ARTICLES

Constructive Ordinal Notation Systems

G: Adam, did you find a good system for naming ordinals?
A: Ordinals? I thought you said "animals."

FREDERICK GASS
Miami University
Oxford, OH 45056

Cantor's ideas for the development of transfinite numbers can be traced back through a sequence of memoirs to his early research on trigonometric series. In this early research, Cantor included a study of the irrational numbers, using rational Cauchy sequences to define them, and it is in this setting that he seems to have drawn the inspiration for sequences of counting numbers that go beyond the integers.

Recall that if S is an increasing Cauchy sequence of rationals, then S belongs to some equivalence class of Cauchy sequences that by definition constitutes a certain number x, either rational or irrational. Viewing the situation geometrically, we identify x with the point on the real line that is the limit point of the sequence S, and we use this image to guide most of our thinking about x. In fact, prior to the time of Weierstrass it had been presumed that this geometrical point of view was a sufficiently rigorous approach to the theory of irrational numbers.

By analogy with this view of the irrationals, Cantor proposed in 1882 (Introduction to [**3**], p. 54) that the sequence, $0, 1, 2 \ldots$ be considered to have a least upper bound or limit, now denoted by ω, and that ω be followed in order by numbers $\omega + 1$, $\omega + 2$, $\omega + 3$, and so on. The process of passing from a number x to its immediate successor $x + 1$ was Cantor's "first principle of generation", and the process of passing from a denumerable increasing sequence of numbers to its limit was his "second principle of generation." If you begin with 0 and repeatedly apply the first principle, you obtain the natural numbers (the nonnegative integers), which Cantor called the "first number class." By applying the second principle once, you obtain ω; then repeated application of the first principle yields $\omega + 1, \omega + 2, \omega + 3, \ldots,$ and then another application of the second principle yields $\omega + \omega$, also called $\omega \cdot 2$. In this way the generating process continues on and on. Incidentally, if you are curious about the use of $\omega \cdot 2$ rather than $2 \cdot \omega$, then you might enjoy reading about the noncommutative arithmetic of these transfinite numbers. (Section 21 of [**9**] is a good reference for this topic. You will learn that $2 \cdot \omega$ is equal to ω, as is $1 + \omega$.)

The set of all numbers generated from 0 through use of both principles is Cantor's "second (cumulative) number class," and it evidently cannot be denumerable, for otherwise it could be extended by application of the second principle. In order to obtain more numbers, Cantor introduced a third principle that would lead from the second number class to its limit, Ω. From there, one obtains $\Omega + 1, \Omega + 2, \ldots, \Omega + \omega, \ldots, \Omega + \Omega, \ldots$ through use of all three principles. And of course one need never stop for want of a new principle.

The objects introduced in this way are called **ordinal numbers** because they serve to describe the sequential order of elements in any well-ordering. Each nonzero ordinal is classified as a **successor** if it is generated by Cantor's first principle, and otherwise it is a **limit** ordinal. For ordinals α and β, $\alpha < \beta$ if and only if α precedes β in the sequence of ordinals. This relation has the well-ordering property in that any non-empty set of ordinals contains a least element.

In order to place his ordinal number theory on a mathematically sound basis, Cantor used an equivalence class approach similar to the familiar one for the reals. Thus in [**2**] he formally introduced ordinals as isomorphism classes of well-ordered sets. For a good description of this approach, see [**10**]. A quite different approach due to von Neumann, described in [7] and [**9**], is preferred by most modern set-theorists. An interesting brief account of Cantor's life and work is contained in [**4**].

The ordinal numbers play a major role in modern set theory, and they are used occasionally in various branches of mathematics as a vehicle for extended inductive proofs. Applications of ordinals occur in topology, the area in which undergraduates are most likely to meet them. For example, Cantor's second number class with its order topology is a countably compact Hausdorff space that is not compact. The second number class is especially useful in constructing examples because every countable subset of the class has its least upper bound in the class ([**15**], pp. 10, 11, and 117). Another topological use of the ordinals is the formation of a hierarchy for Borel sets: at the 0th level of the hierarchy are the basic open sets. Given the αth level, we define the $\alpha+1$st level to contain all sets that are either a countable union of α-level sets or a countable intersection of α-level sets or the difference of two α-level sets. Finally, if λ is a limit ordinal, then the λth level contains all sets that belong to any previous level, so λ is a sort of gathering-together level. There are other ways of organizing the Borel sets into a hierarchy, but this way is perhaps the simplest to describe. As an exercise you can show that only ordinals of the second number class are needed here, because no new sets are obtained beyond the Ωth level ([7], pp. 123, 124).

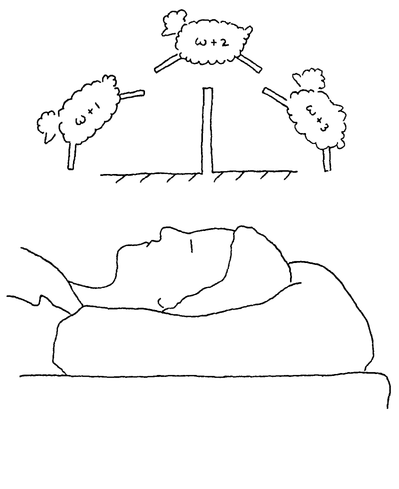
Ordinals are used in a similar fashion to index more extensive hierarchies of sets and functions in modern set theory. In the hierarchy of constructible sets ([7], Chapter V), each ordinal belongs to and represents a certain stage at which sets of a particular complexity are formed. Consequently the ordinals are often viewed as the backbone of the hierarchy, with each ordinal a vertebra.

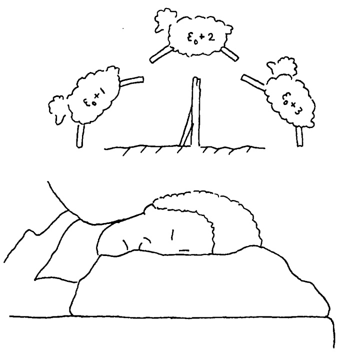
Polynomials in ω

Let us imagine the ordinals being generated by Cantor's first two principles and list several of them at strategically-chosen points so as to keep track of the growth process:

$$0,1,2,\ldots,\omega,\omega+1,\omega+2,\ldots,\omega\cdot 2,\omega\cdot 2+1,\omega\cdot 2+2,\ldots,\omega\cdot 3,$$
$$\omega\cdot 3+1,\omega\cdot 3+2,\ldots,\omega\cdot n,\omega\cdot n+1,\omega\cdot n+2,\ldots,\omega^2,\ldots,$$
$$\omega^3,\ldots,\omega^n,\ldots.$$

In the sequence indicated here, ordinals written in the form $\omega\cdot n$ are followed by a sequence $\omega\cdot n+1$, $\omega\cdot n+2$, $\omega\cdot n+3$, and so on, with $\omega\cdot n+m$ understood to mean $(\omega\cdot n)+m$. The limit of this sequence is $\omega\cdot(n+1)$, from whence another sequence leads to $\omega\cdot(n+2)$. The ordinals $\omega,\omega\cdot 2,\omega\cdot 3,\ldots$ form a denumerable increasing sequence whose limit we call $\omega\cdot\omega$ or ω^2.





Repeating the process, beginning with ω^2 rather than with 0, leads up to $\omega^2 \cdot 2$, and similarly we obtain $\omega^2 \cdot 3, \omega^2 \cdot 4, \ldots$, a sequence whose limit is called $\omega^2 \cdot \omega$ or ω^3. If you experiment by writing out names for some of the other ordinals, the following ideas will probably occur to you.

First, the notation assigned to the typical ordinal is a kind of polynomial in ω with positive integer coefficients, such as $\omega^2 \cdot 5 + \omega + 2$. (The ordinal mentioned here is quite far down our list. To reach it, go out to the limit ordinal $\omega^2 \cdot 5$ and start counting off numbers from there as if from 0: $1, 2, 3, \ldots, \omega, \omega + 1, \omega + 2$. The last one counted will be $\omega^2 \cdot 5 + \omega + 2$.) Second, this process of generating ordinals can go on indefinitely, but the system of polynomials in ω eventually gives out. For example, the increasing sequence $\omega, \omega^2, \omega^3, \ldots$ has a limit ordinarily called ω^ω, and one can go on to larger ordinals with progressively more complicated exponential notations, such as

$$\omega^{\omega^\omega}, \quad \omega^{\omega^{\omega^\omega}}, \ldots. \qquad (*)$$

What would you call the limit of the sequence indicated in $(*)$? The usual name is ε_0.

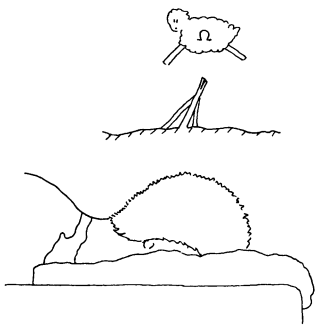
We call any expression of the form $\omega^{m_1} \cdot n_1 + \cdots + \omega^{m_k} \cdot n_k + n_{k+1}$ a (finite) **polynomial in** ω, provided that the coefficients n_i and exponents m_i are natural numbers, and the exponents are arranged in decreasing order. Given a polynomial in ω that denotes an ordinal β, you can quickly determine certain information about β that I shall call "the essential information":

1. You can tell from the polynomial whether β is 0, a successor, or a limit ordinal.
2. If β is a successor, then you can give a polynomial for its immediate predecessor, $\beta - 1$.
3. If β is a limit ordinal, then you can tell how to write polynomials that name a denumerable increasing sequence of ordinals with limit β (called a **fundamental sequence for** β).

To illustrate item 3, suppose that β is $\omega^3 \cdot 5 + \omega^2 \cdot 4$. Then $\omega^3 \cdot 5 + \omega^2 \cdot 3 + \omega \cdot n$, for $n = 1, 2, 3, \ldots$, gives a fundamental sequence for β. Although we can determine other information from the polynomial for β (such as the polynomial notation for β's immediate successor $\beta + 1$), we confine our attention to facts that show the genealogy of β with respect to Cantor's principles.

Since there are only countably many polynomials in ω, this notation system provides names for only a countable subset of Cantor's second number class. We wish to extend the system to a more comprehensive one that still conveys the essential information about each ordinal. The least ordinal for which there is no polynomial in ω is the ordinal ω^ω. One obvious option for extending the polynomial system is to continue with notations like ω^ω that allow the use of exponents greater than or equal to ω. This option works fine up to the number ε_0, as I suggested earlier. In order to extend the notation system beyond that point, one can simply admit ε_0 as a new symbol and allow it to be used in polynomial expressions such as $\varepsilon_0, \varepsilon_0 + 1, \varepsilon_0 + 2, \ldots, \varepsilon_0 + \omega, \ldots, \varepsilon_0 \cdot 2$, and $\varepsilon_0^2 + \omega^\omega \cdot 5 + \omega^3 \cdot 4 + \omega \cdot 14 + 85$.

With more explicit information about the polynomial expressions in this extended system, you could check for yourself that all the essential information is conveyed. But you can readily see that even this system is subject to extension. The major question formalized and answered in the remainder of this paper concerns the *existence of a maximal ordinal notation system*—one that



conveys the essential information and provides notations for the largest possible set of ordinals.

Maximal systems do exist, although at this point your intuition might strongly suggest otherwise. Indeed if λ is the least ordinal not named by some particular maximal system, could not the system be extended by adjoining a special symbol to denote λ? After we achieve the main results of this paper, we shall resolve this apparent problem.

Algorithms

Reviewing the essential information that is to be contained in a notation system, we notice the phrases "you can tell" and "you can give." The evident intent of these phrases is not that one can achieve something by means of luck or ingenuity, but rather that anyone who can follow directions can do it. In other words, there are routine procedures—algorithms, to use the standard modern term—for accomplishing the stated tasks.

To be a bit more specific about the nature of these algorithms, I ask you to pick some general-purpose programming language like BASIC or Pascal—one that you will be able to use or just imagine using for the rest of this article—and assume that all the algorithms are expressed in that language. It may seem rather vague to leave this choice open, but the specifics of the language that you pick will not really matter. Furthermore, it has been established that all standard programming languages have the same theoretical capability when it comes to expressing mathematical procedures ([**13**], p. 114, presents a technical version of this fact. See also Chapters 6–8 of [**1**]).

There is an important question that should be considered, though: whether every algorithm can indeed be expressed in terms of a computer program. (We are speaking of algorithms for manipulating symbols, of course; not algorithms for physical tasks such as tying shoelaces.) An early version of this question was central to the pioneering work of logicians in the 1930's who invented systems of computation equivalent to modern programming languages. Turing machines and recursive functions are perhaps the most famous of those systems. Since that time, every proposed procedure that is evidently algorithmic has been shown to be Turing programmable, so that now virtually all logicians and computer scientists accept the thesis that algorithmic procedures are precisely the programmable ones. The original version of this thesis is credited to Church or to Church and Turing jointly. See Chapter 1 of [**14**] for more discussion of this topic. The branch of mathematics that has grown out of these considerations is called "recursive function theory" or "the theory of effective computability," which I abbreviate as "computability theory."

If F is a computer program and x is an input, then $F(x)$ denotes the output, if any, when F is run with input x. If there is no output, then $F(x)$ is undefined. If F and G are programs, then $F(x) = G(x)$ means that either $F(x)$ and $G(x)$ are both defined and equal, or they are both undefined.

It is traditional and still fairly common practice in computability theory to restrict oneself to natural numbers as the inputs and outputs of algorithms. Natural numbers are also used to identify whole algorithms, in much the way that serial numbers identify appliances; sets of natural numbers are used to provide ordinal notations. Besides being traditional, this natural number approach ties many ideas together neatly, and so it is the approach we shall follow.

In the first place, then, we shall assume that all programs mentioned in this article are intended for use with natural number inputs and outputs. Any nonnumerical outputs will be considered the same as no output at all. Also, "natural number" will be abbreviated as simply "number."

The decision to use only numbers as ordinal notations may seem too drastic or oversimplifying until you see how systems like our polynomials in ω can be transformed into strictly numerical systems without any loss of information. One way to transform the polynomials is to consider the symbols $0,1,2,\ldots,8,9,+,\cdot,$ and ω to be the digits of a base-13 numeration systems. Then, for instance, the polynomial $\omega^2\cdot 5+\omega\cdot 15+2$ can be rewritten slightly as $\omega 2\cdot 5+\omega\cdot 15+2$ and identified with the number that it represents as a base-13 numeral. In this example, the number would be

$$12\times 13^{10}+2\times 13^{9}+11\times 13^{8}+5\times 13^{7}+10\times 13^{6}$$
$$+12\times 13^{5}+11\times 13^{4}+13^{3}+5\times 13^{2}+10\times 13+2.$$

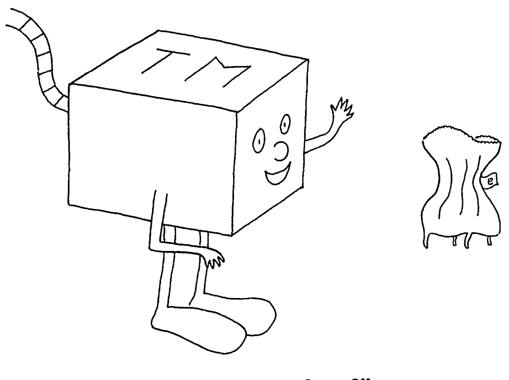
In this way each polynomial is associated with a unique number, and from that number the original polynomial can be recovered via straightforward arithmetic.

In a similar fashion one can associate a unique number with each line of a computer program written in your language. (Let $0,1,2,\ldots,8,9,+,\ldots$ be the symbols of your language and consider them to be the digits of a numeration system. Then each line can be interpreted as a numeral in the system.) If $L_1, L_2,\ldots,L_k$ are the lines of a program and $n_1, n_2,\ldots,n_k$ are the corresponding numbers, then we combine the n_i's into a single number e that represents the entire program. One way to combine them is to let

$$e=2^{n_1}\cdot 3^{n_2}\cdot 5^{n_3}\cdots p^{n_k},$$

where p is the kth prime number. Given the number e, we can reverse the process by factoring e into distinct prime powers, observing the exponents, then decoding each exponent into the program line it represents.

The idea of associating numbers with programs or formulas or any set of formal expressions is credited to Gödel, who used this technique in his famous paper [8] on undecidability. **Gödel number** is now a standard term for the numbers that result. In this article, if e is the Gödel number of a computer program, then I shall write $\{e\}$ to denote that program. If the number e is not the Gödel number of a program, then $\{e\}$ will denote some particular program (for you to choose) that never gives an output, regardless of the input. So now every number e represents some program $\{e\}$, and every program is represented by at least one number. I say "at least one" because the particular program that I invited you to choose will be $\{e\}$ for many numbers e. Furthermore, if you consider a program to be unchanged by certain minor rearrangements, then many programs will be represented by more than one e.



"Gödel number, please?"

Constructive Ordinal Notation Systems

S. Kleene gave the following definition that unites most of the ideas discussed so far, using the term "r-system" ([**11**]). In the definition, L is the set of ordinal notations and f matches the notations with the ordinals they name, while K, P and Q provide the essential information.

DEFINITION. A **constructive ordinal notation system** (CONS) is a pair (L, f) in which L is a set of natural numbers, f is a function from L into the ordinal numbers, and there are programs K, P and Q having the properties listed below.

1. If $f(x) = 0$, then $K(x) = 1$,
   if $f(x)$ is a successor ordinal, then $K(x) = 2$, and
   if $f(x)$ is a limit ordinal, then $K(x) = 3$.
2. If $f(x)$ is a successor ordinal $\beta + 1$, then $P(x)$ is a notation for the immediate predecessor β.
3. If $f(x)$ is a limit ordinal λ, then $Q(x)$ is the Gödel number of a program, and the outputs $\{Q(x)\}(0), \{Q(x)\}(1), \{Q(x)\}(2), \ldots$ are notations for a fundamental sequence for λ.

EXAMPLE. Let L be the set whose elements are all numbers of the form 2^n or $2^n \cdot 3$ for $n \geqslant 0$. We shall take $1, 2, 2^2, 2^3, \ldots$ as notations for the finite ordinals $0, 1, 2, 3, \ldots$, respectively, and $3, 2 \cdot 3, 2^2 \cdot 3, 2^3 \cdot 3, \ldots$ as notations for $\omega, \omega + 1, \omega + 2, \omega + 3, \ldots$, respectively. Consequently the ordinal-assigning function f is given by

$$f(x) = \begin{cases} n & \text{if } x = 2^n \\ \omega + n & \text{if } x = 2^n \cdot 3. \end{cases}$$

The programs K and P should be constructed so as to have the following output features:

$$K(x) = \begin{cases} 1 & \text{if } x = 1 \\ 3 & \text{if } x = 3 \\ 2 & \text{if } x = 2^n \text{ or } 2^n \cdot 3 \text{ for } n \geqslant 1, \end{cases} \qquad P(x) = \begin{cases} 2^n & \text{if } x = 2^{n+1} \\ 2^n \cdot 3 & \text{if } x = 2^{n+1} \cdot 3. \end{cases}$$

We have considerable leeway in constructing K and P, because their behavior on inputs not belonging to L is irrelevant. For instance, we could construct P so that $P(x) = [x/2]$ for all x, using the greatest-integer function to insure integer outputs.

Since 3 is the only notation for a limit ordinal ($f(3) = \omega$), $Q(3)$ is the only output that must be carefully planned when you construct program Q for this CONS. Find a program that prints out the value 2^n for each input number n, and suppose that e is the Gödel number of the program. Then $\{e\}$ generates notations for a fundamental sequence for ω, and therefore Q could be any program constructed so that $Q(3) = e$.

The next example is patterned after the system S_1 defined by Kleene in [**11**]. It will turn out to be a maximal CONS.

EXAMPLE. $\mathcal{S} = (\mathcal{L}, \mathcal{F})$. I shall describe $\mathcal{L}$ and $\mathcal{F}$ by indicating for each ordinal number β which numbers belong to $\mathcal{L}$ as notations for β, that is, by describing the set $\mathcal{F}^{-1}(\beta)$. $\mathcal{L}$ is then the union of all the nonempty sets $\mathcal{F}^{-1}(\beta)$.

(i) 0 is the unique notation for 0.
(ii) 2^x is a notation for $\alpha + 1$ if and only if x is a notation for α.
(iii) 3^e is a notation for the limit ordinal λ if and only if e is a Gödel number such that $\{e\}(0), \{e\}(1), \{e\}(2), \ldots$ are notations for a fundamental sequence for λ.

In $\mathcal{S}$, then, the finite ordinals $0, 1, 2, 3, 4, 5, 6, \ldots$ receive as their unique notations the numbers

$$0, 1, 2, 4, 16, 2^{16}, 2^{(2^{16})}, \ldots. \tag{1}$$

Beginning with ω, however, each ordinal has many notations. The notations for ω are numbers of the form 3^e, where $\{e\}$ is a program whose successive outputs $\{e\}(0), \{e\}(1), \{e\}(2), \ldots$ are an increasing subsequence of (1). For each of these notations 3^e, the number $2^{(3^e)}$ is a notation for $\omega + 1$, and so on for $\omega + 2, \omega + 3, \ldots$. Likewise, $\omega \cdot 2$ has many notations, all of the form 3^e, and from them we obtain notations for succeeding ordinals.

The following theorem is a companion to the definition of $\mathscr{S}$, for it insures that $\mathscr{S}$ and $\mathscr{F}$ are well defined. The proofs are examples of ordinal induction. They show that the properties ascribed to ordinal β in the theorem are indeed true for all ordinals. The approach is: show that the property is true for 0; show that the property is true for $\alpha + 1$, if it is true for α; show that the property is true for a limit ordinal λ, if it is true for all ordinals less than λ.

THEOREM. *Let β be any ordinal. In the definition of $\mathscr{S}$,*
(a) *the set $\mathscr{F}^{-1}(\beta)$ is defined, and*
(b) *$\mathscr{F}^{-1}(\beta)$ is disjoint from $\mathscr{F}^{-1}(\delta)$ for all $\delta \neq \beta$.*

Proof. (a) By part (i) of the definition of $\mathscr{S}$, $\mathscr{F}^{-1}(0) = \{0\}$. Next, assume $\mathscr{F}^{-1}(\alpha)$ is defined. Then $\mathscr{F}^{-1}(\alpha + 1) = \{2^x \colon x \in \mathscr{F}^{-1}(\alpha)\}$. Finally, assume that $\mathscr{F}^{-1}(\alpha)$ is defined for each ordinal α less than the limit ordinal λ. Then $\mathscr{F}^{-1}(\lambda) = \{3^e \colon \{e\}(0), \{e\}(1), \{e\}(2), \ldots$ belong respectively to sets $\mathscr{F}^{-1}(\alpha_0), \mathscr{F}^{-1}(\alpha_1), \mathscr{F}^{-1}(\alpha_2), \ldots$, where $\alpha_0, \alpha_1, \alpha_2, \ldots$ is a fundamental sequence for $\lambda\}$.

(b) Clearly $\mathscr{F}^{-1}(0)$ is disjoint from $\mathscr{F}^{-1}(\delta)$ for all $\delta \neq 0$. Next, assume that $\mathscr{F}^{-1}(\alpha)$ is disjoint from $\mathscr{F}^{-1}(\delta)$ for all $\delta \neq \alpha$; we show that $\mathscr{F}^{-1}(\alpha + 1)$ must also have the disjointness property. Let $\mathscr{F}^{-1}(\alpha + 1)$ and $\mathscr{F}^{-1}(\delta)$ have some notation in common. That notation must be of the form 2^x, where $x \in \mathscr{F}^{-1}(\alpha)$. Furthermore, δ must be a successor ordinal, and $x \in \mathscr{F}^{-1}(\delta - 1)$. Consequently $\delta - 1$ must equal α by our assumption about $\mathscr{F}^{-1}(\alpha)$, and so $\delta = \alpha + 1$.

Finally, assume that each ordinal less than the limit ordinal λ has the disjointness property described in the theorem. To see that λ must also have the property, let $\mathscr{F}^{-1}(\lambda)$ and $\mathscr{F}^{-1}(\delta)$ have some notation in common. That notation must be of the form 3^e, where $\{e\}(0), \{e\}(1), \{e\}(2), \ldots$ are notations for a fundamental sequence for λ and also for δ. By assumption, each of the notations $\{e\}(n)$ denotes a unique ordinal. Furthermore, it is a set-theoretical fact that the limit of a fundamental sequence is unique. Therefore it must be the case that $\delta = \lambda$.

The last thing we do here, and the simplest, is to verify that $\mathscr{S}$ is in fact a CONS. It's really a do-it-yourself verification. Using your chosen programming language, you can construct programs K, P and Q such that

$$K(x) = \begin{cases} 1 & \text{if } x \text{ is } 0 \\ 2 & \text{if } x \text{ is positive and even} \\ 3 & \text{otherwise,} \end{cases}$$

$$P(x) = y \quad \text{if } x = 2^y,$$
$$Q(x) = e \quad \text{if } x = 3^e.$$

Programs with those characteristics can serve as the auxilary programs for the CONS $\mathscr{S}$.

Three theorems from computability theory will be crucial to the proof of $\mathscr{S}$'s maximality. The programs mentioned in the first two theorems are of fundamental importance, as is evidenced by the fact that [**13**] calls a programming system "acceptable" if and only if it contains them. The third theorem, the Recursion Theorem, is satisfied by an acceptable programming system, and it serves to justify program descriptions in which certain outputs are affected by others (the recursion aspect). To quote [**14**], p. 179, "It is a deep result in the sense that it provides a method for handling, with elegance and intellectual economy, constructions that would otherwise require extensive, complex treatment."

Given any two numbers e and n, and possibly considerable patience, one can determine the program $\{e\}$, apply it with input n, and give the output $\{e\}(n)$ when and if the computation ever halts. As a matter of fact, this whole process can be carried out by a sort of universal program U whose existence is stated below in the Enumeration Theorem. The title of the theorem is prompted by the way U "enumerates" all possible programs $\{e\}$ as e ranges through the natural numbers. The program U acts something like a modern day compiler.

ENUMERATION THEOREM. *There is a program U such that $U(e, n) = \{e\}(n)$ for all numbers e and n.* ([**13**], p. 82; [**14**], p. 22, ϕ_e means $\{e\}$.)

Another task that can be handled by a special program is that of forming compositions of given programs.

COMPOSITION THEOREM. *There is a program C such that for all numbers z and e, $C(z, e)$ is a Gödel number of a program, and $\{C(z, e)\}(n) = \{z\}(\{e\}(n))$ for all numbers n.* ([**13**], p. 83; [**14**], p. 24.)

The fact that the next theorem has many useful applications is indicated by the wide variety of forms in which it has been expressed. The one given here is best suited to the application I shall make of it with respect to $\mathscr{S}$.

RECURSION THEOREM. *If F is any program using two inputs, then there is a number z_0 such that $F(z_0, x) = \{z_0\}(x)$ for all numbers x.* ([**5**], p. 176, Theorem 7.4)

This result can be viewed as a fixed-point theorem, if we let F_z denote the one-input program obtained by fixing z as the first input of F. According to the Recursion Theorem, the mapping $\{z\} \to F_z$ has a fixed point.

The Main Theorem

As an exercise you might prove by ordinal induction that the ordinals named by a CONS form an initial segment of the ordinal numbers. In other words, if β receives a notation from (L, f), then so do all ordinals less than β. The next result shows that $\mathscr{S}$, defined in our previous section, is maximal in that it covers all the ordinals named by any other CONS. It even claims the existence of a program T that transforms the notations of a given CONS into corresponding ones of $\mathscr{S}$.

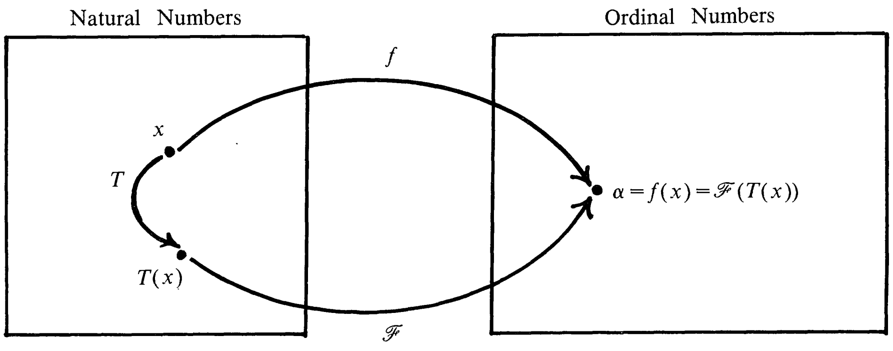
THEOREM. *If (L, f) is a given* CONS, *then there is a program T such that $T(L) \subseteq \mathscr{S}$ and $\mathscr{F}(T(x)) = f(x)$ for each x in L.*

Proof. Let K', P' and Q' be the auxiliary programs of the given CONS. With the definitions of CONS and $\mathscr{S}$ in mind, we will seek a program T with the following features:

$$T(x) = \begin{cases} 0 & \text{if } x \text{ denotes } 0 \\ 2^{T(P'(x))} & \text{if } x \text{ denotes a successor} \\ 3^{e} & \text{if } x \text{ denotes a limit ordinal,} \end{cases} \tag{2}$$

where e is a Gödel number, and $\{e\}(n) = T(\{Q'(x)\}(n))$ for all numbers n. The proof that such a program T (if it exists) would satisfy this theorem is by induction on the ordinal β denoted by x, and is left as an exercise. Whether or not you perform that exercise, you should inspect the description of T closely enough to see how it provides $\mathscr{S}$-notations for the ordinals named by (L, f). Also notice that recursion is involved in the second and third lines of (2), where $T(x)$ is determined by T values at notations for previous ordinals.

The following construction of T shows a typical application of the Recursion Theorem. In definition (3) below, think of the z's as candidates for being a Gödel number of T. When we get a fixed point z_0 for (3), $\{z_0\}$ will have exactly the properties in (2).



The relationships among f, $\mathscr{F}$ and T

Suppose that F is a program with the following features, where U and C are the programs mentioned in the Enumeration Theorem and the Composition Theorem.

$$F(z,x) = \begin{cases} 0 & \text{if } K'(x) = 1 \\ 2^{U(z,P'(x))} & \text{if } K'(x) = 2 \\ 3^{C(z,Q'(x))} & \text{if } K'(x) = 3. \end{cases} \tag{3}$$

A brief inspection of (3) suggests how programs K', P', Q', U and C are used as subroutines in constructing F.

Now let z_0 be the special number that is predicted in the Recursion Theorem, and let's rewrite the information of (3) in other terms.

$$F(z_0,x) = \{z_0\}(x) = \begin{cases} 0 & \text{if } x \text{ denotes } 0 \\ 2^{\{z_0\}(P'(x))} & \text{if } x \text{ denotes a successor} \\ 3^e & \text{if } x \text{ denotes a limit ordinal,} \end{cases}$$

where $e = C(z_0, Q'(x))$ is a Gödel number of $\{z_0\}$ composed with $\{Q'(x)\}$. This program $\{z_0\}$ has all the features described in (2), and so it is the T we are after.

Let λ be the least ordinal not provided with a notation by $\mathscr{S}$. You can check that λ would have to be a limit ordinal. Now, it would seem easy enough to extend $\mathscr{S}$ by simply adjoining a notation for λ, but since $\mathscr{S}$ is maximal, there is presumably no way to make a CONS that extends it. Suppose we attempt an extension by selecting the number 5 to represent λ. Then $\mathscr{F}(5) = \lambda$, $K(5) = 3$, $P(5)$ is irrelevant, and $Q(5) = e$ where $\{e\}$ is a program whose outputs name a fundamental sequence for λ. But in that case 3^e would already belong to $\mathscr{S}$ as a notation for λ, which contradicts our definition of λ. There are other strategies that one might use in attempting an extension (e.g., allow new notations for ordinals less than λ so that $\{e\}$ can approach λ by a new route), but the maximality of $\mathscr{S}$ ensures that all of them will fail. $\mathscr{S}$ reaches as far into the transfinite as is possible for a CONS.

Incidentally, the ordinals for which $\mathscr{S}$ provides notations are called the **constructive ordinals.** Since $\mathscr{S}$ is a CONS, they form a countable initial segment of Cantor's second number class.

Questions of Recursiveness

A final question I want to raise about CONS' in general and $\mathscr{S}$ in particular is a global one about the nature of the whole set of notations. One requirement that we might have considered adding to the essential information is "You can tell whether a given expression is an ordinal notation." The question here is one of recognizing a notation when you see one. For example, a polynomial in ω is certainly recognizable as such. On the other hand, it isn't clear how easily one could determine whether or not a given number 3^e is an ordinal notation in $\mathscr{S}$.

Similar questions of recognition occur with regard to Gödel numbers for programs. For example, "Can one recognize whether a given number e is the Gödel number of a program?" and "Can one recognize whether e is the Gödel number of a program that gives an output for every input n?"

To put it another way, we are concerned here with the complexity of a set of numbers, whether it be a set of ordinal notations or a set of Gödel numbers. One way to make the question more rigorous is to ask whether the given set is **recursive**, which is to say whether there exists an algorithm for distinguishing the members of that set from all other numbers. It turns out that for $\mathscr{S}$ or any other maximal CONS, the set of ordinal notations is not recursive, and so in this sense one must sacrifice recognition in order to have maximality. (This result is a corollary of facts derived in [**12**] about the complexity of another maximal CONS called $\mathscr{O}$.)

Incidentally, the answer to the first Gödel number question I mentioned above depends partly upon the language (or dialect thereof) that you assume, and partly upon what you require of a bona fide program. In the case of extremely lean and simple programming systems such as Turing machines, the answer is "yes" ([**5**], p. 60, item (11)). The second Gödel number question above is similar to the famous Halting Problem of computability theory and has a negative answer (Section 1.9 of [**14**], especially Theorem VIII). See [**14**], Section 2.2, for notes about similar questions in other branches of mathematics, including **Hilbert's tenth problem**. A good up-to-date exposition of that famous problem is contained in [**6**].

Recursive Ordinals

The two basic ingredients in our approach to the constructive ordinals have been algorithms, as embodied in computer programs, and natural numbers. The former give rise to the term "constructive," while the latter are a convenient though not really essential choice as ordinal notations. Another way to combine these ingredients in the study of ordinals is via the well-ordering concept, as follows. An ordinal is called **recursive** if it is the order type of an algorithmic well-ordering of natural numbers. In other words, the ordinal must be order-isomorphic to some well-ordering W whose field is a set of natural numbers and whose ordered pairs can be distinguished from all others by some algorithm.

For example, to show that $\omega + 2$ is recursive, one might produce the sequence

$$2,3,4,5,\ldots,0,1 \tag{4}$$

in which 0 and 1 follow all the other numbers, then define W to be $\{(x, y)\colon x \text{ precedes } y \text{ in } (4)\}$ and show by means of a program that W is algorithmic. More generally, it is not hard to show that the set of all recursive ordinals is a countable (since there are only countably many algorithms) initial segment of Cantor's second number class, and the least nonrecursive ordinal is a limit ordinal.

This recursive ordinal concept is a fairly straightforward constructive analogue of Cantor's well-ordering approach to the ordinals, whereas the main thrust of this article has been to constructivize Cantor's principles of ordinal generation. Remarkably, these two avenues lead to the same destination, for the recursive ordinals turn out to be precisely the constructive ordinals ([**14**], Section 11.8). This result is particularly satisfying because it gives evidence that the set of constructive ordinals is a natural one, being stable under two quite different characterizations.

Acknowledgement

I wish to thank the editor and a referee for valuable suggestions in the preparation of this article. I also wish to thank my friend and former teacher and advisor Don Kreider for introducing me to this subject matter.

References

[1] G. Boolos and R. Jeffrey, Computability and Logic, 2nd ed., Cambridge Univ. Press, 1980.

[2] G. Cantor, Beiträge zur Begründung der Transfiniten Mengenlehre, Math. Ann., 46 (1895) 481–512 and 49 (1897) 207–246.

[3] ______, Contributions to the Founding of the Theory of Transfinite Numbers, Dover, 1947, English translation with introduction and notes by P. E. B. Jourdain.

[4] W. Dauben, George Cantor and the origins of transfinite set theory, Scientific American, 248 (1983) 122–131.

[5] M. Davis, Computability and Unsolvability, McGraw-Hill, 1958.

[6] ______, Hilbert's tenth problem is solvable, Amer. Math. Monthly, 80 (1973) 233–269.

[7] K. Devlin, Fundamentals of Contemporary Set Theory, Springer-Verlag, 1979.

[8] K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I, Monatshefte fur Mathematik und Physik, 38 (1931) 173–198.

[9] P. Halmos, Naive Set Theory, Van Nostrand, 1960.

[10] E. Kamke, Theory of Sets, Dover, 1950.

[11] S. C. Kleene, On notation for ordinal numbers, J. Symbolic Logic, 3 (1938) 150–155.

[12] ______, On the forms of predicates in the theory of constructive ordinals (second paper), Amer. J. Math., 77 (1955) 405–428.

[13] M. Machtey and P. Young, An Introduction to the General Theory of Algorithms, North-Holland, 1978.

[14] H. Rogers, Jr., Theory of Recursive Functions and Effective Computability, McGraw-Hill, 1967.

[15] S. Willard, General Topology, Addison-Wesley, 1968.

Our Mathematical Alphabet

As I recite the alphabet to one who's only three
The world of mathematics is opened up to me.

For a, b, c are constants or parameters assigned
And D is a determinant or distance undefined.

The image which e gives to me is one I can't erase
For it can only mean for me the logarithmic base.

F, G, H are functions with appropriate domain
And i's a unit vector in the Gauss or complex plane.

J's a Bessel function and another kind is K.
L's a linear operator or inductance one could say.

M and N are integers but m could be a mass.
O is the number zero but $\varnothing$'s the empty class.

P and Q give odds that you will win or lose a bet.
R gives correlation of two variables you have met.

I see before me Einstein's world when I hear S and T
For they make me think of space and time and relativity.

At this point I'm so deep in thought of time and space and such
That velocity components u, v, w don't seem much.

Perhaps some day that three-year old may learn when fully grown
Why x, y, z imply for me how much there is unknown.

—M. R. Spiegel