +_o m = 0$ for some $m \neq 0$, which is impossible. Clause (ii) cannot apply either, for otherwise $x +_o y = x \neq 1$, a contradiction. Therefore, Clause (i) must apply. Hence, $x = 1$ and all '$\Leftarrow$' implications are true as claimed at the beginning of the proof of this theorem.

Inductive Case: $w \neq 1$ and $w \in O$.

Inductive Hypothesis (IH): For all $x, y$ such that $x +_o y <_o w$, (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ hold.

We will show that for all $x, y$ such that $x +_o y =_o w$, the '$\Leftarrow$' implications listed in the IH hold. Suppose $x$ and $y$ are such that $x +_o y =_o w$. We consider subcases based on which clause of $+_o$ applies to define $w$.

Subcase 1: $x = 1$.

Then, as stated earlier in this proof, all the '$\Leftarrow$' implications are true.

Subcase 2: $y = 1 \ \wedge \ x \neq 0 \ \wedge \ x \neq 1$.

In this case, $w = x +_o y = x$. Clearly, $x +_o y \in O$ implies that $x \in O$. Therefore, (a)$\Leftarrow$ holds.

Suppose $z$ is such that $x +_o z <_o x +_o y$. Therefore, by IH(a)$\Leftarrow$, $z \in O$. Using (c)$\Rightarrow$, we get $x \leq_o x +_o z$. Also, by the definition of $+_o$, $x +_o y = x +_o 1 = x$. Therefore, $x +_o z \geq_o x +_o y$ and hence (d)$\Leftarrow$ holds.

All that remains in this case is to prove (e)$\Leftarrow$.

*Subsubcase (i):* $x$ is of the form $2^m$, for some $m \neq 0$.

Suppose $x +_o z =_o x +_o y = x = 2^m$. Then, only Clauses (ii) and (iii) in the definition of $+_o$ can apply to give $x +_o z = x$. If Clause (ii) applies, then $z = 1$ and (e)$\Leftarrow$ holds. If Clause (iii) applies, then $z = 2^n$ for some $n \neq 0$. Also, $x +_o z = 2^{x +_o n} = x$. Since $x +_o z \in O$, $2^{x +_o n} \in O$. Therefore, $x +_o n \in O$. Also, $x +_o n <_o x +_o z = x +_o y = w$. Hence, by IH(a)$\Leftarrow$, $n \in O$. Further, we have $x +_o n <_o x$, a contradiction to (c)$\Rightarrow$. Hence, (e)$\Leftarrow$ holds.

*Subsubcase (ii):* $x$ is of the form $3 \cdot 5^q$.

Suppose $x +_o z =_o x +_o y = x = 3 \cdot 5^q$. Then, only Clauses (ii) and (iv) can apply to give $x +_o z = 3 \cdot 5^q$. If Clause (ii) applies, then $z = 1$ and (e)$\Leftarrow$ holds. If Clause (iv) applies, then $z = 3 \cdot 5^p$ for some $p$. Also, $x +_o z = 3 \cdot 5^q = x$. Since $3 \cdot 5^q \in O$, for all $n$, $x +_o \varphi_p(n) \in O$ and $x +_o \varphi_p(n) <_o x +_o y = w$. Therefore, by IH(a)$\Leftarrow$, for all $n$, $\varphi_p(n) \in O$. Further, we have, for all $n$, $x +_o \varphi_p(n) <_o x$, a contradiction to (c)$\Rightarrow$. Hence, (e)$\Leftarrow$ holds.

Subcase 3: $y = 2^m \ \wedge \ m \neq 0 \ \wedge \ x \neq 1$.

In this case, $w = x +_o y = 2^{x +_o m}$. If $x +_o y \in O$, then $x +_o m \in O$ and hence, by IH(a)$\Leftarrow$, $x$ and $m \in O$. Therefore, (a)$\Leftarrow$ holds.

Suppose $x +_o z <_o x +_o y$. Then, $x +_o z \leq_o x +_o m$. By IH(d)$\Leftarrow$ and IH(e)$\Leftarrow$, $z \leq_o m$; so (d)$\Leftarrow$ holds.

Suppose $x +_o z =_o x +_o y$. Since $x \neq 1$, Clause (i) cannot apply. Clause (ii) cannot apply either, since in that case, $x +_o z = x = x +_o y$, a contradiction to (c)$\Rightarrow$ and the fact (by IH(a)$\Leftarrow$) that $y \in O$. Therefore, Clause (iii) must apply; so $z = 2^{m'}$ for some $m' \neq 0$ and $x +_o z = 2^{x +_o m'}$. Since $2^{x +_o m'} = 2^{x +_o m}$, we get $m' = m$; so (e)$\Leftarrow$ holds.

Subcase 4: $y = 3 \cdot 5^p \ \wedge \ x \neq 1$.

In this case, $x +_o y = 3 \cdot 5^q$, where, for all $n$, $\varphi_q(n) = x +_o \varphi_p(n)$. If $x +_o y \in O$, then for all $n$, $\varphi_q(n) \in O$ and $\varphi_q(n) <_o \varphi_q(n+1) <_o 3 \cdot 5^q$. Hence, for all $n$, $x +_o \varphi_p(n) \in O$; so, by IH(a)$\Leftarrow$, $x \in O$ and for all $n$, $\varphi_p(n) \in O$. Similarly, by IH(d)$\Leftarrow$, for all $n$, $\varphi_p(n) <_o \varphi_p(n+1)$. Therefore, $y \in O$ and (a)$\Leftarrow$ holds.

Suppose $x +_o z <_o x +_o y = 3 \cdot 5^q$. Then, by Fact 7(e), let $n$ be such that $x +_o z <_o \varphi_q(n) = x +_o \varphi_p(n)$; therefore, by IH(d)$\Leftarrow$, (d)$\Leftarrow$ holds.

Suppose $x +_o z =_o x +_o y = 3 \cdot 5^q \in O$. Clause (i) cannot apply to give $x +_o z = 3 \cdot 5^q$ since $x \neq 1$. Clause (ii) cannot apply either, since in that case, $x +_o z = x = x +_o y$, a contradiction to (c)$\Rightarrow$ and the fact (by IH(a)$\Leftarrow$) that $y \in O$. Therefore, Clause (iv) must apply; so $z = 3 \cdot 5^{p'}$ for some $p'$. Hence, $x +_o 3 \cdot 5^{p'} = x +_o 3 \cdot 5^p = 3 \cdot 5^q$. Since, in the definition of $+_o$, $q$ is a 1–1 function of $p$ and $x$, $p' = p$; so (e)$\Leftarrow$ holds.

As stated earlier, (f)$\Leftarrow$ follows from (c)$\Rightarrow$, (d)$\Rightarrow$ and (a)$\Leftarrow$.

$\square$ ('$\Leftarrow$' IMPLICATIONS OF THEOREM 13)

Thus, all the '$\Leftarrow$' implications hold. Finally, we prove the 'furthermore' part of the theorem.

Suppose by way of contradiction that $x \leq_o z <_o x +_o y$ and there exist $y'' \neq y'$ such that $y' <_o y \ \wedge \ x +_o y' =_o z$ and $y'' <_o y \ \wedge \ x +_o y'' =_o z$. By part (e)$\Leftarrow$, $y' = y''$, a contradiction. Hence, the 'furthermore' part holds. $\square$ (THEOREM 13)

We restate Theorem 14 next.

**Theorem 14** *There is a computable function $\times_o$ such that, for all $x, y \in N$,*

$$x \times_o y = \begin{cases} 1, & \text{if } y = 1 \ \vee \ x = 1; \text{ (clause i)} \\ y, & \text{if } x = 2 \ \wedge \ y \neq 1; \text{ (clause ii)} \\ (x \times_o m) +_o x, & \text{if } y = 2^m \ \wedge \ m \neq 0 \ \wedge \\ & \quad x \neq 1 \ \wedge \ x \neq 2; \text{ (clause iii)} \\ 3 \cdot 5^q, & \text{if } y = 3 \cdot 5^p \ \wedge \ x \neq 1 \ \wedge \ x \neq 2, \text{ (clause iv)} \\ & (\text{where } (\forall n)[\varphi_q(n) = x \times_o \varphi_p(n)] \ ); \\ 7, & \text{otherwise. (clause v)} \end{cases}$$

*Furthermore, $q$ as in Clause (iv) is a computable, 1–1 function of $x$ and $p$.*

Again, as mentioned earlier, the 1–1-ness of $q$ just mentioned is crucial for proving the right-to-left implication of part (e) of Theorem 15, which in turn is necessary for proving many results that follow.

We next restate and prove Theorem 15.

**Theorem 15** *For all $x, y$ and $z \in N$,*

*(a) $x, y \in O \ \vee \ x = 1 \ \vee \ y = 1 \Leftrightarrow x \times_o y \in O$.*
*(b) $x, y \in O \Rightarrow |x \times_o y|_o = |x|_o \times |y|_o$.*
*(c) $x >_o 1 \ \wedge \ y >_o 2 \Rightarrow x <_o x \times_o y$.*
*(d) $x >_o 1 \ \wedge \ y \in O \ \wedge \ z <_o y \Leftrightarrow x \times_o z <_o x \times_o y$.*
*(e) $x = 1 \ \vee \ (x \in O \ \wedge \ z =_o y) \Leftrightarrow x \times_o z =_o x \times_o y$.*
*(f) $y \in O \ \wedge \ x \leq_o z <_o x \times_o y \Leftrightarrow (\exists x' <_o x)(\exists y' \mid 1 <_o y' <_o y)[z =_o (x \times_o y') +_o x']$.*
*Furthermore, in (f)$\Rightarrow$, $x'$ and $y'$ are both unique.*

Proof of Theorem 15.

If $x = 1$, then $x \times_o y = 1$ and all the parts of the theorem follow easily. Similarly, if $x = 2$, it is easy to verify that all the parts of the theorem hold. Also, the '$\Rightarrow$' implications are trivially true if any of $x$, $y$ or $z \notin O$. So, it is sufficient to prove the '$\Rightarrow$' implications for cases where $x$, $y$ and $z \in O$, $x \neq 1$ and $x \neq 2$. We do this first by transfinite induction on $y$ w.r.t. $<_o$-order.

Proof of '$\Rightarrow$' implications of Theorem 15.

Base Case: $y =_o 1$.

Since $y = 1$, $x \times_o y = 1$. Hence, (a)$\Rightarrow$, (b)$\Rightarrow$ and (e)$\Rightarrow$ hold. (c)$\Rightarrow$ is trivial since $y >_o 2$ is false. (d)$\Rightarrow$ and (f)$\Rightarrow$ are vacuously true by Fact 7(c).

Inductive Case: $y = v \neq 1$, where $v \in O$.

Inductive Hypothesis (IH): For all $x$, $y$ and $z \in O$ such that $y <_o v$, (a)$\Rightarrow$, (b)$\Rightarrow$, (c)$\Rightarrow$, (d)$\Rightarrow$, (e)$\Rightarrow$, and (f)$\Rightarrow$ hold.

We will show that for all $x, z \in O$, for $y =_o v$, the implications (a)$\Rightarrow$, (b)$\Rightarrow$, (c)$\Rightarrow$, (d)$\Rightarrow$, (e)$\Rightarrow$ and (f)$\Rightarrow$ hold. To do this, we consider the following cases based on the form of $y$.

Subcase 1: $y =_o 2^m$.

If $x = 1$ or $x = 2$, then all the '$\Rightarrow$' implications hold, as proved earlier. So we may assume that $x >_o 2$.

Therefore, $x \times_o y = (x \times_o m) +_o x$. Since $x$, $y \in O$, we get $m \in O$ and $m <_o y$. By IH(a)$\Rightarrow$, $x \times_o m \in O$. Therefore, by Theorem 13(a), $x \times_o y \in O$. Therefore (a)$\Rightarrow$ and (e)$\Rightarrow$ hold. Also, by Theorem 13(b) and IH(b)$\Rightarrow$, $|x \times_o y|_o = (|x|_o \times |m|_o) + |x|_o = |x|_o \times (|m|_o + 1) = |x|_o \times |2^m|_o = |x|_o \times |y|_o$. Therefore, (b)$\Rightarrow$ holds.

We have $x >_o 1$. Suppose $y >_o 2$. Then, $2^m >_o 2$. Hence, $m \geq_o 2$. Also, $x \times_o y = (x \times_o m) +_o x$. The following two subcases are possible.

*Subsubcase (i): $m = 2$.*

Then, $x \times_o m = (x \times_o 1) +_o x = 1 +_o x = x$. So, $(x \times_o m) +_o x = x +_o x$. By Theorem 13(c), $x <_o x +_o x$. Therefore, $x <_o x \times_o y$.

*Subsubcase (ii): $m >_o 2$.*

Then, by IH(c)$\Rightarrow$, $x <_o x \times_o m$. Therefore, by Theorem 13(c), $x \times_o m <_o (x \times_o m) +_o x$. Therefore, $x <_o x \times_o y$.

Therefore, (c)$\Rightarrow$ holds.

We have $x >_o 1$ and $y \in O$. Suppose $z <_o y$. Then, $x \times_o y = (x \times_o m) +_o x$. Also, $y = 2^m$ and $m <_o y$. Hence, $z <_o y$ implies that $z \leq_o m$. By IH(d)$\Rightarrow$ and IH(e)$\Rightarrow$, $x \times_o z \leq_o x \times_o m$. Hence, by Theorem 13(c), $x \times_o m <_o (x \times_o m) +_o x$. Therefore, $x \times_o z <_o x \times_o y$. Hence, (d)$\Rightarrow$ holds.

Suppose $x \leq_o z <_o x \times_o y$. Hence, $x \leq_o z <_o (x \times_o m) +_o x$. Therefore, $m >_o 1$; for otherwise, $x \leq_o z <_o 1 +_o x = x$, a contradiction. By Fact 7(f) applied to $(z <_o (x \times_o m) +_o x) \ \wedge \ (x \times_o m <_o (x \times_o m) +_o x)$, we conclude that either $x \times_o m \leq_o z$ or $z <_o x \times_o m$.

*Subsubcase (iii):* $x \times_o m \leq_o z$.

Then, $x \times_o m \leq_o z <_o (x \times_o m) +_o x$. By Theorem 13(f), $\exists x' <_o x$ such that $z =_o (x \times_o m) +_o x'$. Therefore, $(\exists y' = m \mid 1 <_o y' <_o y)(\exists x' <_o x)[z =_o (x \times_o y') +_o x']$.

*Subsubcase (iv):* $z <_o x \times_o m$.

Then, $x \leq_o z <_o x \times_o m$. By IH(f), $(\exists y' \mid 1 <_o y' <_o m)(\exists x' <_o x)[z =_o (x \times_o y') +_o x']$.

Therefore, (f)$\Rightarrow$ holds.

<u>Subcase 2:</u> $y =_o 3 \cdot 5^p$.

Again, we may assume that $x \neq 1$ and $x \neq 2$. In this case, for all $n$, $\varphi_p(n) \in O$ and $\varphi_p(n) <_o \varphi_p(n+1) <_o y = 3 \cdot 5^p$. Also, $x \times_o y = 3 \cdot 5^q$, where for all $n$, $\varphi_q(n) = x \times_o \varphi_p(n)$.

Since $x \in O$, by IH(a)$\Rightarrow$, for all $n$, $\varphi_q(n) \in O$. By our assumption, $x >_o 1$. Therefore, by IH(d)$\Rightarrow$, for all $n$, $\varphi_q(n) <_o \varphi_q(n+1)$. Therefore, $3 \cdot 5^q \in O$. Hence, (a)$\Rightarrow$ and (e)$\Rightarrow$ hold. Also, for all $n$, $\varphi_q(n) <_o x \times_o y$.

Since $x \in O$, $|x \times_o y|_o = |3 \cdot 5^q|_o = \lim_{n\to\infty}(|x \times_o \varphi_p(n)|_o)$, = (by IH(b)$\Rightarrow$) $\lim_{n\to\infty}(|x|_o \times |\varphi_p(n)|_o) = |x|_o \times \lim_{n\to\infty}(|\varphi_p(n)|_o) = |x|_o \times |y|_o$. Therefore, (b)$\Rightarrow$ holds.

$x >_o 1$, and clearly, $y = 3 \cdot 5^p >_o 2$. By Fact 7(e), let $n$ be such that $2 <_o \varphi_p(n)$. Hence, by IH(c)$\Rightarrow$, $x <_o x \times_o \varphi_p(n) <_o x \times_o y$. Therefore, (c)$\Rightarrow$ holds.

We have $x >_o 1$. Suppose $z <_o y = 3 \cdot 5^p$. By Fact 7(e), let $n$ be such that $z <_o \varphi_p(n)$. By IH(d)$\Rightarrow$, $x \times_o z <_o x \times_o \varphi_p(n)$. Therefore, $x \times_o z <_o x \times_o y$. Therefore, (d)$\Rightarrow$ holds.

Suppose $x \leq_o z <_o x \times_o y = 3 \cdot 5^q$. Let $n$ be such that $z <_o x \times_o \varphi_p(n)$. By IH(f)$\Rightarrow$, $(\exists y' \mid 1 <_o y' <_o \varphi_p(n))(\exists x' <_o x)[z =_o (x \times_o y') +_o x']$. Therefore, $(\exists y' \mid 1 <_o y' <_o y)(\exists x' <_o x)[z =_o (x \times_o y') +_o x']$. Therefore, (f)$\Rightarrow$ holds.

This completes the proof of all the '$\Rightarrow$' implications of Theorem 15.

$\square$ ('$\Rightarrow$' IMPLICATIONS OF THEOREM 15)

PROOF OF '$\Leftarrow$' IMPLICATIONS OF THEOREM 15.

We note that if $x \times_o y \notin O$, (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ are trivially true. Also, as we shall show later, (f)$\Leftarrow$ can be easily proved using Theorem 13 and the other parts of this Theorem. Hence, we next prove (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ by transfinite induction on $w =_o x \times_o y$ w.r.t. $<_o$-order. Since 1 is the least notation in $O$ (w.r.t. $<_o$ order) in the range of $\times_o$, we begin the induction with $w = 1$.

<u>Base Case:</u> $w = 1$.

$w = 1$ is possible only via Clause (i) of the definition of $\times_o$. This is true by the following reasoning. Clearly, Clauses (iv) and (v) are ruled out. Clause (ii) cannot hold since that would give $y = 1$ and $y \neq 1$, a contradiction. If Clause (iii) were to hold, then $y = 2^m$ for some $m \neq 0$, $x \neq 1$ and $x \neq 2$; also, $(x \times_o m) +_o x = 1$; by the definition of $+_o$, this is possible only if $x \times_o m = 1$ and $x = 1$, a contradiction[27].

Hence, only Clause (i) can hold to give $x \times_o y = 1$. Hence, $x = 1$ or $y = 1$. If $x = 1$, as claimed earlier, all the '$\Leftarrow$' implications are true. Hence suppose $y = 1$. Then, clearly, (a)$\Leftarrow$ holds. (d)$\Leftarrow$ is vacuously true by Fact 7(c). To prove (e)$\Leftarrow$, suppose $x \times_o z =_o 1$. By reasoning similar to that in the previous paragraph, either $x = 1$ or $z = 1$. In either case, (e)$\Leftarrow$ holds.

[27] PROOF. Only Clauses (i), (ii) and (iii) in the definition of $+_o$ can apply to give $(x \times_o m) +_o x = 1$. Clause (ii) is eliminated since in that case we get $x \times_o m \neq 1$ and $x \times_o m = 1$. If Clause (iii) were to apply, then $x \times_o m \neq 1$, $x = 2^{m'}$ for some $m' \neq 0$ and $(x \times_o m) +_o m' = 0$. Only Clause (i) can apply to give $(x \times_o m) +_o m' = 0$, in which case we get $x \times_o m = 1$, a contradiction. $\square$

Inductive Case: $w \neq 1$ and $w \in O$.
Inductive Hypothesis (IH): For all $x, y$ such that $x \times_o y <_o w$, (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ hold.

We will show that for all $x, y$ such that $x \times_o y =_o w$, (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$ hold. Suppose $x$ and $y$ are such that $x +_o y =_o w$. We consider subcases based on which clause of $\times_o$ applies to define $w$.

Subcase 1: $x = 2 \ \wedge \ y \neq 1$.

Since $x = 2$, as claimed earlier, all the '$\Leftarrow$' implications hold.

Subcase 2: $y = 2^m \ \wedge \ m \neq 0 \ \wedge \ x \neq 1 \ \wedge \ x \neq 2$.

In this case, $w = x \times_o y = (x \times_o m) +_o x$. Since $x \times_o y \in O$, by Theorem 13(a), $x \times_o m \in O$ and $x \in O$. By Theorem 13(c), $x \times_o m <_o w$; therefore, by IH(a)$\Leftarrow$, $m \in O$. Hence, $2^m = y \in O$. Therefore, (a)$\Leftarrow$ holds.

Suppose $x \times_o z <_o x \times_o y$. By (a)$\Leftarrow$, $x$ and $y \in O$. By IH(a)$\Leftarrow$, $z \in O$. Also, $m <_o y$. Therefore, by (d)$\Rightarrow$, $x \times_o m <_o x \times_o y$. By Fact 7(f), either $x \times_o m <_o x \times_o z$ or $x \times_o z \leq_o x \times_o m$.

If $x \times_o m <_o x \times_o z$, then by IH(d)$\Leftarrow$, $m <_o z$; hence, $y \leq_o z$. So, by (d)$\Rightarrow$ and (e)$\Rightarrow$, we get $x \times_o y \leq_o x \times_o z$, a contradiction. Therefore, $x \times_o z \leq_o x \times_o m$. Hence, by IH(d)$\Leftarrow$ and IH(e)$\Leftarrow$, $z \leq_o m$. Therefore (d)$\Leftarrow$ holds.

Suppose $x \times_o z =_o x \times_o y$. By (d)$\Rightarrow$, since $m <_o y$, we get $x \times_o m <_o x \times_o y$. Also, $x \times_o z = (x \times_o m) +_o x$. By (a)$\Leftarrow$, $x$ and $y \in O$. If $z$ is of the form $3 \cdot 5^p$ for some $p$, then Clause (iv) of the definition of $\times_o$ applies and $x \times_o z$ is of the form $3 \cdot 5^q$, for some $q$; hence $(x \times_o m) +_o x$ is also equal to $3 \cdot 5^q$, for the same $q$. However, this is a contradiction to the fact that the range of $q$ of $+_o$ is disjoint from the range of $q$ of $\times_o$ (see Remark 68). Hence, Clause (iv) cannot apply; so only Clause (iii) can apply.

Hence, $z = 2^n$ for some $n \neq 0$. So, $x \times_o z = (x \times_o n) +_o x \in O$. Therefore, by Theorem 13(a), $x \times_o n \in O$.

Also, by Theorem 13(c), $x \times_o n <_o (x \times_o n) +_o x$. Therefore, $x \times_o n <_o x \times_o z = x \times_o y$. By IH(d)$\Leftarrow$, $n <_o y$. Therefore, $2^n = z \in O$. Also, $z = 2^n \leq_o y$. If $z <_o y$, then by (d)$\Rightarrow$, $x \times_o z <_o x \times_o y$, a contradiction. Hence, $z = y$. Therefore, (e)$\Leftarrow$ holds.

Subcase 3: $y = 3 \cdot 5^p \ \wedge \ x \neq 1 \ \wedge \ x \neq 2$.

In this case, $w = x \times_o y = 3 \cdot 5^q$, where, for all $n$, $\varphi_q(n) = x \times_o \varphi_p(n)$.

Since $w = x \times_o y \in O$, we have for all $n$, $\varphi_q(n) \in O$. Therefore, for all $n$, $x \times_o \varphi_p(n) \in O$. By IH(a)$\Leftarrow$, $x \in O$ and for all $n$, $\varphi_p(n) \in O$. Also, for all $n$, $\varphi_q(n) <_o \varphi_q(n+1)$; so, by IH(d)$\Leftarrow$, for all $n$, $\varphi_p(n) <_o \varphi_p(n+1)$. Therefore, $3 \cdot 5^p \in O$. Therefore, (a)$\Leftarrow$ holds.

Suppose $w = x \times_o z <_o x \times_o y$. By (a)$\Leftarrow$, $x, y \in O$. By IH(a)$\Leftarrow$, $z \in O$. By Fact 7(e), let $n$ be such that $x \times_o z <_o x \times_o \varphi_p(n)$. By IH(d)$\Leftarrow$, $z <_o \varphi_p(n)$. Since $\varphi_p(n) <_o y$, we have $z <_o y$; hence (d)$\Leftarrow$ holds.

Suppose $w = x \times_o z =_o x \times_o y$. Then, $x \times_o z = 3 \cdot 5^q$. By our assumption that the range of $q$ in Theorem 12 is disjoint from the range of $q$ in Theorem 14 (see Remark 68), only Clause (iv) in Theorem 14 can hold to give $x \times_o z = 3 \cdot 5^q$. Therefore, for some $r$, $z = 3 \cdot 5^r$, where for all $n$, $\varphi_q(n) = x \times_o \varphi_r(n)$.

Since, in the definition of $\times_o$, $q$ is a 1–1 computable function of $x$ and $p$, we conclude that $r = p$. Hence, $3 \cdot 5^r = 3 \cdot 5^p$. Therefore, $z = y$. Therefore, (e)$\Leftarrow$ holds.

This completes the proof of (a)$\Leftarrow$, (d)$\Leftarrow$ and (e)$\Leftarrow$. We next prove (f)$\Leftarrow$.

Suppose $(\exists x' <_o x)(\exists y' \mid 1 <_o y' <_o y)[z =_o (x \times_o y') +_o x']$. Therefore, by Theorem 13(a) and (a)$\Leftarrow$, $x, x'$ and $y \in O$. Clearly, by Theorem 13(c), $x \times_o y' \leq_o (x \times_o y') +_o x'$. Also, $x >_o 1$ and $y' >_o 1$. If $y' >_o 2$, then by (c)$\Rightarrow$, $x <_o x \times_o y'$. If $y' = 2$, then by the definition of $\times_o$, $x \times_o y' = x$.

Therefore, $x \leq_o x \times_o y'$. Hence, $x \leq_o (x \times_o y') +_o x' = z$.

If $x = 2$, then $x' = 1$. Hence, $z = y'$ and $x \times_o y = y$. Therefore, $z <_o x \times_o y$. If $x >_o 2$, then $x \times_o 2^{y'} = (x \times_o y') +_o x$, which, by Theorem 13(d), is $>_o (x \times_o y') +_o x'$. Also, since $y' <_o y$, $2^{y'} \leq_o y$. Hence, by (d)$\Rightarrow$ and (e)$\Rightarrow$, $x \times_o 2^{y'} \leq_o x \times_o y$. Therefore, $(x \times_o y') +_o x' <_o x \times_o y$.

Hence, (f)$\Leftarrow$ holds. Therefore, all the '$\Leftarrow$' implications hold.

□ ('$\Leftarrow$' IMPLICATIONS OF THEOREM 15)

Finally, we prove the 'furthermore' part of the theorem.

Suppose by way of contradiction that there exists $x, x', x'', y, y', y''$ and $z$ such that $y \in O$, $x \leq_o z <_o x \times_o y$, $(x' \neq x'' \ \vee \ y' \neq y'')$, $x' <_o x$, $x'' <_o x$, $1 <_o y' <_o y$, $1 <_o y'' <_o y$ and $z =_o (x \times_o y') +_o x' =_o (x \times_o y'') +_o x''$.

If $y' = y''$, then by (e)$\Rightarrow$, $x \times_o y' = x \times_o y''$. Hence, $z = (x \times_o y') +_o x' = (x \times_o y'') +_o x' = (x \times_o y'') +_o x''$. Therefore, by Theorem 13(e), $x' = x''$, which is a contradiction to our assumption that either $y' \neq y''$ or $x' \neq x''$.

Therefore, suppose without loss of generality that $y' <_o y''$. Hence, $1 <_o y' <_o y'' <_o y$. Clearly $x >_o 1$. If $x =_o 2$, then $x' = x'' =_o 1$; also, from the definition of $\times_o$, we get $x \times_o y' = y'$ and $x \times_o y'' = y''$. Therefore, from the definition of $+_o$, we have $z = (x \times_o y') +_o x' = y' +_o x' = y' +_o 1 = y'$, since $y' \neq 1$. Similarly, $z = (x \times_o y'') +_o x'' = y'' +_o 1 = y''$, a contradiction to our assumption that $y' <_o y''$. So suppose $x >_o 2$. Since $y' \neq 0$, from the definition of $\times_o$, we have $x \times_o 2^{y'} = (x \times_o y') +_o x$. Also, since $y' <_o y''$, we have $2^{y'} \leq_o y''$. Therefore, by (d)$\Rightarrow$ and (e)$\Rightarrow$, $x \times_o 2^{y'} \leq_o x \times_o y''$. Since $x' <_o x$, using Theorem 13, we have $z = (x \times_o y') +_o x' <_o (x \times_o y') +_o x = x \times_o 2^{y'} \leq_o x \times_o y'' \leq_o (x \times_o y'') +_o x'' = z$, a contradiction.

This proves the 'furthermore' part of the theorem. □ (THEOREM 15)

Theorem 16 is restated below.

**Theorem 16** *There is a computable function $\{\cdot^{\cdot}\}_o$ such that, for all $x, y \in N$,*

$$\{x^y\}_o = \begin{cases} 1, & \textit{if } x = 1 \ \wedge \ y \neq 1; \text{ (clause i)} \\ 2, & \textit{if } x = 2 \ \vee \ (y = 1 \ \wedge \\ & \quad x \neq 1 \ \wedge \ x \neq 2); \text{ (clause ii)} \\ \{x^m\}_o \times_o x, & \textit{if } y = 2^m \ \wedge \ m \neq 0 \ \wedge \\ & \quad x \neq 1 \ \wedge \ x \neq 2; \text{ (clause iii)} \\ 3 \cdot 5^q, & \textit{if } y = 3 \cdot 5^p \ \wedge \ x \neq 1 \ \wedge \ x \neq 2, \text{ (clause iv)} \\ & (\textit{where } (\forall n)[\varphi_q(n) = \{x^{\varphi_p(n)}\}_o] \ ); \\ 7, & \textit{otherwise.} \text{ (clause v)} \end{cases}$$

*Furthermore, $q$ as in Clause (iv), is a computable, 1–1 function of $x$ and $p$.*

As mentioned earlier, the 1–1-ness of $q$ just mentioned is crucial for proving the right-to-left implication of part (e) of Theorem 17.

We restate and prove Theorem 17 next.

**Theorem 17** *For all $x, y$ and $z \in N$,*
*(a) $x, y \in O \ \wedge \ (x \neq 1 \ \vee \ y \neq 1) \Rightarrow \{x^y\}_o \in O$.*
*(b) $x, y \in O \ \wedge \ (x \neq 1 \ \vee \ y \neq 1) \Rightarrow |\{x^y\}_o|_o = |x|_o^{|y|_o}$.*
*(c) $x >_o 2 \ \wedge \ y >_o 2 \Rightarrow x <_o \{x^y\}_o$.*
*(d) $x >_o 2 \ \wedge \ z <_o y \Rightarrow \{x^z\}_o <_o \{x^y\}_o$.*
*(e) $x >_o 2 \ \wedge \ z = y \in O \Rightarrow \{x^z\}_o = \{x^y\}_o \in O$.*
*(f) $x >_o 1 \ \wedge \ x' <_o x \ \wedge \ y \in O \Rightarrow \{x^y\}_o \times_o x' <_o \{x^{(y +_o 2)}\}_o$.*

As mentioned earlier, with modifications for special cases, the right-to-left implications of parts (a), (d), (e) and (f) of Theorem 17 are true, but not necessary for this work. So also is a property similar to Theorem 15(f).

Proof of '$\Rightarrow$' implications of Theorem 17.

If $x = 1$, then for $y \neq 1$, $\{x^y\}_o = 1$ and all the parts of the theorem can easily seen to be true. Similarly, if $x = 2$, then for all $y$, $\{x^y\}_o = 2$ and all the parts of the theorem easily follow. Also, if any of $x$, $y$ or $z \notin O$, all the implications are true; so we only need to prove the implications in cases where $x$, $y$ and $z \in O$, where $x \neq 1$ and $x \neq 2$.

For such cases, we will prove parts (a), (b), (d) and (e) by transfinite induction on $y$ w.r.t. $<_o$. Parts (c) and (f) will then be proved using the other parts.

Base Case: $y =_o 1$.

Since $x \neq 1$ and $x \neq 2$, from Clause (ii) of the definition of $\{\cdot\}_o$, $\{x^y\}_o = 2 \in O$. Hence (a) and (e) hold. $|\{x^y\}_o|_o = |2|_o = 1$; $|x|_o^{|y|_o} = |x|_o^0 = 1$, since $|x|_o \neq 0$. Hence, (b) holds. (d) is vacuously true by Fact 7(c).

Inductive Case: $y =_o v \neq 1$, where $v \in O$.

Inductive Hypothesis (IH): For all $x$, $y$ and $z \in O$ such that $y <_o v$, (a), (b), (d) and (e) hold.

We will show that for all $x$, $z$, for $y =_o v$, (a), (b), (d) and (e) hold. To do this, we consider the following subcases based on the form of $y$.

Subcase 1: $y =_o 2^m$ and $m \neq 0$.

Therefore, $m \in O$ and $m <_o y$. Also, $\{x^y\}_o = \{x^m\}_o \times_o x$.

Suppose $x \in O$. Then, by IH(a), $\{x^m\}_o \in O$. Therefore, by Theorem 15(a), $\{x^m\}_o \times_o x \in O$. Hence (a) and (e) hold. Also, $|\{x^y\}_o|_o = |\{x^m\}_o \times_o x|_o$, which by Theorem 15(b), $= |\{x^m\}_o|_o \times |x|_o$. By IH(b), $|\{x^m\}_o|_o = |x|_o^{|m|_o}$. Therefore, $|\{x^y\}_o|_o = |x|_o^{|m|_o} \times |x|_o = |x|_o^{|m|_o+1} = |x|_o^{|2^m|_o} = |x|_o^{|y|_o}$. Hence (b) holds.

To prove (d), suppose $x >_o 2$ and $z <_o y$. Therefore, $z \leq_o m$. Also, $\{x^y\}_o = \{x^m\}_o \times_o x$.

Also, $\{x^m\}_o >_o 1$; otherwise, $\{x^m\}_o = 1$ and $m = 2^n$, for some $n \neq 0$ and $\{x^n\}_o \times_o x = 1$, which in turn would imply that $\{x^n\}_o = 1$, and so on ad infinitum, a contradiction to Fact 8(b). Therefore, $\{x^m\}_o >_o 1$.

By IH(d) and IH(e), $\{x^z\}_o \leq_o \{x^m\}_o$. Also, since $\{x^m\}_o >_o 1$, by Theorem 15(c), $\{x^m\}_o <_o \{x^m\}_o \times_o x$. Hence, $x \times_o z <_o x \times_o y$. Therefore, (d) holds.

Subcase 2: $y = 3 \cdot 5^p \in O$.

In this case, $\{x^y\}_o = 3 \cdot 5^q$, where for all $n$, $\varphi_q(n) = \{x^{\varphi_p(n)}\}_o$. By IH(a), for all $n$, $\{x^{\varphi_p(n)}\}_o \in O$. Also, by IH(d), for all $n$, $\{x^{\varphi_p(n)}\}_o <_o \{x^{\varphi_p(n+1)}\}_o$. Therefore, by IH(a), for all $n$, $\varphi_q(n) \in O$, and by IH(d), $\varphi_q(n) <_o \varphi_q(n+1)$. Therefore, $3 \cdot 5^q \in O$. Hence, (a) and (e) hold. Also, for all $n$, $\{x^{\varphi_p(n)}\}_o <_o \{x^y\}_o$.

Also, $|\{x^y\}_o|_o = |3 \cdot 5^q|_o = \lim_{n\to\infty} |\varphi_q(n)|_o =$
$\lim_{n\to\infty} |\{x^{\varphi_p(n)}\}_o|_o$, which by IH(b), $= \lim_{n\to\infty}(|x|_o^{|\varphi_p(n)|_o}) =$
$|x|_o^{\lim_{n\to\infty} |\varphi_p(n)|_o} = |x|_o^{|y|_o}$. Hence, (b) holds.

Suppose $x >_o 2$ and $z <_o y$. Then, using Fact 7(e), let $n$ be such that $z <_o \varphi_p(n)$. By IH(d), $\{x^z\}_o <_o \{x^{\varphi_p(n)}\}_o$, which is $<_o \{x^y\}_o$. Hence (d) holds.

This completes the proof of parts (a), (b), (d) and (e). We next prove part (c) and then part (f).

Suppose $x >_o 2$ and $2 <_o y$. By (d), $\{x^2\}_o <_o \{x^y\}_o$. But $\{x^2\}_o = \{x^1\}_o \times_o x = 2 \times_o x = x$. Therefore, $x <_o \{x^y\}_o$. Therefore, (c) is true.

Suppose $x' <_o x$ and $y \in O$. Therefore, $x >_o 1$. We consider the following two cases.
Case 1: $x = 2$.

Therefore, $x' = 1$. Hence, $\{x^y\}_o = \{2^y\}_o = 2$. Hence, $\{x^y\}_o \times_o x' = 2 \times_o x' = 2 \times_o 1 = 1$. Also, $\{x^{(y +_o 2)}\}_o = \{2^{(y +_o 2)}\}_o = 2$. Hence, $\{x^y\}_o \times_o x' <_o \{x^{(y +_o 2)}\}_o$.
Case 2: $x >_o 2$.

Then, $\{x^y\}_o >_o 2 >_o 1$. Therefore, using Theorem 15(d), we get $\{x^y\}_o \times_o x' <_o \{x^y\}_o \times_o x$. But $\{x^{(y +_o 2)}\}_o = \{x^{(2^y)}\}_o = \{x^y\}_o \times_o x$. Hence, $\{x^y\}_o \times_o x' <_o \{x^{(y +_o 2)}\}_o$.

Hence (f) it true. $\square$ ('$\Rightarrow$' IMPLICATIONS OF THEOREM 17)

Therefore, all the parts of Theorem 17 are true $\square$ (THEOREM 17)

## References


[Add65] J. W. Addison. The method of alternating chains. In J. W. Addison, Leon Henkin, and Alfred Tarski, editors, *Theory of Models (Proc. 1963 International Symposium, Berkeley, Calif.)*, pages 1–16, Amsterdam, 1965. North–Holland.

[AFS99] A. Ambainis, R. Freivalds, and C. H. Smith. Inductive inference with procrastination: Back to definitions. *Fundamenta Informaticae*, 40:1–16, 1999.

[AJS99] A. Ambainis, S. Jain, and A. Sharma. Ordinal mind change complexity of language identification. *Theoretical Computer Science*, 220(2):323–343, 1999.

[Amb95] A. Ambainis. The power of procrastination in inductive inference: How it depends on used ordinal notations. *Lecture Notes in Computer Science*, 904:99–111, 1995.

[Ang80] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[Aps94] Kalvis Apsītis. Derived sets and inductive inference. In Setsuo Arikawa and Klaus P. Jantke, editors, *Algorithmic Learning Theory, Proc. 4th International Workshop on Analogical and Inductive Inference (AII'94) and the 5th International Workshop on Algorithmic Learning Theory (ALT'94), October 10-15, 1994, Reinhardsbrunn Castle, Germany*, volume 872 of *LNAI*, pages 26–39. Springer-Verlag, 1994.

[ASY92] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, 95:97–113, 1992.

[BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[Beh97] Libor Behounek. Ordinal calculator, 1997. Web document at: http://www.ff.cuni.cz/ behounek/ordinalc.htm.

[Blu67] M. Blum. A machine independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

[BM95] I. Bratko and S. Muggleton. Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70, 1995.

[BUV96] A. Brazma, E. Ukkonen, and J. Vilo. Discovering unbounded unions of regular pattern languages from positive examples. In *Proceedings of the Seventh International Symposium on Algorithms and Computation (ISAAC'96)*, volume 1178 of *Lecture Notes in Computer Science*, pages 95–104, 1996. Osaka, Japan.

[Cas74] J. Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8:15–32, 1974.

[Cas94] J. Case. Infinitary self-reference in learning theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:3–16, 1994.

[Che82] K. Chen. Tradeoffs in inductive inference of nearly minimal sized programs. *Information and Control*, 52:68–86, 1982.

[CJK+01] J. Case, S. Jain, S. Kaufmann, A. Sharma, and F. Stephan. Predictive learning models for concept drift. *Theoretical Computer Science*, 268:323–349, 2001. Special Issue for *ALT'98*.

[CJLZ99] J. Case, S. Jain, S. Lange, and T. Zeugmann. Incremental concept learning for bounded data mining. *Information and Computation*, 152:74–110, 1999.

[CJS95] 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.

[CJS96] J. Case, S. Jain, and A. Sharma. Machine induction without revolutionary changes in hypothesis size. *Information and Computation*, 128:73–86, August 1996.

[CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[CS03] J. Case and M. Suraj. Characterizing Ershov hierarchies by algorithmic $O$-count down, 2003. Working paper.

[EHK81] R. L. Epstein, R. Haas, and R. L. Kramer. Hierarchies of sets and degrees below $\mathbf{0}'$. In *Logic Year 1979–80*, volume 859 of *Lecture Notes in Mathematics*, pages 32–48, Heidelberg, 1981. Springer–Verlag.

[Ers68a] Yu. L. Ershov. A hierarchy of sets, I. *Algebra i Logika*, 7(1):47–74, 1968. In Russian (English translation in *Algebra and Logic*, 7:25–43, 1968).

[Ers68b] Yu. L. Ershov. A hierarchy of sets, II. *Algebra i Logika*, 7(4):15–47, 1968. In Russian (English translation in *Algebra and Logic*, 7:212–232, 1968).

[Ers70] Yu. L. Ershov. A hierarchy of sets, III. *Algebra i Logika*, 9(1):34–51, 1970. In Russian (English translation in *Algebra and Logic*, 9:20–31, 1970).

[Fre75] R. Freivalds. Minimal Gödel numbers and their identification in the limit. *Lecture Notes in Computer Science*, 32:219–225, 1975.

[Fre90] R. Freivalds. Inductive inference of minimal programs. In M. Fulk and J. Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 3–20. Morgan Kaufmann Publishers, Inc., August 1990.

[FS93] R. Freivalds and C. Smith. On the role of procrastination in machine learning. *Information and Computation*, 107(2):237–271, 1993.

[Ful85] M. Fulk. *A Study of Inductive Inference Machines*. PhD thesis, SUNY at Buffalo, 1985.

[FW79] R. Freivalds and R. Wiehagen. Inductive inference with additional information. *Electronische Informationverarbeitung und Kybernetik*, 15:179–195, 1979.

[GD01] A. Gale and R. Downey. On genericity and Ershov's hierarchy. *Mathematical Logic Quarterly*, 47(2):161–182, 2001.

[Gol67] E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[GS95] W. I. Gasarch and C. H. Smith. Recursion theoretic models of learning: some results and intuitions. *Annals of Mathematics and Artificial Intelligence*, 15(2):151–166, 1995.

[Joc82] C. G. Jockusch, Jr. Review of "Hierarchies of Sets and Degrees Below $\mathbf{0}'$" by Richard L. Epstein and Richard Haas and Richard L. Kramer. *Mathematical Reviews*, 1982. MR 82k:03073.

[JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Mass., second edition, 1999.

[JS97] S. Jain and A. Sharma. Elementary formal systems, intrinsic complexity, and procrastination. *Information and Computation*, 132:65–84, 1997.

[JS99] S. Jain and A. Sharma. Mind change complexity of learning logic programs. In P. Fischer and H.U. Simon, editors, *Fourth European Conference on Computational Learning Theory*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 198–213. Springer-Verlag, Berlin, 1999.

[JS01] S. Jain and A. Sharma. On a generalized notion of mistake bounds. *Information and Computation*, 166:156–166, 2001.

[Kin74] E. Kinber. On the synthesis in the limit of almost minimal Gödel numbers. *Theory Of Algorithms and Programs, LSU, Riga*, 1:221–223, 1974.

[Kle38] S. C. Kleene. On notation for ordinal numbers. *Jounal of Symbolic Logic*, 3:150–155, 1938.

[Kle44] S. C. Kleene. On the forms of predicates in the theory of constructive ordinals. *American Journal of Mathematics*, 66:41–58, 1944.

[Kle55] S. C. Kleene. On the forms of predicates in the theory of constructive ordinals (second paper). *American Journal of Mathematics*, 77:405–428, 1955.

[KM67] K. Kuratowski and A. Mostowski. *Set Theory.* North-Holland, 1967.

[KMU95] P. Kilpeläinen, H. Mannila, and E. Ukkonen. MDL learning of unions of simple pattern languages from positive examples. In Paul Vitányi, editor, *Second European Conference on Computational Learning Theory*, volume 904 of *Lecture Notes in Artificial Intelligence*, pages 252–260. Springer-Verlag, 1995.

[LD94] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.

[Mit97] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[MR94] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:669–679, 1994.

[MY78] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms.* North Holland, New York, 1978.

[Nix83] R. Nix. Editing by examples. Technical Report 280, Department of Computer Science, Yale University, New Haven, CT, USA, 1983.

[Odi99] P. Odifreddi. *Classical Recursion Theory*, volume II. Elsivier, Amsterdam, 1999.

[Put63] H. Putnam. Probability and confirmation. *Voice of America, Forum on Philosophy of Science*, 10, 1963.

[Put65] H. Putnam. Trial and error predicates and the solution to a problem of Mostowski. *Journal of Symbolic Logic*, 30:49–57, 1965.

[Rog58] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967. Reprinted, MIT Press, 1987.

[Roy87] J. Royer. *A Connotational Theory of Program Structure*, volume 273 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1987.

[SA95] T. Shinohara and A. Arikawa. Pattern inference. In Klaus P. Jantke and Steffen Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 259–291. Springer-Verlag, 1995.

[Sac90] G. Sacks. *Higher Recursion Theory.* Springer-Verlag, 1990.

[Sal94a] A. Salomaa. Patterns (The Formal Language Theory Column). *EATCS Bulletin*, 54:46–62, 1994.

[Sal94b] A. Salomaa. Return to patterns (The Formal Language Theory Column). *EATCS Bulletin*, 55:144–157, 1994.

[Sch98] M. Schaefer. A guided tour of minimal indices and shortest descriptions. *Archives for Mathematical Logic*, 18:521–548, 1998.

[Sel84] V. L. Selivanov. On a hierarchy of limiting computations. *Sibirskii Mathematicheskii Zhurnal*, 25(5):146–156, 1984. In Russian (English translation in *Siberian Mathematical Journal*, 25:798-806, 1984).

[Sha71] N. Shapiro. Review of "Limiting recursion" by E.M. Gold and "Trial and error predicates and the solution to a problem of Mostowski" by H. Putnam. *Journal of Symbolic Logic*, 36:342, 1971.

[Shi83] T. Shinohara. Inferring unions of two pattern languages. *Bulletin of Informatics and Cybernetics*, 20:83–88., 1983.

[Sie65] W. Sierpinski. *Cardinal and ordinal numbers.* PWN –Polish Scientific Publishers, 1965. Second revised edition.

[Smu61] R. Smullyan. *Theory of Formal Systems.* Annals of Mathematics Studies, No. 47. Princeton University Press, 1961.

[Soa96] Robert I. Soare. Computability and recursion. *Bulletin of Symbolic Logic*, 2(3):284–321, 1996.

[SSS+94] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara, and S. Arikawa. Knowledge acquisition from amino acid sequences by machine learning system BONSAI. *Trans. Information Processing Society of Japan*, 35:2009–2018, 1994.

[SSV97] A. Sharma, F. Stephan, and Y. Ventsov. Generalized notions of mind change complexity. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pages 96–108. ACM Press, 1997.

[Wri89] K. Wright. Identification of unions of languages drawn from an identifiable class. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory, Santa Cruz, California*, pages 328–333. Morgan Kaufmann Publishers, Inc., 1989.