

Applications of the Analysis of Infinities

Julie A. Theobald^{*†}

James S. Walker^{*†}

theobaja@uwec.edu

walkerjs@uwec.edu

December 2, 2005

Abstract. This paper surveys applications of Cantor's analysis of infinities to important problems in a wide variety of fields. Using Cantor's diagonal method, proofs are given of the unsolvability of the Halting Problem in computer science, and of an Incompleteness Theorem in logic. Examples are given of applications of Cantor's idea of different sizes of infinities in the fields of biology, linguistics, economics, and philosophy.

^{*}Department of Mathematics; University of Wisconsin–Eau Claire; Eau Claire, WI 54702–4004; Phone: 715–836–3301; Fax: 715–836–2924.

[†]We express our thanks for a UWEC ORSP grant which partially supported our research.

Introduction

Cantor's analysis of the different sizes of infinities is one of the crown jewels of mathematics. The basic ideas behind his analysis have been synthesized within elementary notions of set theory to such a degree that now they can be taught at a basic level (see [10], [3], or [6]). In this article, we explore some of the ways that the analysis of infinities has been applied in other fields.

The article begins with a detailed discussion of the application of Cantor's diagonal method, which he used for proving the uncountability of the real numbers, to proving the unsolvability of the *Halting Problem* in computer science. In discussing the Halting Problem, we use many of the original notions that Cantor created and touch on important developments in logic, such as Gödel numbers and Incompleteness Theorems. We then provide a summary of applications of the notion of uncountable infinities to a wide variety of other fields: biology, linguistics, economics, and philosophy. This summary illustrates how Cantor's ideas have permeated all of modern intellectual thought.

This paper arose from a *Lesson Study* for an introductory course, *Introduction to Mathematical Thinking*, for Liberal Arts students at our university, which uses [10] as text. Lesson Study is a method for practical research and development of new methods of teaching mathematics, proposed by James Stigler and James Hiebert in their excellent book, *The Teaching Gap* [12]. It involves concentrated studying and researching of an individual topic and collectively developing a method for teaching the topic. This involves a cycle of *study and research, lesson development, teaching, revision, and renewed teaching* leading towards a profound understanding of the topic and how to teach it. In [12], it is emphasized that teachers formally write up their results in Lesson Study

and publish those results for sharing with other teachers of the same material—this article is the fruit of our research in connection with teaching Chapter 3 of [10] on Infinity. Additional teaching material can be found by visiting the website listed in [13].

Applications to Computer Science and Logic

Through the years, computer science has split off from mathematics. Yet, the two fields are still very connected. There are computer programs that give an output to a wide variety of problems. A program that outputs in a finite time an answer to a particular problem is said to *halt*. The question then arises if there is a program that can analyze in advance whether or not any given program will halt (anyone who has suffered an unending loop bug when computer programming can see the value of such an analyzing program). This question is called the *Halting Problem*.

In a landmark article [2], the computer scientist Gregory Chaitin described three proofs of the unsolvability of the Halting Problem, i.e. that no such analyzing program for halting of other programs exists. We shall provide a more detailed discussion of the first proof that Chaitin gave, since it touches on several important mathematical topics: such as, Cantor's diagonal method, Gödel numbers, and Incompleteness Theorems.

Unsolvability of the Halting Problem

We will show that the Halting Problem is unsolvable for a restricted class of computer programs: programs that are designed to output a natural number when their input is a natural number. We shall refer to these programs as *computable functions*. To begin, observe that *all computable*

functions are a countable set. To a computer scientist, this last statement is obvious because a program is compiled in machine language to a finite string of bits. The set of all such finite strings of bits is countable via the following one-to-one correspondence with a subset of the natural numbers (here $p_1, p_2, \dots, p_n, \dots$ denote the prime numbers in increasing order):

$$\text{(string of bits): } b_1 b_2 \dots b_n \mapsto p_1^{b_1+1} \times p_2^{b_2+1} \times \dots \times p_n^{b_n+1}$$

$$\text{e.g.: } 1\ 0\ 0\ 1\ 1 \mapsto 2^2 \times 3^1 \times 5^1 \times 7^2 \times 11^2.$$

The one-to-one nature of this mapping is an immediate consequence of the uniqueness of prime factorizations. To see one-to-oneness, one could also just pair each compiled program's string of bits with a unique integer in base 2 notation, but we gave the proof above first because it is a simple version of the idea of Gödel numbers that we shall discuss later. We shall also provide a more mathematical, *machine-independent* proof below, which highlights the connection with Gödel numbers in mathematical logic. Since the computable functions are a countable set, we shall denote them by $\{f_n\}_{n=1}^{\infty}$.

Suppose that the Halting Problem were solvable. We will show that that assumption leads to a contradiction. The solvability of the Halting Problem implies that we can construct a computable function F as follows:

$$\text{for each } n: F(n) = \begin{cases} f_n(n) + 1 & \text{if } f_n \text{ halts on input } n \\ 1 & \text{if } f_n \text{ does not halt on input } n. \end{cases} \quad (1)$$

Note how the construction of F makes use of the solvability (computability) of the Halting Problem. The definition of F is analogous to Cantor's diagonal proof of the uncountability of decimal

expansions—we elaborate on this point below, and we also elaborate on the computability (in a finite number of steps for each input n) of the function F .

Now, since F is a computable function, it would have to belong to our list $\{f_n\}_{n=1}^{\infty}$. Therefore, $F = f_k$ for some k . But this leads us to a contradiction, by considering the definition of $F(k)$. If f_k halts on input k , then $F = f_k$ allows us to write $F(k) = f_k(k)$, *but in (1) we defined* $F(k) = f_k(k) + 1$. Thus we have a contradiction; forcing us to conclude that f_k does not halt on input k . But in that case, $F(k) = 1$ and therefore, because $F = f_k$, we get $f_k(k) = 1$. However, $f_k(k) = 1$ says that f_k halts on input k (it outputs 1). Again, we have a contradiction.

We have thus shown that the existence of the computable function F leads to an absurdity. Since the definition of F was predicated on the solvability of the Halting Problem, we must conclude that *the Halting Problem is unsolvable*.

Connection of the proof with Cantor's diagonal method

It is not hard to see the connection of the proof above with Cantor's diagonal method of proving that the reals are uncountable. Let's briefly review the gist of Cantor's diagonal method (our proof finesses the problem of non-unique decimal expansions for reals). Consider the set \mathcal{D} of all real numbers between 0 and 1 whose decimal expansions do not end in an infinite string of 0's or 9's. To see that \mathcal{D} is uncountable, suppose that $\{x_n\}_{n=1}^{\infty}$ were a list of all the elements of \mathcal{D} . Let $d_n[x]$ stand for the n^{th} digit in the decimal expansion of a number x between 0 and 1. Then an unlisted

number $U \in \mathcal{D}$ is defined as follows:

$$\text{For each } n: d_n[U] = \begin{cases} d_n[x_n] + 1 & \text{if } 0 \leq d_n[x_n] \leq 7 \\ 1 & \text{if } d_n[x_n] = 8 \text{ or } 9. \end{cases} \quad (2)$$

Clearly $U \in \mathcal{D}$; and, by construction: $d_n[x_n] \neq d_n[U]$ for each n . Thus, $U \notin \{x_n\}_{n=1}^{\infty} = \mathcal{D}$, contradicting $U \in \mathcal{D}$. Hence, \mathcal{D} is uncountable. The non-countability of \mathcal{D} , of course, implies the non-countability of the reals.

A philosophical observation

The close analogy between formulas (1) and (2) raises an important philosophical point. Regardless of whether or not one believes in the objective reality of different infinitely-sized sets,¹ *because Cantor's diagonal argument was available as a method of proof to computer scientists, it provided a simple resolution of the Halting Problem*, an important problem in computer science. Theoretical problems in mathematics often provide tools to other disciplines; here we have just one example, we shall briefly review other examples below.

Numbering computable functions, and Incompleteness Theorems

As promised above, we shall provide a *machine-independent* description of how to count computable functions. We shall see that this counting also provides an elementary proof of an *Incom-*

¹For the record, we believe that infinite sets of different sizes do exist objectively *within the realm of human thought and discourse* (our ability to agree, and logically converse about, the existence of such sets is one hallmark distinguishing humans from other animals), but they have no existence in some otherworldly Platonic realm for which there is no empirical evidence. See [3], [5], and [6].

pleteness Theorem. The first such Incompleteness Theorem was the famous theorem of Gödel, published in 1931. As pointed out by Chaitin in [2], advances in the theory of computation since that time allow for results of wider scope and easier proofs than Gödel's original Incompleteness Theorem.

The counting method involves assigning natural numbers, called Gödel numbers, to each computable function. The basic ideas, phrased in the context of mathematical logic, are well-explained in the standard exposition [8, Chap. VII], so we shall be brief. The idea is to assign exponents to the building blocks of computable functions, the elementary computations with variables shown in Table 1, and illustrated in Table 2. (We have also included some elementary logical and set operations because they will be used in proving an Incompleteness Theorem.)

The formation of a Gödel number for a computable function, the doubling of any natural number, is illustrated in Table 2. Because of the uniqueness of prime factorizations, we see that this Gödel numbering scheme assigns in a one-to-one fashion a natural number to each computable function. Thus the computable functions are countable.

Note in passing that the computer operations in the first row of Table 1 allow for iteration of multiple inputs as well as comparisons of sizes of natural numbers. Those operations provide enough of a foundation for designing a program which reads in elements from a list. In particular, the list of computable functions can be read in sequentially and evaluated—that is how we know that the function F defined above is computable.

Having demonstrated the countability of computable functions, we are now prepared to prove an Incompleteness Theorem.

Incompleteness Theorems

Recall that Gödel's Incompleteness theorem states that there are formally undecidable propositions in a consistent mathematical logic that is elaborate enough to include number-theoretic statements.²

Consider the set of all provably computable functions, i.e. those which can be proven to return a natural number for each given natural number as input using the *Computable Functions* syntax summarized in Table 1, which we shall refer to from now on as CF-syntax. Note that CF-syntax is also assumed to include basic axioms of set theory, e.g., the Axiom of Infinity specifying that the natural numbers \mathbb{N} exist. Furthermore, it is important to note that (as explained in [8, Chap. VII]) the proof that a proposition holds in the logical system generated by CF-syntax is subsumable under factorization of Gödel numbers—a *computable process using the logical and program atoms summarized in Table 1*. This set of provably computable functions in CF-syntax is countable, by the formation of Gödel numbers as described above. It is also the *sine qua non* of number theory.³

We now outline the proof that there is a computable function *that is not provably computable using CF-syntax*, and that this yields incompleteness of the logical system generated by CF-syntax. Let \mathcal{F} denote the countable set of all provably computable functions using CF-syntax. The beauty of Cantor's diagonal method is that it generates a function that does not belong to any list of functions, in this case a listing of \mathcal{F} . We will now show that there is a computable function G that simultaneously lists all elements of \mathcal{F} *and*, by construction, does not belong to \mathcal{F} .

The function G is computed as follows. For each number $n = 2, 3, 4, \dots$, in succession,

²Gödel reformulated the famous Liar paradox statement (see [15]): *This statement is false* into *This statement is not provable*. This latter statement is then either true (so there is incompleteness) or false (so there is inconsistency).

³Paraphrasing Hilbert: '*Gentle ones, let us compute!*'

decompose n into its prime factorization. If this prime factorization is a Gödel number of a proof of computability of a function $g \in \mathcal{F}$,⁴ then add g to the list as $g = g_{k[n]}$ and put

$$G(k[n]) = g_{k[n]}(k[n]) + 1. \quad (3)$$

This procedure simultaneously computes G , lists the elements of \mathcal{F} (repetitions allowed), and constructs G so that G does not appear in the generated listing of all the elements of \mathcal{F} . Therefore, G is non-halting and computable, but it cannot be proven to be so using CF-syntax. In particular, the statement

$$n \in \mathbb{N} \Rightarrow G(n) \in \mathbb{N} \quad (4)$$

is known to be true, but cannot be proven using the statements of CF-syntax.

To summarize, we have sketched the proof of the following Incompleteness Theorem.

Theorem *The logical system generated by CF-syntax is (1) large enough to include elementary number theory, (2) either inconsistent or consistent but incomplete.*

Further Applications

The ramifications of Cantor's ideas go well beyond just pure mathematics and computer science. Indeed, they permeate much of modern intellectual life. We conclude this article with a brief survey of some tantalizing references to Cantor's ideas in other fields.

⁴Note: Here we are implicitly assuming that the logical system generated by CF-syntax is consistent, so that a proof of $g \in \mathcal{F}$ does, in fact, imply that $g \in \mathcal{F}$.

Biology

In [7, p. 49], the biologist Richard Lewontin uses the concept of an uncountable infinity in his critique of the abstract concept of an ecological niche:

The concept of an empty ecological niche cannot be made concrete. There is a non-countable infinity of ways in which the physical world can be put together to describe an ecological niche, nearly all of which would seem absurd or arbitrary because we have never seen an organism occupying such a niche.

Linguistics

The linguist, Stephen Anderson, contrasts the countable infinity of the rationals with the uncountable infinity of the reals in a fascinating comparison of the dance language of bees with human language ([1, pp. 78–79]):

The waggle dance [of honey bees], however, conveys its message of location in terms of three parameters (distance, direction, and quality), and each of these parameters can in principle have any value within a range of possibility . . .

In mathematical terms, a parameter that ranges over a continuous interval can take on as many values as there are real numbers, a value represented symbolically as \aleph . A human language such as English, in contrast to the bees' system, produces new messages not by choosing points on a continuous interval but rather by combining discrete elements in new ways. As a result, the number of sentences in such a system is the same as the number of rational numbers, or \aleph_0 . Because \aleph_0 is less than \aleph , the

communication system of the waggle dance is — in principle — actually richer than English, in terms of the sheer number of potentially distinct messages.⁵

Economics

In [9, pp. 43–44], the economist Paul Ormerod, uses the concept of an uncountable infinity in his critique of orthodox mathematical economics:

To take just one example, the phrase ‘assume a continuum of traders’ will be encountered in many theoretical papers on the idealised market economy. . . .

But what does this phrase ‘continuum’ actually mean? It sounds quite innocuous, yet spelt out in words it might lead people to query the realism of any academic paper based on this assumption, or even begin to doubt whether the article was worth writing in the first place. For the phrase means that the number of people, whether as individuals or as firms, carrying out trade in this theoretical economy is not just large but quite literally infinite. In fact, to be strictly accurate, it even means rather more than this. If one were to start to count the whole numbers — one, two, three and so on — one could go on for ever. There is an infinite number of them. . . . But mathematicians have the apparently bizarre but nevertheless logical concept of infinities which are even bigger than this infinity! A continuum is exactly one of these. In other words, it is assumed

⁵It should be noted that no claim is made in [1] that the waggle dance language of bees is *actually* richer than English, since Anderson goes on to state that “Bees are not capable of arbitrarily fine discrimination in perception and production of various elements of the dance, though, so this discussion of continuous-valued parameters is undoubtedly irrelevant in practice, at least as the basis for estimating the expressive capacity of the dance system.”

that there is not just an infinite number of traders, in the sense that the set of whole numbers is infinite, but there is an even bigger number of them than this.

Philosophy

We conclude our survey of examples from other fields with some examples from philosophy. The following quotation from a student essay [11] is a fine example of how the notion of separate infinities leads to philosophical ruminations:

there is a profound aesthetic quality—something universal, beyond numbers—to the notion of separate infinities. One example I can see here is a correlation to the human mind, with one mind, for example, possessing its own infinite set of possibilities, determined by a combination of its unique genetic structure and the myriad unique impressions on it over a lifetime, then another mind having, again, another infinite set of possible outcomes, and so on.

In considering this passage, we observe that it highlights the interaction of a discrete language, the language of genes, which could only generate a countable infinity of instructions, with the environment, which generates an uncountable infinity (a continuum) of interactions with the organism. That brings us full circle, back to biology: see [7, Figure 1.10d, p. 35] for an illustration of how the continuum of effects of developmental noise interacts with the countable infinity of messages from the genetic code to produce the endless variety of nature's organisms.

We conclude with some wisdom from Buddhist philosophy, a portion of a Zen meditation chant from the Ring of Bone Zendo [4, p. 103]:

Beings are numberless: I vow to enlighten them

Obstacles are countless: I vow to cut them down

Dharma gates are limitless: I vow to master them

and recall the famous rejoinder of Hilbert:

We shall never leave this paradise Cantor has created for us.

References

- [1] S.R. Anderson, *Doctor Dolittle's Delusion*. Yale University Press, 2004.
- [2] G. Chaitin, Gödel's Theorem and Information. *International Journal of Theoretical Physics*, **21**, 941–954 (1982). Reprinted in [14], 300–311 (1998).
- [3] P.J. Davis and R. Hersh, *The Mathematical Experience*. Houghton Mifflin, 1982.
- [4] J. Halper (Ed.), *Gary Snyder: Dimensions of a Life*. Sierra Club Books, 1991.
- [5] R. Hersh, *What is Mathematics, Really?* Oxford University Press, 1997.
- [6] G. Lakoff and R.E. Núñez, *Where Mathematics Comes From*. Basic Books, 2000.
- [7] R.C. Lewontin, *The Triple Helix*. Harvard University Press, 2000.
- [8] E. Nagel and J.R. Newman, *Gödel's Proof*. New York University Press, 1958.
- [9] P. Ormerod, *The Death of Economics*. St. Martin's Press, 1995.

- [10] M. Starbird and E.B. Burger, *The Heart of Mathematics, Second Edition*. Key College Publishing, 2005.
- [11] J. Steans, Final essay for UWEC Summer 2005 course: Math 106, Introduction to Mathematical Thinking.
- [12] James W. Stigler and James Hiebert, *The Teaching Gap*. The Free Press, 1999.
- [13] Julie A. Theobald and James S. Walker, Webpage on the Analysis of Infinities at
http://www.uwec.edu/walkerjs/Lesson_Study/Infinity/
- [14] T. Tymoczko (Ed.), *New Directions in the Philosophy of Mathematics*. Princeton University Press, 1998.
- [15] J.S. Walker, An elementary resolution of the liar paradox. *College Math Journal*, **35**, 105-111 (2004).

Table 1: Assigning Gödel exponents to components of the *Computable Function* syntax.

There are an infinite number of memory variables (assigning as exponents the primes larger than 20) and an infinite number of input variables (assigning as exponents the squares of primes larger than 20).

<i>Program atoms:</i>	Input	Output	Do	Until	Loop	<	+	-	0	1
<i>Gödel exponents:</i>	1	2	3	4	5	6	7	8	9	10
<i>Logical/Set Atoms*:</i>	\vee	\sim	\Rightarrow	\exists	ϕ	\mapsto	()	,	\in
<i>Gödel exponents:</i>	11	12	13	14	15	16	17	18	19	20
<i>Memory Variables:</i>	X	Y	Z	X_1	Y_1	Z_1	X_2	Y_2	Z_2	...
<i>Gödel exponents:</i>	23	29	31	37	41	43	47	53	59	...
<i>Input Variables:</i>	k	m	n	k_1	m_1	n_1	j_2	k_2	n_2	...
<i>Gödel exponents:</i>	23^2	29^2	31^2	37^2	41^2	43^2	47^2	53^2	59^2	...

*The symbol \vee stands for logical *or*, the symbol \sim stands for logical *not*, the symbol \Rightarrow stands for logical implication, and the symbol \exists stands for the logical statement *there exists*. The symbol ϕ stands for the empty set, the symbol \mapsto stands for the computer command *store in* (e.g., the statement $n \mapsto X$ means *store the value of n in the memory variable X*), the symbol \in stands for *is an element of* in set notation.

Table 2: Gödel numbering of a program that doubles a natural number k .

<u>Statements</u>	<u>Exponents</u>
Input k	1, 23^2
$(k + k) \mapsto X$	17, 23^2 , 7, 23^2 , 18, 16, 23
Output X	2, 23
<i>Gödel number:</i> $2^1 \times 3^{23^2} \times 5^{17} \times 7^{23^2} \times 11^7 \times 13^{23^2} \times 17^{18} \times 19^{16} \times 23^{23} \times 29^2 \times 31^{23}$	