UNIVERSITY OF CALIFORNIA,
IRVINE

Mathematical Limits and Philosophical Significance of Transfinite Computation

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Philosophy

by

Benjamin Gilead Rin

2014

DEDICATION

To my loving parents and siblings.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

In my time at the University of California, Irvine, I have benefitted from the support of a Social Science Merit Fellowship from the School of Social Sciences. In the fall of 2013, I was funded by an Associate Dean's Fellowship, as well as a grant from the *Analysis Trust.* In the summers of 2008 and 2011-2014, I was supported by Regents' Summer Fellowships. I am grateful to all of these sources of funding which facilitated my ability to conduct this research.

I am also grateful to the members of my dissertation committee, without whom none of this would have happened. It was Jeff Barrett who first introduced me to the subject of transfinite computation. Since then, he has served as an excellent mentor. His experience and perspective led me through some of those most difficult parts of this journey. Sean Walsh introduced me to the areas in philosophy of math which this dissertation addresses. He has worked tirelessly to shape my research to the place where it is now and I am forever indebted for all that he's done. Kai Wehmeier has been an exceptional role model and teacher. Separately from his role on my dissertation committee, he also initiated a co-authored project ('Actuality in Propositional Modal Logic', *Studia Logica,* June 2013) which, though not part of my thesis, was an invaluable experience for me.

In addition to the above, I would like to thank the following individuals for their helpful conversations regarding the material in Chapter 2: Kyle Banick, Philip Ehrlich, Ethan Galebach, Peter Koepke, Christopher Menzel, Charles Parsons, Erich Reck, Sam Roberts, Sam Sanders, Claudio Ternullo, John Wigglesworth, and two anonymous referees. I would also like to thank the following individuals for their suggestions regarding the material in Chapter 3: Merlin Carl, Joel Hamkins, Bob Lubarsky, Benjamin Seyfferth, Philip Welch, and an anonymous referee. Further, I thank the journals *Synthese* and *Annals of Pure and Applied Logic* for their permission to use my published work in this thesis.

Finally, I would like to thank some other important people who have contributed to my development as a philosopher and thinker. This list is by no means exhaustive: Ethan Bloch, Mary Coleman, Sam Fletcher, Garry Hagberg, Jeremy Heis, Bennett Holman, Kent Johnson, Penelope Maddy, David Malament, Bob Martin, Bob McGrail, Lauren Rose.

CURRICULUM VITAE

Benjamin Gilead Rin

EDUCATION

Ph.D., Philosophy	**2014**
University of California, Irvine	*Irvine, California*
M.A., Philosophy	**2011**
University of California, Irvine	*Irvine, California*
B.A., Mathematics	**2006**
Bard College	*Annandale-on-Hudson, New York*

Areas of Specialization
Philosophical Logic, Mathematical Logic, Philosophy of Mathematics

Areas of Competence
Philosophy of Language, Philosophy of Science, 20$^{\text{th}}$-century Analytic Philosophy

PAPERS

'Transfinite Computation and Recursion in the Iterative Conception of Set."
' To appear in *Synthese.*

"The Computational Strengths of α-tape Infinite Time Turing Machines."
Annals of Pure and Applied Logic. Volume 165, Issue 9, September 2014, p.1501-1511.

"Actuality in Propositional Modal Logic."
Studia Logica, June 2013 (with Kai Wehmeier and Allen Hazen).

AWARDS AND GRANTS

Social Sciences Merit Fellowship	**2007–2014**
Associate Dean's Fellowship	**Fall 2013**
Analysis Trust Grant	**Fall 2013**
Regents' Summer Fellowship	**2008, 2011-2014**

ABSTRACT OF THE DISSERTATION

Mathematical Limits and Philosophical Significance of Transfinite Computation

By

Benjamin Gilead Rin

Doctor of Philosophy in Philosophy

University of California, Irvine, 2014

Professor Jeffrey A. Barrett, Co-chair

Assistant Professor Sean Walsh, Co-chair

The general aim of this thesis is to explore applications of transfinite computation within the philosophy of mathematics—particularly the foundations and philosophy of set theory—and to contribute to the young but rapidly growing literature on the technical capabilities and limits of transfinite computation.

The material is divided into two main parts. The first concerns transfinite recursion, an essential component of the practice of set theory. Here we seek intrinsically justified reasons for believing in transfinite recursion and the notions of higher computation that surround it. In doing so, we consider several kinds of recursion principle and prove results concerning their relation to one another. Next, philosophical motivations are considered for these formal principles, coming from the idea that computational notions lie at the core of our conception of set. This is significant because, while the iterative conception of set has been widely recognized as insufficient to establish Replacement and recursion, its supplementation by considerations pertaining to algorithms suggests a new and philosophically well-motivated reason to believe in such principles.

The second part addresses technical questions on the mathematical capabilities and limits of a family of models of transfinite computation. These models resemble the infinite-time Turing machine (ITTM) model of Hamkins and Lewis, except with α-length tape (for some ordinal $\alpha \geq \omega$). Let T_α denote the machine model of tape length α (so T_ω is equivalent to the ITTM model). Define that T_α is *computationally stronger* than T_β (abbreviated $T_\alpha \succ T_\beta$) precisely when T_α can compute all T_β-computable functions on binary input strings of size $\min(\alpha, \beta)$, plus more. The following results are found: (1) $T_{\omega_1} \succ T_\omega$. (2) There are countable ordinals α such that $T_\alpha \succ T_\omega$, the smallest of which is precisely γ, the supremum of ordinals clockable by T_ω. In fact, there is a hierarchy of countable T_αs of increasing strength corresponding to the transfinite (weak) Turing-jump operator $^\nabla$. (3) There is a countable ordinal μ such that neither $T_\mu \succeq T_{\omega_1}$ nor $T_\mu \preceq T_{\omega_1}$—that is, the models T_μ and T_{ω_1} are computation-strength incommensurable (and the same holds if countable $\mu' > \mu$ replaces μ). A similar fact holds for any larger uncountable device replacing T_{ω_1}. (4) Further observations are made about countable T_αs, including the existence of incommensurabilities between them.

Chapter 1

Introduction

1.1 Overview

The general aim of my work is to explore applications of transfinite computation within the philosophy of mathematics—particularly the foundations and philosophy of set theory—and to contribute to the young but rapidly growing literature on the technical capabilities and limits of transfinite computation.

This dissertation is divided into two parts. The first is largely philosophical, though with a few technical results, and argues for applications of transfinite notions of computation within the foundations of math. The second part is primarily technical; in it, a new type of variant of transfinite computing machine is defined, and theorems are proved concerning the computational strength of the resulting models of computation. The parts of this thesis are written in such a way that they can be read as standalone papers and still be understood. Accordingly, the reader's familiarity with the other part is not assumed. That said, the order in which they are presented is chosen to maximize the motivation for the technical work of the second part.

The philosophical and mathematical aims of this dissertation are distinct but connected. In order to understand fully the ramifications of transfinite computation on issues within philosophy, it is crucial to have a solid body of theory on the mathematical features of the computational models in question. I say ‘models’ (in the plural) because there are many types of non-equivalent transfinite machines that are *prima facie* viable candidates for modeling what ought to count as transfinite computation. This is in contrast to the situation found in classical computability theory. There, the concept of computability is widely taken to be fully captured by the Turing machine model (and its equivalents). For this is, more or less, what the celebrated Church-Turing thesis tells us. While a transfinite rendition of the Church-Turing thesis may be possible—indeed, such a rendition was recently proposed in [10], as I discuss in Part I—none has yet gained widespread currency. As a result, the technical work in the second part of this dissertation addresses only a small fragment of the possible models of transfinite computing. It should go without saying that the theorems proved here are far from the last word on the matter of mathematically understanding transfinite computation. While the philosophical questions raised in Part I are given a preliminary attempt at an answer, a full resolution may yet require the mathematical community’s development of a better understanding of what transfinite machines are mathematically capable of. Part II comprises my contribution to this understanding. It turns out that my work in Part I only makes direct use of a model of computation already explored in the literature, not the ones newly defined in Part II. Yet the results of Part II are still indirectly relevant, for reasons that will be explained in the conclusion.

1.2 What Is Transfinite Computation?

Computation is standardly taken as a finitary notion. The input, the output, the speed of computation and the memory used are all assumed to be finite. In transfinite computation, this assumption is abandoned. In this setting, a machine's resources—and hence the possible input and output information with which the machine operates—can be infinite. The driving intuition goes: if we somehow could have access to an infinitely fast computer, what could we do with it? What, if any, would be its limitations? The literature of the last fifteen years has sought to provide answers to these questions.

Let us briefly give an informal description first of the finitary notion of computation as canonically represented by a Turing machine. Then it will be possible to describe its transfinite analogues. Some of the technical definitions will be omitted from this introduction and saved until they are needed (in Part II).

A Turing machine device is a theoretical construct capable of computing precisely the recursive functions. It includes a tape with ω many cells, indexed by natural numbers, and a moving head that occupies one cell at a time. Each cell must contain a 1 or 0 at all times. At the beginning of computation, the head rests on the cell at the far left (cell 0). The *input* in a computation is the starting values of the cells before computation begins. At every step of computation, the machine reads the value at the current head location, and based on this reading and the program's instructions, changes the value or leaves it and then moves either left or right. After this, it repeats the process. If and when the program instructs the machine to *halt*, computation stops. By stipulation, a halting Turing computation must be finite in duration (measured by the number of computational steps carried out). The content of the tape after halting is known as *output.* If, for some function f on the natural numbers, a program causes a computer to consistently output $f(n)$ for each possible input n, the program is said to *compute* the function f. If a program exists that computes a given function f, then f is called *computable.*

Some computations fail to halt, either because they force the machine into a loop or else because the machine is performing an endless task (e.g., searching indefinitely for ever larger prime numbers). The *halting problem* is the task (representable as a function) of reliably outputting ‘yes’ or ‘no’ (coded as a 1 or 0) depending on whether or not a given (input value numerically representing a) program halts when run on null input (input $\vec{0}$). It turns out that there is no program that solves the halting problem on a Turing machine. Thus there is no way to program a machine to tell in advance whether any given computation will halt. This is an example—indeed, the most well-known example—of an incomputable function.

The functions that are computable by a Turing machine turn out to be exactly those which are recursive. A thorough technical account of Turing machines and recursive functions can be found in [56, Ch. 1]. For our present purposes, what matters most is that recursivity is a way to characterize which functions are and are not Turing-computable.

With respect to transfinite computation, the first model to be conceived and described in print is the infinite-time Turing machine model from [21]. It is intuitively intended to represent an infinitely fast computer. It is called an infinite *time* Turing machine not because it takes infinitely many seconds to compute (it does not), but because, in contrast to the classical Turing machine, it is permitted to run for infinitely many steps of computation before halting. In this sense, time is conceived not in terms of seconds or minutes, but in computational steps.

Whereas the classical finitary Turing machine is considered to “fail” in some sense if it does not successfully halt within a finite number of steps, an ITTM may compute through steps $1, 2, \ldots, \omega, \omega+1, \omega+2, \ldots, \omega+\omega, \ldots$ and halt after (say) $3\omega^4 + 2\omega + 6$ steps. Indeed, it can even halt after an ordinal number of steps too large to be defined recursively.

The model is defined in [21] similarly to the classical Turing machine, with a few key changes. First, as just mentioned, computations need not halt in finite time, hence the rules of computation must be defined slightly differently to accomodate this. While the rules are the same as those of the Turing machine at finite and successor ordinal steps, at limit ordinal steps such as ω the machine is set to a special program state called the *limit* state, the cell’s values are set to the *inferior limit*[1] of their values at prior steps, and the position of the head is returned to the leftmost cell (i.e., cell 0).

The second change is that [21] allows for an infinite length of tape to be used, because computations are permitted to run for transfinitely many steps (i.e., the machine has more than enough time to read and operate with ω-length strings of bits). Thus the device can be thought of as computing functions on the reals.[2]

It is known that the ITTM model of computation is strictly more powerful than the Turing model; it can compute any Turing-computable function, plus decide the classical halting problem. In fact, as proved in [21], it is powerful enough to decide membership in any set of reals up to, and in some cases beyond, complexity Π^1_1. That said, the ITTM model's computational power has limits. This is made clear from the fact that a *program*, by definition, is a finite set of commands satisfying certain conditions; an entire program can be coded by a single Gödel number. Hence there are countably many programs, thus countably many ITTM-computable functions. For cardinality reasons alone, there are functions on the reals that are not ITTM-computable. Indeed, the ITTM has its own analog of the halting problem, which is undecidable but semi-decidable by the ITTM, by exactly the same diagonalization type of argument as in the classical case. It has been found that all ITTM-semi-decidable sets of reals fall under Δ^1_2, giving an upper bound on the ITTM's strength (see [21]).

Research in the study of ITTMs and related models has recently been very active. Broadly, there have been two main directions of research being done in the area. One direction looks at

[1]The inferior limit, or $\liminf$, of previous cell values is calculated by setting each cell to value 1 if it had stabilized at 1 from some prior successor step onward, and all other cells to 0—in other words, every cell that stabilized at 1 (i.e., had value 1 without changing from some point onward) takes on value 1, and every cell that stabilized at 0 or never stabilized at all (i.e., had its value change between 0 and 1 unboundedly often) is set to 0. This rule, together with the ordinary Turing rules that govern cell values at successor steps, completely determines a unique value for the machine's cell contents at any transfinite ordinal time (assuming one knows which program p and initial input string x was chosen to be run on the machine). The original paper [21] uses a *superior limit* rule instead of the $\liminf$ rule described here, but subsequent authors, e.g. [33], use the latter. The resulting models of computation are equivalent in either case.

[2]As in [21], we refer to ω-length strings (members of Cantor space ${}^{\omega}2$) as *reals*.

the theoretical possibility of physically instantiating such a machine, based on the limitations imposed by our best physical theories (relativity and quantum mechanics in particular). Examples of work in this vein include [2], [15], [16], and [17]. Discussion touches on the use of rockets moving at near-light speeds, arranged so that what is infinite time for one frame of reference corresponds to finite time for another. A normal computer in the process of falling through a black hole in a particular way can also serve the purpose. Elsewhere, the literature considers Pitowsky spacetimes, Malament-Hogarth spacetimes, and Etesi-Németi models to provide other ways to perform such computations. For a partial survey, see [61]. Part of what has drawn this level of interest is that the degree to which these machines are physically possible, and the extent to which they are mathematically more powerful than classical Turing machines, together determine to what extent, if any, the resulting models of computation present a threat to the Church-Turing thesis. (Or at least, to a physical rendition of the Church-Turing thesis.)

The second vein of discussion centers around the mathematical, rather than physical, limits on what transfinite machines can accomplish. It is with this vein that the second part of my dissertation research is concerned. As mentioned, even though the machines exceed the classical machines in power, they possess computational shortcomings of their own. Further, there are many non-equivalent ways to mathematically define models of transfinite computation, of which the ITTM is just one. An alternative is the *infinite-time register machine* (ITRM), which extends the classical register machine into the transfinite context (see [11], [34], and [61]). Yet another type of model extends the ITTM by considering what happens when the tape length is increased beyond size ω. Throughout the dissertation, I focus on this latter way of defining models in particular.

1.3 Applications and Limits

In Part I of this dissertation, I examine the consequences of infinitary models of computation for the philosophy of set theory. The primary target of discussion is the *iterative conception of set*. Under the iterative conception, the sets in set theory begin with the empty set (and "individuals", if such exist), and generate further sets in stages: there are all the sets of individuals (if any) and a set containing the empty set as a member (i.e., $\{\emptyset\}$), there are the sets of sets of individuals and sets like $\{\{\emptyset\}\}$ and $\{\{\emptyset\},\emptyset\}$, sets containing those, sets containing *those*, and so on transfinitely.

The iterative conception is standardly taken as a powerful intuitive picture on which to ground set theory. Some would say it gives us reason to believe in the axioms. However, it is widely recognized that the conception fails to justify the existence of even "small" infinities: those that were present in Cantor's work and around which contemporary set theory revolves. As elaborated in Part I, this is roughly because sets on this conception are just extensionalizations of properties formed in a cumulative process, none of which need to be large. However, many of the relevant types of infinity are computable according to recent infinitary models of computation. Indeed, such models are capable of performing infinitary recursive procedures, which can generate large infinities but are problematic to justify using the iterative conception of set. Because of this, I argue that the iterative conception ought to be supplemented by the idea that intensional entities like algorithms determine infinitary functions just as intensional entities like classes determine sets. Since the approach advocated here is novel, the scope of the present work is only to outline some preliminary ideas and provide a few initial results. Thus a great deal of work remains to be done in future research to develop this program forward. Key is determining an appropriate way to make formal the informal idea that algorithms determine functions. But further, as stated before, the project would also benefit from new mathematical results shaping our understanding of transfinite computation.

In Part II, I fill a void in the technical literature on machines of the type defined there, by proving results on their relative computational strengths. These machines are related, though not identical, to the model employed in Part I. The respective machine models in Part I and Part II have the common feature that they are ITTM-like, but with a longer infinite tape.

Yet there are two substantial differences. The first is that the increased infinite tape lengths are not the same in the two cases. The machine model used in Part I, called the *Ordinal Turing Machine* (OTM) model (originally defined in [33]), has one cell in it for every transfinite ordinal β. This parallels a Turing machine having one cell for every natural number n. The models defined in Part II, however, each contain one cell for every $\beta < \alpha$, where α is some predefined ordinal bounding the tape length, analogous to having a 1-cell Turing machine, a 2-cell Turing machine, a 3-cell Turing machine, etc. The second difference is that the OTM model allows for the use of ordinal parameters in its computations, whereas the machines in Part II do not. The significance of these differences will be discussed in the conclusion. There I will explain to what extent the results in Part II, while relevant in their own right to the interests of an ordinal computability theorist, might also inform the philosophical project in Part I.

Chapter 2

I — Transfinite Computation and Recursion in the Iterative Conception

2.1 Introduction

It has been often noted that the notion of infinity that we find in pre-19th century sources is vastly different from the notion that we inherit from Cantor and that is tacit in the axioms of ZFC set theory. For instance, pre-Cantorian notions tend to accept the principle that a proper subset of the natural numbers should be smaller than the natural numbers themselves, a principle which the Cantorian theory of cardinality happily violates. In contrast to the pre-Cantorian conception, a distinctive feature of the Cantorian conception of the infinite is that we can apply induction to ordinal and cardinal numbers. That said, induction was not the only feature of the natural numbers which was transferred to the infinite: for, in Cantor's works and practice, we find that the notion of recursion was simultaneously transferred to the infinite.

While much in the extant literature on the philosophy of set theory has attended to the idea of a well-order inherent in the Cantorian conception of the infinite, less attention has been paid to the notion of recursion, and this paper seeks to fill this gap in the literature. To be sure, part of this lack of attention can be traced to the fact that the iterative conception of set is sometimes thought not to justify reflection and Replacement, either of which may be used to give the standard proof of transfinite recursion. Our guiding questions here are thus: what notions of recursion and computation are available for Cantor's infinite collections, and of what philosophical principles about the notion of set can we avail ourselves to validate these principles?

Part of the difficulty in answering these questions resides in the comparative opacity of the conception of computation even in the ordinary setting of the natural numbers. There we have a confluence of different models (Turing programs, general recursive functions, etc.), but this confluence does not seemingly deliver an axiomatic conception of computation in the same way that the Peano axioms deliver an axiomatic conception of number. Despite this, it is suggested in this paper that some light can be shed on these conceptual questions by asking after the pre-theoretic ideas implicit in recent work by Hamkins, Koepke, and Welch on infinite time and tape Turing machines. So, in this paper, we proceed by contrasting these pre-theoretic ideas to those behind Kreisel and Sacks' work on higher recursion theory and some related ideas of Jensen and Karp on transfinite analogues of primitive recursion.

Section 2.2 begins with some historical examples of uses of transfinite recursion. While the standard proof of the theorem of transfinite recursion using Replacement is of course correct, it is natural to ask after reasons for believing in transfinite recursion that are more strongly rooted in our foundational notions of set and ordinal. Next we consider Kanamori's history of the Axiom of Replacement and suggest that it does not provide a convincing case for the Axiom to someone who is interested in strong intrinsic reasons for accepting such principles. In section 2.3, we define some axioms to serve as a basic initial attempt at formalizing principles to potentially ground recursion. Realizing how closely these relate to computational notions, we look in section 2.4 at the possibility of grounding recursion on metacomputational notions falling from Koepke's so-called ordinal Turing machines (OTMs). It is argued that such machines present good candidates for answering a challenge once cast by Kreisel concerning the possibility of a notion of higher computation corresponding to his (and later Sacks') notion of higher recursion. We conclude by considering the principle, "Algorithms determine functions," and offer some initial attempts to formalize it. The intent is to advance this as a foundational principle intrinsic to our fundamental conception of set.

Ultimately, the aim of this paper will be less to provide definitive answers to the guiding questions, and more to provide a reasonable starting point that draws attention to an area which until now has received less notice than it deserves. The attempts here to formalize the intuition that algorithms determine functions will be seen to encounter challenges whose solutions are non-trivial and warrant further research. Still, it is hoped that the philosophical intuition proposed here will spur further consideration that eventually leads to the right kind of formal theory.

2.2 The Practice and Justification of Transfinite Recursion

The first historical use of transfinite recursion is due to Cantor, in the proof of what is now called the Cantor-Bendixson Theorem.[1] The theorem states that every closed uncountable set X of real numbers contains a perfect subset. A *perfect set* is a set containing no isolated points.[2] Cantor's proof of the theorem uses a procedure that we now recognize as transfinite

[1]For a modern treatment, see [24, p. 40]. This result is sometimes called the perfect set theorem.

[2]A real number within a set X is *isolated in X* if none of its neighbors are in X—that is, if there exists an open interval of reals containing it such that no other numbers in the interval are in X.

recursion. First, he begins with the original set X, then removes all the isolated points from it. Naïvely, it would seem that the task is finished, but the resulting set—call it X_1—may well have points in it that were not isolated in X but have now become isolated thanks to the removal of some of its surrounding points. So Cantor removes all the now-isolated points from X_1, resulting in a new set X_2, and so on.

Cantor's insight was that this process may be iterated recursively (indeed, transfinitely recursively), and if it is, it will eventually reach a stage X_α when no isolated points remain to be removed. When this happens, the process can stop. The recursive function he defined that drives the proof is formally given by: $X_0 = X$, $X_{\alpha+1} = X_\alpha \setminus I(X_\alpha)$ and $X_\alpha = \bigcap_{\beta<\alpha} X_\beta$ for $X \subseteq \mathbb{R}$, where $I(Y)$ is the set of isolated points in Y. One can argue that the perfect set theorem marked the birth of modern (Cantorian) set theory. For it was here that transfinite ordinals were first used, in order to define a recursive process that does eventually halt but not at any finite stage.

Another historical example of transfinite recursion from the same time period is in the definitions of the basic operations of ordinal arithmetic.[3] Ordinal addition is defined by transfinite iteration of the "successor" operation. Ordinal multiplication is defined by transfinite iteration of the addition operation. Ordinal exponentiation is defined by transfinite iteration of multiplication. These definitions parallel and generalize their finitary counterparts (their restrictions to finite numbers giving exactly the recursive definitions of $+$, $\times$ and exp in Peano Arithmetic). What separates them is of course their behavior at the infinite limit step.

A more recent example of transfinite recursion, familiar to many philosophers, is seen in Kripke's iterative definition of truth (c.f. [23, Ch. 9]). Kripke's truth predicate famously uses a recursive procedure to evaluate sentences like, "It is true that it is true that it is false that it is true that snow is white," from the inside on out. The recursion necessarily caps at some countably infinite ordinal stage, proving that a three-valued language as Kripke defines

[3]See, e.g., [24, p. 23].

it can, unlike Tarski's, contain its own truth predicate. Problem sentences like the Liar do not receive a true or false value at any stage.

As a final, and rather iconic, couple of examples of transfinite recursion, we have the definitions of the stages of V, the Von Neumann universe of all sets, and of L, the Gödel universe of constructible sets. Beginning with the empty set $\emptyset = V_0 = L_0$, we recursively define $V_{\alpha+1} = \mathcal{P}(V_\alpha)$ and $L_{\alpha+1} = \mathrm{Def}(L_\alpha)$, where $\mathcal{P}$ is the power set operation and $\mathrm{Def}(Y)$ is the set of definable sets of Y.[4] At limit stages we define $V_\alpha = \bigcup_{\beta<\alpha} V_\beta$ and $L_\alpha = \bigcup_{\beta<\alpha} L_\beta$.[5]

The examples of definitions above share the common feature that each depends on applying an iterated procedure that takes the output of one iteration as the input for the next. This results in a function whose existence and well-definedness are guaranteed by the theorem of transfinite recursion (TR). The theorem TR, in turn, is standardly justified in ZFC by a proof reliant on the Axiom of Replacement. Replacement says that the range of a function whose domain is a set must also be an existing set.[6] Its role in the proof of TR is in helping to show inductively that for all ordinals δ, there exists an approximation of the desired recursive function accurate to the first δ output values. In brief, this is made to work because Replacement asserts the existence of the range of a function from domain δ to a set of smaller approximations (each $\delta' < \delta$ maps to an approximation accurate to δ' many values), whose union is then the desired δ-approximation.[7]

However, though Replacement might be a good reason to think that recursive definitions are justified, it is natural to ask: is this the reason why we in fact think that they are justified? Certainly Cantor did not justify his argument this way. Historically, the Axiom of Replacement was not introduced until 1922, and it was not until Von Neumann later made the

[4]I.e., $\mathrm{Def}(Y) = \Big\{\{y \mid y \in Y \text{ and } (Y, \in) \models \Phi(y, z_1, \ldots, z_n)\} \Big| \Phi \text{ is a first-order formula and } z_1, \ldots, z_n \in Y.\Big\}$

[5]See, e.g., [24, p. 175].

[6]There are other equivalent formulations, and in first-order set theory the axiom is standardly represented as an infinite schema.

[7]See [40, p. 25] or [24, p. 22].

connection between Replacement and recursion that the two were formally associated.[8] As a matter of actual practice, definitions by recursion require no understanding of or reference to the Axiom of Replacement at all. A mathematician working in topology can perfectly well understand the definition of Cantor's function without this formal grounding. Similarly, a philosopher working on formal theories of truth with little background in set theory can readily understand Kripke's construction without Replacement. Replacement is in practice quite divorced from transfinite recursion, despite the formal entailment.

Of course, to count as being the proper justification for something, an axiom or other principle need not be constantly borne in mind during its practical application. No one questions Cauchy sequences or Dedekind cuts as being legitimate justifications for the existence of real numbers by objecting that earlier mathematicians did not think about them in practice. But nonetheless, where multiple justifications are present, it is natural to think that some are better explanations than others, and that some enhance our understanding better than others. To continue with the analogy with the real numbers, it seems that part of the success of Cauchy sequences (or Dedekind cuts) is that $\{3, 3.1, 3.14, 3.141, 3.1415, \dots\}$ usefully *tells us something* about π. A present-day student—or a 19th-century analyst—who encounters Cauchy sequences would not merely gain a more rigorous knowledge of the existence of π in this way, but would learn something about the nature of π, namely that it is a limit of a certain rational approximation. The properties of such rational approximations can then be studied in numerous contexts, like that of transcendental number theory. It is expected, then, that a study of different reasons to believe in transfinite recursion will show that some of these reasons would enhance our understanding of the underlying concept of set, while others would merely justify definitions by transfinite recursion.

[8]See [28].

To make this point clearer, it is important to become more precise about just what is meant here by 'justifying a definition'. After all, stipulative definitions usually need no justification. So what could this possibly mean?

One (minimal) answer is that it means being sure that our definition actually succeeds at *being a definition*. That is, we want to be sure that there exists a function satisfying the description in the definition (and be sure that there is exactly one such function). Consider the following recursive instruction for computing $f(n)$:

$$f(n) = \begin{cases} \frac{5}{12}, & \text{if } n = 0 \\ \frac{1}{f(n-1)} - 2, & \text{if } n > 0 \end{cases}$$

The equation here offers a recursive description of what looks to be an unobjectionable mathematical function with domain the natural numbers and with outputs in the rationals. To keep the example simple, only regular (finitary) recursion is used, with no transfinite limit step. The recursive process here begins with $f(0) = \frac{5}{12}$. Subsequent values are calculated from previous answers such that $f(1) = \frac{1}{\frac{5}{12}} - 2 = \frac{2}{5}$, from which it follows that $f(2) = \frac{1}{2}$, and thus $f(3) = 0$. Here, though, we encounter a problem: we see straightforwardly that $f(4)$ has an undefined value. The equation above, which initially seemed to define a recursive function, in fact defines nothing. Based on this example, at the bare minimum we should require that justifying a recursive definition means making sure this type of thing does not happen. In short: definitions must define!

But it may be that one could ask for more than this. One might want not only a guarantee that there is a function satisfying the given description, but also for the function to be able to be "read off" the description in some uniform way. One could want, in other words, for there to be a single way to take descriptions of this kind and understand from them the behavior of the function described.

As an even stronger demand, one could also require that the definition be the 'right' one based on considerations arising within the most general relevant areas of mathematical practice. For example, consider how we define a prime number. In [57], Tappenden contrasts two competing definitions. The first is that $a \neq 1$ is prime if its factors are precisely 1 and a. The second is that $a \neq 1$ is prime if it divides b or c whenever it divides bc. These definitions are equivalent in $\mathbb{N}$—they have the same extension $\{2, 3, 5, 7, \ldots\}$—but are not equivalent in alternate or generalized contexts (e.g., in certain instances of modular arithmetic). It turns out that in these other contexts, the first definition fails to entail key results. On this, Tappenden writes:

> Once the two options are recognized, we need to say which will be canonical. [...] The word *'prime'* is given the *second* definition. [...] The reason for counting the second definition as the proper one is straightforward: The most significant facts about prime numbers turn out to depend on it. [...] The original definition of prime number fixes a set $\{2, 3, 5, 7, \ldots\}$ with interesting properties. In the original domain $\mathbb{N}$ it can be picked out by either definition. The new definition is the more important, explanatory property, so it is the natural one. [57, 268]

The justification of the second definition over the first connects strongly with the definition's usefulness and explanatory power in general mathematical practice, e.g., in ring theory. In contrast, the first serves well only in its native setting $\mathbb{N}$. Along these lines, a strong general criterion for justifying definitions could require that a justification for transfinite recursion ensure the definition captures the most general relevant properties of the function defined.

Regardless of what notion of justification is at play here, it is fair to ask: why should one feel the need to justify recursions in the first place? An answer that one could be tempted to give is that we need recursion for various results.[9] However, it is not trivial to formulate

[9]Returning to the motivating examples earlier in this section, one may notice that there is actually a certain asymmetry between the Cantor-Bendixson example and the others. In the Cantor-Bendixson case,

clearly what is meant by 'needing' transfinite recursion to do something. What one might have in mind in saying TR is 'needed' for a given result may be that one could choose an appropriate background set theory without it, say ZFC minus Replacement (ZC), and show that it follows formally from the given result. But it's wrong to think that this would be the only type of reason we should give in order to motivate the project of justifying transfinite recursion. If it turned out somehow that we did not need transfinite recursion for anything, even so, the question of its justification is of interest. The goal we have is to pinpoint what good reasons we have for believing in it, not whether it is the only route to this or that result.[10]

One kind of reason that can be offered has to do with to our conception of the ordinals. To the extent that the class Ω of ordinals is thought of as generalizing the notion of natural number, the possibility of performing recursive procedures on them should be viewed as an inherent part of what they are. Just as it is difficult to picture what the naturals $\mathbb{N}$ would 'look like' if it were impossible to perform recursion upon them, it should be equally difficult and bizarre to picture what the ordinals would look like if it were impossible to perform (transfinite) recursion upon *them.* Ordinal numbers *just are* the type of thing that one can use for performing transfinite recursion.

It is well known that the theory ZC can be modeled by the structure $(V_{\omega+\omega}, \in)$. Thus one might picture the collection Ω^- of ordinals-if-recursion-upon-them-were-impossible as simply the class $\omega+\omega$—now a proper class. But I hardly think that anyone believes this to be what

we are going "down" and not "up", in the sense that the sets being defined at each stage are smaller than those at previous stages. In contrast, the definitions of V_α and L_α, as well as the operations of transfinite arithmetic, result in objects that are unboundedly larger or higher ranked than their predecessors in the recursive construction. One may want to distinguish between these two types of cases, which arguably warrant different treatment. In 1922, the same year that Replacement was introduced, Kuratowski [42] actually gave a general procedure for how to do proofs with "downward" examples like the Cantor-Bendixson function without using recursion (see [20, p. 253*ff.*]). This method certainly does not work for cases like V_α.

[10]In section 2.3, we prove some results showing that Replacement is actually provable by route of certain types of recursion and other principles. This, of course, is useful. But the point is that even if we did not have results like these, what we are ultimately interested in are reasons for believing in such principles. These we talk about in sections 2.4 and 2.5.

the class of all ordinals truly looks like. Later in this paper, it will be seen that there exist recursion principles weaker than TR that are still able to surmount the $\omega+\omega$ barrier. But to eschew recursion above the finite altogether is to invite a very limited picture of the class Ω.

Having said that, the smallness of the model $(V_{\omega+\omega}, \in)$ has historically been used to motivate the Axiom of Replacement. As Kanamori recounts in [28], Fraenkel and Skolem pointed out in the '20s to Zermelo that the theory Z, with or without Choice, does not entail the existence of certain sets, such as $\mathcal{P}^{\omega}(\omega) =_{\text{df}} \{\omega, \mathcal{P}(\omega), \mathcal{P}(\mathcal{P}(\omega)), \ldots\}$. This set is formed, in modern terms, at stage $V_{\omega+\omega}$. In a letter to Fraenkel, partly reproduced in [28], Zermelo agreed that since Z cannot prove the existence of such sets, it cannot by itself be considered a complete theory of sets. But with Replacement, it becomes possible to construct the ordinal $\omega+\omega$ as a set using the function $n \mapsto \omega + n$ for $n \in \omega$; since the domain ω is a set, so must be its range $(\omega+\omega) \setminus \omega$, the union of which is $\omega+\omega$. One constructs the set $\mathcal{P}^{\omega}(\omega)$ from the function $n \mapsto \mathcal{P}^{n}(\omega)$ similarly.

However, it has since been recognized that while Replacement solves the $\omega+\omega$ problem, it is not well motivated by the iterative conception of set. For the iterative conception has it that sets are formed at "stages" out of the elements formed in previous stages. Beginning with the empty set, we transfinitely continue forming subsets of a given stage and collecting these subsets together to comprise the next stage of our universe of sets. But in [5], [6], and [50], we see that certain natural attempts at formalizing the iterative conception are in fact satisfiable by models like $(V_{\omega+\omega}, \in)$. Thus the iterative conception, at least under the best current attempts at formalization, fails to justify Replacement. Relatedly, as Martin has expressed: "There are certainly unclear points about this iterative notion of set. What is meant by 'subset' of $[V_\alpha]$? How long does the transfinite process continue?" [48, p. 112]. The latter question points at the underdetermination of the size of the universe by a conception that excludes Replacement (though this type of problem persists even in full ZFC when considering large cardinals).

Given that the iterative conception does not justify Replacement, some might call this a strike against Replacement. Kanamori's attitude seems to be: so much the worse for the iterative conception. Or rather, so much the worse for using it as the sole principle from which to generate a mature set theory.

Kanamori defends his view in [28], offering a history of the Axiom of Replacement with the specified intention of arguing for and highlighting its significance—even centrality—among the axioms of set theory. He aims to push back against the "hesitation and even skepticism [that] have been voiced about the importance and even the need for Replacement" by, e.g., Potter, Boolos, and others. Within his historical account, he cites numerous results that depend on Replacement, in the sense that they are not provable in ZC—for instance, Borel Determinacy, the existence of the ordinal $\omega + \omega$, and of course the theorem of transfinite recursion. In doing this, he compares ZC with ZFC, without attention to other theories that might entail these results.

The main contributions of Replacement, on Kanamori's account, are "functional substitution, recursion, and indifference to identification." The first and last of these draw on the fact that Replacement lets us literally *replace* elements of one set with other existing objects one for one and still be left with a set, allowing us familiar constructions like $\{t_i \mid i \in I\}$ given some specified correlation of elements t_i to members i of an index set I. Thus Replacement allows for a sort of structuralist-style view of mathematical objects such as ordered pairs, real numbers, ordinals, and so forth:

> What Replacement does is to allow for articulations that these representations are not necessary choices and to mediate generally among possible choices. Replacement is a corrective for the other axioms, which posit specific sets and subsets, by allowing for a fluid extensionalism. The deepest subtlety here is also on the surface, that through functional correlation one can shift between tokens

> (instances, representatives) and types (extensions, classes), and thereby shift the ground itself for what the types are. ([28, p. 47])

Some remarks on this are in order. First, the flexibility offered by Replacement is indeed very powerful. To be indifferent among alternative ways to define real numbers, ordinals, and so on, is mathematically quite convenient. But one is much more likely to find this option necessary or even desirable if one antecedently has structuralist leanings. And in the context of set theory, there is hardly any structure at all to sets *in general*, in contrast to, say, groups, well-orderings, or natural number systems. The only structural component in a universe of sets is its relation of elementhood, $\in$. An "isomorphism" between two sets is nothing more than a bijection. Thus a powerful principle like Replacement provides for the existence of sets based solely on their size, the only structural feature that sets in general share. The argument for Replacement based on indifference to identification therefore offers little more than what is offered by a "limitation of size" principle.[11]

But this sort of principle brings with it known difficulties. As discussed in, e.g., [6] and [20, Ch. 5]), there is a tension between limitation of size as a principle of set existence and the iterative conception of set. On the typical characterization, limitation of size says that sets come from classes (or properties) that aren't "too big." If a property is satisfied by a class of elements that is too big, then the elements are just too numerous to be consistently collected into a set. For if they were so collected, we would fall to the paradoxes of naïve set theory. This principle would be a reasonable grounding for Replacement, which the iterative conception fails to justify alone. However, limitation of size is difficult to reconcile with the Axiom of Power Set, as $\mathcal{P}(x)$ can grow enormously large with respect to x, even unpredictably so (see [20, p. 205-208]).[12]

[11] See also [50, p. 231].

[12] An important distinction to keep in mind here is between limitation of size and the related but alternative principle Hallett has called Cantorian finitism. Roughly, Cantorian finitism is the idea that infinite sets should be treated as much as possible in the same way that we treat finite sets. One of the tenets is that any set, even an infinite one, should be conceived as a unit. Another one is extensionalism: sets (even infinite ones)

In connection with limitation of size, one might also try to defend Replacement based on the naïve picture, briefly noted earlier, that one simply ought to be able to take an existing set and exchange (or *replace*) its elements one for one with other members of the universe without sacrificing its set-hood. But ultimately, this amounts just to a restatement of what the Axiom of Replacement says. While similar intuitions have been used before to motivate Separation, Separation is weaker than Replacement, and so it is not obvious why a reason for believing that should thereby be a reason to believe in the truth of Replacement.

In contrast to all these attempts, Kanamori's approach is to push aside the issue of needing a single foundational, pre-theoretic idea behind *Set* that would justify both Power Set and Replacement intrinsically. Instead, he points to Replacement's vast usefulness in practice and accepts these two competing principles together on extrinsic grounds. But for one who still asks for a single pre-theoretic idea, Kanamori's philosophical argument—the quality of his historical account notwithstanding—does not offer much that is new. One is left in the same position as when, in Potter's words, "Even von Neumann himself admitted that the axiom of replacement, just regarded as an axiom without regard to its useful consequences, 'goes a bit too far' " [51, 182].

If one were to try to extract from Kanamori's account an intrinsic reason for believing Replacement, not just extrinsic reasons, then the structuralist-style argument discussed above should really be characterized as not about sets, but classes. To speak of sets as the objects being structurally compared is loose. For sets in set theory are not comparable to, say, groups in group theory; universes are. If we are being precise, the justification should take the form: some classes are "large" and some classes are "small," and Replacement holds because if one class is small and another class is structurally similar (i.e., similarly sized), then the other class should also be small.

consist precisely of their members. (For more on this, see [20, p. 7-9 and 32-40].) With respect to the present discussion, Cantorian finitism can justify the Axiom of Power Set on the grounds that since finite sets always have power sets, so should infinite sets. This sort of argument is independent of arguments based on limitation of size, and so should not be viewed as providing reason for them.

The difficulty is that it is not obvious that "smallness" is a structural property of classes. Classes are usually taken as intensional entities—if they are taken as entities at all. It is not completely clear that one can apply structuralist-style thinking to classes, in particular to draw coherent conclusions about their "size." Furthermore, if one is inclined to be structuralist at all, it is questionable whether set theory is an appropriate foundation to be using in the first place. There are other systems, such as category theory, that fit more harmoniously with structuralist intuitions (see, e.g., [1] and [43]).

As a further issue, Linnebo has argued that structuralism, at least in its stronger forms, is actually inconsistent with the iterative conception of set. The main incompatibility comes from the structuralist theses that individual objects in a structure are dependent on one another and/or dependent on the structure as a whole. If sets are formed according to the iterative conception, he says, then they should be dependent only on their members (and perhaps members of those members, etc.)—not on higher-ranked sets. It is especially odd to say that the singleton set $\{a\}$ depends on anything other than just a, especially anything high up in the hierarchy:

> This asymmetric dependence [of sets upon their members, and not vice versa] is in fact a very good thing, as there are all kinds of difficult questions about the higher reaches of the hierarchy of sets. How far does the hierarchy extend? Are the different stages rich enough for the continuum hypothesis to fail? It would be a pity if very simple sets, such as the empty set and its singleton, depended on the entire hierarchy of sets, and their identities could therefore not be completely known before these hard questions had been answered. But fortunately the situation is the reverse. In particular, we can give an exhaustive account of the identity of the empty set and its singleton without even mentioning infinite sets. [46, p. 73]

Thus, to sum up, Kanamori's attitude towards axiomatic set theory is one that preferences extrinsic justification principles over intrinsic ones. He is not attached to the iterative conception as a sole foundational idea generating the set-theoretic universe. If Replacement is not amenable to intrinsic justification, then those who endorse the axiom (i.e., nearly everyone) must search elsewhere.

Of course, outside structuralism and limitation of size, there have been some recent attempts to revitalize arguments for Replacement based on reflection principles, which are arguably better motivated by the iterative conception. Such principles follow the intuition that the universe of sets is "a very rich structure with no conceivable end" [14, p. 123], too large to be uniquely characterized, hence any property that might be ascribed to V must also be ascribable to some initial segment V_α. Some formalizations of this notion entail Replacement.[13]

But there are difficulties with this approach. Why are reflection principles justified? How do they follow from our concept of set? On Burgess' account, reflection is motivated by the principle, "The xs form a set unless ... they are indefinably or indescribably many" [9, p. 205]. Put differently, the universe of sets is so large and comprehensive that it must be indescribable. But what is meant here by 'indescribable'? The answer given is that "any statement about all of [the members of the universe] continues to hold if reinterpreted to be not about all of them but just about some of them, few enough to form a set." In other words, there is no way to distinguish the universe from its initial segments by means of a first-order sentence that is true of the universe and not its earlier stages. In Drake's words, "it always seems a plausible step, in view of the reflection principle, to take a property of the whole universe ... and postulate that it already holds at some level V_α" [14, p. 124]. But this answer is just a restatement of the principle of reflection, not a justification.

[13]See, e.g., [4] (of which a recent reconstruction is given in [9]), building on [45]. For a history of these, see [26, Sec. 2] and [27].

Another answer is that 'indescribable' means something else. But in that case, it's not clear *what* it means without further explanation. A robust account of indescribability is needed before reflection can properly ground axioms like Replacement. If the notion of indescribability is taken too generally, such as, "not describable or distinguishable by any broadly linguistic means," then the statement of reflection is clearly false: after all, the universe V contains all sets, but its initial segments V_α do not. So the challenge here is to formulate a notion of 'indescribable' that renders the justification of reflection neither question-begging nor false.

An additional hurdle is that some formulations of higher-order reflection are inconsistent. For instance, Koellner offers examples of reflection principles that are proved inconsistent in [31]. Higher-order reflection principles, as noted by Reinhardt, are also inconsistent: "Since it is the presence of the classes as parameters in Bernays' schema which appears to be responsible for its great strength, we would eventually like to continue by formulating a third-order reflection principle which allows third-order objects as parameters. It does not appear to be possible to generalize ... in this direction very directly" [53, p. 196]. While not every statement of reflection is inconsistent, one must meet the challenge of explaining why its justification, whatever that is, provides for the consistent versions and not the inconsistent ones.

Regardless of the status of arguments for reflection, I suggest that the principle of transfinite recursion is so natural and basic that it should not be held hostage to the success of these more ambitious programs. The approach I propose and pursue in the next sections is a new and yet entirely natural approach. Rather than begin with a strong principle such as reflection and use it to justify Replacement, here we begin with notions related to recursion and combine them with other principles to obtain Replacement, or at least fragments of it. So, on this picture, Replacement is not independently justified as a special instance of one larger principle, but rather as a consequence of several principles which we already believe.

Even if this specific proposal does not work out, it seems like a natural and yet hitherto unexplored route: instead of justifying Replacement from 'above' by stronger principles, we try to build up to it from 'below' by a collection of principles with *primae facie* intrinsic support. Even if the specific proposal below encounters problems, it is surprising that this natural idea has not been pursued in the extant literature on the justification of principles like Replacement. Obviously, one of the problems which this proposal encounters is that it must argue that some recursion-like notions are built into our concept of set. This point is addressed further in sections 2.4-2.5. The next section is concerned with making precise the idea of building up from 'below' to versions of Replacement.

2.3 Calibrating Thresholds: Some Reverse Set Theory

Above we have motivated the question of how to justify transfinite recursion and explained why extant proposals are wanting. In this section, we'd like to come to some better understanding of how to formalize transfinite recursion as a single principle and investigate how it relates to other standard principles like Replacement.

Recall that Z is Zermelo's Z—the subsystem of ZFC obtained by removing the Axiom Schema of Replacement and the Axiom of Choice—and that ZC is just Z plus Choice[14]. Further, recall that these theories have obvious second-order variants, wherein the schemes are replaced by single axioms and wherein one adds the full second-order comprehension schema.[15]

[14]We actually will not be appealing to the Axiom of Choice as such, but rather the well-ordering theorem. For convenience, we will abbreviate this by C.

[15]While first-order treatments of set theory are more common, our aim is to formalize principles yielding recursion, which is then used to justify part of Replacement. Recursion is most naturally formulated as a second-order axiom. In future work, I hope to investigate whether these results can be replicated in first-order logic or related systems like Lavine's schematic logic [44]. This would be important because there are at least two well-known concerns with the invocation of second-order logic in the context of set theory. First, some suggest that the standard semantics for second-order logic is already set-theoretically "entangled". (The metaphor of mathematical entanglement is originally from Charles Parsons. For further use of it, see [32]). Second, there is the concern that an appeal to second-order logic is inconsistent with the iterative conception (see [8]).

Definition 1. Let *Ordinal Recursion* (Ω-Rec) be the following axiom in the signature of second-order ZC: for every function $G : V \to V$ there is a function $F : \Omega \to V$ such that for every ordinal α, the class $F \restriction \alpha$ forms a set and $F(\alpha) = G(F \restriction \alpha)$.

This is essentially the theorem of transfinite recursion stated as an axiom. As an attempt to define an axiomatic system that includes recursion without Replacement, Ω-Rec would seem a reasonable place to start. As Proposition 3 will show, though, we can recover Replacement from Ω-Rec, provided we also have the following principle.

Definition 2. Let *Ordinal Abstraction* (Ω-Abs) be the following axiom in the signature of second-order ZC: if X is a set and $(X, <)$ is a well-ordering, then there is an ordinal α and an order-preserving bijection $f : \alpha \to X$.

The axiom Ω-Abs is a way of formally expressing the natural idea that every well-ordered set determines an ordinal. Before Zermelo and Von Neumann, Cantor's conception of ordinal consisted of just an arbitrary representative of the class of well-orderings of a given order-type.[16] The name *Ordinal Abstraction* reflects the idea that we can abstract away from the particularities of the class of ordinals Ω as ordered by the $\in$-relation to any isomorphic well-order. If we believe this, then by Proposition 3 we simply recover ZFC:

Proposition 3. *Against the background of second-order ZC, the following are equivalent:*

1. *The Axiom of Replacement*
2. *The conjunction of Ω-Rec and Ω-Abs*

Proof. First suppose the Axiom of Replacement, so that we are working in second-order ZFC. The usual proof of transfinite recursion from ZFC carries verbatim to show the axiom Ω-Rec holds, and similarly for the axiom Ω-Abs. Second, suppose that we are working in

[16]See, e.g., [18].

second-order ZC plus the axioms Ω-Rec and Ω-Abs. Let $H : X \to V$ be a class function with domain a set X. It must be shown that the range of H is also a set. Let $(X, <)$ be a well-ordering. By the axiom Ω-Abs, choose an order-preserving bijection $g : \gamma \to X$ for some ordinal γ.

Now define $G : V \to V$ as follows. If f is the graph of a (set-) function with domain $\alpha < \gamma$, then set $G(f)$ equal to the union of f with the singleton of $H(g(\alpha))$; otherwise, set G equal to zero. Then apply the axiom Ω-Rec to obtain the $F : \Omega \to V$ associated to G. Then $F \restriction \gamma$ forms a set, so that $F"\gamma$ forms a set, and we claim that $\cup(F"\gamma) = H"X$. Clearly by construction of G and the relation between F and G, we have $\cup(F"\gamma) \subseteq H"X$. Conversely, if $y = H(x)$ where $x \in X$, then since $g : \gamma \to X$ is a bijection, choose $\delta < \gamma$ with $g(\delta) = x$. Then we have that $y = H(x) = H(g(\delta)) \in F(\gamma)$ and hence that $y \in \cup(F"\gamma)$. □

As an alternative to Ω-Rec, we now define σ-Rec, an axiom intended as a weakening of Ω-Rec that only affirms the existence of functions built by adding to the function's graph the suprema of previous values. This is a generalization of primitive recursion on the natural numbers.

Definition 4. Let *Supremum Recursion* (σ-Rec) be the following axiom in the signature of second-order ZC: for all ordinals γ_0 and all functions $G_s : \Omega \to \Omega$ there is a unique function $F : \Omega \to \Omega$ such that

$$F(0) = \gamma_0, \qquad F(\alpha + 1) = G_s(F(\alpha)), \qquad F(\lambda) = \sup_{\alpha<\lambda} F(\alpha) \tag{2.1}$$

wherein λ is a limit.

Remark 5. Note that ZC does not prove σ-Rec. For instance, consider the model $(V_{\omega+\omega}, \in)$, which satisfies ZC. Suppose it were a model of σ-Rec. Then define $\gamma_0 = \omega$ and $G_s(\alpha) = \alpha+1$. Let $F : \Omega \to \Omega$ be the function associated to γ_0 and G_s. Then $F(\omega) = \sup_{n<\omega}(\omega+n) = \omega+\omega$, which is not an element of $V_{\omega+\omega}$. This shows that σ-Rec is not already part of ZC.

Moreover, the system ZC + Ω-Abs does not prove σ-Rec. Let κ be the least fixed point of the $\beth$-function—that is, $\kappa = \beth_\kappa$. Then we can show that $(V_\kappa, \in) \models$ ZC + Ω-Abs+$\neg\sigma$-Rec. To show $(V_\kappa, \in) \models \Omega$-Abs, let $x \in V_\alpha$ for some $\alpha < \kappa$. Then in V there is an ordinal β and bijection $f\colon x \to \beta$. So $|\beta| = |x| \leq |V_\alpha| \leq \beth_\alpha$. Since $\alpha < \kappa$, we have $\beth_\alpha < \beth_\kappa = \kappa$, so $|\beta| < \kappa$. Thus $\beta < \kappa$, so $\beta \in V_\kappa$. Since x and β are both in V_κ, so is f by closure under Union, Pairing, and Power Set.

Now suppose for contradiction that $(V_\kappa, \in) \models \sigma$-Rec. Then there exists a function F given by:

$$F(0) = \omega \qquad F(\alpha + 1) = \beth_{F(\alpha)} \qquad F(\lambda) = \sup_{\alpha<\lambda} F(\gamma).$$

Now we claim that $F(\omega) = \beth_{F(\omega)}$, contradicting the supposition that κ is minimal. For $F(\omega) = \sup_{n<\omega} F(n)$, and hence $\beth_{F(\omega)} = \beth_{\sup_{n<\omega} F(n)} = \sup_{n<\omega} \beth_{F(n)} = \sup_{n<\omega} F(n+1) = \sup_{n<\omega} F(n) = F(\omega)$. This completes the argument.

One can show further that while κ is known to be a singular cardinal (since it is the limit of the ω-sequence $\{\beth_0, \beth_{\beth_0}, \beth_{\beth_{\beth_0}}, \ldots\}$), any regular cardinal κ' for which $(V_{\kappa'}, \in) \models$ ZC+Ω-Abs is inaccessible. One can show this by letting $\lambda < \kappa'$. Then we have $\lambda \in V_{\kappa'}$, so $\mathcal{P}(\lambda) \in V_{\kappa'}$. By C, there is a well-order $(\mathcal{P}(\lambda'), <)$, so by Ω-Abs there is an $\alpha \in V_{\kappa'}$ that is order-isomorphic to the well-order. So $|\mathcal{P}(\lambda)| = |\alpha| < \kappa'$.

Remark 6. Unlike Ω-Rec, the axiom σ-Rec does not imply Replacement in ZC + Ω-Abs (assuming consistency), which we show below in Proposition 12. It is, however, equivalent to what could be called Ω-Replacement:

Definition 7. Let Ω-Replacement, or Ω-Rep, be the axiom asserting that for every α, the range of every function $G'\colon \alpha \to \Omega$ is a set.

Proposition 8. *In ZC + Ω-Abs, the axiom σ-Rec is equivalent to Ω-Replacement.*

Proof. For the right-to-left direction, a similar argument as in the usual proof of transfinite recursion shows the existence of functions defined by σ-Rec using just the fragment of Replacement found in Ω-Rep. For the other direction, suppose that the sentences of $\mathrm{ZC} + \Omega\text{-Abs} + \sigma\text{-Rec}$ are true. Let α be an ordinal and $G' \colon \alpha \to \Omega$. What we must show is that the range of (G') exists as a set. Define $G \colon \Omega \to \Omega$ by letting $G(\delta)$ be the least ordinal above δ in the range of G', or 0 if there are no such. Suppose for contradiction that the proposition were false and $\mathrm{rng}(G')$ were a proper class. By the ZC axioms, the universe is closed under Separation, Union, Pairing, and Power Set, so the function G is increasing unboundedly in Ω (rather than eventually reaching a point γ at which $G(\delta) = 0$ for all $\delta \geq \gamma$). But this would then imply that the following function definable by σ-Rec is strictly increasing unboundedly in Ω:

$$F(0) = \min(\mathrm{rng}(G')), \quad F(\delta+1) = G(F(\delta)), \quad F(\lambda) = \sup_{\delta<\lambda} F(\delta).$$

By Power Set, there exists a cardinality $|\alpha|^+$ immediately above $|\alpha|$. The axiom σ-Rec ensures that $F \restriction |\alpha|^+$ is a set function, and hence so is $F \restriction |\alpha|^+ \setminus \{\text{limits}\}$, the restriction of F to 0 and successor ordinals of size bounded by $|\alpha|^+$. Note that $F \restriction |\alpha|^+ \setminus \{\text{limits}\}$ injects into $\mathrm{rng}(G')$. (The same cannot be said for $F \restriction |\alpha|^+$, since the images of the limit ordinals are not in general in $\mathrm{rng}(G')$.) From this injection one can now obtain a contradiction by constructing an injection from $|\alpha|^+ \setminus \{\text{limits}\}$ to α. Let $H \colon \mathrm{rng}(G') \to \alpha$ be given by $H(\zeta) = \min(\{\theta \mid G'(\theta) = \zeta\})$. Then $H \circ F \restriction |\alpha|^+ \setminus \{\text{limits}\}$ injects into α. □

Definition 9. Let the *Axiom of Ordinals* denote the assertion that V_α exists for every α.

This name is found in Potter's [50, p. 218]. There, as in our setting, the Axiom of Ordinals is defined in a system without the Axiom of Replacement.

Proposition 10. *In ZC + Ω-Abs, the Axiom of Ordinals + σ-Rec + "$\forall x \exists \alpha\, x \in V_\alpha$" together imply Replacement.*

Proof. Suppose these axioms hold. Then let $H : X \to V$ be a class function with domain a set X. We must show that the range of H is also a set. By Ω-Abs, let $g : \gamma \to X$ be an order-preserving bijection for some ordinal γ. So $H \circ g\colon \gamma \to V$. Let $R\colon \mathrm{rng}(H) \to \Omega$ be defined by $R(s) = \mathrm{rnk}(s)$ (the rank of s exists by assumption). So $R \circ H \circ g\colon \gamma \to \Omega$. By σ-Rec and Proposition 8, we have Ω-Rep, so $\mathrm{rng}(R \circ H \circ g)$ exists as a set. Hence its supremum β exists as a set, which means $\beta + \omega$ exists. Thus $V_{\beta+\omega} =_{\mathrm{df}} \bigcup_{\lambda<\beta+\omega} V_\lambda$ exists by the axioms and contains every member of $\mathrm{rng}(H)$. □

We have seen now that σ-Rec entails the existence of many ordinals that otherwise would not necessarily exist, precisely the ones that exist under a weakened version of Replacement restricted to ordinal functions. Since the converse of Proposition 10 is obviously true, these observations suggest a way to characterize what the gap is between σ-Rec/Ω-Replacement and full Ω-Rec/Replacement, from the perspective of ZC + Ω-Abs—namely, the "width" of the universe, as measured by the existence of all V_α (for every ordinal α), which together contain all sets. But given a sufficiently "slim" model, in the spirit of [49], we can show that ZC + σ-Rec + Ω-Abs alone cannot imply Replacement:

Definition 11. For finite α, let $D_\alpha = V_\alpha$. For $\alpha \geq \omega$, let $D_{\alpha+1} = \{x \mid \exists y, y' \in D_\alpha$ s.t. $x = \mathcal{P}(y) \;\vee\; x \subseteq y \;\vee\; x = \bigcup y \;\vee\; x = \{y, y'\}\}$ and let $D_\lambda = \lambda \cup \bigcup_{\alpha<\lambda} D\gamma$.

Proposition 12. *For any inaccessible $\kappa > \omega$, $D_\kappa \models$ ZC + σ-Rec + Ω-Abs + $\neg$Replacement.*

Proof. Since D_κ is closed under Union, Pairing, and Power Set, it is straightforward to see that $D_\kappa \models$ ZC. Further, since κ is inaccessible and all the ordinals of V_κ are in D_κ, we have $D_\kappa \models \Omega$-Abs. Likewise, the regularity of κ guarantees Ω-Replacement and hence also σ-Rec (indeed, σ-Rec follows from closure under suprema). It remains to be shown that D_κ fails Replacement. We argue that for every $x \in D_\kappa$, there exist only finitely many Zermelo ordinals[17] in its transitive closure $tc(x)$; thus $V_\omega \notin D_\kappa$ and Replacement fails. Suppose this

[17]For reference, the Von Neumann ordinal $\alpha + 1 =_{\mathrm{df}} \alpha \cup \{\alpha\}$, whereas the Zermelo ordinal $\alpha + 1 =_{\mathrm{df}} \{\alpha\}$. So, e.g., the set of finite Zermelo ordinals is $\{\{\}, \{\{\}\}, \{\{\{\}\}\}, \dots\}$.

were false. Then let α be the least ordinal such that D_α contains a counterexample x. Note that α cannot be a limit, for at limits the only new element added is a Von Neumann ordinal, which contains exactly two Zermelo ordinals ($\emptyset$ and $\{\emptyset\}$, neither of which is new anyway) and is identical to its own transitive closure. So $\alpha = \beta + 1$ for some β. Thus $x = \mathcal{P}(y)$ or $x \subseteq y$ or $x = \bigcup y$ or $x = \{y, y'\}$ for some $y, y' \in D_\beta$. But since $tc(x)$ contains infinitely many Zermelo ordinals, so must y or y', contradicting the minimality of α. □

To summarize, in this section we have considered two recursion principles as axioms: Ordinal Recursion (Ω-Rec), which is just the theorem of transfinite recursion stated as an axiom, and Supremum Recursion (σ-Rec), a weakened version. We showed that the first of these is nearly strong enough by itself to prove Replacement in ZC; all one needs to add is Ordinal Abstraction, the principle that there is an ordinal corresponding to every well-ordering. Then we showed that σ-Rec is equivalent to a fragment of Replacement, namely Replacement on ordinals. We saw that full Replacement can then be recovered from this weaker recursion principle by further adding "V_α exists for every α" (the Axiom of Ordinals) and the basic assumption that every set appears on some stage. These results collectively provide reason to believe that recursion principles are plausible candidates for foundational axioms of a workable set theory. In the section that follows, we consider a third notion of recursion. Even if it should turn out that these particular formalizations fall short in some way, it is suggested here that some alternate approach within the same spirit should be promising.

2.4 Transfinite Computation and Algorithms Determine Recursions

Having seen the strength and the weakness of Supremum Recursion (Definition 4), we now look at a related and potentially useful notion from Jensen and Karp [25]:

Definition 13 (Jensen, Karp 1971). A function $f : \Omega \to \Omega$ is a *primitive recursive ordinal function* iff it is generated by the following scheme:

- Initial Functions: $P_{n,i}(x_1, ..., x_n) = x_i$ for $1 \leq i \leq n < \omega), F(x) = 0, F(x) = x + 1$
- Case Distinction: $C(x, y, u, v) = x$ if $u < v$, else $C(x, y, u, v) = y$
- Substitution: $F(x, y) = G(x, H(y), y), F(x, y) = G(H(y), y)$
- Primitive Recursion $F(z, x) = G(\bigcup\{F(u, x) \mid u \in z\}, z, x)$.

Further, f is *ordinal recursive* iff it is generated by the above plus:

- Minimization Rule: $F(x) = min\{\xi \mid G(\xi, x) = 0\}$, provided $\forall x \exists \xi \; G(\xi, x) = 0$.

These are clearly intended as generalizations of primitive and total recursion on the natural numbers, respectively. Following this definition, a result is obtained that succinctly characterizes the ordinal recursive functions:

Theorem 14 (Jensen, Karp 1971). *$f : \Omega \to \Omega$ is ordinal recursive iff it is $\Delta_1(L)$—i.e., it is definable by a Δ_1 formula with quantifiers ranging over L.*

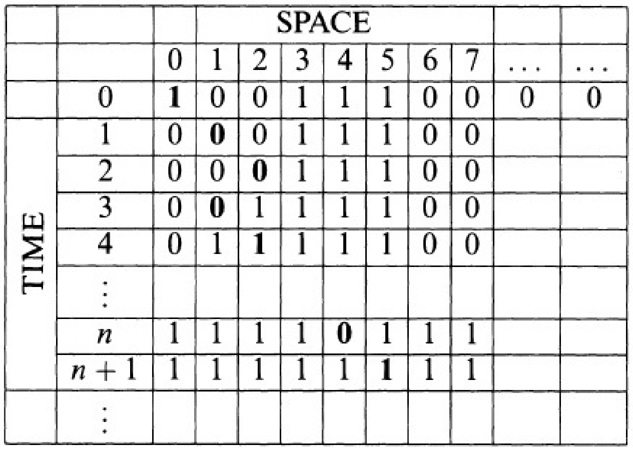
A set-theoretic formula is Δ_1 if it is equivalent to some Σ_1 formula (a formula with only one unbounded existential quantifier) and also to some Π_1 formula (a formula with only one unbounded universal quantifier).

Theorem 14 now gives us two ways of generalizing upon the finitary notion of primitive recursion into a transfinite setting: σ-Rec (Definition 4) and the Jensen-Karp primitive recursive ordinal functions. We also have a definition of *ordinal recursion* that generalizes not just on the finitary notion of primitive recursion but on the full recursive functions of the sort computable by a Turing machine (primitive recursion being of course just a subset of these).

In a recent, rapidly growing body of literature, there have been a number of models of computation generalizing upon Turing machines into a transfinite setting, among which is the *ordinal Turing machine* (OTM) model of P. Koepke. It turns out that by a recent result of Koepke and Seyfferth ([35, Thm. 7]), the notion of ordinal recursive functions coincides precisely with the functions computable by an OTM. That is, a function is ordinal recursive if and only if it is computable by an OTM, given finitely many ordinal parameters. Given our guiding question, "What principles of recursion are available for the transfinite?", such machines are worth considering further.

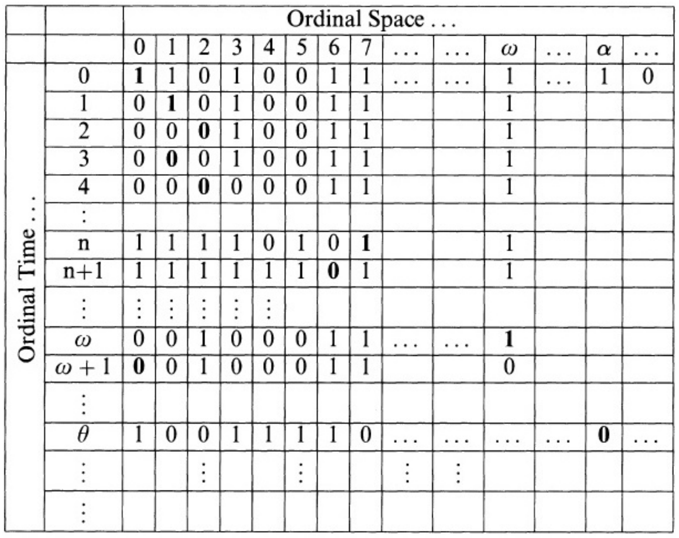
In broadest terms, the workings of an OTM are relatively simple to picture. Rather than computing functions on the natural numbers, as a Turing machine does, it computes on ordinals. The below diagrams from [33] demonstrate the difference. Figure 2.1 depicts a Turing machine computation, with the vertical axis representing time and each row representing the tape content after n number of steps. The bolded number indicates where the head is positioned. In Figure 2.2, a similar representation of an OTM shows the computation continuing through θ many steps and eventually stopping after some further ordinal number (unless the computation is non-halting). A formal definition of how the machine operates is available in [33].

Given the equivalence between ordinal Turing machines and ordinal recursive functions, and in light of our earlier characterization of recursion as a type of "iterated procedure," a new answer to the question of justifying recursion suggests itself: perhaps set-theoretic functions ought to be understood algorithmically. The idea is to identify an intrinsically justified axiom or axiom schema linking the iterative conception of set with algorithms, either in the OTM sense or otherwise. It is natural to picture the process of generating new ranks of the cumulative hierarchy as an essentially algorithmic procedure.



		SPACE									
		0	1	2	3	4	5	6	7	…	…
	0	**1**	0	0	1	1	1	0	0	0	0
TIME	1	0	**0**	0	1	1	1	0	0		
	2	0	0	**0**	1	1	1	0	0		
	3	0	**0**	1	1	1	1	0	0		
	4	0	1	**1**	1	1	1	0	0		
	⋮										
	n	1	1	1	1	**0**	1	1	1		
	$n+1$	1	1	1	1	1	**1**	1	1		
	⋮										

Figure 2.1: A Turing Machine Computation



		Ordinal Space …														
		0	1	2	3	4	5	6	7	…	…	ω	…	α	…
Ordinal Time …	0	**1**	1	0	1	0	0	1	1	…	…	1	…	1	0
	1	0	**1**	0	1	0	0	1	1			1			
	2	0	0	**0**	1	0	0	1	1			1			
	3	0	**0**	0	1	0	0	1	1			1			
	4	0	0	**0**	0	0	0	1	1			1			
	⋮														
	n	1	1	1	1	0	1	0	**1**			1			
	n+1	1	1	1	1	1	1	**0**	1			1			
	⋮	⋮	⋮	⋮	⋮	⋮									
	ω	0	0	1	0	0	0	1	1	…	…	**1**			
	$\omega+1$	**0**	0	1	0	0	0	1	1			0			
	⋮														
	θ	1	0	0	1	1	1	1	0	…	…	…	…	**0**	…
	⋮			⋮			⋮			⋮	⋮				
	⋮														

Figure 2.2: An Ordinal Turing Machine Computation

The idea of identifying algorithms as the source of corresponding functions is not new. In 1971, Kreisel suggested the possibility of privileging calculations over the functions so determined by these calculations:

> Which comes first: the calculations or the functions that result from the calculations? [In the "calculations first" approach], the processes or rules come first, ideally determined by a phenomenological analysis of possible instructions (understandable by the means considered); the functions are defined in terms

of these processes, and one may look for other systems of rules which compute the functions of the same class. Evidently one must expect a possible conflict of interest here: for a complete *description* of possible processes we want lots of rules; for *deciding* some specific question about the computable functions we often want few rules (provided only that they generate all the functions) ([38] p. 151).

Two natural questions present themselves: first, is the claim that algorithms determine functions even formalizable in an axiomatic language? Second, if it is, how do we avoid the "conflict of interest" that Kreisel mentions, namely, the issue of the "completeness" of the axioms?

What Kreisel means by the possible conflict seems to be: in axiomatizations we face the conflicting choice between trying to have a manageable and understandable axiom-set on the one hand, and a complete axiom-set on the other hand. The axioms of Peano Arithmetic are effective and very easy to understand, but they are incomplete. We could, if we wanted, change the system so that the set of axioms is just simply *the set of all true sentences about numbers in the language of number theory.* But this would be a useless axiom system. For starters, the axiom set is not effectively computable; we have no way of knowing from the description whether a given sentence is an axiom or not. In fact, if we did—that is, if we were smart enough to know antecedently of every number-theoretic sentence whether it is true—then we would have no need for an axiom system. Hence the desire for completeness on the one hand, and for usefulness on the other, stand in tension. How should we balance these?

In the case of number theory, the prevailing formal system errs on the side of effectiveness rather than completeness. The same is true for set theory. It is notable that the idea of using the set of second-order validities as axioms for set theory to settle problems like CH has not been seriously pursued—nor has some comparable but weaker system been seriously proposed—because there is no decision procedure to determine which formulas belong within that set. But in the case of computation, we do not have a canonical axiom system to consider.

It turns out that Kreisel's own answer to the original question, "Which comes first... ?" is actually that functions come before calculations. Taking the notion of a recursive function in the finitary setting and looking at transfinite analogues, Kreisel's brand of generalized recursion works its way up the constructible hierarchy and constructs higher recursive functions through definability criteria (see, e.g., [37] and [39]). The literature in this field, such as presented in Sacks' [54], clearly prioritizes functions. But Kreisel is fully aware of the possibility of what he calls *metacomputation*, taking a machine-theoretic approach that prioritizes calculations. He suggests that this possibility is "controversial":

> Here it is to be remarked that, in connection with metarecursion theory, I intentionally excluded any discussion of the notion of 'metacomputation' for conceptual analysis ... There was no need to bring in controversial matters. [38, p. 180]

In light of OTMs and other models of higher computation, surely the idea is far less controversial now. At the time, Kreisel's main challenge was:

> Is there such a thing as a general concept of computation? [O]r the related question: Is there such a thing as an extension of Church's Thesis to general (abstract) structures? ... If the answer is positive, we have to do with the *analysis* of a concept, specifically a phenomenological analysis of the kind given by Turing for mechanical computations of number theoretic functions. [38, p. 144]

To place higher algorithms in a position prior to the functions they define, he says, we must have a conceptual analysis akin to Church's Thesis, with a phenomenological description of *what it is like* to be a computer.[18]

It just so happens that there is now a project being developed in precisely this direction. Though not evidently with Kreisel in mind, Merlin Carl formulates and defends in [10] a version of Church's Thesis that is extended to include transfinite models of computation, specifically the OTM model. He articulates a pre-theoretic concept of higher computation, conducted by what he calls an Idealized Agent Machine (IAM). First he presents a philosophical argument that the IAM model is computationally equivalent to Kitcher's notion of an idealized mathematician. Then he proves formally that an IAM is equivalent to an OTM. The phenomenology of an IAM is described thoroughly:

> At each time, he has a complete memory of his earlier computational activity. Also, he has a working memory where he may store information. We assume that the working memory consists of separate 'places', each containing one symbol from a finite alphabet. The agent is working [according to] instructions that determine his activity ... given by some finite expressions. [10, p. 6]

These descriptions are later on made more precise. Included in the IAM definition is also the ability to look back at the course of past computation and ask questions of it, especially logically complex ones such as, "Does there exist a time such that, for all times after, the value of cell α did not change?" The answers to such questions are critical for taking inferior limits of cell values, which is needed for emulating the operations of an OTM.[19] In the next

[18]Turing's original definition of a 'computer' in [58] depicted a human being, not a machine, following the rote instructions of a program in "a desultory manner". For this reason, Sieg [55] writes 'computor' to indicate a human agent calculating in this way and 'computer' to indicate a machine in the word's contemporary sense. In these terms, Kreisel's challenge is to provide a general concept of computation and an associated Church's Thesis, while also offering an account of the phenomenology of such a comput*or*.

[19]An OTM works similarly to a classical Turing machine at successor ordinal-numbered steps of computation, but at limit ordinal steps the cells' contents must be calculated by taking some sort of limit of previous

section, Carl discusses the agent's perception and use of time and space, and argues that the correct way to understand these in both cases is by considering them as discretized, ordered units, indexed by the transfinite ordinals. From this paper, and the rest of the literature on transfinite computation, we have an answer to Kreisel's once open question:

> Mechanical [i.e., finitary] operations are appropriate (fitting) if the objects on which we operate are given in a finitistic (spatio-temporal) way. ... The question ... concerning the existence of ... a general concept of computation becomes the question whether, on examination, it will be possible to isolate a general notion from the mixture which we associate with the idea of operations proper to (or: implicit in) the nature of the objects, and whether this (hypothetical) notion admits of a useful theory ([38] p. 179-80).

Kreisel claims here that just as we have a solid criterion for evaluating whether we can compute on some given objects in Turing's sense—namely, the objects must be finitary—so too do we need to have an equally solid criterion for whether a given object is appropriate for computing upon in a generalized (transfinite) sense. It is evident that we do now have such a criterion. Corresponding to *finitary/finitistic* in the setting of Turing machines, we now have *set-sized*, or perhaps *ordinal-sized*, in the setting of OTMs/IAMs. And this leads to a natural idea of ordinal computation. (Of course, with Choice there is no great distinction between saying something is set sized or ordinal sized, but the latter puts emphasis on the fact that the objects are ordered, which coheres with Carl's argument that infinitary space and time are best represented by ordinals.)

values, implemented here as the *lim inf*. This means the machine must be able to answer for each cell the Σ_2 question of whether there *exists* a step before the limit such that, *for all* steps after, the cell value was stabilized at 1. If so, the cell's limit value becomes 1, and otherwise (i.e., if it diverged or it stabilized at 0) it is set to 0.

All this is to say that we are in a much better position now to entertain the notion that algorithms determine functions than we were a few decades ago.[20] Before exploring this possibility further, however, I want to address a potential objection. One might suggest that the claim, “Algorithms determine functions,” is suspect because of the “determination” notion contained therein. What does it really mean to say that an algorithm *determines* a function? Consider, though, the analogous principle of Ordinal Abstraction (Ω-Abs) from Definition 2 of section 2.3. This axiom naturally captures the intuitive thought that any well-order determines an ordinal. If we wanted, we could formalize this thought in a way that entirely avoids any “determination” language. In particular, we can recast the determination talk in terms of a function from one thing to another that preserves the relevant properties. This possibility suggests that blanket scepticism about determination talk is misplaced.

So here is a first attempt at formalizing the claim in the particular form, “OTM algorithms determine ordinal functions.” One might simply try to describe, in patently set-theoretic language, the workings of the ordinal Turing machines, using the usual set-theoretic reductions, and use this to write down axioms that effectively say, “If such-and-such OTM algorithm working on the ordinals is total, then there is a corresponding function on ordinals that has the same values.” The difficulty with this proposal is that in defining “halting computation,” we must use transfinite recursion, or at least some significant fragment of it. Defining how the OTM computes cell values after a limit ordinal number of operations requires taking the inferior limits of all prior cell values, and the same is true of the machine’s head position and program state. So if we are working in ZC and looking for a principle to add, we cannot directly express the notion of halting computation without already having access to more recursion than is available in the theory. Not only does ZC not include recursion, it also has models without ordinals $\omega + \omega$ or larger; hence there is no clear reason to believe it makes

[20]Since not every computation halts, it is more precise really to say that *some* algorithms determine functions.

sense to consider a machine with tape or computational run-time that long. In short, ZC is too weak to talk about OTMs.

An alternative approach would be to try to avoid this issue by referring not to machine computations directly, but rather to the complexity of the sets computable by such machines. As mentioned earlier, we have that OTM computability coincides precisely with ordinal recursivity. This comes from Theorem 14 together with the following result:

Theorem 15 (Koepke, Seyferrth 2009)**.** *A set of ordinals is OTM-computable if and only if it is $\Delta_1(L)$—i.e., it is definable by a Δ_1 formula with quantifiers ranging over L.*

This theorem is the natural analogue in the OTM setting of the folklore result that a function on natural numbers is Turing-computable if and only if its graph is Δ_1-definable over the standard model of the natural numbers.[21] Using this result (and the same definition of $\Delta_1(L)$ as after Theorem 14), the proposal is that for each Σ_1-formula $\varphi(x) \equiv \exists\ y\ \varphi_0(x,y)$ and Π_1-formula $\psi(x) \equiv \forall\ y\ \psi_0(x,y)$, we state an axiom of the following form:

$$\begin{aligned}[][\forall x\ (L,\in) \models \varphi(x) \leftrightarrow \psi(x)] \quad \& \quad & [\forall \alpha\ \exists!\ \beta\ (L,\in) \models \varphi(\langle \alpha,\beta\rangle)]] \\ \Longrightarrow \quad & [\exists F\ \forall \alpha,\beta\ F(\alpha)=\beta \leftrightarrow \varphi(\langle\alpha,\beta\rangle)]\end{aligned} \tag{2.2}$$

Unfortunately for this proposal, the antecedent of the axiom contains reference to the inner model L and what holds true and false within it. If we are considering new axioms for ZC, which does not already prove transfinite recursion, then there is no obvious way to express the class L, as L is defined transfinite-recursively. So the same circularity problem appears as before. One attempted response is to suggest we simply replace L by V, for one can easily refer to all sets in V without appeal to recursion. The problem with this suggestion is that we don't know that the resulting axiom schema is actually true.

[21]See, e.g., [56, Ch. 2 and Ch. 4].

Another attempt would be to move the arrow in the conditional to outside the theory. That is, let us consider the theory consisting of all sentences $\exists F\ \forall\alpha, \beta\ F(\alpha) = \beta \leftrightarrow \varphi(\langle\alpha, \beta\rangle)$ such that the two conjuncts in (2.2) above hold. The difficulty is that while this is technically a theory—it is a set of sentences—it is (highly) non-computable, and so it runs us into the second horn of Kreisel's dilemma. Here, then, we have a theory that is not a system you and I can reasonably adopt.

The challenges with expressing formally the idea that algorithms determine functions on the approaches mentioned so far suggest that a different type of strategy is called for. This paper will not develop a formal system that definitively settles the question of how to do this. But one promising idea is to articulate a formal language in which the notion, "OTM-algorithm e halts on input α and returns β," is treated as primitive. The proposal, then, is to develop axioms for this primitive notion that deliver recursion principles such as σ-Rec, Ω-Rec, or similar. It turns out that such a procedure has already been developed in the computability theory literature for Turing computation in [3, Ch. VI], using PCAs (partial combinatory algebras). In the setting of OTMs, we can emulate much of this same method. The merit of this approach is that it circumvents the formal worries above. The downside is that we must find a way to philosophically motivate the primitive.

2.5 Transfinite Algorithms and the Iterative Conception

We have looked at a few basic proposals for how to formalize the notion that algorithms determine functions. These particular attempts may or may not lead us in the right direction, but the project overall is important if our goal is to give an intrinsically justified principle behind transfinite recursion. The hope is that unlike Replacement, recursion can be justified directly from the iterative conception by appeal to the algorithmic nature of the formation of stages in the universe.

This idea has not been deeply explored, but its roots can be found in canonical texts on the iterative conception. In [5], for instance, Boolos invokes what greatly resemble algorithmic notions in his description of the intuitive picture behind the iterative conception. Describing how the stages are formed, he repeatedly uses recursive language:

> At stage one, form all possible collections of individuals and sets formed at stage zero. [...] At stage two, form all possible collections of individuals, sets formed at stage zero, and sets formed at stage one. At stage three, form all possible collections of individuals and sets formed at stages zero, one, and two. At stage four, form all possible collections of individuals and sets formed at stages zero, one, two, and three. Keep on going in this way [...]. [5, p. 221]

The phrase, "Keep on going in this way," appears twice more on the page: once after the discussion of stages $\omega + 1, \omega + 2, \omega + 3, \ldots$, and once after stages $\omega \times 2, \omega \times 3, \ldots, \omega \times \omega, \ldots$. This manner of speaking comes out very naturally in this setting.[22] What is being described is an iterated procedure of set formation—in short, a recursion.

In the later [6], he writes:

> The fact that it takes time to give such a sketch, and that certain sets will be mentioned before others might easily enough be (mis-) taken for a quasi-temporal feature of sets themselves, and one might be tempted to say that sets coming earlier in the description actually *come earlier*, that sets cannot exist *until* their members do, that they *come into being* only after their members do, and that they are *formed* after all their members are.

[22]The phrase chosen later in [6] is "And so it goes," but the meaning is much the same.

> In any case, for the purpose of explaining the conception, the metaphor is thoroughly unnecessary, for we can say instead: there are the null set and the set containing just the null set, sets of all those, sets of all *those*, sets of all *Those*, ... There are also sets of all **THOSE**. Let us now refer to these sets as "those." Then there are sets of those, sets of *those*, ... Notice that the dots "..." of ellipsis, like "etc.," are a demonstrative; both mean: *and so forth,* i.e., in *this* manner forth.

Boolos spells out the connection between sets and recursive procedures even more explicitly in a description of Gödel's views from 1951:

> He then proceeds to lay out the "iterative" or "cumulative" hierarchy of sets: we begin with the integers and iterate the power-set operation through the finite ordinals. This iteration is an instance of a general *procedure* for obtaining sets from a set A and well-ordering R: starting with A, iterate the power-set operation through all ordinals less than the order type of R (taking unions at limit ordinals). Specializing R to a well-ordering of A (perhaps one whose ordinal is the cardinality of A) yields a *new operation* whose value at any set A is the set of all sets obtained from A at some stage of this procedure, a set far larger than the power set of A. We can require that this new operation, and indeed *any set-theoretic operation, can be so iterated,* and that there should also always exist a set closed under our *iterative procedure* when applied to any such operation. [19, p. 291, emphasis added]

In this light, it should not be so foreign to think that recursive algorithms lie at the heart of our very conception of set. Indeed, I believe it is no accident that the 1971 formalization of the conception results in a system producing precisely the recursive ordinals. As recounted in [6]: "In 'The Iterative Conception of Set', I claimed that not even the existence of a stage corresponding to the first non-recursive ordinal is guaranteed by a formalization of the iterative conception and therefore that replacement does not follow from the iterative conception."[23] There are many ordinals that could have been the stopping point of the iterative universe without implying Replacement: $\omega + \omega$, for one. Any ordinal below the least fixed point of the $\beth$-function, or even any accessible cardinal, would have done the trick. But it is telling that the upper bound was instead the first non-recursive ordinal. If recursive algorithmic notions have nothing to do with our conception of set, then this fact constitutes a very striking coincidence.

In the last section, we saw some initial attempts to express formally that (some) algorithms determine functions. The difficulty encountered in that setting, without access to Replacement, is reminiscent of the difficulty one has in trying to express the meta-principle that (some) classes determine sets. The latter is patently true, but also difficult to make formal; it is not expressible in first-order ZFC, for instance, because classes are not referable objects in the theory. As we have seen, the former is similarly difficult to make formal in ZC, because algorithms and equivalent notions seem to be dependent on recursion. Each of these principles is thus a pre-theoretic intuition whose formalization is non-trivial.

There is substantial parallel between the two. If we consider how an algorithm that fails to determine a total function fails, we can see it does so in one of two ways: either it fails to produce a total function because some inputs cause it to loop infinitely, or else because some inputs provoke a never-ending search for some unfulfillable halting condition (e.g., in the finite case, a search for an even prime greater than 2). When a class fails to determine a set, the failure can likewise be characterized in some cases as a kind of looping behavior, and in other cases as a neverending search. The Russell class $R \coloneqq \{x \mid \neg x \in x\}$, for instance, can on this picture be described as failing to determine a set not because it is too large, as in the limitation of size conception, but rather because an algorithmic procedure that attempts

[23]Here Boolos is referring to [5, fn. 13], which says that R_{δ_1} models the given sketch of the iterative conception, where δ_1 is the first non-recursive ordinal (commonly denoted ω_1^{CK}—see [54, p. 10]).

to produce it must loop once it asks whether $R \in R$. The answer depends on whether $\neg R \in R$, which in turn depends on the answer to whether $R \in R$. Any set of instructions we may intuitively imagine giving a transfinite computer that would potentially produce R as a set would only cause the device to loop. A class, on this construal, is a recipe for the construction of a set, and some recipes (algorithms) just don't produce an output.

Another canonical example of a class that doesn't determine a set is Ω, bringing to mind Burali-Forti. The failure to produce a set in this case can be diagnosed as an unending search for an ordinal that does not exist. Because Ω would have to contain Ω, as the well-worn paradox goes, it would have the property $\Omega < \Omega$, which is provably impossible. A transfinite algorithm that attempts to compute a code for such a set would require Ω-many steps to complete the task, and thus cannot do so. By analogy, a Turing machine cannot in finitely many steps produce a code for ω, and any attempt to do so would classically be characterized as "going on forever." The class Ω, by its very nature, takes longer to create than any halting algorithm can possibly run.

If we incorporate into the iterative conception an algorithmic concept of set construction, we are provided with a way to understand why classes can fail to produce sets. The fusion of algorithmic procedures into the iterative conception is, I suggest, a natural and well-motivated way to do this.

Depending on how this can be done, there are many possible recursion principles that could come out of such a theory. If a sufficiently strong principle of recursion is available, then by Proposition 3 in section 2.3, or a similar result, we may derive Replacement as a consequence. Conversely, if only some weaker principle like σ-Rec is available, then the theory will not contain all of Zermelo-Fraenkel set theory ZFC. Still, this would be a significant step forward from existing formalizations of the iterative conception with strength on the order of Zermelo's set theory Z.

Chapter 3

II — The Computational Strengths of α-tape ITTMs

3.1 Background

Over the past few years, interest has grown in the study of models of transfinite computation—theoretical machines that extend classical computibility theory into an infinitary context. A number of distinct models have been devised, typically resembling one of the classical machines (Turing machines, register machines, etc.) but with some modification that takes the operations into the transfinite. For a detailed survey, see [60] or [61]. Of the machines so defined during this recent wave, particular attention has been given to the infinite time Turing machines of [21], which were the first to see print.

Infinite time Turing machines, or ITTMs, are a set of theoretical computing machines similar to classical Turing machines, with the exception that halting computations are not assumed to run for only finitely many steps. Instead, computations are permitted to run for transfinitely many steps before halting. Whereas a classical finitary Turing machine is considered to “fail” in some sense if it does not successfully halt within a finite number of steps, an ITTM may compute through steps $1, 2, \ldots, \omega, (\omega+1), (\omega+2), \ldots, (\omega+\omega), \ldots$ and halt after, say, $\omega^4 + \omega + 17$ steps. It can even halt after a non-recursive ordinal number of steps.

In [21], the machines are conceived as having three tapes: an *input tape*, an *output tape*, and a *scratch tape* intended for calculations. Each machine has a read-write *head* that at any given time occupies the n^{th} cell C_n of each of these tapes simultaneously (the leftmost cell is C_0, and the tapes have ω-many cells extending infinitely to the right). In the present work, we instead imagine just a single tape for input and output, with a parallel tape for scratch work. This type of machine is equivalent.[1]

It will be convenient to use the fact from [22, 2.4] that it is possible to use the single scratch tape of a machine to simulate having a machine with multiple parallel tapes. As described in [33], there are “canonical but tedious” translations of programs on machines with n tapes to programs on machines with one tape. We also recall from [33] that one can use finite binary strings as codes for ‘symbols’, and thus at times may treat the tape as containing a sequence of arbitrary symbols from an alphabet rather than binary bits.

Since an ITTM may run for transfinitely many steps, it has time to read and process infinite-length input, and to write infinite-length output. Hence ITTMs can be understood as computing (partial) functions on the reals ${}^{\omega}2$. (We can still represent a natural number k by a string consisting of all 0s except C_k.) The functions computable by an ITTM form a strict superset of the Turing-computable functions.

To define the machine’s behavior fully, [21] stipulates that on successor ordinal steps, cell values are calculated identically to the way they are determined in a Turing machine, but

[1]Indeed, by [22], it is sufficient to have just *one cell* for scratch work accompanying the input/output tape to guarantee the same capabilities as the three-tape ITTM. Without this, a machine can compute most, though curiously not all, of the functions computable by three-tape machines.

at limit ordinal steps, cell values take on the *lim sup* of their values at previous stages.[2] In the present paper, as in [33] and elsewhere, the *lim inf* is used instead. The difference is immaterial as the resulting models of computation are equivalent. We also set the machine's *head position* and *program state* at the limit stage to their respective *lim inf*s from prior stages.[3]

For ITTMs there are both computable and noncomputable functions $f\colon {}^{\omega}2 \to {}^{\omega}2$. Any recursive function is computable, as is the classical halting problem. An ITTM can solve the latter by running any Turing machine computation in parallel with some program known to halt after ω steps, checking whether the simulated Turing computation has halted or not by that stage. The model's computational power goes far beyond this, however. It is capable of deciding membership in any set of reals up to complexity Π^1_1—in particular, it is able to determine whether a give real $a \in {}^{\omega}2$ codes[4] a well-ordered relation on $\mathbb{N}$ (see [21]). However, all ITTM-decidable sets of reals (indeed, all ITTM semi-decidable sets of reals) fall below complexity Δ^1_2. For an example of non-computable functions, there are ITTM analogs of the classical halting problem, defined below.

While it is clear that the ITTM model is strictly stronger than the finite Turing model, it is natural to ask whether it can be made even stronger. In [21], there is some analysis of extensions of the model using oracles. In [33], the model is extended by lengthening the tape from having ω-many cells to having a proper class of cells (indexed by ordinals $\alpha \in \mathbf{Ord}$). In [12] and [35], machines with fixed α-length tape are considered, with α assumed to be admissible. For these, the assumption is also made that computation duration is limited to

[2]That is, the value of cell n at limit ordinal step α is 0 (respectively, 1) if it had stabilized at value 0 (respectively, 1) by some step $\beta < \alpha$ (in other words, it had that value at step β and did not change after), but if the value of cell n did not stabilize (i.e., it changed its value infinitely often cofinally before step α), then at the limit α it defaults to value 1. If [21] had used *lim inf* instead (as we do here), then it would behave the same except default instead to value 0.

[3]Rather than these stipulations, in [21] there is a special program state designated for limit steps called the *limit state*, and the head position is automatically set to C_0 at limit steps, but it again makes no material difference.

[4]Every real r is considered to *code* the relation $\lhd$ on the set $\mathbb{N}$ given by: $a \lhd b$ iff the $\langle a, b\rangle^{\text{th}}$ digit of r is 1, where $\langle \cdot, \cdot \rangle$ refers to the Gödel pairing function.

be below α-many steps. In analogy with Turing computation, which can be thought of as running on ω-length tapes for some duration below ω-many steps, these machines run with α-length tape for some duration below α-many steps. What results is a theory of transfinite computability in which the computational behavior of the machines bears a relationship to Sacks' α-recursion theory from [54].

In the present paper, the machines under consideration are also of fixed α-length tape, but without restriction of the tape length to any particular type of ordinal and without bound on the potential duration of computations. Rather, the α-machines here will parallel infinite time Turing machines, whose computational run times have no fiat limitation. It will emerge instead as a consequence of these machines' design that, just as all ITTM computations that halt will do so within at most a countable number of steps (Settling Theorem, [21, Thm 1.2]), so too will a machine of length α halt within at most $|\alpha|$-many steps. We allow for inadmissible ordinal-sized machines—indeed, for machines with successor ordinal size—out of a motivation to describe the most general and inclusive class of machines possible, even though it happens that the most interesting results occur with machines of admissible length.[5]

In analogy with ITTMs computing on real numbers (taken as elements of Cantor space ${}^{\omega}2$), a machine with α-length tape computes on α-length binary strings ${}^{\alpha}2$. At times it will be useful to refer to the "hardware" of a machine (the tape and head) without mention of any particular choice of program. We refer to such hardware as a *device*. Thus a device together with a choice of program $p \in \mathbb{N}$ constitutes a *machine*, and a device without mention of any such p can be identified with the class of machines characterized by having a specified tape length. Let T_α denote the device with tape length α. On this notation, T_ω can be understood as the class of ITTMs.

[5]In addition to allowing tapes of non-admissible ordinal size and letting computations run longer than α-many steps, the machines in this paper also contrast from those of [12] and [35] insofar as we do not assume that computations are allowed to have ordinal parameters. This has consequences on our results, particularly those of section 3.2.4.

The behavior of a device T_α can be described as follows. If τ is a successor ordinal, the head position at time τ is defined identically as in the Turing machine case, with one proviso: when the device is instructed to move the head somewhere undefined, its position defaults to C_0. This occurs when an instruction to move *left* occurs while the head resides on a limit ordinal cell, as there is no determinate "next cell to the left" in such cases. The head position would also be undefined if $\alpha = \beta + 1$ and T_α's head is instructed to move *right* while on the rightmost cell β. It is again natural to stipulate in such cases that the head returns to cell 0. Similarly, if α is a limit ordinal and the head is instructed to move *right* for α-many steps on T_α, then the head, having nowhere further to go, defaults to C_0. In general, at limit ordinal times τ the head position is determined by taking the *lim inf* of prior head positions.[6]

If we let $D(\tau)$ denote the configuration of a machine after exactly τ steps of computing (that is, a complete description of the head position, tape contents, program state, and choice of program), then the above completely defines a model of computation whose configuration at all times is deterministically and uniquely given by the initial configuration $D(0)$. When context requires a distinction, we will use subscripts—e.g., $D_{\alpha,p,x}(\tau)$ for the configuration of $\alpha_p(x)$ at step τ. For reference, the description just given of the machines' behavior is formalized below in definitions 3.1-3.1, which resemble definitions found in [33] except for appropriate modifications to 3.1.

Definition A *command* is a 5-tuple (s, c, c', m, s') such that $s, s' \in \mathbb{N}$ and $c, c', m \in \{0, 1\}$. The values of s are called *program states* and the values of m are interpreted as move left/right instructions. A command is to be understood intuitively as instructing a device, "If the current program state is s and the cell content at the current head position is c, then write c' and move left (or right) and change the program state to s'."

[6]That is, the position at step τ is the limit of prior head positions if such is defined, and the leftmost cell among those occupied cofinally often in the steps prior to τ if not.

Definition A *program* is a finite set P of commands satisfying the following conditions:

1. If $(s, c, c', m, s') \in P$, then there is some $(s, d, d', n, t) \in P$ with $c \neq d$.

2. If $(s, c, c', m, s') \in P$ and $(s, c, c'', m', s'') \in P$, then $(c', m, s') = (c'', m', s'')$.

These two conditions together guarantee that for every program state appearing in a program, there exists a unique command informing the device what to do when the cell content at the current head position is 0 and another unique command for when the cell content at the current head position is 1.

Let $\mathit{states}(P) = \{s | (s, c, c', m, s') \in P\}$ be the set of program states in P. Because there are finitely many states, there are countably many possible programs, so we will commonly identify programs by Gödel numbers $p \in \mathbb{N}$.

Definition Let P be a program (coded by p). A T_α*-computation* $\alpha_p(x)$ is a triple $\quad S\colon \theta \to \mathbb{N}, \quad H\colon \theta \to \alpha, \quad T\colon \theta \to {}^\alpha 2 \quad$ such that:

1. θ is a successor ordinal or $\theta = \mathbf{Ord}$. We call θ (and, informally, $\theta - 1$) the *length* of the computation (equally, the *duration* or *run time*).[7]

2. $S(0) = 0$ and $H(0) = 0$. We call $S(\theta)$ and $H(\theta)$ respectively the *state* and *head position* at time θ.

3. If $t < \theta$ and $S(t) \notin P$, then $\theta = t + 1$; the computation *halts* if the machine's state is not a program state of P.

[7]In [21], computations are treated as being able to potentially have a limit ordinal length. In contrast, [33] defines computations in such a way that they may only halt at successor ordinal steps. The present paper uses an approach more similar to that of [33], and while this choice leads to no functional difference in computational outcomes, it does influence statements one can make about run times. For instance, results that are otherwise interesting, such as [21, 8.8] ('No admissible ordinal is clockable') become trivial when computations are defined in such a way that they cannot halt *at* limit ordinals but only one step later. In [36, Def. 3], the definition of *clockable* is modified to address this, but here we instead use 'length' and related expressions to refer to $\theta - 1$. Context will make clear if we ever mean θ instead. In short, we pretend when convenient that we followed [21] in the formalism.

4. If $t < \theta$ and $S(t) \in \mathit{states}(P)$ then $t+1 < \theta$. Let (s, c, c', m, s') be the unique command in P such that $S(t) = s$ and $T(t)_{H(t)} = c$. Then:

   (a) $S(t+1) = s'$

   (b)
$$H(t+1) = \begin{cases} H(t)+1 & \text{if } m = 1 \text{ and } H(t) < \alpha \\ H(t)-1 & \text{if } m = 0 \text{ and } H(t) \text{ is a successor ordinal} \\ 0 & \text{otherwise} \end{cases}$$

   (c)
$$T(t+1)_\xi = \begin{cases} c' & \text{if } \xi = H(t) \\ T(t)_\xi & \text{otherwise} \end{cases}$$

5. At limit ordinal times $t < \theta$, the machine's configuration takes on inferior limits of previous values:

$$S(t) = \liminf_{r \to t} S(r)$$

$$H(t) = \begin{cases} 0 & \text{if } \liminf\limits_{s \to t, S(s)=S(t)} H(s) \geq \alpha \\ \liminf\limits_{s \to t, S(s)=S(t)} H(s) & \text{otherwise} \end{cases}$$

   For all $\xi \in \mathbf{Ord}, T(t)_\xi = \liminf\limits_{r \to t} T(r)_\xi$.

If a program p run on T_α with input $x \in {}^\alpha 2$ halts with output $y \in {}^\alpha 2$, we write $\alpha_p(x) \downarrow y$. We say a partial function $f\colon {}^\alpha 2 \rightharpoonup {}^\alpha 2$ is T_α-*computable* if there is a program p such that, for any input $x \in \mathrm{dom}(f)$ (represented as a binary string), we have $\alpha_p(x) \downarrow f(x)$. For our purposes, if $\alpha_0 < \alpha_1$, we identify $a \in {}^{\alpha_0} 2$ with $a_1 \in {}^{\alpha_1} 2$ if a_1 is identical to a concatenated solely with 0s. Hence even if $\beta \neq \alpha$ we allow some $f'\colon {}^\beta 2 \rightharpoonup {}^\beta 2$ to be considered T_α-computable if 0s are appropriately curtailed or appended. A set $A \subseteq {}^\alpha 2$ is called T_α-*decidable* if χ_A is T_α-computable.

For an example of incomputable functions, consider the undecidable sets $\mathbf{H}^\alpha = \{(p,x) \mid \alpha_p(x)\downarrow\}$ and $h^\alpha = \{p \mid \alpha_p(0)\downarrow\}$. Both sets are undecidable by T_α, by the same argument as in the classical case.[8] These will prove useful toward the aim of comparing the relative strengths of varying T_α, as we will be able to show that one device is computationally stronger than another by showing it is *at least* as strong and is also able to decide the other device's halting problem(s). Formally, we define "strength" by:

Definition We say $T_\alpha \succeq T_\beta$ ("T_α is at least as strong as T_β, with respect to functions on ${}^{\min(\alpha,\beta)}2$") if T_β computes no functions $f\colon {}^{\min(\alpha,\beta)}2 \to {}^{\min(\alpha,\beta)}2$ uncomputable by T_α.[9]

We write $T_\alpha \cong T_\beta$ when $T_\alpha \succeq T_\beta$ and $T_\beta \succeq T_\alpha$. We write $T_\alpha \succ T_\beta$ when $T_\alpha \succeq T_\beta$ and $T_\alpha \ncong T_\beta$. The default way to show that $T_\alpha \succ T_\beta$ will be to show that T_α can emulate any computation on T_β and can also decide h_β.

3.2 Results

3.2.1 Reachability

Note that it does not go without saying that $T_\alpha \succeq T_\beta$ whenever $\alpha > \beta$. Nor does the converse hold. In case $\alpha > \beta$, we provide in Proposition 3.2.1 a necessary and sufficient condition for

[8]The classical analogs of both these halting problems are of equivalent Turing difficulty, so they are often not heavily distinguished, but in the infinitary case they are distinct. The "strong" halting problem $\mathbf{H}^\alpha$ is much more complex than the "weak" problem h^α. See [21] for discussion of these points with respect to $\mathbf{H}^\omega$ and h^ω in particular.

[9]We could have defined $\succeq$ without the restriction to functions $f\colon {}^{\min(\alpha,\beta)}2 \to {}^{\min(\alpha,\beta)}2$, comparing instead how two devices T_α and T_β fare on *all* functions on binary strings. In this case, we would trivially have that $T_\alpha \ncong T_\beta$ whenever $\alpha \neq \beta$, since if (without loss) $\alpha > \beta$, the device T_α can compute functions $f\colon {}^\alpha 2 \to {}^\alpha 2$ not computable by T_β by the brute fact that the input and output strings do not fit on T_β's tape. We find the resulting definition of 'strength' not to capture the notion we are interested in as effectively as the one above. We want, for example, to say $T_\omega \cong T_{\omega+1}$, considering the fact that T_ω and $T_{\omega+1}$ compute precisely the same functions $f\colon {}^\omega 2 \to {}^\omega 2$ (see 3.2.6 below) and the $T_{\omega+1}$-computable functions $f'\colon {}^{\omega+1}2 \to {}^{\omega+1}2$ are not capable of accomplishing anything of mathematical interest not already accomplished by the functions f. Moreover, it would be strange to have a notion of computational strength whereby the addition of *one cell* to an already infinitely long tape causes a meaningful increase of strength.

$T_\alpha \succeq T_\beta$. Later, in Lemma 3.2.6, we provide a sufficient condition applicable for any α and β.

Definition Suppose $\alpha > \beta$. We say the cell C_β is *T_α-reachable* if there exists a program p such that $\alpha_p(0) \downarrow$ with final head position C_β.

To say that a cell is reachable is not to say merely that the head can be placed there. Otherwise, every cell would be 'reachable' in that sense. Instead, reachability refers to a device's ability to be programmed to find a particular cell with its head, recognize that it is there, and then halt—in a sense, to *put its finger* on that cell. As will be seen, some devices have unreachable cells, imposing strong limitations on their computational abilities.

Proposition 3.2.1. Suppose $\alpha > \beta$. Then $T_\alpha \succeq T_\beta$ iff C_β is T_α-reachable. In particular, C_ω is always T_α-reachable for any $\alpha > \omega$, so $T_\alpha \succeq T_\omega$.

Proof. Assume that all computations described here begin with input string 0.

For the forward direction, suppose that $T_\alpha \succeq T_\beta$. Note that T_β, like any other device, can be programmed to fill its entire tape with 1s and then halt. To do this it just needs to print a 1 and then move right, then repeat this process until the value at the current head position is already 1. At this point the routine can stop, since the head must have by now traversed the tape's full length and returned around to cell 0 (because the head always returns to cell 0 whenever its position is otherwise undefined). Thus T_β can be programmed to print a strings of 1s of exactly length β and then stop. Since $T_\alpha \succeq T_\beta$, there must therefore exist a program causing T_α to print a string of 1s of exactly length β. Let it do so, then seek out C_β (the first cell not containing a 1), place the head there and halt.

For the other direction, first have T_α run a routine that reaches C_β, except that the halting command is replaced by an instruction to mark the current location (C_β) with a unique symbol on the scratch tape. Any T_β-computation can now be simulated on T_α by adding a program instruction to return the head immediately to C_0 every time it reads this symbol (i.e., occupies C_β).

To show C_ω is always T_α-reachable for all $\alpha > \omega$, we note that we can repeatedly move the output tape's head to the right, while simultaneously reserving a "flag" cell on the scratch tape that initially has a 1 and then flashes off and then on again each time the head moves. After exactly ω steps, the flag will read 0 by the *lim inf* operation, signaling the machine to halt. □

3.2.2 Settling

In the literature, a computation is said to be *settled* when it has reached a point at which it has either halted or entered into an endlessly repeating loop.[10] It is proven in [21] that all computations on T_ω necessarily settle within countably many steps—thus, if any T_ω computation runs for an uncountable number of steps and has not yet halted, then it never will. A similar result is true in general for all T_α. We state the result without proof, as the argument is essentially identical to [21, 1.2].

Theorem 3.2.2. *(α-Settling Theorem)* For all cardinals κ, computation on any device T_α such that $|\alpha| = \kappa$ will either halt or enter an endless loop within a number of steps of cardinality κ.

This places an upper bound on the maximum run time of halting T_α-computations. For a lower bound, observe that there is a program—call it LoopAround—that halts after exactly α steps when run with input 0 on any T_α. In pseudo-code:

[10]Here 'loop' is taken in a strong sense, namely that the machine is in the same configuration $D_{\alpha,p,x}$ at two different limit ordinal stages *and* no cell with value 1 at the earlier stage ever subsequently changes to 0, a stipulation without which the machine might break out of the repetition at the second limit stage through the *lim inf* operation.

```
        MoveRight;
        WHILE (HeadPosition > 0) MoveRight;
        end;
```

The LoopAround algorithm moves the head to the right exactly α times, then halts. One might notice that in this code there is an implicit assumption that the machine is capable of recognizing when the head is located on C_0. One of many ways to accomplish this is to include, e.g., an instruction at the beginning of the algorithm to print a special symbol on C_0 of a scratch tape, then use this as a reference point during later computation.

Definition An ordinal δ is called *T_α-clockable* (or just *α-clockable*) if there exists a program p such that $\alpha_p(0) \downarrow$ after exactly δ steps.[11] Let $\gamma(\alpha)$ denote the supremum of T_α-clockable ordinals.[12]

The above remarks mean that T_α can clock α, and also that $\gamma(\alpha)$ has cardinality $|\alpha|$ (because there are countably many programs p). Therefore, we have:

Proposition 3.2.3. $T_{\omega_1} \succ T_\omega$.

In fact, we have in general:

Theorem 3.2.4. If C_β is α-reachable and $|\alpha| > |\beta|$, then $T_\alpha \succ T_\beta$.

Proof. Consider the following algorithm.[13] Given a T_β-computation, simulate it on a scratch tape (since C_β is α-reachable, we immediately have $T_\alpha \succeq T_\beta$). Simultaneously, run a

[11]The special case of T_ω-clockable ordinals was first defined in [21], which called them the *clockable ordinals*. The structure of these ordinals is explored in [21] and subsequent work by P. Welch.

[12]The existing literature refers to $\gamma(\omega)$ as γ. We use both here interchangeably, and also let $\gamma^0(\alpha) := \alpha$ and $\gamma^{\beta+1}(\alpha) := \gamma(\gamma^\beta(\alpha))$, with analogous definition for limit $\gamma^\beta(\alpha)$.

[13]An equally effective approach would be to use instead the algorithm in [47, thm. 3], which lets the model of [33] decide $\mathbf{H}^\omega$. By the Settling Theorem, T_{ω_1} is sufficient for the same purpose. Both procedures work by using the extra space on the tape to outlast the simulated computation's maximum possible halting time.

LoopAround clock on another scratch tape. If the T_β-simulation halts, print 'yes'. If it does not halt before the LoopAround does, then print 'no', as α steps have passed and the simulation must be in a loop. This decides the strong halting set $\mathbf{H}^\beta$, hence $T_\alpha \succ T_\beta$ strictly. $\square$

The above procedure could be called the *outlasting* algorithm, as it works by employing the machine's uncountable tape as a sort of clock to outlast the possible halt time of T_ω-computations by the Settling Theorem, analogous to the way that T_ω decides the classical halting problem by outlasting possible halt times of Turing computations. In a certain way, one could accomplish the same thing with a device that is just an ITTM but affixed with a sort of 'oracle clock', an alarm that reads 0 until precisely θ number of steps have passed for some (otherwise unclockable) ordinal θ, then reads 1.

A perfectly natural question that now arises is: is ω_1 the smallest ordinal α such that $T_\alpha \succ T_\omega$? In other words, are all countable devices equally strong? In general, if $|\alpha| = |\beta|$, must $T_\alpha \cong T_\beta$? We provide a negative answer. There is a countable device of strictly greater computational strength than T_ω—namely, $T_\gamma := T_{\gamma(\omega)}$.

Theorem 3.2.5. *$T_\gamma \succ T_\omega$.*

Proof. The outlasting algorithm above decides the weak halting set h^ω on T_γ. $\square$

We show in Corollary 3.2.7 that T_γ is the *smallest* device such that $T_\gamma \succ T_\omega$. We also show that $T_{\gamma(\gamma)}$ is the smallest device stronger than that, and so on (within the constraints of Section 3.2.4).

3.2.3 Writability

The literature on transfinite computation makes frequent use of the property of *writability* in the context of various computational models. Applied in the context of T_α-machines, it has direct consequences on comparisons of devices' strengths.

Definition A real $x \in {}^\omega 2$ is called T_α*-writable* if there exists a program p such that $\alpha_p(0) \downarrow x$. An ordinal δ is T_α*-writable* if there exists a T_α-writable real x that codes a well-ordering of order type δ. Denote the supremum of T_α-writable ordinals by $\lambda(\alpha)$.[14] Let $\lambda = \lambda(\omega)$.

Lemma 3.2.6. If β is T_α-writable, then $T_\alpha \succeq T_\beta$.

Proof. Suppose β is T_α-writable. If $\beta < \alpha$, then it is sufficient to show C_β is α-reachable. First, have T_α write β on a scratch tape, then 'count through' the coded well-ordering $\lhd$ (as in the proof that WO is infinite time decidable from [21, thm 2.2]). Simultaneously, move the head on a second scratch tape to the right. When the count-through stops, halt. The second head is on C_β.

If $\beta \geq \alpha$, then let $f\colon {}^\alpha 2 \to {}^\alpha 2$ be any function computable by T_β. We show f is also T_α-computable. Let p be the program such that $\beta_p(x) \downarrow f(x)$ for all $x \in {}^\alpha 2$. We construct a program p' such that $\alpha_{p'}(x) \downarrow f(x)$. First, print a code for β on a scratch tape. Then, run a routine on scratch tape that is just like p except that each 'move right/left' instruction is replaced by 'move one cell higher/lower in accordance with the $\lhd$-ordering (and to C_0 if undefined)'. When the routine halts, unscramble the output string so that the output is $f(x)$ by writing on the output tape the cell contents of the first α cells under $\lhd$. $\square$

Since there are countably many programs, there are countably many T_α-writable ordinals. The ordinal $\lambda(\omega)$ in particular (denoted λ in [21], [59], and elsewhere) is a large countable

[14]The same notational shortcut from fn. 12 will be adopted here.

ordinal well studied in the literature. It is far larger than the Church-Kleene ordinal ${\omega_1}^{ck}$. Indeed, it is shown in [21] that it is an admissible limit of admissible ordinals and is indescribable by Π^1_1 properties. It is shown that if β is T_ω-writable, then so is any $\beta' < \beta$, so the T_ω-writable ordinals have no gaps and $\lambda(\omega) = \{\delta \mid \delta \text{ is } T_\omega\text{-writable}\}$. By contrast, the T_ω-clockable ordinals do have gaps. If α is countable, the arguments from [21] in support of these observations can be straightforwardly applied to T_α-writable and T_α-clockable ordinals in general. Moreover, it is shown in [21] that $\lambda \leq \gamma$, because any writable ordinal λ', once written on the tape, can be "counted through" in a halting procedure lasting at least λ' steps. This result also straightforwardly generalizes to $\lambda(\alpha) \leq \gamma(\alpha)$.

In [59], it is proven that $\lambda \geq \gamma$, hence $\gamma = \lambda$ (indeed, [59] shows $\gamma^A = \lambda^A$, where A is any oracle). As a corollary, every T_ω-clockable ordinal is T_ω-writable. In light of Lemma 3.2.6 and the remarks above, this establishes the earlier claim that T_γ (which is identical to T_λ) is the smallest device strictly stronger than T_ω:

Corollary 3.2.7. For all $\alpha, \beta < \gamma$, we have $T_\alpha \cong T_\beta$.

Proof. This follows directly from Lemma 3.2.6 together with the fact that $\gamma = \lambda = \{\delta \mid \delta \text{ is } T_\omega\text{-writable}\}$. □

Remark Some readers might find it a surprise that the smallest tape length to confer an increase of strength above T_ω is γ, as opposed to either ω_1, being the smallest higher cardinal, or ω_1^{ck}, the smallest non-recursive ordinal. In [29] and [30], however, it is shown that γ behaves in certain ways as the analogue of ω_1^{ck} in the context of transfinite computation. For instance, when a set (called $\mathcal{O}^+$) is constructed analogously to Kleene's $\mathcal{O}$ except with the role of Turing computation replaced by ITTM computation, the resulting tree has height γ. Moreover, just as ω_1^{ck} is the first ordinal not describable by recursive properties, so too is γ the first ordinal indescribable by ITTM-decidable properties ([59][3.8]).

Corollary 3.2.7 can be broadened to say that if $\lambda(\alpha) = \gamma(\alpha)$ holds for some countable α, then for any T_α-clockable ordinal β we have $T_\alpha \cong T_\beta$. Does the equivalence $\lambda(\alpha) = \gamma(\alpha)$ hold for all countable α? It will be seen that it holds for $\alpha = \gamma(\omega)$ and for much larger choices of α, making good on the claim following theorem 3.2.5 that $T_{\gamma(\gamma)}$ is the smallest device such that $T_{\gamma(\gamma)} \succ T_\gamma$ etc. For that matter, since the writable ordinals are gapless and T_α can always clock α with a LoopAround, in view of Lemma 3.2.6 one can see that a countable T_α for which the $\lambda(\alpha) = \gamma(\alpha)$ condition is true is equivalent to a corresponding oracle ITTM device (that is, a ${T_\omega}^A$ for which A contains a code for α). Such devices are already relatively well understood (see [21]), so the set of these T_α can be equally well understood. Since these T_α can write a code for α, we can think of a so-called "γ jump" operation for them that corresponds precisely with the transfinite Turing jump operator $^\triangledown$ defined in [21]—that is, $T_{\gamma(\alpha)} \cong T_{\alpha^\triangledown}$ and $T_\omega \prec T_{\gamma(\omega)} \prec T_{\gamma(\gamma(\omega))} \prec \ldots \prec T_{\gamma^\omega(\omega)} \prec T_{\gamma^{\omega+1}(\omega)} \prec \ldots$.

However, it turns out that the equivalence $\gamma(\alpha) = \lambda(\alpha)$ does not hold in general for all countable T_α. We show below that the parallel between T_α machines and oracle ITTMs breaks down at some stage.

3.2.4 Incommensurability

Once it is seen that devices can increase in strength as their tapes become longer, it is normal to expect that a longer tape in general results in a device that is always at least as strong as the smaller device, and sometimes stronger. That is, given a hardware device, one would naturally expect that adding cells onto the tape would make it no worse off. So if $\alpha < \beta$ and you can compute some function with T_α, then you can also compute that function with T_β. As hinted earlier in the paper, however, this is not so. We do not have that $T_\alpha \succeq T_\beta$ either; we are able to show that two devices can have mutually incomparable strength.

Theorem 3.2.8. (Incommensurability) There exists $\mu < \omega_1$ such that neither $T_\mu \preceq T_{\omega_1}$ nor $T_\mu \succeq T_{\omega_1}$— that is, T_μ and T_{ω_1} have incommensurable strength.

Proof. Since there are only countably many programs, there are only countably many T_{ω_1}-reachable cells. Let $\mu \coloneqq \sup\{\mu^* \mid \mu^* \text{ is } T_{\omega_1}\text{-reachable}\}$. By 3.2.1, we do not have $T_{\omega_1} \succeq T_\mu$. To be explicit, the constant function that sends every input to a μ-length string of 1s is T_μ-computable but not T_{ω_1}-computable.

At the same time, there is a slice of $\mathbf{H}^\mu$ undecidable by T_μ but decidable by T_{ω_1}, namely the set $\{p \mid \mu_p(a) \downarrow\}$, where a is a μ-length string of 1s. It is T_μ-undecidable for the usual (classical diagonal) reasons, and T_{ω_1}-decidable by the outlasting algorithm from thm. 3.2.4 (after using the string a as reference to reach and then emulate T_μ). $\square$

The above shows the counterintuitive fact that it is possible for a smaller device to have some capability not possessed by a larger device. Indeed, the proof given above applies equally well to any countable $\mu' \geq \mu$, so one can say that of the devices shorter than T_{ω_1}, it is not just some but "almost all" of them that are not commensurable with T_{ω_1}. All this said, the reader may find the machine's ability to print a μ-length string of 1s to be a relatively unimpressive feat for T_μ to hold over T_{ω_1}. Of what utility is a μ-length tape filled with 1s? It would be more remarkable if a small device could instead be shown, e.g., to write a real x not writable by a larger device. Later we show such an example.

Having seen that T_{ω_1} is not at least as strong as all smaller devices—indeed, as *most* of them—one is naturally drawn to ask whether this phenomenon is possible strictly between two countable devices. That is, if α and β are both countable, is it possible for T_α and T_β to be incommensurable? Or are all countable devices commensurable, forming a monotonically increasing hierarchy of strength as tape length increases? We have the following answer:

Proposition 3.2.9. There are countable α and β such that neither $T_\alpha \succeq T_\beta$ nor $T_\alpha \preceq T_\beta$.[15]

Proof. We argue that there is a countable elementary substructure V' of the universe V in which there exist two incommensurable devices (just as in V), and then show by absoluteness that in V, these devices, which we know are countable, must be incommensurable still.

Let $\alpha_p(x){\downarrow_\theta}[H{=}\xi]$ denote that if T_α is running program p on input x then it will halt *in θ many steps and with head position ξ*. We begin by observing that $V \models \phi(\mu, \omega_1, \omega_2)$, where $\phi(\xi, \alpha, \beta)$ stands for the below Δ_0 formula for the statement, "$\xi < \alpha < \beta$ and C_ξ is not reachable on T_α." The formula actually states that for some limit ordinal θ_1, cell ξ is not reachable on T_α *before step* θ_1 and T_α must halt or enter a loop (in the sense of fn. 10) no later than this step, which implies that C_ξ is unreachable:

$$\xi{<}\alpha{<}\beta \;\wedge\; \exists\theta_1,\theta_2{\in}\mathrm{Lim} \quad (\theta_1{<}\theta_2{<}\beta \;\wedge \forall p{\in}\mathbb{N}\; \forall\tau{<}\theta_1 \neg\alpha_p(0){\downarrow_\tau}[H{=}\xi] \;\wedge\; \mathit{looping}(\alpha, p, 0, \theta_1, \theta_2)),$$

where $\mathit{looping}(\alpha, p, 0, \theta_1, \theta_2)$ stands for

$$D_{\alpha,p,0}(\theta_1){=}D_{\alpha,p,0}(\theta_2) \wedge \forall\theta'(\theta_1{<}\theta'{<}\theta_2 \rightarrow (C_\delta(\theta_1){=}1 \rightarrow C_\delta(\theta'){=}1))).$$

All of the notions abbreviated in the above by shorthand are defined by recursion from Δ_0 notions, and are therefore absolute for transitive models of $ZF - P$ (c.f. [41, IV 5.6]). Thus we have, e.g., that $H(\omega_3) \models \phi(\mu, \omega_1, \omega_2)$.

Now let M^* be a countable elementary substructure of $H(\omega_3)$. So for some $\nu^*, \alpha^*, \beta^* \in |M^*|$, we have $M^* \models \phi(\nu^*, \alpha^*, \beta^*)$. As M^* is well founded, take a Mostowski collapse $\pi\colon M^* \longrightarrow M$ into a transitive model M. Now for some $\nu, \alpha, \beta \in |M|$, we have $M \models \phi(\nu, \alpha, \beta)$. Since this formula is absolute for transitive models of $ZF - P$ such as M, we have that $V \models \phi(\nu, \alpha, \beta)$.

[15]In conversations before and during the CiE 2012 conference, Joel Hamkins and Robert Lubarsky independently stated this observation, in both cases supporting it by similar argument sketches. The proof in this paper follows the general strategy sketched out in these discussions and fills in some further details. Thanks are owed to them as well as to Sean Walsh and the anonymous referee for helpful conversations and insightful comments.

By construction, the formula $\phi(\nu, \alpha, \beta)$ guarantees that there exist countable limit ordinals $\theta_1 < \theta_2 < \beta$ such that T_α cannot reach cell ν before step θ_1 and will enter a loop in any computation performed on input 0 still running at θ_1. Since T_α is in a loop, it will never escape and halt on C_ν. □

Since even countable devices can be incommensurable, the hierarchical "jumping" structure described at the end of the previous section continues through the countable ordinals up to a certain point, after which the trend must stop. At what ordinal does this happen? That is, how large is the smallest device T_δ that cannot reach all its cells, i.e., cannot emulate all devices beneath it? This is currently an open question, though it seems as though it should not be difficult. In proposition 3.2.10, it is established that at the very least this ordinal is larger than ζ (defined in [21] and below).

Definition If a T_α-computation $\alpha_p(x)$ is in a loop while its tape content y is unchanging, we write $\alpha_p(x) \uparrow y$. If there exists a program p such that $\alpha_p(0) \uparrow y$, then we say that y is *eventually* T_α*-writable*. An ordinal is *eventually* T_α*-writable* if it is coded by an eventually α-writable real. Denote the supremum of eventually α-writable ordinals $\zeta(\alpha)$.[16] Let $\zeta = \zeta(\omega)$.

Remark Like the writable ordinals, the eventually writable ordinals have no gaps. It is clear from the definition that we have $\lambda \leq \zeta$, and in [21] it is shown that $\lambda < \zeta$. We can assert the generalization $\lambda(\alpha) < \zeta(\alpha)$ by much the same argument. It is known that just as λ is very large and possesses a significant number of strong closure properties, the same is all the more true for ζ.

Proposition 3.2.10. Let δ be as above. Then δ is larger than ζ.

Proof. By the previous remark, it suffices to show that T_ζ can write a real coding ζ, since this implies that T_ζ can write codes for all smaller ordinals and thus reach all its cells

[16]Let the same shortcut be used for ζ as for λ and γ.

by Proposition 3.2.1 and 3.2.6. We code ζ by first simulating the universal T_ω machine described in [59]. Its behavior is to run a nonhalting computation that prints increasingly large ordinals (until inevitably it enters into an endless loop). After ζ steps, the tape content has a code for ζ. While this T_ω universal machine simulation is running, the device T_ζ can simultaneously run a ζ-clock with LoopAround, and halt the simulation as soon as the clock halts. □

We conclude with the promised demonstration that there is a real unwritable by a (countable) device that is writable by a smaller device.

Proposition 3.2.11. There exist countable $\alpha < \alpha'$ such that there is a real x that is T_α-writable but not $T_{\alpha'}$-writable.

Proof. We know that there exists a countable device that is unable to write a code for its own tape length (for the ability to do so implies the ability to write codes for all smaller ordinals, hence to emulate the computations of all smaller devices by 3.2.6). Let T_δ be the smallest such device. Note that for any device, an ordinal θ that is both clockable and eventually writable must be writable: simply run an algorithm that eventually writes θ while constantly checking the output tape to see whether its content can be "counted down" for a number of counts equal to the duration of a simulated θ-clock. Thus $\lambda(\delta) < \delta$, because $\lambda(\delta)$ is not writable but is eventually writable (c.f. [21][3.8]). By construction, $T_{\lambda(\delta)}$ can write a code for $\lambda(\delta)$ but T_δ can't. □

Chapter 4

Conclusion

I would like to recap briefly what has been done so far in the last two parts of this dissertation. Then, I will explain the connection between the two parts and describe some of the available avenues for further research not already described in their concluding sections.

In Part I, exploration of the existing justifications for set-theoretic recursion found them wanting. The iterative conception, limitation of size, reflection, Replacement, and structuralism all fail to give us everything we want in a justification for recursion principles. Accordingly, recursion itself was postulated as a viable foundational principle. Several candidate versions were proposed and some formal results were proven. To develop a reason for believing recursion principles, algorithms were then suggested as the source. Just as classes can be said to determine sets (some of them, at least), so too can algorithms be said to determine functions (again, the ones that are well behaved). This idea was put into context next to comments once made by Kreisel on the necessary conditions for giving calculations (algorithms) primacy over functions. It was suggested that the recent swath of literature on transfinite computation, together with conceptual advances such as Carl's transfinite version of the Church-Turing thesis, ought to mitigate Kreisel's challenge. Finally, some formalizations of the claim that algorithms determine functions were offered, but the difficult problem of finding the right formal system is far from settled. Part I concluded, however, that the proposal to incorporate algorithms into the iterative conception of set is really quite a natural one, arguably already at the core of Boolos' and Gödel's intuitions about how sets are formed. Hence the idea is well worth continued pursuit in further research.

In Part II, transfinite Turing machine devices were evaluated according to their relative computational strengths as their tape lengths varied. It was found that increasing the tape length even a countable number of cells is sufficient in some cases to provide an increase in strength. In other cases, at small enough countable infinities, adding cells to the tape has no effect on strength. However, there were also shown to be cases in which two devices could be incommensurable in strength (even two countable ones), in the sense that each can compute a function that the other can't. This counterintuitive result is a consequence of the exclusion of ordinal parameters briefly mentioned in the Introduction and in Part II, footnote 5.

To explain this further, ordinal parameters can be likened to an oracle that provides a code for a finite set of ordinals (this could just be a string of 0s save for some 1s in those cells C_β for which β is in the set). A function is said to be *computable with finitely many ordinal parameters* by a device if there is a program that computes the function with help from such an oracle. It is common in the literature on machines with α-length tape to include ordinal parameters in the notion of computability, or to consider both computability with and without parameters. If parameters are allowed, it is not difficult to see that reachability concerns are no longer at play; for one can always search for any given cell on the tape if there is a parameter for it. This gives the machines a great deal more power and means that no two devices would be computationally incommensurable.

One immediate place to begin further inquiry is to allow for parameters in T_α computations. Call the resulting devices U_α. Given an admissible α, the strength of U_α would compare to that of the α-machine model in [35] in the way that the strength of ITTMs compares to that of Turing machines. The α-machines in [35] (a special case of which is the OTM, when Ω takes the place of α) are essentially T_α machines, except that ordinal parameters are allowed, computation duration is stipulated to be bounded above by α, and α is assumed to be admissible. By setting an upper bound on computation duration equal to the tape length α, the α-machines resemble Turing machines, which also have an equal bound in both dimensions—namely ω. Let us label these types of devices *square*, or *symmetric*. In contrast, ITTMs are *rectangular*, or *asymmetric*, in that the computations may run for more steps than the tape is long. While T_α machines are rectangular like ITTMs, one could argue that U_α machines would make better candidates for generalizing ITTMs to larger tapes, just as α-machines with parameters make for better generalizations of Turing machines than without. The work done here on T_α machines can be thought of as the null case (with parameter set $\emptyset$—or equivalently, with parameters consisting only of reachable ordinals).

This is not the only thing one can do to the definition of the machine resources to create alternate mathematically interesting models of computation. Another way to modify the machines is to change the rules of operation that they follow when reaching a limit ordinal computation step. The procedure stipulated so far is that the cell value take the $\liminf$ of their previous values, but there is nothing sacred about this stipulation. For example, under the current rule, if a particular cell's values over time reads (01011011101111...) the $\liminf$ operation would set the cell to value 0 at step ω, even though intuitively the cell is converging to value 1. In [62], Welch has general results concerning what happens to the computational strength of ITTMs when limiting rules other than the standard $\liminf$ rule are used. A possible avenue of continued research is to consider the same question for devices other than the ITTM model.

There are certainly innumerable ways to vary the machines under consideration. Hence there are innumerable open questions to answer on the absolute and comparative strengths of the resulting models of computation. Further, there are questions not related to computational strength directly, but to surrounding issues such as computational efficiency and computational complexity ($\mathbf{P} = \mathbf{NP}$, space usage, etc.) addressed for the ITTM context in [47], [13] and elsewhere that are worth addressing for other classes of machines, such as the ones discussed here.

The above comprises a partial list of further technical questions worth pursuing. Philosophically, I believe they have at least an indirect, if not direct, relevance to the project advanced in Part I. While only the OTM is considered there, results concerning variations of the models inform us about one another. One can see, for example, that the inclusion of ordinal parameters is not dispensable for the purposes of foundational set theory, as then the issue of reachability becomes a real hindrance on the resultant recursion principle(s) generated by such a weakened OTM device.

Further, the preliminary answers to the philosophical questions in this thesis are by no means the full story, as Part I made clear. Hence in spite of Carl's suggestion that OTMs are the canonical model of transfinite computation in the way that Turing machines are canonical for finite computation, it could well turn out that some other model better suits our purposes. I don't necessarily consider this possibility likely, but it would be premature to rule it out. These points, together, motivate not only the work on the above open technical questions, but also the original choice to study the T_αs. Independently of their inherent mathematical interest, all of these models have potential to contribute meaningfully to the justification of recursion principles in foundational mathematics.

Bibliography

[1] S. Awodey. An answer to Hellman's question: 'does category theory provide a framework for mathematical structuralism?'. *Philosophia Mathematica*, 12(1):54–64, 2004.

[2] J. A. Barrett and W. Aitken. A note on the physical possibility of ordinal computation. *British Journal for the Philosophy of Science*, 61(4):867–874, 2010.

[3] M. J. Beeson. *Foundations of Constructive Mathematics*, volume 6 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, Berlin, 1985.

[4] P. Bernays. On the problem of schemata of infinity in axiomatic set theory. In G. H. Müller, editor, *Sets and Classes on The Work by Paul Bernays*, volume 84 of *Studies in Logic and the Foundations of Mathematics*, pages 121 – 172. Elsevier, 1976.

[5] G. Boolos. The iterative conception of set. *Journal of Philosophy*, 68(8):215–231, 1971.

[6] G. Boolos. Iteration Again. *Philosophical Topics*, 17:5–21, 1989. Reprinted in [7].

[7] G. Boolos. *Logic, Logic, and Logic*. Harvard University Press, Cambridge, MA, 1998. Edited by Richard Jeffrey.

[8] J. P. Burgess. Reviews. *The Journal of Symbolic Logic*, 50:544–547, 6 1985.

[9] J. P. Burgess. *E pluribus unum*: Plural Logic and Set Theory. *Philosophia Mathematica*, 12(3):193–221, 2004.

[10] M. Carl. Towards a Church-Turing-thesis for infinitary computations. *Preprint*, 2013.

[11] M. Carl, T. Fischbach, P. Koepke, R. Miller, M. Nasfi, and G. Weckbecker. The basic theory of infinite time register machines. *Arch. Math. Log.*, 49(2):249–273, 2010.

[12] B. Dawson. Ordinal time Turing computation, 2009. Phd thesis, Bristol.

[13] V. Deolaliker, J. D. Hamkins, and R. Schindler. P$\neq$NP$\cap$co-NP for infinite time Turing machines. *J. Logic Comput.*, 15(5):577–592, 2005.

[14] F. R. Drake. *Set Theory: An Introduction to Large Cardinals*, volume 76 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1974.

[15] J. Earman. *A Primer on Determinism*. D. Reidel Publishing Company, 1986.

[16] J. Earman and J. D. Norton. Forever is a day: Supertasks in pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science*, 60(1):22–42, Mar. 1993.

[17] G. Etesi and I. Nemeti. Non-Turing computations via Malament-Hogarth spacetimes. *International Journal of Theoretical Physics*, 41(2):341–370, 2002.

[18] K. Fine. Cantorian Abstraction: A Reconstruction and Defense. *The Journal of Philosophy*, 95(12):599–634, 1998.

[19] K. Gödel. *Kurt Gödel, Collected Works, Volume III: Unpublished Essays and Lectures.* Oxford University Press, USA, May 2001.

[20] M. Hallett. *Cantorian Set Theory and Limitation of Size*, volume 10 of *Oxford Logic Guides.* The Clarendon Press, New York, 1984.

[21] J. Hamkins and A. Lewis. Infinite time Turing machines. *The Journal of Symbolic Logic*, 65(2 (Jun., 2000)):567–604, 2000.

[22] J. D. Hamkins and D. E. Seabold. Infinite time Turing machines with only one tape. *Math. Log. Q.*, 47(2):271–287, 2001.

[23] L. Horsten. *The Tarskian Turn.* The MIT Press, Cambridge, 2011.

[24] T. Jech. *Set Theory.* Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. The Third Millennium Edition.

[25] R. B. Jensen and C. Karp. Primitive recursive set functions. In D. S. Scott, editor, *Axiomatic Set Theory*, volume XIII (Part 1) of *Proceedings of Symposia in Pure Mathematics*, pages 143–176, Providence, Rhode Island, 1971. American Mathematical Society.

[26] A. Kanamori. Levy and set theory. *Annals of Pure and Applied Logic*, 140(1):233–252, 2006.

[27] A. Kanamori. Bernays and set theory. *Bulletin of Symbolic Logic*, 15(1):43–69, 03 2009.

[28] A. Kanamori. In praise of replacement. *Bulletin of Symbolic Logic*, 18(1):46–90, 03 2012.

[29] A. Klev. Extending Kleene's O using infinite time Turing machines, 2007. ILLC Master of Logic Thesis.

[30] A. Klev. Infinite time extensions of Kleene's O. *Archive for Mathematical Logic*, 48(7):691–703, October 2009.

[31] P. Koellner. On Reflection Principles. *Annals of Pure and Applied Logic*, 157(2-3):206–219, 2009.

[32] P. Koellner. Strong logics of first and second order. *Bulletin of Symbolic Logic*, 16(1):1–36, 03 2010.

[33] P. Koepke. Computations on ordinals. *The Bulletin of Symbolic Logic*, 11(3 (Sep., 2005)):377–397, 2005.

[34] P. Koepke and R. Miller. An enhanced theory of infinite time register machines. *Lecture Notes in Computer Science: Logic and Theory of Algorithms*, 5028:303–315, 2008.

[35] P. Koepke and B. Seyfferth. Ordinal machines and admissible recursion theory. *Annals of Pure and Applied Logic*, 160(3):310 – 318, 2009.

[36] P. Koepke and B. Seyfferth. Towards a theory of infinite time Blum-Shub-Smale machines. In S. Cooper, A. Dawar, and B. Löwe, editors, *How the World Computes*, volume 7318 of *Lecture Notes in Computer Science*, pages 405–415. Springer Berlin Heidelberg, 2012.

[37] G. Kreisel. Set theoretic problems suggested by the notion of potential totality. *Infinitistic Methods*, pages 103–140, 1961.

[38] G. Kreisel. Some Reasons for Generalizing Recursion Theory. In R. Gandy and C. Yates, editors, *Logic Colloquium '69*, volume 61 of *Studies in Logic and the Foundations of Mathematics*, pages 139–198. North-Holland, Amsterdam, 1971.

[39] G. Kreisel and G. E. Sacks. Metarecursive sets. *Journal of Symbolic Logic*, 30(3):318–338, 09 1965.

[40] K. Kunen. *Set theory : an introduction to independence proofs*. Studies in logic and the foundations of mathematics. Elsevier science, Amsterdam, Lausanne, New York, 1980. Fifth impression, 1992.

[41] K. Kunen. *Set Theory: An Introduction to Independence Proofs*. Elsevier Science Publishers B.V., 1980.

[42] C. Kuratowski. Une méthode d'élimination des nombres transfinis des raisonnements mathématiques. *Fundamenta Mathematicae*, 3(1):76–108, 1922.

[43] E. Landry. Category theory as a framework for mathematical structuralism. *The 1998 Annual Proceedings of the Canadian Society for the History and Philosophy of Mathematics*, pages 133–142, 1999.

[44] S. Lavine. Skolem was Wrong. Unpublished. Dated June, 1999.

[45] A. Lévy. Axiom schemata of strong infinity in axiomatic set theory. *Pacific Journal of Mathematics*, 10(1):223–238, 1960.

[46] Ø. Linnebo. Structuralism and the Notion of Dependence. *The Philosophical Quarterly*, 58(230):59–79, 2008.

[47] B. Löwe. Space bounds for infinitary computation. *Logical Approaches to Computational Barriers*, pages 319–329, 2006.

[48] D. A. Martin. Review of [52]. *The Journal of Philosophy*, 67(4):111–114, 1970.

[49] A. R. D. Mathias. Slim models of zermelo set theory. *J. Symb. Log.*, 66(2):487–496, 2001.

[50] M. Potter. Set theory and its philosophy, 2004.

[51] M. D. Potter. Iterative set theory. *The Philosophical Quarterly*, 43(171):pp. 178–193, 1993.

[52] W. Quine. *Set Theory and Its Logic.* Revised edition. Harvard University Press, Cambridge, 1969.

[53] W. N. Reinhardt. Remarks on Reflection Principles, Large Cardinals, and Elementary Embeddings. In T. Jech, editor, *Axiomatic Set Theory*, pages 189–205. American Mathematical Society, Providence, 1974.

[54] G. Sacks. *Higher Recursion Theory.* Perspectives in Mathematical Logic. Springer-Verlag GmbH, 1990.

[55] W. Sieg. Church without dogma: Axioms for computability. 2005.

[56] R. I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer, Berlin, 1987.

[57] J. Tappenden. Mathematical concepts and definitions. In P. Mancosu, editor, *The Philosophy of Mathematical Practice*, pages 256–275. OUP Oxford, 2008.

[58] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[59] P. D. Welch. The length of infinite time Turing machine computations. *Bull. London Math. Soc.*, 32:129–136, 2000.

[60] P. D. Welch. Characteristics of discrete transfinite time Turing machine models: Halting times, stabilization times, and normal form theorems. *Theoretical Computer Science*, 410(4–5):426–442, 2009. Computational Paradigms from Nature.

[61] P. D. Welch. Discrete transfinite computation models. *Computability in Context*, pages 371–410, 2011.

[62] P. D. Welch and S.-D. Friedman. Hypermachines. *The Journal of Symbolic Logic*, pages 620–636, 2011.