An overview of the ordinal calculator

Paul Budnik
Mountain Math Software
paul@mtnmath.com

Abstract

An ordinal calculator has been developed as an aid for understanding the countable ordinal hierarchy and as a research tool that may eventually help to expand it. A GPL licensed version is available in C++. It is an interactive command line calculator and can be used as a library. It includes notations for the ordinals uniquely expressible in Cantor normal form, the Veblen hierarchies and a form of ordinal projection or collapsing using notations for countable admissible ordinals and their limits. The calculator does addition, multiplication and exponentiation on ordinal notations. For a recursive limit ordinal notation, α, it can list an initial segment of an infinite sequence of notations such that the union of the ordinals represented by the sequence is the ordinal represented by α. It can give the relative size of any two notations and it determines a unique notation for every ordinal represented. Input is in plain text. Output can be plain text and/or LaTeX math mode format. This approach is motivated by a philosophy of mathematical truth that sees *objectively true* mathematics as connected to properties of recursive processes. It suggests that computers are an essential adjunct to human intuition for extending the combinatorially complex parts of objective mathematics.

Contents

List of Figures

List of Tables

Introduction

An ordinal calculator has been developed as an aid for understanding the countable ordinal hierarchy and as a research tool that may eventually help to expand it. A GPL licensed version is available in C++[1]. It is an interactive command line calculator and can be used as a library. It includes notations for the ordinals uniquely expressible in Cantor normal form (that are $< \varepsilon_0$), the Veblen hierarchies and a form of ordinal projection or collapsing using notations for countable admissible ordinals and their limits (see section 5).

[1]The source code and executable can be downloaded from `https://sourceforge.net/projects/ord`. The downloads include a users manual and a document describing the underlying mathematics and documenting the code.

The calculator does addition, multiplication and exponentiation on ordinal notations. For a recursive limit ordinal notation, α, it can list an initial segment of an infinite sequence of notations such that the union of the ordinals represented by the sequence is the ordinal represented by α. It can give the relative size of any two notations and it determines a unique notation for every ordinal represented. Input is in plain text. Output can be plain text and/or LaTeX math mode format.

Loosely speaking there are two dimensions to the power of axiomatic mathematical systems: definability and provability. The former measures what structures can be defined and the latter what statements about these structures are provable. Provability is usually expanded by extending definability, but there are other ways to expand provability. In arguing for the necessity of large cardinal axioms a number of arithmetic statements have been shown to require such axioms to decide[8]. This claim is relative to the linear ranking of generally accepted axiom systems. However any arithmetic (or even hyperarithmetic) statement can be decided by adding to second order arithmetic a finite set of axioms that say certain integers do or do not define notations for recursive ordinals in the sense of Kleene's $\mathcal{O}$[11][2] This follows because Kleene's $\mathcal{O}$ is a Π^1_1[3] complete set[4][18] and a TM (Turing machine) with an oracle that makes a decision must do so after a finite number of queries.

Large cardinal axioms are needed to decide some questions because it has not been possible to construct a sufficiently powerful axiom system about notations for recursive ordinals. This can change. Any claim that large cardinal axioms are needed to decide arithmetic statements is relative to the current state of mathematics.

Large cardinal axioms seem to implicitly define large recursive ordinals that may be beyond the ability of the unaided human mind to define explicitly. Thus the central motivation of this work is to use the computer as a research tool to augment human intuition with the enormous combinatorial power of today's computers.

There is the outline of a theory of objective mathematical truth that underlies this approach in Section 7. This theory sees objective mathematics as logically determined by a recursive enumerable sequence of events. (The relationship between these events may be complex[5], but these events, by themselves, must decide the statement.) Objective mathematics includes arithmetic and hyperarithmetic statements and some statements requiring quantification over the reals.

This paper's intended readership is anyone with an interest in the recursive ordinals or the foundations of mathematics that has a basic understanding of programming (at least knows what a TM is and how it can be programmed) and a basic understanding of set theory and ordinal numbers such as might be obtained in an introductory course in set theory or its equivalent.

[2]Kleene's $\mathcal{O}$ is a set of notations for all recursive ordinals. It obtains this completeness by a definition that requires quantifying over the reals and thus these notation are not recursively enumerable. Notations for any initial segment of these ordinals $\leq \alpha$, a recursive ordinal, are recursively enumerable from any member of $\mathcal{O}$ which is a notation for α.

[3]A Π_n statement starts with a universal quantifier ($\forall$) and contains $n-1$ alternations between universal and existential ($\exists$) quantifiers. A Π^1_n statement has a similar definition for quantifiers over the reals.

[4]A Π^1_1 complete set, if it is encoded as a TM oracle, allows the TM to decide any Π^1_1 statement.

[5]The valid relationships cannot be precisely defined without limiting the definition beyond what is intended.

1 Program structure and interactive mode

This section gives a brief overview of aspects of object oriented programming in C++ and how they are used to structure the program. It then gives a brief description of the interactive mode of the calculator. It is intended to keep this article self contained for those with a limited knowledge of programming and to introduce the interactive calculator.

1.1 Program structure

C++ connects data and the procedures that operate on that data by defining a `class`[6]. Both data structures and member functions that operate on those structures are declared within a `class`. Instances of a `class` are created with a special member function called a constructor. This helps to insure that all instances of a `class` are properly initialized.

C++ `class`es can form a hierarchy with one `class` derived from another. The first or base `class` has only the operations defined in it. A derived `class` has both the base `class` operations and its own operations. In the ordinal calculator there is a base `class Ordinal` for notations less than ε_0. Larger ordinals are derived in a hierarchy that has `Ordinal` as its base `class`. All members of this hierarchy can be referenced as `Ordinals`. Some procedures, like `compare` that determines the relative size of two notations, must be rewritten for each new derived `class`. By using `virtual` functions with the same name, `compare`, the correct version will always be called even though the `class` instance is only referenced as an `Ordinal` in the code. Programs which call `Ordinal` member functions like `compare` are written with an instance of the `Ordinal class` followed by dot and the member function and its parameters. For example `ord1.compare(ord2)` compares `Ordinals` `ord1` and `ord2`. This example returns -1,0 or 1 if `ord1` is $<, =$ or $>$ `ord2`.

1.2 Interactive mode

The ordinal calculator has a command line interactive mode that supports most functions without requiring C++ coding. In this mode one can assign ordinal expressions to a variable. These expressions can include ordinal notation variables. Aside from reserved words[7], all names starting with a letter are variables that can be assigned notations. Typing a name assigned to a notation will display the notation in plain text format (the default) and/or LaTeX format[8].

The calculator includes symbols for addition '`+`', multiplication '`*`', exponentiation '`^`' and parentheses to group subexpressions. To compare the relative size of two ordinal expressions use the operators, `<`, `<=`, `>`, `>=` and `==`. To list the first n notations for ordinals in an infinite

[6] C++ constructs and plain text calculator input and output are in `typewriter font`.

[7] The reserved words in the calculator are `help` and the commands listed by typing "`help cmds`", `w` and `omega` representing ω, `epsilon` representing ε, `gamma` representing Γ, `psi` representing φ, `w1CK` representing $\omega_1^{\rm CK}$ (the ordinal of the recursive ordinals) and `eps0` representing ε_0.

[8] The command `opts` followed by `text`, `tex` or `both` controls the output format. In addition there is a member function, `.cpp`, that outputs an ordinal notation as C++ code and command `cppList` that outputs all user defined notations in C++ code. These are useful in writing C++ code using the calculator C++ `class`es directly.

sequence that have the ordinal represented by α as there union write α.`listElts(n)`[9]. The `help` command provides online documentation.

2 Recursive ordinal notations and the Cantor normal form

The ordinal calculator assigns strings as notations to an initial fragment of the recursive ordinals. It contains a recursive process for deciding the relative size ($<, =,$ or $>$) of the ordinals represented by each string and a recursive process for deciding if a given string represents an ordinal. Section 5 describes an expansion of this recursive ordinal notation system that represents ω_1^{CK} (the Church-Kleene ordinal, the ordinal of the recursive ordinals) and larger countable ordinals. There are gaps in the ordinals with notations in this expanded structure although the set of *all notations in the system* is recursively enumerable and recursively ranked.

Greek letters represent both notations, the finite strings that represent ordinals, and the ordinals themselves. The relative size of notations is the relative size of the ordinals they represent. Notations are successors or limits if the ordinal they represent are.

There is a `virtual Ordinal` member function `limitElement(n)` on the integers that outputs an increasing sequence of ordinal notations with increasing `n`. If notation α represents recursive ordinal, the union of the ordinals represented by the outputs of α.`limitElement(n)` is the ordinal represented by α. Every ordinal, α, can be represented as shown in expression 1

$$\alpha_1 > \alpha_2 > \alpha_3 > ... > \alpha_k$$

The α_k are ordinal notations and the n_k are integers > 0.

$$\omega^{\alpha_1} n_1 + \omega^{\alpha_2} n_2 + \omega^{\alpha_3} n_3 + ... + \omega^{\alpha_k} n_k \tag{1}$$

The calculator input format represents ω as `w`. The above expression is written as: `a=w^a1*n1+w^a2*n2+w^a3*n3+...+w^ak*nk`. The `n1...nk` are integers and the `a1...ak` are variables for previously defined ordinal notations or notations in parenthesis. The equal sign assigns the specified notation to variable `a`.

$\varepsilon_0 = \bigcup \omega, \omega^\omega, \omega^{\omega^\omega} \omega^{\omega^{\omega^\omega}}$ ... and cannot be reached with ω^α, the integers and ω. The Cantor normal form gives unique representation only for ordinals $< \varepsilon_0$. Each term in expression 1 is represented by a member of `class CantorNormalElement`. The terms are linked in decreasing order in `class Ordinal`. This base `class` can represent any ordinal $< \varepsilon_0$. The integers used to define finite ordinals[10] are scanned and processed with a library that supports arbitrarily large integers[11]. The `Ordinal` instance representing ω is predefined. Larger ordinals

[9]Sometimes mathematical notation is combined with C++ code. In this example the C++ definition of a member function is combined with a Greek letter to represent the C++ object (an `Ordinal` notation) that the subroutine is called from.

[10]The syntax for defining the ordinal 12 named '`a`' is '`Ordinal a(12);`' in C++ and '`a=12`' in the interactive ordinal calculator.

[11]The package used is MPIR, (Multiple Precision Integers and Rationals) based on the package GMP (GNU Multiple Precision Arithmetic Library). Either package can be used, but only MPIR is supported on Microsoft operating systems.

in the base class are constructed using the integers, ω and three ordinal operators, $+$, $\times$ and exponentiation.

3 The Veblen hierarchy

The Veblen hierarchy[22, 14, 9, 16] extends the Cantor normal form by defining functions that grow much faster than ordinal exponentiation. These define ordinal notations much larger than ε_0. The Veblen hierarchy is developed in two stages. The first involves expressions of a fixed finite number of parameters. The second involves functions definable as limits of sequences of functions of an increasing number of parameters.

3.1 Two parameter Veblen function

The Veblen hierarchy starts with a function of two parameters, $\varphi(\alpha_1, \alpha_2)$ based on $\varphi(\alpha) = \omega^\alpha$. $\varphi(1, \alpha_2)$ is defined as the α_2 fixed point of ω^α and is written as ε_{α_2}.

$$\varphi(2,0) = \bigcup \varepsilon_1, \varepsilon_{\varepsilon_1+1}, \varepsilon_{\varepsilon_{\varepsilon_1+1}+1}, ...$$

and

$$\varphi(2, \alpha_2 + 1) = \bigcup \varphi(2, \alpha_2), \varphi(1, \varphi(2, \alpha_2) + 1), \varphi(1, \varphi(1, \varphi(2, \alpha_2) + 1) + 1), ...$$

Each element of the sequence, past the first, takes the previous element as the second parameter. If `limitElement` computes a limit ordinal as a parameter, it often adds 1 to that parameter to avoid fixed points. This is reflected in the examples.

The ordinal calculator plain text format for $\varphi(\alpha_1, \alpha_2)$ is `psi(a1,a2)`. The calculator uses three common substitutions in expressions in the Veblen hierarchy. These are: $\omega^\alpha = \varphi(\alpha)$, $\varepsilon(\alpha) = \varphi(1, \alpha)$ and $\Gamma(\alpha) = \varphi(1, 0, \alpha)$. These substitutions are used in tables 2, 3, 4 and 6. The plain text versions are `w^a` for ω^α, `epsilon(a)` for $\varepsilon(\alpha)$ and `gamma(a)` for $\Gamma(\alpha)$.

Table 1 defines the two parameter Veblen function. In some cases the table gives an inductive definition on the integers. It defines $\varphi(...)^0$ and then $\varphi(...)^{n+1}$ using $\varphi(...)^n$. Finally $\varphi(...)$ is defined as an infinite union over the integers.

If α_2 a limit in $\varphi(\alpha_1, \alpha_2)$ then the limit of the expression is expanded by expanding the limit α_2. If the least significant nonzero parameter is a limit, then it must be expanded first by `limitElement` and related functions. Consider $\varphi(\omega, \omega)$. $\forall_{n\in\omega}\varphi(n + 1, 0) > \varphi(n, \omega)$ and thus $\bigcup_{n\in\omega} \varphi(n, \omega) = \bigcup_{n\in\omega} \varphi(n + 1, 0) = \varphi(\omega, 0)$. With this in mind, if the least significant parameter is a successor and the the next least significant parameter is a limit, one must exercise care to make sure both parameters affect the result. Examples of the two parameter Veblen function are in in Table 2[12].

[12]In the ordinal calculator an exit code is assigned to all code fragments that compute the sequences that define the value of a limit ordinal. These are included in some tables as the right most column labeled X. These are documented in [3] and used to verify that the regression tests include all cases. They are available in the interactive calculator using member function `lec`. They are included here to connect examples of sequences that define limit notations with the rules they come from and to insure the tables are complete.

Definition of $\varphi(\alpha_1, \alpha_2)$ L is lines in Table 2. X is an exit code (see Note 12).			L	X
$\varphi(\alpha_2)$	$= \omega^{\alpha_2}$.		1	C
$\varphi(1, \alpha_2)$	$= \varepsilon_{\alpha_2}$.	α_2 a successor	3	FD
		α_2 a limit	4	FL
If α_1 and α_2 are successors, define				
$\varphi(\alpha_1, \alpha_2)^0$	$= \varphi(\alpha_1, \alpha_2 - 1)$ and define			
$\varphi(\alpha_1, \alpha_2)^{n+1}$	$= \varphi(\alpha_1 - 1, \varphi(\alpha_1, \alpha_2)^n) + 1$ then by induction on n			
$\varphi(\alpha_1, \alpha_2)$	$= \bigcup_{n \in \omega} \varphi(\alpha_1, \alpha_2)^n$ which expands to			
$\varphi(\alpha_1, \alpha_2)$	$= \bigcup \varphi(\alpha_1, \alpha_2 - 1),\ \varphi(\alpha_1 - 1, \varphi(\alpha_1, \alpha_2 - 1) + 1) + 1,$ $\varphi(\alpha_1 - 1, \varphi(\alpha_1 - 1, \varphi(\alpha_1, \alpha_2 - 1) + 1) + 1) + 1, ...,.$		10	FD
If α_2 is a limit, then				
$\varphi(\alpha_1, \alpha_2)$	$= \bigcup_{\beta \in \alpha_2} \varphi(\alpha_1, \beta)$.		8	FL
If α_1 is a limit and α_2 is a successor then				
$\varphi(\alpha_1, \alpha_2)$	$= \bigcup_{\beta \in \alpha_1} \varphi(\beta, \varphi(\alpha_1, \alpha_2 - 1) + 1)$.		12	FN

Table 1: Two parameter Veblen function definition

	α	α.`limitElement(n)`			
		`n=1`	`n=2`	`n=3`	X
1	ω^{ω^2}	ω^ω	$\omega^{\omega 2}$	$\omega^{\omega 3}$	C
2	ε_1	ε_0	ω^{ε_0+1}	$\omega^{\omega^{\varepsilon_0+1}}$	FD
3	ε_2	ε_1	ω^{ε_1+1}	$\omega^{\omega^{\varepsilon_1+1}}$	FD
4	ε_ω	ε_1	ε_2	ε_3	FL
5	$\varphi(2, 1)$	$\varphi(2, 0)$	$\varepsilon_{\varphi(2,0)+1}$	$\varepsilon_{\varepsilon_{\varphi(2,0)+1}+1}$	FD
6	$\varphi(2, 2)$	$\varphi(2, 1)$	$\varepsilon_{\varphi(2,1)+1}$	$\varepsilon_{\varepsilon_{\varphi(2,1)+1}+1}$	FD
7	$\varphi(2, 5)$	$\varphi(2, 4)$	$\varepsilon_{\varphi(2,4)+1}$	$\varepsilon_{\varepsilon_{\varphi(2,4)+1}+1}$	FD
8	$\varphi(2, \omega)$	$\varphi(2, 1)$	$\varphi(2, 2)$	$\varphi(2, 3)$	FL
9	$\varphi(3, 1)$	$\varphi(3, 0)$	$\varphi(2, \varphi(3, 0) + 1)$	$\varphi(2, \varphi(2, \varphi(3, 0) + 1) + 1)$	FD
10	$\varphi(3, 2)$	$\varphi(3, 1)$	$\varphi(2, \varphi(3, 1) + 1)$	$\varphi(2, \varphi(2, \varphi(3, 1) + 1) + 1)$	FD
11	$\varphi(\omega, 1)$	$\varepsilon_{\varphi(\omega,0)+1}$	$\varphi(2, \varphi(\omega, 0) + 1)$	$\varphi(3, \varphi(\omega, 0) + 1)$	FN
12	$\varphi(\omega, 9)$	$\varepsilon_{\varphi(\omega,8)+1}$	$\varphi(2, \varphi(\omega, 8) + 1)$	$\varphi(3, \varphi(\omega, 8) + 1)$	FN
13	$\varphi(\omega, \omega)$	$\varphi(\omega, 1)$	$\varphi(\omega, 2)$	$\varphi(\omega, 3)$	FL
14	$\varphi(\omega, \omega + 2)$	$\varepsilon_{\varphi(\omega,\omega+1)+1}$	$\varphi(2, \varphi(\omega, \omega + 1) + 1)$	$\varphi(3, \varphi(\omega, \omega + 1) + 1)$	FN

Table 2: Two parameter Veblen function exanples

3.2 Finite parameter Veblen functions

The finite parameter Veblen function generalizes the two parameter Veblen function to n parameters for *all integers n*. This section defines how it is computed. The definition of a notation for a recursive limit ordinal must define a recursively enumerable sequence of notations such that the union of ordinals they represent is the ordinal represented by the original notation. This sequence uses the original notation modified by replacing parameters. The replacement can use the previous notation in the sequence. One or two parameters are replaced and they are usually the least significant parameters and/or the least significant nonzero parameters. For the Veblen hierarchy the most significant parameters always occur first in reading from left to right.

Following are the rules that apply to the finite parameter Veblen hierarchy.

1. The hierarchy starts with $\varphi(\alpha) = \omega^{\alpha}$ and $\varphi(1, \alpha) = \varepsilon_{\alpha}$.

2. In some of the substitutions, when the substituted parameter is a limit, 1 is added to avoid fixed points.

3. If notation α has least significant nonzero parameter β, a limit, then α represents the union of a sequence of ordinals with the same notation as α except β is replaced by a sequence of notations that have β as their limit. See lines 7 and 11 in Table 4 (exit code FL).

4. If notation α has least significant parameter γ which is a successor and the next most significant nonzero parameter, β, is a limit, then the sequence replaces β with a sequence that has β as its limit and the parameter to its immediate right with α with one subtracted from the γ parameter. See lines 5 and 6 in Table 4 (exit code FN).

5. If notation α has least significant nonzero notation β, a successor, and one or more less significant parameters of 0, then α represents the ordinal that is the limit of the sequence starting with the notation for α with 1 subtracted from β and the next (to the right) parameter changed from 0 to 1 to support the special case of a subtraction result of 0. Subsequent sequence values have a similar replacement with the next (to the right) parameter replaced with the previous element in the sequence. See lines 8 and 12 in Table 4 (exit code FB). Line 12 is the special case when the only nonzero parameter is 1 and all members of the sequence that define α have one less parameter than α.

6. If notation α's least significant parameter, β, is a successor, the next to the least most significant parameter is zero and the next to the least most significant *nonzero* parameter, γ, is a successor, then the first element in the sequence that defines α is similar to α with 1 subtracted from β. Subsequent members of the sequence have one subtracted from γ and the parameter to the *immediate right* is replaced with the previous element in the sequence. See lines 9 and 10 in Table 4 (exit code FD).

7. Items 4 and 6 above include cases where the least significant nonzero parameter has no effect on the limit that defines the sequence. Those parameters are left unchanged

α	α.`limitElement(n)`		
	`n=1`	`n=2`	`n=3`
$\varphi(1,1,0)$	Γ_1	Γ_{Γ_1+1}	$\Gamma_{\Gamma_{\Gamma_1+1}+1}$
$\varphi(1,1,1)$	$\varphi(1,1,0)$	$\Gamma_{\varphi(1,1,0)+1}$	$\Gamma_{\Gamma_{\varphi(1,1,0)+1}+1}$
$\varphi(1,1,2)$	$\varphi(1,1,1)$	$\Gamma_{\varphi(1,1,1)+1}$	$\Gamma_{\Gamma_{\varphi(1,1,1)+1}+1}$
$\varphi(1,1,\omega)$	$\varphi(1,1,1)$	$\varphi(1,1,2)$	$\varphi(1,1,3)$
$\varphi(1,2,1)$	$\varphi(1,2,0)$	$\varphi(1,1,\varphi(1,2,0)+1)$	$\varphi(1,1,\varphi(1,1,\varphi(1,2,0)+1)+1)$
$\varphi(1,2,2)$	$\varphi(1,2,1)$	$\varphi(1,1,\varphi(1,2,1)+1)$	$\varphi(1,1,\varphi(1,1,\varphi(1,2,1)+1)+1)$
$\varphi(1,2,5)$	$\varphi(1,2,4)$	$\varphi(1,1,\varphi(1,2,4)+1)$	$\varphi(1,1,\varphi(1,1,\varphi(1,2,4)+1)+1)$
$\varphi(1,\omega,0)$	$\varphi(1,1,0)$	$\varphi(1,2,0)$	$\varphi(1,3,0)$
$\varphi(1,\omega,1)$	$\varphi(1,1,\varphi(1,\omega,0)+1)$	$\varphi(1,2,\varphi(1,\omega,0)+1)$	$\varphi(1,3,\varphi(1,\omega,0)+1)$
$\varphi(2,2,\omega)$	$\varphi(2,2,1)$	$\varphi(2,2,2)$	$\varphi(2,2,3)$
$\varphi(3,1,1)$	$\varphi(3,1,0)$	$\varphi(3,0,\varphi(3,1,0)+1)$	$\varphi(3,0,\varphi(3,0,\varphi(3,1,0)+1)+1)$
$\varphi(4,3,2)$	$\varphi(4,3,1)$	$\varphi(4,2,\varphi(4,3,1)+1)$	$\varphi(4,2,\varphi(4,2,\varphi(4,3,1)+1)+1)$
$\varphi(\omega,3,2)$	$\varphi(\omega,3,1)$	$\varphi(\omega,2,\varphi(\omega,3,1)+1)$	$\varphi(\omega,2,\varphi(\omega,2,\varphi(\omega,3,1)+1)+1)$

Table 3: Three parameter Veblen function examples

	α	α.`limitElement(n)`		
		`n=1`	`n=2`	`n=3`
1	$\varphi(1,0,0,0)$	Γ_0	$\varphi(\Gamma_0+1,0,0)$	$\varphi(\varphi(\Gamma_0+1,0,0)+1,0,0)$
2	$\varphi(1,0,0,1)$	$\varphi(1,0,0,0)$	$\varphi(\varphi(1,0,0,0)+1,0,1)$	$\varphi(\varphi(\varphi(1,0,0,0)+1,0,1)+1,0,1)$
3	$\varphi(\omega,0,0,0)$	$\varphi(1,0,0,0)$	$\varphi(2,0,0,0)$	$\varphi(3,0,0,0)$
4	$\varphi(\omega,0,0,1)$	$\varphi(1,\varphi(\omega,0,0,0)+1,0,1)$	$\varphi(2,\varphi(\omega,0,0,0)+1,0,1)$	$\varphi(3,\varphi(\omega,0,0,0)+1,0,1)$
5	$\varphi(\omega,0,0,4)$	$\varphi(1,\varphi(\omega,0,0,3)+1,0,4)$	$\varphi(2,\varphi(\omega,0,0,3)+1,0,4)$	$\varphi(3,\varphi(\omega,0,0,3)+1,0,4)$
6	$\varphi(\omega,0,0,5)$	$\varphi(1,\varphi(\omega,0,0,4)+1,0,5)$	$\varphi(2,\varphi(\omega,0,0,4)+1,0,5)$	$\varphi(3,\varphi(\omega,0,0,4)+1,0,5)$
7	$\varphi(\omega,0,0,\omega)$	$\varphi(\omega,0,0,1)$	$\varphi(\omega,0,0,2)$	$\varphi(\omega,0,0,3)$
8	$\varphi(\omega,5,0,0)$	$\varphi(\omega,4,1,0)$	$\varphi(\omega,4,\varphi(\omega,4,1,0)+1,0)$	$\varphi(\omega,4,\varphi(\omega,4,\varphi(\omega,4,1,0)+1,0)+1,0)$
9	$\varphi(\omega,5,0,1)$	$\varphi(\omega,5,0,0)$	$\varphi(\omega,4,\varphi(\omega,5,0,0)+1,1)$	$\varphi(\omega,4,\varphi(\omega,4,\varphi(\omega,5,0,0)+1,1)+1,1)$
10	$\varphi(\omega,5,0,9)$	$\varphi(\omega,5,0,8)$	$\varphi(\omega,4,\varphi(\omega,5,0,8)+1,9)$	$\varphi(\omega,4,\varphi(\omega,4,\varphi(\omega,5,0,8)+1,9)+1,9)$
11	$\varphi(\omega,\varepsilon_0,0,0)$	$\varphi(\omega,\omega,0,0)$	$\varphi(\omega,\omega^\omega,0,0)$	$\varphi(\omega,\omega^{\omega^\omega},0,0)$
12	$\varphi(1,0,0,0,0)$	$\varphi(1,0,0,0)$	$\varphi(\varphi(1,0,0,0)+1,0,0,0)$	$\varphi(\varphi(\varphi(1,0,0,0)+1,0,0,0)+1,0,0,0)$

Table 4: More than three parameter Veblen function examples

in the ordinal calculator for the cases that do not matter. Examples include lines 5, 6 and 10 in Table 4.

Three parameter Veblen function examples are shown in Table 3 and larger examples in Table 4.

3.3 Transfinite Veblen functions

The limit of the finite parameter Veblen functions is the union of the sequence $\varphi(1)$, $\varphi(1,0)$, $\varphi(1,0,0)$, ...,. To represent this and larger ordinals, the notation for finite parameter Veblen functions is expanded with the ordinal notation subscript γ in the following expression.

$$\varphi_\gamma(\alpha_1, \alpha_2, ..., \alpha_n) \tag{2}$$

The ordinal calculator plain text format for this is `psi_{g}(a1,a2,...,an)`. The above sequence is defined to be φ_1.

Definition of $\varphi_\gamma(\alpha)$ L is lines in Table 6. X is an exit code (see Note 12).	L	X
$\varphi_1 \quad = \bigcup \varphi(1),\ \varphi(1,0),\ \varphi(1,0,0), ..., .$	1	IG
$\varphi_{\gamma+1} \quad = \bigcup \varphi_\gamma(\varphi_\gamma+1),\ \varphi_\gamma(\varphi_\gamma+1,0),\ \varphi\gamma(\varphi_\gamma+1,0,0), ...,$	9	IG
If γ is a limit and $\alpha = 0$ then $\varphi_\gamma \quad = \bigcup_{\beta\in\gamma} \varphi_\beta.$	7	IJ
If γ and α are successors then $\varphi_\gamma(\alpha) \quad = \bigcup \varphi_\gamma(\alpha-1)+1, \varphi_{\gamma-1}(\varphi_\gamma(\alpha-1)+1,0), \varphi_{\gamma-1}(\varphi_\gamma(\alpha-1)+1,0,0), ...,.$	5	II
If γ is a limit and α is a successor then $\varphi_\gamma(\alpha) \quad = \bigcup_{\beta\in\gamma} \varphi_\beta(\varphi_\gamma(\alpha-1)+1).$	10	IK

Table 5: Definition of $\varphi_\gamma(\alpha)$

The transfinite Veblen function is built on the finite parameter Veblen function. If ζ is a transfinite Veblen notation, then the rules for defining ζ`.limitElement(n)` include the numbered rules in Section 3.2. When those rules are applicable, the γ parameter is unchanged and copied from ζ to ζ`.limitElement(n)`. γ is changed only if the single nonzero α parameter is a successor and the least significant α parameter. The rules for this are given in in Table 5. The right column, X, gives the exit code described in Note 12. Table 6 gives examples of the transfinite Veblen function, the associated first 3 values of `limitElement(n)` and the corresponding exit code. These codes each refer to either Table 5 or the enumerated list in Section 3.2.

The C++ coding of `limitElement` when the same rules are used for two `class`es of notations (like the finite and transfinite Veblen functions) involves two steps. First the higher level `limitElement` explicitly calls the lower `class` version to handle some cases. In C++ this is done by explicitly referencing the lower `class` as in *lower_class_name*`::limitElement(n)`. The second step, in the lower `class` function, creates an output `Ordinal` of the required `class`. This is done by calling a virtual function of the `Ordinal` instance that `limitElement` was originally called from. The creating virtual function calls the constructor for the `Ordinal` derived sub`class` being generated Before calling this constructor. it fills in parameters defined only at the higher `class` level and that are not modified in computing `limitElement(n)`. As the ordinal calculator was expanded to define more ordinal notation `class`es, this approach was increasingly helpful.

Finally it is worth noting that the constructor is not called directly. The constructor is called from a function that evaluates possible fixed points generated by the parameters of the notation to creates a unique notation for each ordinal represented in the system. Direct calls to the constructor can create non unique notations.

4 Limitations of ordinal notations

There are at least two ways one can develop and extend the recursive ordinal hierarchy. One is with a recursively enumerable set of ordinal notations and a recursive function that determines the ranking of any two notations in the system. This ranking can be used to recursively compute a unique notation for every ordinal represented in the system. The

	α	α`.limitElement(n)`			
		`n=1`	`n=2`	`n=3`	X
1	φ_1	ω	ε_0	Γ_0	IG
2	$\varphi_1(1,0,0)$	$\varphi_1(1,0)$	$\varphi_1(\varphi_1(1,0)+1,0)$	$\varphi_1(\varphi_1(\varphi_1(1,0)+1,0)+1,0)$	FB
3	$\varphi_1(1,0,1)$	$\varphi_1(1,0,0)$	$\varphi_1(\varphi_1(1,0,0)+1,1)$	$\varphi_1(\varphi_1(\varphi_1(1,0,0)+1,1)+1,1)$	FD
4	$\varphi_3(1)$	φ_3+1	$\varphi_2(\varphi_3+1,0)$	$\varphi_2(\varphi_3+1,0,0)$	II
5	$\varphi_3(2)$	$\varphi_3(1)+1$	$\varphi_2(\varphi_3(1)+1,0)$	$\varphi_2(\varphi_3(1)+1,0,0)$	II
6	$\varphi_5(\omega)$	$\varphi_5(1)$	$\varphi_5(2)$	$\varphi_5(3)$	FL
7	φ_ω	φ_1	φ_2	φ_3	IJ
8	$\varphi_\omega(1)$	$\varphi_1(\varphi_\omega+1)$	$\varphi_2(\varphi_\omega+1)$	$\varphi_3(\varphi_\omega+1)$	IK
9	$\varphi_{\omega+5}$	$\varphi_{\omega+4}(\varphi_{\omega+4}+1)$	$\varphi_{\omega+4}(\varphi_{\omega+4}+1,0)$	$\varphi_{\omega+4}(\varphi_{\omega+4}+1,0,0)$	IG
10	$\varphi_{\omega^\omega}(8)$	$\varphi_\omega(\varphi_{\omega^\omega}(7)+1)$	$\varphi_{\omega^2}(\varphi_{\omega^\omega}(7)+1)$	$\varphi_{\omega^3}(\varphi_{\omega^\omega}(7)+1)$	IK

Table 6: Transfinite Veblen function examples

Veblen hierarchy[22] and its extensions, that includes ordinal collapsing (see Section 5.2), are examples. This type of notation is used in the ordinal calculator.

Kleene's $\mathcal{O}$, defines a notation for every recursive ordinal. However the set of all these notations is not recursively enumerable. Every infinite recursive ordinal has multiple notations in $\mathcal{O}$ and no recursive algorithm can determine the relative size of all notations in $\mathcal{O}$. Both notation systems, the Veblen hierarchy and Kleene's $\mathcal{O}$, have a limit on effective notation systems for recursive ordinals. In the first case the limit is of the notations defined in the system. In the case of Kleene's $\mathcal{O}$ it is the limit of unique notations in recursive and hyperarithmetical progressions of notations in $\mathcal{O}$[15].

One cannot construct a recursive system of recursively ranked notations for all recursive ordinals. The ordinal calculator constructs notations for an initial segment of the recursive ordinals and for the ordinal of the recursive ordinal, ω_1^{CK}, and many larger ordinals. Such a system must be incomplete with many gaps starting with the gap between the limit of the recursive ordinals defined in the system and ω_1^{CK}.

The ordinal calculator notations $\geq \omega_1^{\mathrm{CK}}$ are based in part on generalizing the idea in $\mathcal{O}$ of indexing the notation for a limit ordinals with the integers or finite ordinals. For any α, a limit ordinal notation in $\mathcal{O}$, one can construct a recursive function on the integers that enumerates an infinite sequence of notations, such that the union of the ordinals represented by notations in the sequence is the ordinal α represents. The ordinal calculator generalizes this idea by defining levels of ordinals indexed by notations at a lower level. The first level is the integers or finite ordinals. The next level is the recursive ordinals. The first level beyond the recursive ordinals has both limits indexed by all recursive ordinal notations and limits indexed by the integers. The notations for limit ordinals must encode the type or parameter they are indexed with. The idea of recursive functions defined on a hierarchy of types is a bit reminiscent of the typed hierarchy of *Principia Mathematica*[23].

Although a domain that includes notations for all recursive ordinals cannot be recursively enumerable, recursive functions operating on that domain can use its properties to insure the output for any valid input will have the required properties. This imposes constraints on `virtual` functions, such as `limitElement` and `compare` to insure new `class`es can be added to expand the notation. For example the `compare` function must check to see if its argument comes from a derived `class`[13] which may not have been defined when this version of `compare` was written. In this case, if `term1` and `term2` are `CantorNormalElement`s in an `Ordinal` expression, then `term1.compare(term2)` returns `-term2.compare(term1)` calling `compare` in the higher level derived `class`.

5 Notations for the Church-Kleene ordinal and beyond

The countable admissible ordinals are the staring point for extending the ordinal calculator to and beyond ω_1^{CK}. The first two admissible ordinals are ω and ω_1^{CK}. Subsequent countable admissible ordinals are defined as Kleene's $\mathcal{O}$ is, but using a TM with an oracle for previously defined notations[20, 21]. There are limits of sequences of admissible ordinals that are not themselves admissible. For example this is true of the first ω admissible ordinals. These limit ordinals are part of the hierarchy of admissible ordinals in the calculator.

5.1 Notations for countable admissible ordinals

The notation for countable admissible ordinals in the ordinal calculator is ω_κ. κ is the index in the admissible hierarchy starting with $\omega_0 = \omega$ and $\omega_1 = \omega_1^{\text{CK}}$. The notation is extended to $\omega_{\kappa,\gamma}(\alpha_1, \alpha_2, ..., \alpha_n)$ to define some of the values between admissible ordinals. The ordinal calculator plain text format is `w_{k,g}(a1,a2,...,an)`[14]. The γ and α_i parameters have a definition based in part on the Veblen hierarchy. Thus, for example, $\omega_1(\omega)$ and $\omega_1(1, 2)$ use rules 3 and 6 with exit codes FL and FD in the list in Section 3.2. Having the same exit code means they are computed by the same code fragment as described in Section 3.3. New rules and code are required only for some instances of $\omega_{\kappa,\gamma}(\alpha)$. If there is more than one α parameter, α_n is a limit or $\gamma > 0$ then the existing rules apply. The new rules are in Table 7.

If there are no parameters except κ (γ and all α are 0) then new rules and an expanded approach to defining notations for limit ordinals is required. The smallest example is ω_1. It is the first limit ordinal notation that cannot be fully indexed by the integers. It must be indexed by notations for recursive ordinals. The `Ordinal` virtual member function `limitOrd(`α`)` is defined to support this. This is used to expand limits somewhat as `limitElement(n)` is[15].

Notations have an associated limit type. For example the limit type of ω_1 is the recursive ordinals, however the limit type of ω_w is the integers since it is the union of $\omega_1, \omega_2, \omega_3, ...,$. If

[13]The kernel version of `compare`, which does most of the work, acts on the terms that make up the Cantor normal form expression of an `Ordinal`. These terms belong to the base `class CantorNormalElement`. The kernel version of `compare` is a member function of this `class`. As the `Ordinal class` is expanded so is `CantorNormalElement`. As new `class`es are derived from this base `class` they are assigned an incremented integer level which can be accessed with `term.codeLevel` where `term` is a `CantorNormalElement`. This facility is used by `compare` to check if its argument is at a higher `class` level.

[14]The calculator plain text output format expands '`w_`' to '`omega_`'. Either version can be input.

[15]`limitElement(n)` is defined for limits that require `limitOrd(`α`)` but it can only compute an initial fragment of the defining notations. Like `limitElement`, `limitOrd` has its own exit codes (see Note 12). These codes are not listed here. They are documented in [3] and are important in insuring that the set of regression tests is complete.

Definition of $\omega_\kappa(\alpha)$ for α a successor		
L is lines in Table 9. X is an exit code (see Note 12).	L	X
If $\kappa = 1$ and α is a successor define $\omega_1(\alpha)^0 \quad = \omega_1(\alpha - 1)$ and define $\omega_1(\alpha)^{n+1} \quad = \varphi_{\omega_1(\alpha)^n+1}(\alpha - 1)$ then $\omega_1(\alpha) \quad = \cup_{n\in\omega}\, \omega_1(\alpha)^n$ which expands to $\omega_1(\alpha) \quad = \omega_1(\alpha - 1), \varphi_{\omega_1(\alpha-1)+1}(\alpha - 1), ...,.$	5	LCCI
If κ and α are successors and $\kappa > 1$ then define $\omega_\kappa(\alpha)^0 \quad = \omega_\kappa(\alpha - 1)$ and define $\omega_\kappa(\alpha)^{n+1} \quad = \omega_{\kappa-1,\omega_\kappa(\alpha)^n+1}(\alpha - 1)$ then $\omega_\kappa(\alpha) \quad = \cup_{n\in\omega}\, \omega_\kappa(\alpha)^n$ which expands to $\omega_\kappa(\alpha) \quad = \cup\, \omega_\kappa(\alpha - 1), \omega_{\kappa-1,\omega_\kappa(\alpha-1)+1}(\alpha - 1), ...,.$	7, 9	LCDP
If κ is a limit and α is a successor $\omega_\kappa(\alpha) \quad = \cup_{\zeta\in\kappa}\, \omega_{\zeta,\omega_\kappa(\alpha-1)+1}.$	11 12	LCEL
Section 3.2 defines $\omega_{\kappa,\gamma}(\alpha_1, \alpha_2, ..., \alpha_n)$ if $\gamma > 0 \vee n > 1 \vee \alpha_n$ a limit.		

Table 7: Definition of $\omega_\kappa(\alpha)$ for α a successor

the least significant nonzero parameter β of an ordinal ζ is a limit, then the limit type of ζ is the limit type of β.

The value of ω_κ`.limitOrd(`η`)` for κ a successor is expressed in the notation $\omega_\kappa[\eta]$. The plain text format for this is `w_{k}[e]`. This is the first case where the presence of a parameter in an `Ordinal` definition leads to a smaller `Ordinal` than if it were absent. This holds for all square bracketed parameters. This syntax with a square bracketed suffix is only meaningful (and accepted in the ordinal calculator) if all previously described parameters except κ are 0 and κ represents a successor. In addition η must meet constraints on limit type.

Limit type is implemented primarily through the member functions `limitType` and `maxLimitType`. `limitType` is determined by the least significant (nonzero sometimes) one or two parameters. `maxLimitType` is determined by the maximum limit type of all parameters and the value of κ[16]. α.`limitOrd`(β) is a valid expression if α.`limitType`() $>$ β.`maxLimitType`(). In addition parameters with a square bracketed suffix (either single or double as described in Section 5.2) are allowed if α.`limitType`() $=$ β.`maxLimitType`() and $\beta < \alpha$.

[16]The limit type of ω_κ is $\kappa+1$ if κ is finite and κ otherwise. The limit type of the integers and all successor notations is 0. Limit ordinals $< \omega_1$ have a limit type of 1. These limits have integer indices. The `limitType` and `maxLimitType` member functions are available in the interactive calculator.

By definition $\omega_\kappa = \bigcup_{\eta<\omega_\kappa} \omega_\kappa[\eta]$. Many values of η will not be defined in a recursive system of notations, but the system should be expandable to support any of them. This is an example of explicit incompleteness. $\omega_\kappa[\eta]$ and thus ω_κ`.limitOrd(`η`)` could be defined to be η. This is the identity function. However the goal is to define notations for as many ordinals as possible. Thus the function diagonalizes what has previously been defined. In doing so it must make sure that $(\eta < \omega_\kappa) \rightarrow (\omega_\kappa$`.limitOrd(`$\eta$`)` $< \omega_\kappa)$. This rules for this are in Table 8.

There are three classes of notations that require ζ`.limitOrd(`β`)`.

1. If the least significant nonzero parameter of ζ is υ a limit notation with limit type greater than the integers, then the parameter υ is replaced with $\upsilon[\beta]$.

2. If the least significant nonzero parameter of ζ, α_m, (one of the α_n in expression 2) represents a successor and the next least significant nonzero parameter, υ (either γ or one of the α_n in expression 2) represents an ordinal with limit type greater than the integers, then υ is replaced with $\upsilon[\beta]$. In addition the most significant parameter (zero or nonzero) less significant than υ is replaced with the original value of α except 1 is subtracted from α_m.

3. If κ is the least significant nonzero parameter of ζ then ζ`.limitOrd(`β`)` $= \zeta[\beta]$.

For notations for ordinals $\geq \omega_1^{\mathrm{CK}}$, the `compare` virtual member function works as defined in Section 1.1. As a consequence the incomplete ordinal notation hierarchy defined at any point in this process has a recursive well ordering implemented in the ordinal calculator. Thus the hierarchy and/or parts of it can be embedded within itself to fill some of the gaps between already defined notations. This is a bit like the Mandelbrot set[13] which repeatedly embeds its entire structure within itself. The embeddings of parts of ordinal notation within itself to fill in the gaps are a form of ordinal projection or collapsing.

5.2 Admissible ordinals and projection

Projection or collapsing uses the names of uncountable cardinals[14, 16] or countable admissible ordinals $\geq \omega_1^{\mathrm{CK}}$[17] to extend the recursive ordinal hierarchy. A similar approach can both expand recursive ordinal notations and partially fill some of the gaps that must occur in a recursive system of notations that represents ordinals $\geq \omega_1^{\mathrm{CK}}$. This section describes an approach to ordinal projection used in the calculator.

This version of ordinal projection prepends a notation, δ, in double square brackets as in $[[\delta]]\omega_\kappa$. The plain text for this is `[[d]]w_{k}`. δ must be a successor $\leq \kappa$. The δ prefix imposes a `limiType` of δ if δ is finite and $\delta - 1$ otherwise[17]. Recall that the `limitType` of ω_κ is $\kappa + 1$ if κ is finite and κ otherwise. The δ prefix also requires that any output from `limitElement` or `limitOrd` that is $\geq \omega_\delta$ must have the same δ prefix prepended to it. In addition a double bracketed suffix is defined such that $[[\delta]]w_\kappa$`.limitOrd(`β`)` $= [[\delta]]w_\kappa[[\beta]]$. The plain text for this is `[[d]]w_{k}[[b]]`. The double bracketed suffix diagonalizes the ordinals definable with a single bracketed suffix. See tables 10 and 11 for the double bracketed prefix and suffix definitions that require new rules.

[17]Note the δ prefix can never be a limit.

Definition of ω_κ and $\omega_\kappa[\eta]$		
L is lines in Table 9. X is an exit code (see Note 12).	L	X
If κ is a successor then $\omega_\kappa \quad = \bigcup_{\beta<\omega_\kappa} \omega_\kappa[\beta]$.	4	LEDC
$\omega_1[1] \quad = \bigcup \omega, \varphi_\omega, \varphi_{\varphi_\omega+1}, \varphi_{\varphi_{\varphi_\omega+1}+1}, \varphi_{\varphi_{\varphi_{\varphi_\omega+1}+1}+1}, \ldots,.$	1	DDBO
If $\eta > 1$ and a successor and $\kappa = 1$ then define $\omega_1[\eta]^0 \quad = \omega_1[\eta-1]$ and define $\omega_1[\eta]^{n+1} \quad = \varphi_{\omega_1[\eta]^n+1}$ then $\omega_1[\eta] \quad = \bigcup_{n\in\omega} \omega_1[\eta]^n$ which expands to $\omega_1[\eta] \quad = \bigcup \omega_1[\eta-1], \varphi_{\omega_1[\eta-1]+1}, \varphi_{\varphi_{\omega_1[\eta-1]+1}+1}, \ldots,.$	2	DDCO
If $\eta = 1$ and $\kappa > 1$ is a successor then define $\omega_\kappa[1]^0 \quad = \omega_{\kappa-1}$ and define $\omega_\kappa[1]^{n+1} \quad = \omega_{\kappa-1,\omega_\kappa[1]^n+1}$ then $\omega_\kappa[1] \quad = \bigcup_{n\in\omega} \omega_\kappa[1]^n$ which expands to $\omega_\kappa[1] \quad = \bigcup \omega_{\kappa-1}, \omega_{\kappa-1,\omega_{\kappa-1}+1}, \omega_{\kappa-1,\omega_{\kappa-1,\omega_{\kappa-1}+1}+1}, \ldots,.$	8	DCDO
If κ and η are successors > 1 define $\omega_\kappa[\eta]^0 \quad = \omega_\kappa[\eta-1]$ and define $\omega_\kappa[\eta]^{n+1} \quad = \omega_{\kappa-1,\omega_\kappa[\eta]^n+1}$ then $\omega_\kappa[\eta] \quad = \bigcup_{n\in\omega} \omega_\kappa[\eta]^n$ which expands to $\omega_\kappa[\eta] \quad = \bigcup \omega_\kappa[\eta-1], \omega_{\kappa-1,\omega_\kappa[\eta-1]+1}, \ldots,.$	6	DCES
If η is a limit then $\omega_\kappa[\eta] \quad = \bigcup_{\zeta<\eta} \omega_\kappa[\zeta]$.	3	DCAL
If κ is a limit then η must be 0 and $\omega_\kappa \quad = \bigcup_{\zeta<\kappa} \omega_\zeta$.	14	LCBL

Table 8: Definition of ω_κ and $\omega_\kappa[\eta]$

	α	α.`limitElement(n)`			
		`n=1`	`n=2`	`n=3`	X
1	$\omega_1[1]$	ω	φ_{ω}	$\varphi_{\varphi_{\omega}+1}$	DDBO
2	$\omega_1[3]$	$\omega_1[2]$	$\varphi_{\omega_1[2]+1}$	$\varphi_{\varphi_{\omega_1[2]+1}+1}$	DDCO
3	$\omega_1[\omega]$	$\omega_1[1]$	$\omega_1[2]$	$\omega_1[3]$	DCAL
4	ω_1	$\omega_1[1]$	$\omega_1[2]$	$\omega_1[3]$	LEDC
5	$\omega_1(12)$	$\omega_1(11)$	$\varphi_{\omega_1(11)+1}(11)$	$\varphi_{\varphi_{\omega_1(11)+1}(11)+1}(11)$	LCCI
6	$\omega_3[3]$	$\omega_3[2]$	$\omega_{2,\omega_3[2]+1}$	$\omega_{2,\omega_{2,\omega_3[2]+1}+1}$	DCES
7	$\omega_3(5)$	$\omega_3(4)$	$\omega_{2,\omega_3(4)+1}(4)$	$\omega_{2,\omega_{2,\omega_3(4)+1}(4)+1}(4)$	LCDP
8	$\omega_5[1]$	ω_4	ω_{4,ω_4+1}	$\omega_{4,\omega_{4,\omega_4+1}+1}$	DCDO
9	$\omega_5(8)$	$\omega_5(7)$	$\omega_{4,\omega_5(7)+1}(7)$	$\omega_{4,\omega_{4,\omega_5(7)+1}(7)+1}(7)$	LCDP
10	$\omega_{5,8}$	$\omega_{5,7}(\omega_{5,7}+1)$	$\omega_{5,7}(\omega_{5,7}+1,0)$	$\omega_{5,7}(\omega_{5,7}+1,0,0)$	IG
11	$\omega_{\omega}(5)$	$\omega_{1,\omega_{\omega}(4)+1}$	$\omega_{2,\omega_{\omega}(4)+1}$	$\omega_{3,\omega_{\omega}(4)+1}$	LCEL
12	$\omega_{\omega}(8)$	$\omega_{1,\omega_{\omega}(7)+1}$	$\omega_{2,\omega_{\omega}(7)+1}$	$\omega_{3,\omega_{\omega}(7)+1}$	LCEL
13	$\omega_{\omega,8}$	$\omega_{\omega,7}(\omega_{\omega,7}+1)$	$\omega_{\omega,7}(\omega_{\omega,7}+1,0)$	$\omega_{\omega,7}(\omega_{\omega,7}+1,0,0)$	IG
14	$\omega_{\varepsilon_{\omega}\omega}$	$\omega_{\varepsilon_{\omega}+1}$	$\omega_{\varepsilon_{\omega}2+1}$	$\omega_{\varepsilon_{\omega}3+1}$	LCBL

Table 9: Example notations $\geq \omega_1[1]$

The ordinal notations in the calculator reference ordinals $\geq \omega_1^{\mathrm{CK}}$ but the notations *defined within a specific recursive system* are recursively well ordered and thus can be used to expand the recursive ordinal hierarchy as well as the gaps between notations for lager ordinals. The δ prefix allows the use of notations with any value of κ to index smaller ordinals. Note $[[\delta]]\omega_\kappa < \omega_\delta < [[\delta+1]]\omega_{\delta+1}$ for all $\kappa \geq \delta$.

Definition of $[[\delta]]\omega_\kappa$, $[[\delta]]\omega_\kappa[[\eta]]$ and $[[\delta]]\omega_\kappa[\eta]$			
δ must be a successor with $\delta \leq \kappa$.			
If $\delta = \kappa$ in $[[\delta]]\omega_\kappa[\eta]$ then the δ prefix is dropped.			
L is example line(s) in Table 12. X is an exit code (see Note 12).			
		L	X
If κ is a successor then			
$[[\delta]]\omega_\kappa$	$= \cup_{\eta<[[\delta]]\omega_\kappa}[[\delta]]\omega_\kappa[[\eta]]$.	3, 8	LEDE
If κ is a limit then			
$[[\delta]]\omega_\kappa$	$= \cup_{\alpha<\kappa\wedge\alpha>\delta}[[\delta]]\omega_\alpha$	13	LEEE
If $\kappa = \delta$ and $\eta = 1$ define			
$[[\delta]]\omega_\delta[[1]]^0$	$= \omega$ and define		
$[[\delta]]\omega_\delta[[1]]^{n+1}$	$= \omega_\delta[\omega_\delta[[1]]^n + 1]$ then		
$[[\delta]]\omega_\delta[[1]]$	$= \cup_{n\in\omega}[[\delta]]\omega_\delta[[1]]^n$ which expands to		
$[[\delta]]\omega_\delta[[1]]$	$= \cup\,\omega, \omega_\delta[\omega], \omega_\delta[\omega_\delta[\omega], \omega_\delta[\omega_\delta[\omega_\delta[\omega]], ...,.$	1, 9	DCBO
If $\kappa > \delta$, κ a successor and $\eta = 1$ define			
$[[\delta]]\omega_\kappa[[1]]^0$	$= [[\delta]]\omega_{\kappa-1}$ and define		
$[[\delta]]\omega_\kappa[[1]]^{n+1}$	$= \omega_{\kappa-1,[[\delta]]\omega_\kappa[[1]]^n+1}$ then		
$[[\delta]]\omega_\kappa[[1]]$	$= \cup_{n\in\omega}[[\delta]]\omega_\kappa[[1]]^n$ which expands to		
$[[\delta]]\omega_\kappa[[1]]$	$= \cup[[\delta]]\omega_{\kappa-1}, \omega_{\kappa-1,[[\delta]]\omega_{\kappa-1}+1},$		
	$\omega_{\kappa-1,\omega_{\kappa-1,[[\delta]]\omega_{\kappa-1}+1}}, ...,.$	11	DCDO
If $\eta > 1$ is a successor then			
$[[\delta]]\omega_\kappa[[\eta]]^0$	$= [[\delta]]\omega_\kappa[[\eta-1]]$ and define		
$[[\delta]]\omega_\kappa[[\eta]]^{n+1}$	$= [[\delta]]\omega_\kappa[[[\delta]]\omega_\kappa[[\eta]]^n + 1]$ then		
$[[\delta]]\omega_\kappa[[\eta]]$	$= \cup_{n\in\omega}[[\delta]]\omega_\kappa[[\eta]]^n$ which expands to		
$[[\delta]]\omega_\kappa[[\eta]]$	$= \cup[[\delta]]\omega_\kappa[[\eta-1]],$		
	$[[\delta]]\omega_\kappa[[[\delta]]\omega_\kappa[[\eta-1]] + 1],$		
	$[[\delta]]\omega_\kappa[[[\delta]]\omega_\kappa[[[\delta]]\omega_\kappa[[\eta-1]] + 1] + 1], ...,.$	2, 5	DCCS
If η is a limit then			
$[[\delta]]\omega_\kappa[[\eta]]$	$= \cup_{\beta<\eta}[[\delta]]\omega_\kappa[[\beta]]$ and	14	DCAL
$[[\delta]]\omega_\kappa[\eta]$	$= \cup_{\beta<\eta}[[\delta]]\omega_\kappa[\beta]$.	10	DCAL

Table 10: Definition of $[[\delta]]\omega_\kappa$, $[[\delta]]\omega_\kappa[[\eta]]$ and $[[\delta]]\omega_\kappa[\eta]$

Definition of $[[\delta]]\omega_\kappa(\alpha)$ δ must be a successor with $\delta \leq \kappa$. L is example line(s) in Table 12. X is an exit code (see Note 12).		
	L	X
$[[1]]\omega_1(1) \quad = \cup [[1]]\omega_1, \varphi_{[[1]]\omega_1+1}, \varphi_{\varphi_{[[1]]\omega_1+1}+1}, ...,.$	4	LECK
If $\kappa = \delta \wedge \kappa > 1$ then $[[\kappa]]\omega_\kappa(1) \quad = \cup [[\kappa]]\omega_\kappa, \omega_{\kappa-1,[[\kappa]]\omega_\kappa+1}, \omega_{\kappa-1,\omega_{\kappa-1,[[\kappa]]\omega_\kappa+1}+1}, ...,.$	6	LECK
If $\kappa > 1 \wedge \kappa > \delta \wedge \alpha = 1$ define $[[\delta]]\omega_\kappa(1)^0 \quad = [[\delta]]\omega_\kappa$ and define $[[\delta]]\omega_\kappa(1)^{n+1} \quad = [[\delta]]\omega_{\kappa-1,[[\delta]]\omega_\kappa(1)^n+1}$ then $[[\delta]]\omega_\kappa(1) \quad = \cup_{n\in\omega}[[\delta]]\omega_\kappa(1)^n$ which expands to $[[\delta]]\omega_\kappa(1) \quad = \cup [[\delta]]\omega_\kappa, [[\delta]]\omega_{\kappa-1,[[\delta]]\omega_\kappa+1},$ $[[\delta]]\omega_{\kappa-1,[[\delta]]\omega_{\kappa-1,[[\delta]]\omega_\kappa+1}+1}, ...,.$	7	LCDP

Table 11: Definition of $[[\delta]]\omega_\kappa(\alpha)$

	α	α.`limitElement(n)`			
		`n=1`	`n=2`	`n=3`	X
1	$[[1]]\omega_1[[1]]$	ω	$\omega_1[\omega]$	$\omega_1[\omega_1[\omega]]$	DCBO
2	$[[1]]\omega_1[[2]]$	$[[1]]\omega_1[[1]]$	$\omega_1[[[1]]\omega_1[[1]]]$	$\omega_1[\omega_1[[[1]]\omega_1[[1]]]]$	DCCS
3	$[[1]]\omega_1$	$[[1]]\omega_1[[1]]$	$[[1]]\omega_1[[2]]$	$[[1]]\omega_1[[3]]$	LEDE
4	$[[1]]\omega_1(1)$	$[[1]]\omega_1$	$\varphi_{[[1]]\omega_1+1}$	$\varphi_{\varphi_{[[1]]\omega_1+1}+1}$	LECK
5	$[[1]]\omega_5[[3]]$	$[[1]]\omega_5[[2]]$	$[[1]]\omega_5[[[1]]\omega_5[[2]]]$	$[[1]]\omega_5[[[1]]\omega_5[[[1]]\omega_5[[2]]]]$	DCCS
6	$[[2]]\omega_2(1)$	$[[2]]\omega_2$	$\omega_{1,[[2]]\omega_2+1}$	$\omega_{1,\omega_{1,[[2]]\omega_2+1}+1}$	LECK
7	$[[2]]\omega_3(1)$	$[[2]]\omega_3$	$[[2]]\omega_{2,[[2]]\omega_3+1}$	$[[2]]\omega_{2,[[2]]\omega_{2,[[2]]\omega_3+1}+1}$	LCDP
8	$[[2]]\omega_8$	$[[2]]\omega_8[[1]]$	$[[2]]\omega_8[[2]]$	$[[2]]\omega_8[[3]]$	LEDE
9	$[[3]]\omega_3[[1]]$	ω	$\omega_3[\omega]$	$\omega_3[\omega_3[\omega]]$	DCBO
10	$[[4]]\omega_8[\omega]$	$[[4]]\omega_8[1]$	$[[4]]\omega_8[2]$	$[[4]]\omega_8[3]$	DCAL
11	$[[5]]\omega_6[1]$	$[[5]]\omega_5$	$[[5]]\omega_{5,[[5]]\omega_5+1}$	$[[5]]\omega_{5,[[5]]\omega_{5,[[5]]\omega_5+1}+1}$	DCDO
12	$[[5]]\omega_7[4]$	$[[5]]\omega_7[3]$	$[[5]]\omega_{6,[[5]]\omega_7[3]+1}$	$[[5]]\omega_{6,[[5]]\omega_{6,[[5]]\omega_7[3]+1}+1}$	DCES
13	$[[5]]\omega_\omega$	$[[5]]\omega_6$	$[[5]]\omega_7$	$[[5]]\omega_8$	LEEE
14	$[[8]]\omega_8[[\omega]]$	$[[8]]\omega_8[[1]]$	$[[8]]\omega_8[[2]]$	$[[8]]\omega_8[[3]]$	DCAL

Table 12: Example notations using projection

6 Ordinal projection with nested embedding

The embedding described in Section 5.2 is nested by expanding the single ordinal notation prefix, δ to a sequence of paired notations written as $\delta \lmoustache \sigma$[18]. The $\lmoustache \sigma$ is optional. The full syntax is in Figure 1. The first δ must be $\leq \kappa$ it has the same effect as the single δ prefix did in limiting the parameters for `limitOrd`. The remaining prefix parameters support nested embedding. The idea is to embed previously defined notations inside themselves to expand the recursive ordinals notations in the system and fill other gaps between notations. The notations in the prefix index this embedding. Thus the prefix, going from left (most significant) to right, contains the most significant parameters in the notation. Both the finite parameter Veblen function and the notation prefix have significance going from left to right but there is an important difference. The most significant parameter for the Veblen function is the number of parameters. $\varphi(1, 0, 0) > \varphi(99, 99)$. This is not true with nested embedding. The most significant parameter is the leftmost value of the prefix. The next most significant parameter is the number of δs in the prefix. Then the remainder of the prefix values going from left to right, followed by κ and the remaining parameters.

In describing these notations, the cases for which there are new rules are those in which the prefix of ζ differs from the prefix in ζ.`limitOrd`(β). This only occurs if $\kappa = \delta_m$ in the notation ζ and one of the following three conditions are met.

1. κ is a limit and the only nonzero parameter not in the prefix.

2. The only nonzero parameters not in the prefix are κ and the single bracketed suffix $[\eta]$ which is a successor.

3. The only nonzero parameters not in the prefix are κ and the least significant α which is a successor.

In all other cases computing $\upsilon = \zeta$.`limitOrd`(β) uses rules described in previous sections.

When new rules are needed to define `limitElement` and `limitOrd`, there are multiple conditions that require the prefix to be decremented and multiple states it may be in that require different algorithms to change it. Both the code and its documentation matches this structure. For example several code fragments with different exit codes call the same subroutines to manipulate the prefix. This documentation has a similar functional structure. The exit codes no longer have a nearly one to one correspondence with rules. Table 13 gives some conventions used in tables 14, 15 and 16. Table 14, describes the conditions that determine when and how a prefix is decremented. Table 15 describes the new rules that involve prefix changes defined in Table 14. These two tables together provide all inductive definitions of sequences that define a limit ordinal notation with prefix changes. Table 15 gives the new rules for prefix changes defined by functions on an ordinal parameter. Examples of all the rules are shown in Table 17. The rules for parameters that leave the prefix unchanged have been defined previously. They are in Section 3.2 and tables 5, 7, 8 and 10.

The strength of nested embedding comes in part from appending a large value of σ when the least significant nonzero δ parameter is decremented and making κ much larger

[18]The character '$\lmoustache$' (`\lmoustache` in LaTeX math mode) was chosen to indicate that the two parameters are connected as a pair.

Nested embedding syntax in LaTeX

$$[[\delta_1\diagup\sigma_1, \delta_2\diagup\sigma_2, ..., \delta_m\diagup\sigma_m]]\omega_\kappa$$

$$[[\delta_1\diagup\sigma_1, \delta_2\diagup\sigma_2, ..., \delta_m\diagup\sigma_m]]\omega_\kappa[[\eta]]$$

$$[[\delta_1\diagup\sigma_1, \delta_2\diagup\sigma_2, ..., \delta_m\diagup\sigma_m]]\omega_\kappa[\eta]$$

$$[[\delta_1\diagup\sigma_1, \delta_2\diagup\sigma_2, ..., \delta_m\diagup\sigma_m]]\omega_{\kappa,\lambda}(\alpha_1, \alpha_2, ..., \alpha_n)$$

The $\diagup\sigma_k$ $k = 1, 2, ..., m$ are optional.

Nested embedding syntax in plain text

```
[[d1/s1,d2/s3,...,dm/sm]]w_{k}
[[d1/s1,d2/s3,...,dm/sm]]w_{k}[[e]]
[[d1/s1,d2/s3,...,dm/sm]]w_{k}[e]
[[d1/s1,d2/s3,...,dm/sm]]w_{k,g}(a1,a2....,am)
```

The "`/sk`" `k=1,2,...,m` are optional.

There are several restrictions on these expressions.

1. $\forall_{k<m}\{(\delta_k < \delta_{k+1}) \vee ((\delta_k = \delta_{k+1}) \wedge (\sigma_k < \sigma_{k+1}))\}$.
2. $\kappa \leq \delta_m$.
3. If κ is a limit then no η parameter is allowed.
4. The most significant δ cannot be a limit.
5. If any other δ is a limit, then the associated σ must be 0.
6. If σ_m is a limit, then no η parameter is allowed.
7. $[[\delta]]\omega_\delta[\eta] = \omega_\delta[\eta]$ and thus the $[[\delta]]$ prefix is deleted if $\delta = \kappa$ in a notation with a single bracketed suffix.

Figure 1: Nested embedding syntax and constraints

1. $S(\zeta) \equiv \zeta$ is a successor notation.
2. $L(\zeta) \equiv \zeta$ is a limit notation.
3. ζ^- is defined if the least significant nonzero parameter in notation ζ is a successor. ζ^- is identical with ζ except the least significant nonzero parameter is decremented by 1.
4. $\zeta[[-]]$ is defined if ζ meets the conditions for defining a next least prefix in Table 14. $\zeta[[-]]$ is the prefix of ζ modified as defined by this table. All definitions in this table that contain double square brackets define prefixes.
5. $D(X)$ means X is defined. This applies to conditional definitions 3 and 4.
6. $\zeta[[-\frown\beta]]$ is identical with $\zeta[[-]]$ except δ_m must the least significant nonzero prefix parameter in ζ and β is appended to the prefix defined in Table 14 as σ_{m+1}. There is an exception. Nothing is appended if the prefix from the table is shorter than the prefix in ζ. A shorter prefix means the only legal way to decrement the prefix is to shorten it.
7. `le` is an abbreviation for `limitElement`.
8. `lo` is an abbreviation for `limitOrd`.
9. `il` is an abbreviation for `increasingLimit`. The routine, `il(`δ_m, γ`)` outputs increasing values $> \delta_m$ for increasing γ. This is needed to insure that the ourput of `le` and `lo` produce increasing outputs for increasing inputs.

Table 13: Conventions used in tables 14, 15 and 16

when the least significant σ is decremented. Doing this requires first decrementing a non prefix parameter and the resulting notation is `limitElement(1)`. If δ_m is decremented, `limitElement(n+1)` contains the decremented prefix with `limitElement(n)` appended as σ_m and the rest of the notation is the same as `limitElement(1)`. If σ_m is decremented, `limitElement(n+1)` contains the decremented prefix. κ in `limitElement(n+1)` is set equal to `limitElement(n)`. These cases are fully described in Table 15.

Figure 2 with notations in LaTeX format and Figure 3 in plain text each summarize the syntax in version 0.3.2 of the ordinal calculator. These figures are references with links to the sections and tables that describe all parts of the syntax. The figures contain the same information except for the notation format.

Next least prefix of $[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]$
This is defined if the least significant prefix parameter, δ_m or σ_m, is a successor and $\delta_m = \kappa$.
See Table 13 for conventions. The L column is line(s) in Table 17.

δ_{m-1} and δ_m	σ_{m-1} and σ_m	Next least prefix	L
$\delta_m = \delta_{m-1}$	$\sigma_m = \sigma_{m-1} + 1$	$[[\delta_1 \nearrow \sigma_1, , ..., \delta_{m-1} \nearrow \sigma_{m-1}]]$	11
$\delta_m = \delta_{m-1}$	$\sigma_m > \sigma_{m-1} + 1$	$[[\delta_1 \nearrow \sigma_1, , ..., \delta_{m-1} \nearrow \sigma_{m-1}, \delta_m \nearrow \sigma_m - 1]]$	5
$(\delta_m > \delta_{m-1}) \vee (m = 1)$	$S(\sigma_m)$	$[[\delta_1 \nearrow \sigma_1, , ..., \delta_{m-1} \nearrow \sigma_{m-1}, \delta_m \nearrow \sigma_m - 1]]$	2,3,15
$(\delta_m + 1 = \delta_{m-1}) \wedge L(\delta_{m-1})$	$\sigma_m = \sigma_{m-1} = 0$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_{m-1}]]$	4
$(\delta_m + 1 = \delta_{m-1}) \wedge S(\delta_{m-1})$	$\sigma_m = 0$	$[[\delta_1 \nearrow \sigma_1, , ..., \delta_{m-1} \nearrow \sigma_{m-1}, \delta_m - 1 \nearrow \sigma_{m-1} + 1]]$	16,23,24

Table 14: Next least prefix

Definition of $\zeta = [[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]\omega_\kappa[\eta]$
if $(\sigma_1 > 0 \vee m > 1) \wedge \eta > 0 \wedge \kappa = \delta_m \wedge D(\zeta^-) \wedge D(\zeta[[-]])$.
See Table 13 for conventions. The L column is example line(s) in Table 17.

Conditions on ζ	ζ`.le(1)`	ζ`.le(n+1)`	L	X
$S(\sigma_m) \wedge \eta > 1$	ζ^-	$\zeta[[-]]\omega_{\zeta.\mathtt{le(n)}}$	6,12,33	DQB,DQE,DQC
$S(\sigma_m) \wedge \eta = 1$	$\zeta[[-]]\omega_\kappa$	$\zeta[[-]]\omega_{\zeta.\mathtt{le(n)}}$	1,5,11,32	DQA,DQB,DQE,DQC
$\sigma_m = 0 \wedge S(\delta_m) \wedge \eta > 1$	ζ^-	$\zeta[[- \nearrow \zeta.\mathtt{le(n)}]]\omega_\kappa$	10,24	DQD,DQD
$\sigma_m = 0 \wedge S(\delta_m) \wedge \eta = 1 \wedge \delta_m > \delta_{m-1} + 1$	$\zeta[[-]]w_\kappa$	$\zeta[[- \nearrow \zeta.\mathtt{le(n)}]]\omega_\kappa$	7	DQD
$\sigma_m = 0 \wedge S(\delta_m) \wedge \eta = 1 \wedge \delta_m = \delta_{m-1} + 1$	$\zeta[[- \nearrow 1]]w_\kappa$	$\zeta[[- \nearrow \zeta.\mathtt{le(n)}]]\omega_\kappa$	16	DQD

Definition of $\zeta = [[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]\omega_\kappa(\alpha)$
if $(\sigma_1 > 0 \vee m > 1) \wedge \kappa = \delta_m \wedge D(\zeta^-) \wedge D(\zeta[[-]])$.

Conditions on ζ	ζ`.le(1)`	ζ`.le(n+1)`	L	X
$S(\sigma_m) \wedge \alpha > 1$	ζ^-	$\zeta[[-]]\omega_{\zeta.\mathtt{le(n)}}$	21,27	PLEC,PLED
$S(\sigma_m) \wedge \alpha = 1$	ζ^-	$\zeta[[-]]\omega_{\zeta.\mathtt{le(n)}}$	30	PLED
$\sigma_m = 0 \wedge S(\delta_m) \wedge \alpha > 1$	ζ^-	$\zeta[[- \nearrow \zeta.\mathtt{le(n)}]]\omega_\kappa$	28	PLEE
$\sigma_m = 0 \wedge S(\delta_m) \wedge \alpha = 1 \wedge \delta_m > \delta_{m-1} + 1$	$\zeta[[-]]w_\kappa$	$\zeta[[- \nearrow \zeta.\mathtt{le(n)}]]\omega_\kappa$	37	PLEE
$\sigma_m = 0 \wedge S(\delta_m) \wedge \alpha = 1 \wedge \delta_m = \delta_{m-1} + 1$	$\zeta[[-]]w_\kappa$	$\zeta[[- \nearrow \zeta.\mathtt{le(n)}]]\omega_\kappa$	36	PLEE

Table 15: Rules that change $[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]$ by decrementing it

Definition of $\zeta = [[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]\omega_\kappa$ for $((L(\delta_m) \wedge \sigma_m = 0) \vee L(\sigma_m)) \wedge \delta_m = \kappa$. See Table 13 for conventions. The L column is example line(s) in Table 17.			
δ and σ	ζ`.limitOrd(`β`)`	L	X
$S(\delta_m) \wedge L(\sigma_m) \wedge \delta_{m-1} < \kappa$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_{m-1} \nearrow \sigma_{m-1}, \kappa \nearrow \sigma_m[\beta]]]\omega_\kappa$	29	NEF
$S(\delta_m) \wedge L(\sigma_m) \wedge \delta_{m-1} = \kappa$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \kappa \nearrow \sigma_{m-1}, \kappa \nearrow (\sigma_{m-1} + \sigma_m[\beta])]]\omega_\kappa$	35	NEF
$L(\delta_m) \wedge \sigma_m = 0$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_{m-1} \nearrow \sigma_{m-1}, (\delta_{m-1} + \kappa[\beta])]]\omega_{\delta_{m-1}+\kappa[\beta]}$	26	NEG
Definition of $\zeta = [[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]\omega_\kappa(\alpha)$ for $S(\alpha) \wedge ((L(\delta_m) \wedge \sigma_m = 0) \vee L(\sigma_m))$.			
δ and σ $(\delta_m = \kappa)$	ζ`.limitOrd(`β`)`	L	X
$S(\delta_m) \wedge L(\sigma_m) \wedge \delta_{m-1} < \kappa$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_{m-1} \nearrow \sigma_{m-1}, \kappa \nearrow \sigma_m[\beta]]]\omega_{\kappa,\alpha^-}$	41	NEB
$S(\delta_m) \wedge L(\sigma_m) \wedge \delta_{m-1} = \kappa$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \kappa \nearrow \sigma_{m-1}, \kappa \nearrow (\sigma_{m-1} + \sigma_m[\beta])]]\omega_{\kappa,\alpha^-}$	34	NEB
$L(\delta_m) \wedge \sigma_m = 0$	$[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_{m-1} \nearrow \sigma_{m-1}, (\delta_{m-1} + \kappa[\beta])]]\omega_{\delta_{m-1}+\kappa[\beta],\alpha^-}$	25	NEC

Table 16: Rules that change $[[\delta_1 \nearrow \sigma_1, \delta_2 \nearrow \sigma_2, ..., \delta_m \nearrow \sigma_m]]$ by taking a limit

	α	α.`limitElement(n)`		
		`n=1`	`n=2`	X
1	$[[1\nearrow 1]]\omega_1[12]$	$[[1\nearrow 1]]\omega_1[11]$	$[[1]]\omega_{[[1\nearrow 1]]\omega_1[11]}$	DQA
2	$[[1,3\nearrow 1]]\omega_3[1]$	$[[1,3]]\omega_3$	$[[1,3]]\omega_{[[1,3]]\omega_3}$	DQB
3	$[[1,3\nearrow 1]]\omega_3[4]$	$[[1,3\nearrow 1]]\omega_3[3]$	$[[1,3]]\omega_{[[1,3\nearrow 1]]\omega_3[3]}$	DQB
4	$[[1,\omega,\omega+1]]\omega_{\omega+1}[1]$	$[[1,\omega]]\omega_{\omega+1}$	$[[1,\omega]]\omega_{[[1,\omega]]\omega_{\omega+1}}$	DQE
5	$[[2,3\nearrow 8]]\omega_3[1]$	$[[2,3\nearrow 7]]\omega_3$	$[[2,3\nearrow 7]]\omega_{[[2,3\nearrow 7]]\omega_3}$	DQB
6	$[[2,3\nearrow 8]]\omega_3[5]$	$[[2,3\nearrow 8]]\omega_3[4]$	$[[2,3\nearrow 7]]\omega_{[[2,3\nearrow 8]]\omega_3[4]}$	DQB
7	$[[2,4]]\omega_4[1]$	$[[2,3]]\omega_4$	$[[2,3\nearrow[[2,3]]\omega_4+1]]\omega_4$	DQD
8	$[[2,5\nearrow\omega_1]]\omega_{\omega_{12}[\omega]+5}$	$[[2,5\nearrow\omega_1]]\omega_{\omega_{12}[\omega]+5}[[1]]$	$[[2,5\nearrow\omega_1]]\omega_{\omega_{12}[\omega]+5}[[2]]$	LEDE
9	$[[2\nearrow 5,4]]\omega_4[1]$	$[[2\nearrow 5,3]]\omega_4$	$[[2\nearrow 5,3\nearrow[[2\nearrow 5,3]]\omega_4+1]]\omega_4$	DQD
10	$[[2\nearrow 5,4]]\omega_4[3]$	$[[2\nearrow 5,4]]\omega_4[2]$	$[[2\nearrow 5,3\nearrow[[2\nearrow 5,4]]\omega_4[2]+1]]\omega_4$	DQD
11	$[[2,3\nearrow 4,3\nearrow 5]]\omega_3[1]$	$[[2,3\nearrow 4]]\omega_3$	$[[2,3\nearrow 4]]\omega_{[[2,3\nearrow 4]]\omega_3}$	DQE
12	$[[2,3\nearrow 4,3\nearrow 5]]\omega_3[6]$	$[[2,3\nearrow 4,3\nearrow 5]]\omega_3[5]$	$[[2,3\nearrow 4]]\omega_{[[2,3\nearrow 4,3\nearrow 5]]\omega_3[5]}$	DQE
13	$[[2,3\nearrow 4,3\nearrow 8]]\omega_3[[1]]$	ω	$[[2,3\nearrow 4,3\nearrow 8]]\omega_3[\omega]$	DCBO
14	$[[3\nearrow\omega]]\omega_3$	$[[3\nearrow 1]]\omega_3$	$[[3\nearrow 2]]\omega_3$	NEF
15	$[[3\nearrow\omega+1]]\omega_3[1]$	$[[3\nearrow\omega]]\omega_3$	$[[3\nearrow\omega]]\omega_{[[3\nearrow\omega]]\omega_3}$	DQB
16	$[[3,4]]\omega_4[1]$	$[[3,3\nearrow 1]]\omega_4$	$[[3,3\nearrow[[3,3\nearrow 1]]\omega_4+1]]\omega_4$	DQD
17	$[[3,\omega]]\omega_\omega$	$[[3,4]]\omega_4$	$[[3,5]]\omega_5$	NEG
18	$[[3,\omega]]\omega_\omega(1)$	$[[3,4]]\omega_{4,[[3,\omega]]\omega_\omega+1}$	$[[3,5]]\omega_{5,[[3,\omega]]\omega_\omega+1}$	NEC
19	$[[3,\omega]]\omega_{\omega^\omega}$	$[[3,\omega]]\omega_{\omega^2}$	$[[3,\omega]]\omega_{\omega^2+\omega}$	LEEE
20	$[[3,\omega 15]]\omega_{\omega^\omega}$	$[[3,\omega 15]]\omega_{\omega 16}$	$[[3,\omega 15]]\omega_{\omega^2+\omega 15}$	LEEE
21	$[[3\nearrow 2,3\nearrow 3]]\omega_3(8)$	$[[3\nearrow 2,3\nearrow 3]]\omega_3(7)$	$[[3\nearrow 2]]\omega_{[[3\nearrow 2,3\nearrow 3]]\omega_3(7)}$	PLEC
22	$[[3\nearrow 5,3\nearrow 7]]\omega_3[[9]]$	$[[3\nearrow 5,3\nearrow 7]]\omega_3[[8]]$	$[[3\nearrow 5,3\nearrow 7]]\omega_3[[[3\nearrow 5,3\nearrow 7]]\omega_3[[8]]]$	DCCS
23	$[[3\nearrow 5,4]]\omega_4[1]$	$[[3\nearrow 5,3\nearrow 6]]\omega_4$	$[[3\nearrow 5,3\nearrow[[3\nearrow 5,3\nearrow 6]]\omega_4+1]]\omega_4$	DQD
24	$[[3\nearrow 5,4]]\omega_4[3]$	$[[3\nearrow 5,4]]\omega_4[2]$	$[[3\nearrow 5,3\nearrow[[3\nearrow 5,4]]\omega_4[2]+1]]\omega_4$	DQD
25	$[[3\nearrow 5,\omega]]\omega_\omega(5)$	$[[3\nearrow 5,4]]\omega_{4,[[3\nearrow 5,\omega]]\omega_\omega(4)+1}$	$[[3\nearrow 5,5]]\omega_{5,[[3\nearrow 5,\omega]]\omega_\omega(4)+1}$	NEC
26	$[[3\nearrow\omega,\omega]]\omega_\omega$	$[[3\nearrow\omega,4]]\omega_4$	$[[3\nearrow\omega,5]]\omega_5$	NEG
27	$[[4,7\nearrow 8]]\omega_7(9)$	$[[4,7\nearrow 8]]\omega_7(8)$	$[[4,7\nearrow 7]]\omega_{[[4,7\nearrow 8]]\omega_7(8)}$	PLED
28	$[[4\nearrow 8,7]]\omega_7(9)$	$[[4\nearrow 8,7]]\omega_7(8)$	$[[4\nearrow 8,6\nearrow[[4\nearrow 8,7]]\omega_7(8)+1]]\omega_7$	PLEE
29	$[[4\nearrow 12,5\nearrow\omega_3]]\omega_5$	$[[4\nearrow 12,5\nearrow\omega_3[1]]]\omega_5$	$[[4\nearrow 12,5\nearrow\omega_3[2]]]\omega_5$	NEF
30	$[[4\nearrow 12,7\nearrow 8]]\omega_7(1)$	$[[4\nearrow 12,7\nearrow 8]]\omega_7$	$[[4\nearrow 12,7\nearrow 7]]\omega_{[[4\nearrow 12,7\nearrow 8]]\omega_7}$	PLED
31	$[[5,6]]\omega_6(9)$	$[[5,6]]\omega_6(8)$	$[[5,5\nearrow[[5,6]]\omega_6(8)+1]]\omega_6$	PLEE
32	$[[5,\omega+1]]\omega_{\omega+1}[1]$	$[[5,\omega]]\omega_{\omega+1}$	$[[5,\omega]]\omega_{[[5,\omega]]\omega_{\omega+1}}$	DQC
33	$[[5,\omega+1]]\omega_{\omega+1}[9]$	$[[5,\omega+1]]\omega_{\omega+1}[8]$	$[[5,\omega]]\omega_{[[5,\omega+1]]\omega_{\omega+1}[8]}$	DQC
34	$[[5\nearrow\omega 3,5\nearrow\omega^2]]\omega_5(12)$	$[[5\nearrow\omega 3,5\nearrow\omega 4]]\omega_{5,[[5\nearrow\omega 3,5\nearrow\omega^2]]\omega_5(11)+1}$	$[[5\nearrow\omega 3,5\nearrow\omega 5]]\omega_{5,[[5\nearrow\omega 3,5\nearrow\omega^2]]\omega_5(11)+1}$	NEB
35	$[[5\nearrow\omega 100,5\nearrow\omega^2]]\omega_5$	$[[5\nearrow\omega 100,5\nearrow\omega 101]]\omega_5$	$[[5\nearrow\omega 100,5\nearrow\omega 102]]\omega_5$	NEF
36	$[[8\nearrow\omega 4,9]]\omega_9(1)$	$[[8\nearrow\omega 4,9]]\omega_9$	$[[8\nearrow\omega 4,8\nearrow[[8\nearrow\omega 4,9]]\omega_9+1]]\omega_9$	PLEE
37	$[[8\nearrow\omega 4,12]]\omega_{12}(1)$	$[[8\nearrow\omega 4,12]]\omega_{12}$	$[[8\nearrow\omega 4,11\nearrow[[8\nearrow\omega 4,12]]\omega_{12}+1]]\omega_{12}$	PLEE
38	$[[12\nearrow 1]]\omega_{12}(3)$	$[[12\nearrow 1]]\omega_{12}(2)$	$[[12]]\omega_{12,[[12\nearrow 1]]\omega_{12}(2)+1}$	PLEB
39	$[[12,\omega_3]]\omega_{\omega_4}(\omega+1)$	$[[12,\omega_3]]\omega_{\omega_4[1]+\omega_3,[[12,\omega_3]]\omega_{\omega_4}(\omega)+1}$	$[[12,\omega_3]]\omega_{\omega_4[2]+\omega_3,[[12,\omega_3]]\omega_{\omega_4}(\omega)+1}$	LCEL
40	$[[15\nearrow 4]]\omega_{15}(7)$	$[[15\nearrow 4]]\omega_{15}(6)$	$[[15\nearrow 3]]\omega_{[[15\nearrow 4]]\omega_{15}(6)}$	PLED
41	$[[15\nearrow\omega_3]]\omega_{15}(12)$	$[[15\nearrow\omega_3[1]]]\omega_{15,[[15\nearrow\omega_3]]\omega_{15}(11)+1}$	$[[15\nearrow\omega_3[2]]]\omega_{15,[[15\nearrow\omega_3]]\omega_{15}(11)+1}$	NEB

Table 17: Nested embed ordinal notations in increasing order

Cantor normal form (See Section 2).

$$\alpha_1 > \alpha_2 > \alpha_3 > ... > \alpha_k$$

α and the α_i are ordinal notations, n_i are nonzero integers

$$\omega^{\alpha_1} n_1 + \omega^{\alpha_2} n_2 + \omega^{\alpha_3} n_3 + ... + \omega^{\alpha_k} n_k \tag{3}$$

Finite parameter Veblen functions (See sections 3.1, 3.2 and tables 1, 2, 3 and 4.)

$$\varphi(\alpha_1, \alpha_2, ..., \alpha_k) \tag{4}$$

Transfinite Veblen functions (See Section 3.3 and tables 5 and 6.)

$$\varphi_\gamma(\alpha_1, \alpha_2, ..., \alpha_m) \tag{5}$$

Countable admissible ordinals (See Section 5.1 and tables 7, 8 and 9.)

$$\omega_\kappa \tag{6}$$

$$\omega_\kappa[\eta] \tag{7}$$

$$\omega_{\kappa,\gamma}(\alpha_1, \alpha_2, ..., \alpha_m) \tag{8}$$

Projection on countable admissible ordinals (See Section 5.2 and tables 10, 11 and 12.)

$$[[\delta]]\omega_\kappa \tag{9}$$

$$[[\delta]]\omega_\kappa[[\eta]] \tag{10}$$

$$[[\delta]]\omega_\kappa[\eta] \tag{11}$$

$$[[\delta]]\omega_{\kappa,\gamma}(\alpha_1, \alpha_2, ..., \alpha_m) \tag{12}$$

Nested ordinal projection (See Section 6, tables 13, 14, 15, 16, 17 and Figure 1.)

$$[[\delta_1 \swarrow \sigma_1, \delta_2 \swarrow \sigma_2, ..., \delta_m \swarrow \sigma_m]]\omega_\kappa \tag{13}$$

$$[[\delta_1 \swarrow \sigma_1, \delta_2 \swarrow \sigma_2, ..., \delta_m \swarrow \sigma_m]]\omega_\kappa[\eta] \tag{14}$$

$$[[\delta_1 \swarrow \sigma_1, \delta_2 \swarrow \sigma_2, ..., \delta_m \swarrow \sigma_m]]\omega_\kappa[[\eta]] \tag{15}$$

$$[[\delta_1 \swarrow \sigma_1, \delta_2 \swarrow \sigma_2, ..., \delta_m \swarrow \sigma_m]]\omega_{\kappa,\gamma}(\alpha_1, \alpha_2, ..., \alpha_k) \tag{16}$$

The $\swarrow \sigma_k$ $k = 1, 2, ..., m$ are optional.

Figure 2: Ordinal calculator notation LaTeX format

Cantor normal form (See Section 2).

$$\alpha_1 > \alpha_2 > \alpha_3 > ... > \alpha_k$$

α and the α_i are ordinal notations, n_i are nonzero integers

`w^a1*n1+w^a2*n2+w^a3*n3+...+w^ak*nk` (3)

Finite parameter Veblen functions (See sections 3.1, 3.2 and tables 1, 2, 3 and 4.)

`psi(al1,al3,...,alk)` (4)

Transfinite Veblen functions (See Section 3.3 and tables 5 and 6.)

`psi_{g}(al1,al2,...,alk)` (5)

Countable admissible ordinals (See Section 5.1 and tables 7, 8 and 9.)

`w_{k}` (6)

`w_{k}[e]` (7)

`w_{k,g}(al1,al2,...,alk)` (8)

Projection on countable admissible ordinals (See Section 5.2 and tables 10, 11 and 12.)

`[[d]]w_{k}` (9)

`[[d]]w_{k}[[e]]` (10)

`[[d]]w_{k}[e]` (11)

`[[d]]w_{k,g}(al1,al2,...,alk)` (12)

Nested ordinal projection (See Section 6, tables 13, 14, 15, 16, 17 and Figure 1.)

`[[d1/s1,d2/s2,...,dm/sm]]w_{k}` (13)

`[[d1/s1,d2/s2,...,dm/sm]]w_{k}[[e]]` (14)

`[[d1/s1,d2/s2,...,dm/sm]]w_{k}[e]` (15)

`[[d1/s1,d2/s2,...,dm/sm]]w_{k,g}(al1,al2,...,alk)` (16)

The "`/sk`" `k=1,2,...,m` are optional.

Figure 3: Ordinal calculator notation plain text format

7 Mathematical truth

A recursive formal mathematical system, such as ZF (Zermelo Frankel set theory), is a recursive process for enumerating theorems. Thus it is subject to analysis with the tools of computer science. However the mathematical content of ZF includes the uncountable and far beyond it. This would seem to be outside of the reach of computer science. The Löwenheim-Skolem theorem proved that a first order formal system, such as ZF, that has a model must have a countable model. This raises questions about the nature of the uncountable in mathematics. The views of three logicians show the range of interpretations offered in response to these issues.

Paul J. Cohen, who proved that the negation of CH (Continuum Hypothesis[19]) is consistent with ZF if ZF is consistent, draws a boundary between sets definable by properties of the integers and other infinite sets[4, p. 10].

> There certainly are some cases in which the use of infinite sets presents no essential difficulties. For example, to say that a property holds for all integers or that it holds for all members of the set of integers, is clearly equivalent. Similarly, to say n belongs to the set of even integers is equivalent to saying that n is even. Thus the use of some sets can be avoided by referring back to a suitable property. If this could always be done we would have no problem.

Cohen goes on in this paper and another published decades later to declare that he is a formalist when it comes to the uncountable[5, p. 2416].

> Does set theory, once we get beyond the integers, refer to an existing reality, or must it be regarded, as formalists would regard it, as an interesting formal game? ... Through the years I have sided more firmly with the formalist position. This view is tempered with a sense of reverence for all mathematics which has used set theory as a basis, and in no way do I attack the work which has been done in set theory.

It is interesting to contrast this view with two alternatives. Solomon Feferman, the principle editor of Gödel's collected works, argues that the objectivity of mathematics stems from inter-subjective *human* conceptions[6].

> ... I think the Platonistic philosophy of mathematics that is currently claimed to justify set theory and mathematics more generally is thoroughly unsatisfactory and that some other philosophy grounded in inter-subjective *human* conceptions will have to be sought to explain the apparent objectivity of mathematics.

In a later paper he describes in detail why he believes the Continuum Hypothesis is too vague to be definitely true or false and elaborates on his proposal of Conceptual Structuralism as a basis for mathematical objectivity. He emphasizes that human conceptual creations, such as money, are objective. He thinks mathematics falls in this category[7].

Finally there is the proposal by Hamkins of a Platonic multiverse[10].

[19]The Continuum Hypothesis states that there is no set larger than the integers (that the integers cannot be mapped onto) and smaller then the reals (and that cannot be mapped onto the reals). Gödel proved that ZF+CH was consistent if ZF is consistent. Cohen proved that ZF+$\overline{\text{CH}}$ was consistent if ZF is consistent.

In this article, I shall argue for a contrary position the *multiverse view*, which holds that there are diverse distinct concepts of set, each instantiated in a corresponding set-theoretic universe, which exhibit diverse set-theoretic truths. Each such universe exists independently in the same Platonic sense that proponents of the universe view regard their universe to exist.

This illustrates the wide range of views among logicians. I think there is a grain of truth in all three of these views starting with the objectivity of properties of the integers (or equivalently TMs). Much of mathematics can be defined as properties of recursive processes including divergent properties that may involve quantification over the reals. Each of these properties is logically determined by a recursively enumerable sequence of events. These sequences can only be generated by computers that run forever error free with unlimited storage. That ideal does not exist, but, as technology progresses, the ideal computer can be approximated for ever more execution steps with ever greater probability of being error free.

For me the objective basis of mathematics lies in physical reality. However the objective relationships between recursively enumerable events are human conceptual creations. For example a TM halting can be a physical event, but the general concept of the halting problem for TMs is a human conceptual creation connected to physical reality, but not corresponding to any specific physical event. The halting problem is at the root of a a hierarchy of relationships determined by a recursively enumerable sequence of events that matches a mathematical hierarchy.

7.1 Properties of the integers

This hierarchy of mathematics starts with the arithmetical and hyperarithmetical hierarchies[20] which have an interpretation as generalizations of the computer halting problem. Every Π_2 statement is equivalent to the question does a particular TM have an infinite number of outputs. This is implied by the $\mathbf{U}$ quantifier. $\mathbf{U}_x r(x)$ is true iff $r(x)$, a recursive relation, is true on an infinite subset of the integers. $\forall_{n_1}\exists_{n_2} r_1(n_1, n_2)$ can be replaced with $\mathbf{U}_n r_2(n)$ where r_1 and r_2 are recursive relations[12]. It is straightforward to generate r_2 from r_1. Further every Π_4 statement is equivalent to the question does a particular TM have an infinite number of outputs an infinite subset of which are the Gödel numbers of TMs that have an infinite number of outputs. This can be generalized to define Π_{2n} statements for any integer n and iterated up to any recursive ordinal to define any hyperarithmetical statement.

Going further, Kleene's $\mathcal{O}$ is defined by properties of TMs, albeit properties that require quantification over the reals. The set of all members of $\mathcal{O}$ is a Π^1_1 complete[21]set[18]. Finally, the limit of the countable admissible ordinals is the first uncountable ordinal and these are defined as Kleene's $\mathcal{O}$ is defined for TMs with an oracle for smaller admissible ordinals[1].

[20]Statements in the arithmetical hierarchy are those with a finite number of quantifiers ($\forall$ and $\exists$) over the integers on a recursive relationship. The hyperarithmetic hierarchy, loosely speaking, iterates this definition up to any recursive ordinal.

[21]A set is Π^1_1 complete if a TM with an oracle for this set can decide any Π^1_1 statement.

7.2 The reals

Does a TM that accepts an arbitrarily long sequence of integer inputs halt for every possible sequence? This question is logically determined by a recursively enumerable sequence: which finite sequences of inputs cause the TM to halt. A nondeterministic TM[22] can enumerate all of these events. If the sequences that halt include an initial segment of every real number than the answer is yes and no otherwise. Some initial segments obviously cover every possible real number. For the binary reals less than 1, the decimal fractions that start with .0 and .1 cover every real number.

Some statements that require quantification over the reals are about properties of a TM along divergent paths such that every event that determines the outcome is logically determined and recursively enumerable. These statements are objectively true or false. This is not true for other statements reals such as the Continuum Hypothesis. Like the three logicians quoted in Section 7, I doubt that that the Continuum Hypothesis has a definite truth value. I draw the line with processes that are logically determined by a recursively enumerable sequence of events. That definition clearly includes many statements and excludes many others. However it is not precise enough to define exactly what statements are included. From Gödel we know that every mathematical system of sufficient strength will be incomplete with respect to provability. I think it will also be incomplete with respect to definability.

In light of the Lowenheim-Skolem theorem and at least relative to an always finite but potentially infinite universe Cantor's proof that the reals are not countable is an incompleteness theorem. It cannot tell us something about the relative size of completed infinite totalities if no such thing exists. This interpretation suggests that the cardinal numbers are not definite sets but reflect ways the countable sets can always be expanded in a "correct" formal system. For example in a Löwenheim-Skolem derived countable model for ZF all the sets that are seen as uncountable within ZF are countable in the model.

7.3 Expanding mathematics

Cohen concludes his second philosophical paper on a pessimistic note[5, p. 2418].

> I believe that the vast majority of statements about the integers are totally and permanently beyond proof in any reasonable system. ...
>
> In this pessimistic spirit, I may conclude by asking if we are witnessing the end of the era of pure proof, begun so gloriously by the Greeks. I hope that mathematics lives for a very long time, and that we do not reach that dead end for many generations to come.

There is a way to explore all objective mathematical truth in an always finite but unbounded universe. With no limits, civilization can try everything. That may seem absurd but it appears to be how the mathematically capable human mind evolved. It is doubtful

[22] A nondeterministic TM simulates all the TMs with Gödel numbers in a specified recursively enumerable sequence. It is straightforward to write a single TM program that does exactly what every one of these individual TMs do for every time step.

that this could have happened in a world that was significantly less diverse than ours. Of course biological evolution has only a small random component. Mostly it builds on itself. Over billions of years, random perturbations have created humanity with the ability to understand where we came from and perhaps to choose where we go.

One can extend mathematics in a similar selective and divergent way. The multiverse of Hamkins may not be a Platonic reality, but a partial map of the possible ways in which mathematics can be extended. It is essential that we build on what exists but extend this with ever expanding diversity. Otherwise we will run into a what I call a Gödel limit. This is a sequence of ever more powerful mathematical models all of which may eventually be discovered in a single path of mathematical exploration and development that extends forever. However all of the results are subsumed by a single more powerful result that will never be discovered inside the Gödel limit. The only way to avoid a Gödel limit is through an unbounded expansion of diversity.

Why would it be worth the effort to explore this mathematics? Existing mathematics goes far beyond what is commonly used in science and engineering[6]. The answer to this question takes us outside of mathematics to questions of ultimate meaning and value. Bertrand Russell in 1927 at the end of the *Analysis of Matter* observed that intrinsic nature and by implication intrinsic value only exists in conscious experience.

> As regards the world in general, both physical and mental, everything that we know of its intrinsic character is derived from the mental side, and almost everything that we know of its causal laws is derived from the physical side. But from the standpoint of philosophy the distinction between physical and mental is superficial and unreal[19, p. 402].

Science first abandoned the fundamental substances of earth, air, fire and water and later Newtonian billiard balls for pure mathematical models lacking any fundamental substance. This is made explicit in set theory where the fundamental entity is the empty set or nothing at all. Intrinsic nature and thus meaning and value exists only in conscious experience.

Nonetheless the evolution of consciousness has been an evolution of structure. Reproducing molecules have evolved to create the depth and richness of human consciousness. They have also evolved to the point where we can take conscious control over future human evolution. Human genetic engineering has already begun as a way to cure or prevent horrible diseases. Over time the techniques will be perfected to the point where one may consider using them for human enhancement. We will need to have a sense of meaning and values that is up to the challenge this capability presents.

The depth and richness of human consciousness seems to require the level of abstraction and self reflection that has evolved. These seem necessary for both richness of human consciousness and the ability to create mathematics. The ordinal numbers are the backbone of mathematics determining what problems are decidable and what objects are definable in a mathematical system. Do they also impose limits on the depth and richness of human consciousness? If so than diversity is critical to the unbounded exploration of possible conscious experience. This possibility is explored in a video *Mathematical Infinity and Human Destiny* and a book[2].

References

[1] Jon Barwise. *Admissible Sets and Structures: An Approach to Definability Theory.* Springer-Verlag, Berlin, 1975. 30

[2] Paul Budnik. *What is and what will be: Integrating spirituality and science.* Mountain Math Software, Los Gatos, CA, 2006. 32

[3] Paul Budnik. A Computational Approach to the Ordinal Numbers: Documents ordCalc 0.3.2, `www.mtnmath.com/ord/ordinal.pdf`. August 2012. 6, 12

[4] Paul J. Cohen. Comments on the foundations of set theory. In Dana S. Scott, editor, *Axiomatic set theory*, pages 9–15. American Mathematical Society, 1971. 29

[5] Paul J. Cohen. Skolem and pessimism about proof in mathematics. *Philisophical Transactions of the Royal Society A*, 363(1835):2407–2418, 2005. 29, 31

[6] Solomon Feferman. Does mathematics need new axioms? *American Mathematical Monthly*, 106:99–111, 1999. 29, 32

[7] Solomon Feferman. Is the Continuum Hypothesis a definite mathematical problem? Stanford web page: `math.stanford.edu/~feferman/papers/IsCHdefinite.pdf`, 2011 DRAFT. 29

[8] Harvey M. Friedman. Finite functions and the necessary use of large cardinals. *ANN.OF MATH.*, 2:148, 1998. 3

[9] Jean H. Gallier. What's so Special About Kruskal's Theorem And The Ordinal Γ_0? A Survey Of Some Results In Proof Theory. *Annals of Pure and Applied Logic*, 53(2):199–260, 1991. 6

[10] Joel David Hamkins. The set-theoretic multiverse. *Review of Symbolic Logic*, to appear, 2012. 29

[11] S. C. Kleene. On notation for ordinal numbers. *Journal of Symbolic Logic*, 3(4), 1938. 3

[12] A. H. Lachlan. The U-quantifier. *Mathematical Logic Quarterly*, 7(11-14):171–174, 1961. 30

[13] Benoît Mandelbrot. Fractal aspects of the iteration of $z \mapsto \lambda z(1-z)$ for complex λ and z. *Annals of the New York Academy of Sciences*, 357:49–259, 1980. 14

[14] Larry W. Miller. Normal functions and constructive ordinal notations. *The Journal of Symbolic Logic*, 41(2):439–459, 1976. 6, 14

[15] Rohit Parikh. On nonuniqueness of transfinite progressions. *The Journal of the Indian Mathematical Society*, XXXI(1):23–32, Jan.-Mar. 1967. 11

[16] Michael Rathjen. The Art of Ordinal Analysis. In *Proceedings of the International Congress of Mathematicians, Volume II*, pages 45–69. European Mathematical Society, 2006. 6, 14

[17] Michael Rathjen. How to develop Proof-Theoretic Ordinal Functions on the basis of admissible ordinals. *Mathematical Logic Quarterly*, 39:47–54, 2006. 14

[18] Hartley Rogers Jr. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967. 3, 30

[19] Bertrand Russell. *The Analysis of Matter.* Paul Kegan, London, 1927. 32

[20] Gerald E. Sacks. Metarecursively Enumerable Sets and Admissible Ordinals. *Bulletin of the American Mathematical Society*, 72(1):59–64, 1966. 12

[21] Gerald E. Sacks. *Higher recursion theory.* Springer Verlag, New York, 1990. 12

[22] Oswald Veblen. Continuous increasing functions of finite and transfinite ordinals. *Transactions of the American Mathematical Society*, 9(3):280–292, 1908. 6, 11

[23] A. N. Whitehead and Bertrand Russell. *Principia Mathematica*, volume 1-3. Cambridge University Press, 1918. 11