Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Mathematics and Computation 178 (2006) 8–24
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

The case for hypercomputation

Mike Stannett

Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, United Kingdom

Abstract

The weight of evidence supporting the case for hypercomputation is compelling. We examine some 20 physical and mathematical models of computation that are either known or suspected to have super-Turing or hypercomputational capabilities, and argue that there is nothing in principle to prevent the physical implementation of hypercomputational systems. Hypercomputation may indeed be intrinsic to physics; recursion 'emerges' from hypercomputation in the same way that classical physics emerges from quantum theory as scale increases. Furthermore, even if hypercomputation were one day shown to be physically infeasible, there would still remain a role for hypercomputation as an organising principle for advanced research.

Keywords: Hypercomputation; Super-Turing machine; Theory of computer science; Models of computation; Philosophy of mathematics; Temporal structure; Natural computation

1. Introduction

For the purposes of this article *hypercomputation* refers to the idea that formal systems can be constructed, and physical systems identified, designed, constructed or exploited, that have capabilities beyond those of the Turing machine. Hypercomputation typically refers to systems that can compute non-recursive functions, but one also talks of *super-Turing* systems, which do not necessarily compute anything non-recursive, but which nonetheless outperform Turing machines in terms of complexity or other measures. In general, however, the terms 'hypercomputational' and 'super-Turing' tend to be used interchangeably, with different disciplines showing a slight preference for one term over the other.

I hope to convince you that there is nothing in mathematics or physics to prevent the implementation of such systems. But in a sense this is a secondary issue, for even if we accepted hypercomputation as having no basis whatsoever in physical reality, it is nonetheless an eminently useful *logical* idea, which offers a more comprehensive model of mathematical, physical and biological processes than its merely computational counterpart. To borrow MacLennan's phrase [1], standard recursion-based computability on its own cannot satisfy the real and pressing need for models with 'orthogonal notions of power', especially insofar as computation by

E-mail address: M.Stannett@dcs.shef.ac.uk

0096-3003/$ - see front matter
doi:10.1016/j.amc.2005.09.067

biological and other bounded physical systems are concerned. Computer science has applications in fields as diverse as concurrent multimedia, financial decision making, the distributed behaviour of foraging ants, and the modelling of abnormal heart function, and it is too much to expect of a single model that it should be ideally adapted to so many different application domains—indeed, hypercomputational models have in some cases specifically arisen as by-products of the drive for better application-oriented models. Though we will not consider them in depth, the concept of hypercomputation also underpins important philosophical arguments, including such central questions as the nature of consciousness and intelligence, questions that have relevance and meaning regardless of the physical status of hypercomputation [2–6].

The suggestion that real-world processes exist that cannot be simulated by standard Turing machines is almost as old as computation theory itself; even Turing suggested that human intelligence requires more than the standard version of computation described by his machines [7,8] (but see [9] for an opposing interpretation of Turing's comments). However, a suggestion is not the same as a fact, and it is important to test the many claims that have appeared in the literature, both for and against hypercomputation. I should explain at the outset that I regard hypercomputation as a physically meaningful working hypothesis, and live in hope of its one day proving exploitable, but I nonetheless find many arguments in support of the idea to be less than convincing. This is unfortunate, for it allows opponents of hypercomputation to focus on specific 'non-examples', and draws attention away from those models which may indeed support the physicality of hypercomputation. Some models—for example, infinite time Turing machines—are very open about the need for 'embedded infinities', and indeed celebrate it, but for others it seems to be a serious impediment. As various authors have pointed out [10,11], some hypercomputational models that appear at first sight to be finite actually rely on embedded infinities to acquire or represent values, or in the way users measure outputs. For example, they may rely on an oracular component that 'just happens' to compute a non-recursive function, without explaining how this component is to be prepared. In other cases, nothing about the model explains how the non-recursive behaviours generated are to be accessed. For example, consider the following simple argument that real-world non-recursive values exist: assuming space to be continuous, almost every point in space is at a non-recursive distance from us, because only countably many of the uncountably many real numbers are computable. While it undeniably follows that almost all real-world distances are non-recursive, this is insufficient for our purposes because it does not necessarily follow from this argument (though it does from others, see Section 3.3) that we can *identify, design, construct or exploit* such distances.

But before regarding such problems with the preparation and measurement of systems as irremediable setbacks, we need to reflect on their underlying assumptions. For example, consider measurement and observation. Does computation actually have to involve measurement? It does not; all we require is that the output of one process should be suitable for use as input to the next—there is no need to 'measure' the output. What do we mean by observation? Standard notions of observation are 'loaded', because they are recursive *by definition*; scientific method requires replicability, and this in turn requires the processes underlying observation and measurement to be rule-driven. Suppose we design a computational system using recursive design tools; must the computations carried out by the system also be recursive? The answer is *no, they do not* (Section 3.3.1).

As these questions and answers illustrate, there are assumptions on both sides of the argument. Those in favour of hypercomputation can sometimes be too quick to assume that their models do not include hidden infinities, while those against hypercomputation are sometimes too quick to assume that what they consider to be the ingredients of computation are in some sense 'necessary'. If computation is our theme, we consider it to be a theme with many variations; as Hogarth [12] astutely observes, it is as meaningless to consider recursion theory more fundamental than hypercomputation as it is to consider Euclidean geometry to be more fundamental than its non-Euclidean counterparts. Rather, all models of computation are relevant in their own terms, and by studying their differences and similarities we can hope to learn more about the fundamentals of computation itself.

1.1. Outline of paper

Section 2 investigates various physical models of hypercomputation, including models based on quantum theory and general relativity. We argue that these models are for the most part eminently feasible; certainly,

there is nothing in the standard model that can logically contradict them. Section 3 extends the discussion to include various mathematical models of computation that are thought to have super-Turing capabilities. Section 4 summarises the main points of the paper, and draws some conclusions.

2. Hypercomputation and physical systems

Although computational behaviours are primarily physical in nature, the discipline that principally lays claim to their study, theoretical computer science, has long regarded computation as an essentially mathematical enterprise. The relationship between computation and physics is in consequence fairly poorly understood, though it is increasingly recognised that the benefits of understanding the physical nature of computation can flow in both directions. Svozil [13], for example, suggests using the Church–Turing thesis as a guiding principle for physics, while Kelly [14] demonstrates that the scheme that many already regard as the established 'guiding principle' for physics—the use of inductive inference to generate theories from experimental observations—is exactly paralleled by the formalist reasoning underpinning Putnam's trial-and-error machines [15] and Gold's limiting recursion [16] (see Section 3.1). In this section, we will examine the other side of the coin. What can physical computation systems tell us about the real-world relevance of the Church–Turing thesis?

While Turing always thought of his 'machine' as having physical as well as theoretical form [17,10], Cleland [18] reminds us that Turing's model of computation was not constructed to be a model of *general* computation. Rather, mathematics in the early twentieth century was in a quandary. In order to avoid the paradoxes inherent in Cantor's new set theory, Hilbert redefined mathematics to be the study of formal systems, but this change of paradigm inevitably introduced a new problem. Formalism, as the paradigm became known, would only make sense as an all-embracing model of mathematics if the validity or otherwise of any claim in formalised mathematics could itself be decided formally, and it is this problem of finding an appropriate decision procedure that Hilbert dubbed the *Entscheidungsproblem*. This is the problem tackled by Church and Turing in their famous papers [19,20], and it was in this context that Turing developed the conceptual machine now named after him, to model the processes at the heart of formalism and formal reasoning. This is reflected in the structure and dynamics of the Turing machine, for just as the formalist agenda requires the application of carefully identified logical rules, one after another, so the Turing machine requires the execution of a formally constructed sequence of formally defined, essentially atomic, transitions. In contrast, natural computation has nothing whatever to do with modelling the procedures that underpin formalism; the goal is instead to find models of computation that best reflect computation as we find it in nature. Not surprisingly, therefore, natural computation models need not agree with those of recursion theory.

2.1. Philosophy of mind

The idea that brains might behave non-recursively has been brought into focus in recent years by Penrose's work in the area, but the general principle goes back at least as far as Lucas [21], who argued from Gödel's first incompleteness theorem that human beings could not be Turing machines. Indeed, Gödel [22] also believed that his incompleteness results entailed the falsity of computationalism (the view that human persons are essentially Turing machines). Penrose took up the argument, but his first presentation, *The Emperor's New Mind* [2], was widely considered to be flawed, and his subsequent presentation [3] has again been the subject of sustained formal refutation [9]. The current status of Penrose's argument is, therefore, unclear.

However, while Penrose's critics refute his arguments, they do not necessarily dispute his conclusions. Bringsjord, for example, refutes Lucas and Penrose's Gödelian arguments [23]. He nonetheless holds that computationalism (the view that human persons are Turing machines) is wrong, and has advanced many arguments [24] that minds are hypercomputational: mathematical expertise cannot be a wholly recursive construct [25]; minds engaged in certain types of logical thinking must be hypercomputational [9]; irreversibility requires that cognition is not computation [26]. Certainly, we cannot simply take it for granted that human reasoning is recursive. For as Bringsjord and van Heuveln point out [27], 'arguably the pivot around which philosophy of mind revolves, these days, is whether or not we *are* Turing machines'. Assuming that minds are physically generated, it is clear that the physicality of hypercomputation cannot simply be rejected out of hand; rather it is a question that underpins much of the philosophy of mind.

2.2. Analog neural networks

It is well known that metaphors for brain behaviour have changed over the centuries to reflect the prevailing preoccupations of contemporary investigators. At the present time, the metaphor we most frequently encounter is that of *brain-as-computer*, and it is in this respect that we can examine the computational properties of neural networks in general, and the mature human brain in particular. *Artificial* neural networks have been studied in computer science since the 1940s [28], but it is only in the last decade that the super-Turing potential of analog networks has become the subject of sustained scientific scrutiny [29–31].

It is often assumed that electronic computers are real-world examples of Turing machines in action, but this is an illusion. As Davis points out [10, p. 196], '[a] crucial and often ignored aspect of this relation is that while the abstract theory is involved essentially with the mathematical infinite, physical computers are necessarily finite objects.' In some senses, however, the argument goes the other way. Turing machines are very specifically *discrete* conceptual devices, and we cannot avoid the fact that real computers are analog devices operating in an analog world. It makes good *industrial* sense, therefore, to ask ourselves how standard conceptual models change if we introduce the physically necessary assumption that even digital systems are analog.

Siegelmann and others have addressed this issue in the field of neural networks by developing a mathematical theory of analog neural networks. Following Kleene's demonstration [32] in the 1950s that finite discrete neural networks with hard thresholds are no more powerful than finite automata, researchers have typically used continuous activation functions and non-integral weights. Taking the weights to be rational, a finite network can simulate universal Turing computation [33,34], and allowing real weights gives super-Turing potential [29]. As Davis [10] points out, however, this super-Turing potential comes about only when weights are allowed to take non-recursive values, for if we insist that a network be initialised with recursive real weights, and that the update functions are also recursive, then the network can only ever display recursive behaviour. It might be questioned, therefore, whether analog neural networks can offer any scope for hypercomputation, for it appears that we only get super-Turing outputs if we first supply super-Turing inputs. However, the issue for us is not what can be computed using a *mathematical model* of analog neural networks, but what can be computed using a *real* one. I can think of no a priori reason to assume that biological networks use Turing-computable update functions, but let us assume for the sake of argument that they do. Then taken together, Siegelmann and Davis' arguments tell us that real networks might well behave hypercomputationally, but only if update computations are inexact: if updates are computed exactly as decreed by the associated recursive functions, the outcomes will again be recursive. But the one thing we know from quantum theory—and some of the processes that operate in biology certainly involve agents small enough for quantum effects to be relevant—is that no update function could *ever* be exact. If the 'recursive specification' calls for the new weight on a connection to be w, all we can really say is that the result of the update will be a new weight drawn from a distribution whose *expected value* is w.

Ultimately, therefore, the hypercomputational status of biological neural networks must rest on the nature of physical observation. Is there some *recursive observation principle* (ROP) in nature that ensures that observations are always recursive; if an experiment produces an observed value drawn from a non-trivial continuous distribution with recursive mean, must that value necessarily be recursive? If not, then brain behaviours are as likely (arguably more likely) to be hypercomputational as not. In fact, the validity of ROP is extremely doubtful. First, physics has so far got along very well without imposing ROP, so Ockham's razor would suggest we reject the principle. Second, if ROP *were* in effect, this would presumably imply the existence of some hidden-variables model of quantum state, contrary to current understanding of quantum theory. Third, ROP is unlikely on statistical grounds—the probability of selecting a *recursive* real from a non-trivial continuous distribution is zero—and the idea that physics might 'just happen' to pick a recursive value is as unconvincing as hypercomputational systems that 'just happen' to pick non-recursive oracles. Overall, then, it seems rather likely that physical systems in general (not just biological ones like brains) should be capable of hypercomputational behaviours—the larger the components of the system, the smaller the effects of quantum uncertainty, and the more the world will appear to be recursive; conversely, reducing component size ought to increase the observability of physical hypercomputation.

I suggest the relationship between recursive and hypercomputational systems exactly parallels the relationship between classical and quantum systems. Just as the classical behaviours we observe in daily life emerge

from lower-level quantum behaviours, so their apparently recursive structure emerges from lower-level hypercomputation. Ironically, proponents of hypercomputation[1] have usually argued in the opposite direction, giving the impression of hypercomputation as a kind of 'limit' of recursive behaviours. The truth, I suggest, may well be the other way round; hypercomputation is the physical norm, and recursion the artefact.

2.3. Field computation

It is traditional to think of neural networks as 'brain-like' structures, but in practice the number of neurones in an artificial net, and the number of interconnections between them, pales into insignificance when compared to the organisation of biological neural networks. It is simply not feasible to build an artificial net comparable to the 1 trillion neurons and 100 trillion synapses of the mature human brain [35]. In the early 1990s, MacLennan [36] proposed treating such massively distributed computations in a different way; rather than model each neuron as a separate structure, he represents their inherent information content as points in a *field*. Computation then amounts to field transformation. However, MacLennan's focus is not so much *hyper*computation as the need for *alternative* models of computation. As we noted at the start of this section, the Turing machine and its counterparts were developed to address questions concerning formal constructibility; they are eminently well suited to that task. MacLennan notes that similar principles have underpinned Western epistemology for millennia: 'scientific knowledge must be expressed in a verbal calculus and processed by means of it'; as far back as Socrates, it has been held that ideas that cannot be verbalized are 'mere experience' [37].

This attitude has an unfortunate consequence, for it means that our models are almost inherently recursive by design. We start with basic atomic ideas and put them together using finite methods to construct more elaborate descriptions of reality. This precludes the construction of continuous constructs, which remain inherently 'uncomputable'. By definition, a model resembles the thing modelled, but equally, models possess properties that are *not* shared by the systems they describe. MacLennan has examined the properties of Turing machines in detail—and finds that they do not match up particularly well with the properties of physical systems he wishes to model. Moreover, the types of question addressed within recursion theory can be fairly irrelevant to natural computation. For example, many recursion theorists are interested in questions of computational complexity, but there is no point determining the best algorithm for a problem under the assumption that resources can be increased indefinitely, when we know at the outset that the systems we are modelling are inherently bounded, and indeed cannot exceed the rather small volume of a mature human skull. Rather, we need to model natural systems as we find them. For the types of natural system studied by MacLennan, this means that Turing machines are inappropriate; they do not adequately represent the continuous processes that occur in our analog world, where 'all information representations are continuous (i.e., they are drawn from continuous spaces). Naturally, continuous quantities can be approximated by discrete quantities, but we must be aware of modeling artifacts resulting from the process of approximation, especially when we are investigating fundamental properties of computation. The more direct—and safer—approach is to use continuous models from the beginning. A discrete approximation is adequate only if it does not alter the phenomena of interest' [1].

2.4. Quantum theory

Although the nature of quantum information has been debated for many decades, the first recognisable descriptions of quantum computers appeared in the 1980s. Benioff's analysis [38–40] indicated from the outset that quantum computers would be at least as powerful as Turing machines. Feynman [41,42] considered a slightly different problem; rather than model universal computation, his interest was universal *simulation*; could he find a system that would reliably simulate any other physical system? The two questions are closely related: given that computers are physical systems, Feynman's simulator would ultimately be a universal computer as well. Deutsch' 1985 paper [43] describing a universal quantum computer (UQC) is widely considered to have launched the current trend in quantum computing, but as Steane points out [44], Deutsch' design is not

[1] Myself included.

universal in the strict sense, because it cannot simulate all other computers with no more than polynomial slow-down.

Deutsch' UQC is essentially a register machine, where each of the finitely many registers is a 2-state quantum system; it is therefore a finite-state system and cannot compute anything non-recursive. Nonetheless, quantum computation is well known to have super-Turing tendencies, in the sense that certain tasks can be performed significantly faster. This first became evident in the mid-1990s, although the idea that quantum theory could be used to out-perform Turing machines had always been one of the key factors motivating their investigation. Following quickly on Simon's [45] description of a highly efficient quantum algorithm, Shor [46] described his now famous algorithm for factorising large integers. Although Deutsch's UQC construction offers no scope for true hypercomputation, the possibility remains that other models of quantum computation may do so. Stannett [47,48] argued in the early 1990s that Deutsch' model is overly restrictive, in that even very simple quantum systems exist in a superposition of infinitely many states. For example, the electron in a neutral Hydrogen atom can occupy any one of infinitely many energy levels. By exploiting the infinite dimensionality of quantum systems, Kieu [49–52] has recently demonstrated an explicitly hypercomputational quantum algorithm. His algorithm, based on the quantum adiabatic theorem, provides a solution to Hilbert's tenth problem (concerning Diophantine equations), a problem which is known to be undecidable by recursive means [53].

2.5. General relativity

Current interest in relativistic models centres on Hogarth's suggestion [54–56] that general relativity might support the implementation of supertasks (Section 3.1).[2] Hogarth's scheme relies on the existence of a certain type of space–time singularity [57]; it is known that certain solutions to Einstein's equations (*Malament–Hogarth space–times*) support the existence of such singularities. If S is a singularity of the type envisaged, it will have the following effect on space–time. If an object follows a trajectory that falls into the singularity, its journey takes forever as measured by a clock moving with the object, but external observers perceive it to have taken only a finite amount of time to complete its journey into S. Secondly, no matter how close an object comes to S, it can always emit material that can escape from S.

Let C be a computing device capable of simulating Turing machines, and suppose C is equipped with a small rocket that can be ejected programmatically. Let O be an observer, and M a complete description of a Turing machine (including input tape and program). The observer O sets C to work simulating M, and then sends it on a journey into the singularity; from the observer's point of view, the device is calculated to reach the singularity at some time t. The observer then sets off for some location P, being sure to arrive later than t. If at any point during the execution of M, the device determines that the computation has terminated, it calculates an appropriate trajectory and sends the rocket along that trajectory so that it arrives at P before O does. Having arrived at P the rocket waits there until O arrives.

This system clearly solves the (Turing) Halting Problem. If O arrives at P to find a rocket there, the simulation of M must have terminated. If there is no rocket, the simulation ran forever. Indeed, the system can be used to solve the halting problem for other types of computation, recursive or otherwise, whose temporal structure can be embedded in $\mathbb{R}$ (see Section 3.1.3). Whatever type of computation we wish to analyse, we simply replace D with a device capable of simulating computations of that type.

However, the scheme is not without its problems, and its physicality is in dispute. Earman and Norton [55,57,58] have studied the proposal in detail, and have identified several significant problems. Nonetheless, Hogarth's scheme offers an intriguing perspective on the relationship between general relativity and hypercomputation.

Setting the issue of physicality to one side, Hogarth's study of singularity-based hypercomputation has prompted him to consider hierarchies of machines, and to investigate their properties. An SAD_1 machine is a singularity-exploiting system of the type just described. A similar space–time structure can be used to chain

[2] The word 'supertask' is possibly misleading in this context. The computing device in Hogarth's scheme never performs more than a finite number of tasks in any finite period of time, as measured by itself, so no supertask is involved. Nonetheless, external observers may *perceive* the completion of a supertask.

together an infinite string of SAD_1 computations; this gives an SAD_2 computation. Having defined SAD_n, we chain an infinite string of SAD_n computations together to form an $SAD_{(n+1)}$ computation. Finally, we form an AD (arithmetic deciding) machine as an infinite chain of the form $TM \to SAD_1 \to SAD_2 \to SAD_3 \to \cdots$.

The SAD hierachy is closely related to the arithmetical hierarchy. For each n, let Σ_n (respectively, Π_n) be the set of relations expressible by a series of n alternating quantifiers, beginning with $\exists$ (respectively, $\forall$), and acting upon a recursive relation. For example, $S(a,b) = \exists x \forall y \exists z.\ F(a,b,x,y,z)$ is a statement in Σ_3. Earman and Norton showed [58] that SAD_1 machines could not decide arbitrary relations with two quantifiers; Hogarth [12] extends this result, showing that SAD_n machines can decide $\Pi_n \cup \Sigma_n$ but not $\Pi_{(n+1)} \cup \Sigma_{(n+1)}$, and that arithmetic is decidable by AD-machines. Nonetheless, AD machines are not omniscient; there are functions that AD machines cannot compute.

3. Mathematical models

In this section, we extend the discussion to include various mathematical models of computation that are known to have super-Turing properties, and consider the extent to which they might be physically realised. However, even where they are deemed to be physically infeasible, we argue that these models have their own intrinsic *mathematical* significance, in that they provide a research platform for meaningful mathematical investigation.

3.1. Supertasks, limit-computations and extended models of time

Supertasks [57,58], limiting recursion [15], accelerating machines [59], trial-and-error machines [16], infinite-time Turing machines (ITTMs) [60,61] and inductive machines [62] are closely related to one another, and the easiest way to distinguish between them is probably to consider the nature of *time* that is presupposed during computation.

3.1.1. Time as ordinal ω_0

The dynamics of Turing machines require time to be represented as the ordinal ω_0 (the set $\mathbb{N}$ carrying its usual ordering and topology). The same is true of supertasks and accelerating machines. The difference between these models rests on the distinction between observer and observed. As observers watch the machine executing its instructions, they also experience time, and there is no a priori reason why their time should agree with that of the machine. Nonetheless, the fact that they *are* observers means that there must exist some injection *observe*: $T_m \to T_o$ mapping machine time into observer time. A machine operates *sequentially relative to* T_o if the injection *observe* is order-preserving. *Concurrent* executions must also preserve ordering, but in this case *observe* need not be a function.[3] In each of the following examples, we take our own version of time (as always) to be $T_o = \mathbb{R}$, and require *observe* to be an order-preserving function.

Standard interpretations of the Turing machine require $T_m = \omega_0$, and insist on *observe* being co-final. That is

$$\sup\{observe(t) | t \in T_m\} = \infty. \tag{1}$$

Supertasks and accelerating machines specifically *ban* co-finality, but they are otherwise identical to the standard TM model. These models require

$$(\exists t_o \in T_o) \sup\{observe(t) | t \in T_m\} = t_o. \tag{2}$$

As Shagrir points out [63], Turing-machines with this temporal structure are no more powerful than standard Turing machines. This is in fact obvious, because there is no way that the machine can 'know' how *observe* embeds T_m within T_o; the temporal embedding is not part of the Turing machine model.

[3] If two independent events occur at the same time in T_m they may be recorded at *different* times in T_o. This scenario does not arise if execution is sequential.

3.1.2. Time as ordinal $\alpha > \omega_0$

When we write a program to compute the decimal expansion of π, we intuitively think of the program producing ever better approximations; in other words, each step of the execution replaces the last solution with a better one. This is precisely the behaviour of a trial-and-error or a super-recursive machine. At no point do we require the computation to signal that it has finished; it is always free to 'change its mind' as to its eventual output. Colloquial usage also assumes that it is meaningful to talk about the program having written the 'entire' expansion if we let it run 'forever'. That is, we envisage a time t_o later than every *observe*(t), and ask ourselves what state of the machine will be observed at that time. The answer is that *no state* will be observed at that time, because there is no t_m satisfying *observe*$(t_m) = t_o$. Trial-and-error machines overcome this problem by introducing such a time into the model's temporal structure. Taking $T_m = \omega_0^+ = \omega_0 \cup \{\omega_0\}$, the machine behaves just like a standard Turing machine at each model time $t < \omega_0$, but the observation at ω_0 itself is defined so as to represent the 'limit' (in some sense) of all the observations that preceed it. Formally, we have to envisage some operator *config*:$T_m \to$ *Config*, where *Config* is a configuration space equipped with suitable topology. This function assigns a machine configuration at each machine-time n, and we require

$$\mathit{config}(n) \to \mathit{config}(\omega_0) \text{ as } n \to \omega_0.$$

In other words, *config* must be a *continuous* function with respect to the given topologies. Clearly, there is no reason why we should stop at ω_0^+, and it is obvious that we can extend this type of computation to higher transfinite ordinals [60]. However, we cannot do so indefinitely if we continue to use $\mathbb{R}$ as our model of observer time. As Kwiatkowska and Stannett [64] observed when using ordinals to model extended true-concurrent behaviours, any ordinal that can be embedded in $\mathbb{R}$ with its order intact must be countable, whence the ordinal-based computations of the kind discussed here are only meaningful if the ordinals in question are countable.

3.1.3. Time as $\mathbb{R}$ and [0, 1]

For analog computation, it is usual to take $T_m = \mathbb{R}$ or $T_m = [0,1]$, but it is sometimes forgotten that these choices are not equivalent. Because these spaces carry non-discrete topologies, it is usual to place additional *continuity constraints* on the computations taking place, reflecting the assumption that the behaviour of analog computers varies continuously; that is

$$(\forall x \in T_m)\ (\forall \text{ nets } \langle x_\alpha \rangle \text{ in } T_m)(\mathit{config}(x_\alpha) \to \mathit{config}(x) \text{ as } x_\alpha \to x).$$

For sequential systems we can reasonably assume that observation is also continuous, so that

$$(\forall x \in T_m)\ (\forall \text{ nets } \langle x_\alpha \rangle \text{ in } T_m)(\mathit{observe}(x_\alpha) \to \mathit{observe}(x) \text{ as } x_\alpha \to x).$$

Since $\mathbb{R}$ and $[0,1]$ are not homeomorphic, we should not expect them to support the same notion of 'computable function'. Writing $C(X)$ for the continuous real-valued functions on X, it is obvious that $C([0,1]) \neq C(\mathbb{R})$, because every function in $C([0,1])$ is bounded,[4] while those in $C(\mathbb{R})$ need not be (including presumably computable functions like $f(x) = x$). On the other hand, it is well known that every $f \in C([0,1])$ extends continuously to an $f^* \in C(\mathbb{R})$; so there is a well-defined sense in which $C([0,1]) \subsetneq C(\mathbb{R})$. Consequently, models operating over $[0,1]$ cannot compute as many functions as those operating over $\mathbb{R}$.

There is an interesting, but as yet unexplored, relationship between ordinal-based and analog computation (cf. [64]). We noted above (Section 3.1.2) that only countable ordinals can be embedded in $\mathbb{R}$, and in fact the converse is also true: it is well known that an ordinal α can be embedded in $\mathbb{R}$ if and only if $\alpha < \omega_1$. Countable ordinals can have very complex structure, and can, therefore, model very complicated convergence patterns. Given a computable $f \in C(\mathbb{R})$, what does f 'look like' from the perspective of the ordinals embedded in $\mathbb{R}$? Certainly, if $\beta \subset \mathbb{R}$, then $f \restriction \beta$ is continuous—but is it also computable as an ordinal-based function? Conversely, if α is any ordinal, say that a function $f \in C(\mathbb{R})$ is α-*computable* if $f \restriction \beta$ is computable when regarded as an ordinal-based function, for every $\beta < \alpha$ and every embedding $\beta \hookrightarrow \mathbb{R}$. Is analog computability the same thing as ω_1-computability?

[4] Suppose $f \in C([0,1])$. The continuous image of a compact space is compact, so the image of f must be a compact subset of $\mathbb{R}$. But compact subsets of $\mathbb{R}$ are both closed and bounded.

3.2. Analog models

We noted in Section 1 that non-Turing models can have a categorical significance quite independent of their physical feasibility, and analog computation is a case in point. True analog models are clearly non-Turing, since they allow time to flow continuously rather than in discrete jumps; real electronic computers are, moreover, clear examples of analog devices (cf. Sections 2 and 3.1.3; see also Section 2.2 for a discussion of the super-Turing potential of analog neural nets).

Moore [65] describes a set of functions over $\mathbb{R}$ which play the same role for the reals as the recursive functions do for $\mathbb{N}$, and shows that this class contains non-recursive functions (in particular, they can be used to solve the Turing halting problem). Significantly, Moore goes on to demonstrate the very real sense in which this set of functions can be regarded as 'computable', by demonstrating a conceptual programming language which allows representations of precisely the 'computable' functions. Moore is quick to note that the model is 'almost certainly unphysical, since energy or other quantities related to the variables and their derivatives would go to infinity during the course of a computation', and goes on to investigate the *degree* to which it is unphysical by introducing a stratification of the newly-enfranchised 'recursive' functions; he shows that the lowest level of the new hierarchy corresponds to Shannon's general purpose analog computer [66]. Moore's model is also instructive in that he identifies the *source* of the model's hypercomputational power: it is the standard topology on $\mathbb{R}$. This topology ensures that every compact subset of $\mathbb{R}$ is bounded, whence the programming language (which includes **for** and **while** loops which run in continuous time) can process any compact subset of $\mathbb{R}$ in finite time. Since $\mathbb{R}$ (unlike $\mathbb{N}$) can be mapped 'computably' into a compact subset of itself, this allows the model to run a search over the whole of $\mathbb{R}$ in finite time. Clearly, this hypercomputational model, while it may be unphysical, nonetheless has independent relevance to several areas of research.

On the other hand, a super-Turing model can be significant without necessarily being hypercomputational, because it may enable computational speed-up. The standard exemplar is quantum computation, where exponential speed-up is widely acknowledged, and indeed forms the basis of an emerging sector of the IT research industry, but speed-up may also be present with classical analog computation. It is important to remember that the question "$\mathbf{P} = \mathbf{NP}$" is still unresolved, so that there are problems which may well be in **P**, but which currently have no deterministic polynomial-time solution. Consequently, we may be able to find super-Turing models which allow the computation in polynomial time of problems whose current solutions have super-polynomial complexity. For example, Siegelmann and Fishman [67] observe in their study of analog dynamical systems that the linear programming problem, whose usual solution has exponential worst-case complexity, has an analog solution with only polynomial-time complexity. Once again, then, super-Turing approaches to computation have real mathematical significance.

3.3. Computable real numbers

Which real numbers are computable? There are (at least) three distinct answers to this question. The different answers matter, because they imply that numbers can be computable by one definition and uncomputable by another. This in turn supports our claim that it makes no sense to point to Turing computation as the 'only true model of computation'. Rather, there are *many* models of computation, each making sense in its own domain. A related question is, *what is the significance of computability for real numbers?* We argue that numbers can be wholly uncomputable and unidentifiable and yet retain pragmatic significance.

3.3.1. Computability by construction

The most obvious definition of computability is presumably the oldest. A number is computable provided it can be constructed by physical means. According to this definition, and assuming the existence of Euclidean surfaces, a value like $\sqrt{2}$ is computable, because it is the length of the diagonal of a square of side 1. Circular constructions can also give us $\sqrt{x}$ for any value x already constructed. Of course, we cannot do these constructions without first declaring a certain object to have unit length, another to have unit mass, and so on, and so there is an arbitrary side to computation by construction. But as long as we never provide two different ways to define the same dimension, this arbitrariness is useful, because constructions can go ahead with no need to worry about exactness. For example, if a unit square is defined to be such by convention, it becomes irrelevant

whether the corner angles are all exactly 90°—they are 90° by *definition*, for it is the corners of the square that define what it means for an angle to be 90° and not the other way around. Similarly, the length of the diagonal obtained by cutting the unit square in half is $\sqrt{2}$ by definition. This arbitrariness is disconcerting, but is no different to the situation that confronts workers in non-Euclidean geometry. A common metaphor in this respect is to imagine someone trying to measure a table with a metal ruler, where the temperature of the table varies from one place to another. As the temperature changes, so the ruler expands and contracts, but rather than say that unit length changes from one location to the next, we can argue that—in every respect that actually matters to the experiment—the table is not flat in the first place, but curved. Similarly, computability by construction requires an acceptance that what we might call the 'geometry of computation' can change from one location to another, depending on the 'stability' of the underlying units.

The values constructible from the rationals by 'ruler and compasses' are well known, but there is no reason to restrict ourselves to just these two tools. We could allow the use of a piece of string, and thereby enable the generation via ellipses of previously unobtainable values, or we could allow the rolling of any one constructible shape (e.g., a line) around another (e.g., a circle), thereby allowing the construction of transcendental values like π. The key question, then, is *what tools are allowed*? Suppose we restrict attention to tools that can be designed by recursive means and keep things simple by assuming a classical Newtonian universe of the type familiar from school physics exams. Do the functions constructed using these recursively-designed tools have to be recursive? The answer, perhaps surprisingly, is *no*. This, to me, is the true significance of Pour-El and Richards' findings in the late 1970s and early 1980s [68,69]. By demonstrating that the wave equation could be configured recursively and yet generate a non-recursive value at a recursively identified location after a recursive period of time, they showed that even if we design a system recursively, that system need not have recursive behaviour. Naturally, actually performing the required measurement is infeasible, because we cannot actually identify that point in time which is *exactly* one second later than another, but this is neither here nor there. We cannot *exactly* measure an object of length π either, but this does not stop us talking about physical algorithms for constructing π recursively. If we are to accept π as having physical recursive constructions despite the inexactitude of real world measurements, it is only reasonable that we should extend the same courtesy to the computation-by-construction at the heart of Pour-El and Richards' wave system.

Another example may help to illustrate the main point. Myhill [70] defines a recursive function (let's call it M) on a compact interval whose continuous derivative (M') is not recursive. Even so, we can recursively design a system that draws a graph of M'. First, we use a CAD tool to cut one edge of a rectangular block of wood into the shape of Myhill's function—choose one side of the block to represent the compact interval on which m is defined, and call this the x-axis; cut the opposite edge to the same shape as the graph of $M(x)$, where we choose the scaling of the y-axis to ensure that this can be done with height to spare. Now fix the block to a flat table so that the shaped edge is uppermost. The table is on a conveyor that moves it from right to left at constant speed u. A device attached to the table moves a rod from left to right along the x-axis, also with speed u. Consequently, the horizontal position of the rod never changes. Various springs within the device ensure that the rod is pushed down horizontally onto the surface of the shaped edge at all times. By attaching an accelerometer to the rod (and performing a few analog computations), we can effectively generate M'', so we supply this to an integrator to generate a graph of M'. In other words, we can draw a graph of a non-recursive function. Again, the analog equipment involved cannot be guaranteed to be error-free (just as physical approximations to Turing machines cannot be guaranteed error-free), but even so this is a valid description of a *conceptual* device. In any case, even if the graph drawn by the system does not exactly match M', it is hard to see why it should be recursive; it is unreasonable to suppose that random physical errors consistently 'correct' for hypercomputation by converting non-recursive graphs into recursive ones.[5]

3.3.2. Sequential computability

The traditional view in theoretical computer science is that a value $x \in \mathbb{R}$ is computable provided the nth digit in its binary expansion can be computed by a recursive function $g(n)$ which halts in recur-

[5] We should not forget, however, that much of modern physics *is* 'unreasonable'.

sive time. We will abuse notation by using x to represent both the number and its digit-generating function.[6]

This is, however, a very odd way to define the computable reals, for it is a wholly unpragmatic definition that can never be used as stated—the definition forces us to use approximations to x instead. If we want to use a computable value x in real life, we first have to construct an algorithm that generates the digits of x one at a time, and then run that algorithm for ever (unless, of course, supertasks are permitted). Only then will x truly be available to us. Clearly this definition is problematic—certainly, no value defined in this way could ever be used in a physical experiment.

There is an obvious objection to this claim. For example, having identified my units (of length, say), how can I say that the number 1 cannot be made available to a physical experiment? Surely we can simply choose an object of unit length, and use that as the input to our experiment. Indeed we can, but this method of generating 1 is *computation by construction*, and has nothing to do with sequential computability! If we are to use sequential computability, we need to generate 1 as the output of an algorithm—we cannot simply use an object and then decide later that its length was indeed 1 unit, because this would require us to give an algorithm for deciding the equality of two real numbers (which is undecidable), and in any case we would once again have to wait forever for the answer. Contrast this with construction. To generate $\sqrt{2}$ we simply take our unit square and cut it in half along the diagonal.

3.3.3. $\mathbb{R}$-recursive numbers

As we saw in Section 3.2, Moore [65] has described recursion theory on the reals and its links to continuous-time computation in some detail. In his model, a function $f : \mathbb{R}^m \to \mathbb{R}^n$ is $\mathbb{R}$-*recursive* if it can be generated from the constants 0 and 1 and the following operators; in each case, if f and g can already be generated, then so can h.

- Composition: $h(\vec{x}) = f(g(\vec{x}))$.
- Differential recursion: $h(\vec{x}, 0) = f(\vec{x}), \partial_y h(\vec{x}, y) = g(\vec{x}, y, h(\vec{x}, y))$.
- Zero-finding: $h(\vec{x}) = \mu_y f(\vec{x}, y) = \inf\{y | f(\vec{x}, y) = 0\}$.
- Tupling: vectors can be defined by defining their components.

The parallel with recursive functions is obvious, and many standard functions are consequently $\mathbb{R}$-recursive, including the projection functions, addition, multiplication, division, and such functions as e^x, $\sin x$ and $\cos x$. Having defined the $\mathbb{R}$-recursive functions, Moore defines an $\mathbb{R}$-recursive number x to be one which can be obtained as $x = f(0)$ for some $\mathbb{R}$-recursive f; the $\mathbb{R}$-recursive reals include the field of algebraic numbers, as well as transcendental values like $\sqrt{2}^{\sqrt{2}}$, e and π. This definition of computable reals is very different to the standard one described in Section 3.3.2: 'Now what we mean here is not that e and π have digit sequences which can be computed sequentially (the traditional definition of computable real) but rather that they can be generated as exact, inherently analog quantities'. Significantly, there are real numbers x which are $\mathbb{R}$-recursive but not recursive in the traditional sense, so this very natural definition of analog computability is in fact hypercomputational. There is an important caveat, however. Because the zero-finding operator seems suspiciously unphysical, Moore stratifies the class of $\mathbb{R}$-recursive functions according to the number of times μ is used. This generates a μ-hierarchy which, like Hogarth's *SAD*-machine hierarchy (Section 2.5), interfaces neatly with the standard arithmetical hierarchy.

3.4. Physical significance of uncomputable reals

Writing I_c for the set of recursive reals in $[0, 1]$, their characteristic function $\chi_c : \mathbb{R} \to \mathbb{R}$ is

$$\chi_c(x) = \begin{cases} 1, & x \in I_c, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

[6] For definiteness, we will assume that no binary expansion is eventually 0. That is, we represent, e.g., 1/2 not as 0.1000,..., but as 0.0111,.... In either case, of course, the expansion is infinite.

Recall the following argument: assuming distance to be a continuous function of the reals, almost every distance is uncomputable. As we noted in Section 1, this is unconvincing as an argument that non-recursive values can be obtained physically. Nonetheless, it is not irrelevant, for in the context of analog computation the argument acquires new significance. The set I_c is both infinite and dense in $[0,1]$, but is essentially irrelevant for analog purposes. For example, given any a, $b \in [0,1]$, we can easily compute the (Lebesgue) integral $\int_a^b \chi_c$ regardless of whether a and b are recursive or non-recursive, and even though χ_c is uncomputable. Because I_c is countable it is null[7] and contributes nothing to the integral, whence $\int_a^b \chi_c \equiv 0$. So, even though none of the non-recursive values $x \in [0,1]\backslash I_c$ can be identified or measured, *their existence can be both detected and exploited*, for it is precisely the behaviour of χ_c on the *non-recursive* values that is important to the computation of $\int_a^b \chi_c$. The behaviour of χ_c on I_c is irrelevant.

3.5. Logical models

All of the mathematical models examined above embody physical metaphors. For example, Turing machines mirror the actions taken by mathematicians and accelerating machines ask what happens if the time taken for each instruction reduces fast enough for a whole program to be executed in finite time. An alternative approach is to consider computation from a purely formalist perspective, so that the status of the Church–Turing Thesis [CTT] becomes a question of pure logic. This approach leads, for example, to Church's formulation of recursion in terms of λ-expressions, or the modelling of computation via combinators.

The status of purely logical results is interesting in its own right, because their relationship to physicality is not always clear. Let us assume that the structure of physics can be expressed as a mathematical theory $\mathscr{T}_{\text{phys}}$, and write $\mathscr{U}_{\text{phys}}$ for the 'universe satisfying the physical laws embodied within $\mathscr{T}_{\text{phys}}$'. We obviously want our model of physics to be consistent with, and indeed extend, the mathematics we are using to reason about it, so if we write $\mathscr{T}_{\text{math}}$ for the underlying mathematical theory we can assume that

$$\mathscr{T}_{\text{math}} \subset \mathscr{T}_{\text{phys}}. \tag{4}$$

As is well known, [CTT] is not a purely mathematical statement, as it describes the equality of two sets of functions, one of which is defined mathematically and the other physically.[8] On the other hand, we have chosen $\mathscr{T}_{\text{phys}}$ to be a model of physics; since, in addition, $\mathscr{T}_{\text{phys}}$ extends $\mathscr{T}_{\text{math}}$, it is entirely reasonable to expect [CTT] to be expressible within $\mathscr{T}_{\text{phys}}$. Following [48], we will write [HC] for the negation of [CTT], which essentially states that 'hypercomputation is feasible'; since [CTT] is expressible, so is [HC]. However, the expression of [HC] will vary from one physical model to another, because [HC], like [CTT], is not a purely mathematical statement. Accordingly, we will write $[\text{HC}]_{\text{phys}}$ for its expression relative to $\mathscr{T}_{\text{phys}}$; $[\text{CTT}]_{\text{phys}}$ is defined likewise. To summarise, $[\text{HC}]_{\text{phys}}$ is the statement 'hypercomputation is feasible' expressed in terms that reflect the underlying structure of $\mathscr{U}_{\text{phys}}$.

All we have done so far is to express [HC] as a logical statement; our goal is to determine the status of this statement within $\mathscr{T}_{\text{phys}}$: is it a theorem, for instance? This goal involves the settling of many related questions. Stannett [72,48] addresses the status of [HC] relative to the standard model ($\mathscr{T}_{\text{std}}$) of physics, and concludes that [HC] is irrefutable within that model, i.e.

$$\mathscr{T}_{\text{std}} \nvdash [\text{CTT}]_{\text{std}}. \tag{5}$$

However, Stannett's proof of (5) involves more than just the physical laws of the standard model; he needs to include *scientific method* as part of the standard model; in effect, the inference rules of $\mathscr{T}_{\text{math}}$ are augmented

[7] $N \subseteq \mathbb{R}$ is *null* if there is a sequence of sets $S_i \subseteq \mathbb{R}$ whose measure converges to 0, and which satisfy $N \subseteq \cap S_i$. Given a countable set $N = \{x_n\}$, let S_{in} be the interval of length $2^{-(n+i+1)}$ centred on x_n, and set $S_i = \cup_n S_{\text{in}}$. Then $N \subseteq S_i$ and $\mu(S_i) = \sum_n 2^{-(n+i+1)} = 2^{-i} \to 0$, so N is null. The Lebesgue integral of χ_c is unaffected if we redefine it to equal 0 on the null set I_c, so χ_c has the same integral as the constant zero function [71].

[8] The statement of [CTT] varies across the literature; technically, it is the claim that functions $f : \mathbb{N} \to \mathbb{N}$ are effectively computable *if and only if* they are recursive/Turing-computable. The debate over hypercomputation arises through the common re-interpretation of 'effective computation' to mean 'computation by physical means', together with the assumption that recursive functions, at least, *can* be computed physically. Thus, the import of axiom [HC] is really the claim that a function can be physically computable even if it is not recursive.

with new rules representing the processes by which experimental results are adjudged to be scientifically valid. Given this extension, he identifies two statements P and Q that are logically distinct, but which no experiment satisfying the rules of scientific method can distinguish.[9] He then argues that P is a theorem of $\mathscr{T}_{\text{std}}$ and that Q entails $[\text{HC}]_{\text{std}}$. Consequently, any experiment that refutes $[\text{HC}]_{\text{std}}$ must also refute the standard model itself.

Some authors, for example Cotogno [73], have attempted to disprove [HC] on purely *logical* grounds.[10] It is, however, difficult to see what such a claim actually constitutes, for as we observed above, [HC] cannot be expressed independently of a physical model. One suggestion would be to take $\mathscr{T}_{\text{phys}} = \mathscr{T}_{\text{math}}$; we imagine a version of physics which tells us nothing we did not already know from mathematics, so that our deductions become purely mathematical. Unfortunately, this approach is untenable, because we know that [CTT] and [HC] cannot be expressed in $\mathscr{T}_{\text{math}}$. Clearly some other choice of $\mathscr{T}_{\text{phys}}$ must be being made, but it must be a choice that can be expressed in terms of $\mathscr{T}_{\text{math}}$ alone. The most likely choice to take for $\mathscr{T}_{\text{phys}}$ would seem to be a higher-order *meta-theory* of mathematics: as well as the standard statements of mathematics, we would include statements *about* mathematics. Even so, it remains difficult to see how [CTT] and [HC] could be expressed within such a theory.

There have nonetheless been some very interesting recent developments associated with logical investigations of [HC]. In particular, Wells [75] has generated what appears to be a hypercomputational family of equational theories [76]:

> The author [Wells] has constructed a family of equational theories that are not Turing-decidable, that is, given one of the theories, no Turing machine can recognize whether an arbitrary equation is in the theory or not. But the theory is called pseudorecursive because it has the additional property that when attention is limited to equations with a bounded number of variables, one obtains, for each number of variables, a fragment of the theory that is indeed Turing-decidable. In a 1982 conversation, Alfred Tarski announced that he believed the theory to be decidable, despite this contradicting CTT.

Unfortunately, Tarski only provided verbal arguments to support his claim. The question currently remains open, although Wells continues the investigate Tarski's claim, and to derive new insights into the logical status of hypercomputation [77].

4. Summary and conclusions

We have reviewed a wide range of physical and mathematical models; we begin this section by briefly summarising the material covered. We then present our main conclusions.

4.1. Summary of Section 2: physical models

We began our discussion with a consideration of *philosophy of mind*. Penrose and Bringsjord are both concerned with the concept; using different, and to some extent opposing, approaches they come to the same conclusion: minds are hypercomputational. Penrose adopts the Gödelian approach of Lucas [21], but his two major works in the field have both been the subject of strenuous refutation. Bringsjord rejects these Gödelian arguments, but not the conclusion. He generates numerous arguments against computationalism (and hence, for hypercomputationalism) by appealing to modal logic, irreversibility and other features of the mathematical mind.

Siegelmann and her colleagues have focussed on the properties of analog neural networks, showing them to be hypercomputational, but the significance of this result is disputed by Davis. Our attempt to unify the two points of view prompts us to re-examine the nature of physical observation, and the role it plays in determin-

[9] Our two scenarios concern the decay of a radioactive sample, and assume (as is required by the standard model) that we have no way of telling in advance when the sample will decay. In the first scenario we assert that the sample *must* eventually decay, even though we cannot know when this will happen. In the second scenario, we allow the possibility that the sample *never* decays. The only way to distinguish the scenarios experimentally is to wait forever, but this is inconsistent with scientific method, which requires experiments to be of finite duration.

[10] Cotogno's proof is refuted by Ord and Kieu [74].

ing whether a process is recursive or hypercomputational. We conjecture that hypercomputation is the 'natural state of affairs', and that recursion emerges from hypercomputation in the same way that classical physics emerges from quantum theory. Evidence for this claim is provided both by the super-efficient algorithms of Shor and Simon, which show that quantum computation is super-Turing, as well as Kieu's solution to Hilbert's tenth problem, which shows that true hypercomputation is potentially also present at the quantum level.

Next we looked at MacLennan's field computation. MacLennan reminds us that models should be suited to their applications. Turing machines were developed to solve a problem with formalist set theory, and are not well suited to physical modelling of continuous and pseudocontinuous processes. Non-recursive models are needed not because they are non-recursive, but simply because they are better adapted to the task in hand; their hypercomputational nature is a free bonus.

Finally, we looked at Hogarth's use of space–time singularities to solve arithmetical decision problems. The basic scheme is already powerful enough to solve the halting problem for Turing machines, and each level of the *SAD* hierarchy extends the power of the construction. SAD_n machines can solve n-quantifier arithmetic, but not $(n+1)$, and the limiting model *AD* is powerful but not too powerful; it can decide the whole of arithmetic, but there are still functions that cannot be computed.

4.2. Summary of Section 3: mathematical models

In this section, we extended the discussion to include several mathematical models of computation, and many of these clearly give different notions of computability to the Turing machine. It is perplexing, therefore, that people still say of the Church–Turing Thesis that 'it must be right because every model so far proposed to capture the intuitive notion of effective computability is equivalent to Church and Turing's models'. This is simply not true. There is nothing intrinsically ineffective about analog computation, and if techniques can be found to exploit Malament–Hogarth space–time, there may be nothing ineffective about trial-and-error and ITTM models either.

We began by considering the relevance of a model's underlying representation of time; the computational power of a model depends crucially upon this representation. We noted also that it is essential to distinguish between time as experienced by the model, and time as experienced by an observer watching its progress. Throughout this section, observer-time was represented as $\mathbb{R}$.

- Standard TMs, accelerating TMs and (single-shot) supertasks all use the same representation, $T_{\mathrm{m}} = \omega_0$, and hence all have the same computational power. Where they differ is in the embedding of T_{m} in T_{o} generated by the observation process. Standard TMs require the embedding to be co-final, while accelerating TMs require the opposite.
- Trial-and-error machines take $T_{\mathrm{m}} = \omega_0^+$, while Burgin's super-recursive and Hamkins and Lewis' ITTMs use larger transfinite ordinals. However, there is a limit to the size of ordinal α that can be used in such models, because we can only embed $\alpha \hookrightarrow \mathbb{R}$ if $\alpha < \omega_1$.
- Analog models typically take $T_{\mathrm{m}} = \mathbb{R}$ or $T_{\mathrm{m}} = [0,1]$. The models generated are not equivalent, because more computable functions can be defined on $\mathbb{R}$ than on $[0,1]$.
- We noted that a relationship exists between analog-computation and ω_1-computation. Are they the same thing?

We asked what it means for a real number x to be computable, and identified three distinct answers. Ancient tradition focusses on the construction of x using various physical tools; we noted that even if these tools are designed recursively, the values they construct need not be recursive. The traditional computer science definition of computability requires the digits in an expansion of x to be recursively generated in recursive time; we argue that this definition is wholly impracticable. Finally, we considered the mathematics of analog computation in some detail (the physical side of the question is considered in Sections 2.2 and 2.3). We noted Moore's important contribution to the field; he provides an intuitively sensible description of analog computation that includes the notion of 'analog programming'; functions are computable in his scheme if and only if they can be implemented by a program. We subsequently considered what it means for a number to be computable ($\mathbb{R}$-recursive in this model), and noted that not every $\mathbb{R}$-recursive value is recursive in the traditional

sense. Writing about the upper echelons of his μ-hierarchy (Section 3.2), Moore explains that the hierarchy is valuable, regardless of its physicality: 'these limits are precisely the ones in which many physical quantities are defined. The critical exponent of a spin system, for instance, requires infinite time and a thermodynamic limit, as well as an infinite series of systems closer and closer to the critical temperature. The μ-hierarchy may be a good tool to classify the various quantities about the world we want to measure'. The same can be said of hypercomputation in general.

4.3. Conclusion

We have examined some 20 physical and mathematical models of computation that are either known or suspected to have super-Turing or hypercomputational capabilities, covering topics as diverse as the philosophy of mind and pseudorecursive equational theories. We have found no evidence that hypercomputation is unphysical; on the contrary, no experiment consistent with the standard model can provide evidence against hypercomputation. Hypercomputation may well be intrinsic to physics, with recursion arising out of hypercomputation in the same way that classical physics emerges from quantum theory as scale increases.

The models have themselves been very different, but each is well-suited to its task. This suggests that no one model of computation deserves to be singled out and treated as somehow 'fundamental'. Rather, all models are relevant in their application domains. Hogarth is right: it is as meaningless to regard one version of computation as somehow more fundamental than those that disagree with it, as it is to consider Euclidean geometry more fundamental than its non-Euclidean counterparts.

Furthermore, even if hypercomputation were one day shown to be physically infeasible, there would still remain a role for hypercomputation as an organising principle for advanced research. The history of science includes many examples of ideas being declared initially 'unrealistic', which subsequently provided solid foundations for acceptable theories (to name but three: the constancy of the speed of light, wave-particle duality, and string theory). It is clear that hypercomputation is prompting detailed research into a wide range of interesting and relevant questions; this research activity is itself enough to justify the quest for hypercomputation, irrespective of its ultimate physicality. But is this physicality really in doubt? We have identified compelling evidence that hypercomputation is physically meaningful. This weight of evidence cannot be ignored; the balance of probabilities rests on the side of hypercomputation.

References

[1] B. MacLennan, Natural computation and non-Turing models of computation, Theor. Comput. Sci. 317 (2004) 115–145.
[2] R. Penrose, The Emperor's New Mind: Concerning Computers, Minds and the Laws of Physics, Oxford University Press, Oxford, 1989.
[3] R. Penrose, Shadows of the Mind: A Search for the Missing Science of Consciousness, Oxford University Press, Oxford, 1994.
[4] T. Ward, S. Bringsjord, An argument for the uncomputability of infinitary mathematical expertise, in: P. Feltovitch, K. Ford, R. Hoffman (Eds.), Expertise in Context, AAAI Press, Menlo Park, CA, 1997.
[5] B. Copeland, Hypercomputation: philosophical issues, Theor. Comput. Sci. 317 (2004) 251–267.
[6] P. Kugel, Towards a theory of intelligence, Theor. Comput. Sci. 317 (2004) 13–30.
[7] A. Turing, Systems of logic based on ordinals, Proc. London Math. Soc., Ser. 2 45 (1939) 161–228, originally produced as [78].
[8] M. Newman, Alan Mathison Turing 1912–1954, Biograph. Mem. Fellows R. Soc. 1 (1955) 253–263.
[9] S. Bringsjord, K. Arkoudas, The modal argument for hypercomputing minds, Theor. Comput. Sci. 317 (2004) 167–190.
[10] M. Davis, The myth of hypercomputation, in: C. Teuscher (Ed.), Alan Turing: Life and Legacy of a Great Thinker, Springer, Berlin, 2004, pp. 195–211.
[11] P. Welch, On the possibility, or otherwise, of hypercomputation, Brit. J. Phil. Sci. 55 (2004) 739–746.
[12] M. Hogarth, Deciding arithmetic using SAD computers, Brit. J. Phil. Sci. 55 (2004) 681–691.
[13] K. Svozil, The Church–Turing Thesis as a guiding principle for physics, in: C. Calude, J. Casti, M. Dineen (Eds.), Unconventional Models of Computation, Springer, Singapore, 1998.
[14] K. Kelly, Uncomputability: the problem of induction internalized, Theor. Comput. Sci. 317 (2004) 227–249.
[15] H. Putnam, Trial and error predicates and the solution to a problem of Mostowski, J. Symbolic Logic 30 (1965) 49–57.
[16] E. Gold, Limiting recursion, J. Symbolic Logic 30 (1) (1965) 28–48.
[17] A. Church, Review of [20], J. Symbolic Logic 2 (1937) 42–43.
[18] C. Cleland, The concept of computability, Theor. Comput. Sci. 317 (2004) 209–225.
[19] A. Church, An unsolvable problem of elementary number theory, Am. J. Math. 58 (1936) 345–363.

[20] A. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math. Soc., Ser. 2 42 (1936) 230–265;
A. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math. Soc., Ser. 2 43 (1936) 544–546, correction.
[21] J. Lucas, Minds, machines, and Gödel, in: A. Anderson (Ed.), Minds and Machines, Prentice-Hall, Englewood Cliffs, NJ, 1967, pp. 43–59. <http://users.ox.ac.uk/~jrlucas/mmg.html>.
[22] K. Gödel, Some basic theorems on the foundations of mathematics and their implications, in: Collected Works (1995), Oxford University Press, Oxford, 1951, pp. 304–323.
[23] S. Bringsjord, H. Xiao, A refutation of Penrose's Gödelian case against artificial intelligence, J. Exp. Theor. Artif. Intel. 12 (2000) 307–329.
[24] S. Bringsjord, M. Zenzen, Superminds: People Harness Hypercomputation, and More, Kluwer Academic Publishers, Dordrecht, 2003.
[25] S. Bringsjord, An argument for the uncomputability of infinitary mathematical expertise, in: K. Ford, P. Hayes, P. Feltovich (Eds.), Expertise in Context, AAAI Press, Menlo Park, CA, 1997, pp. 475–497.
[26] S. Bringsjord, M. Zenzen, Cognition is not computation: the argument from irreversibility, Synthese 113 (1997) 285–320.
[27] S. Bringsjord, B. van Heuveln, The 'mental eye' defense of an infinitized version of Yablo's paradox, Analysis 66 (1) (2003) 61–70.
[28] W. McCulloch, W. Pitts, Logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
[29] H. Siegelmann, E. Sontag, Analog computation via neural networks, Theor. Comput. Sci. 131 (1994) 331–360.
[30] H. Siegelmann, Computation beyond the Turing limit, Science 268 (5210) (1995) 545–548.
[31] H. Siegelmann, Neural Networks and Analog Computation, Birkhäuser, Basel, 1999.
[32] S. Kleene, Representation of events in nerve nets and finite automata, in: C. Shannon, J. McCarthy (Eds.), Automata Studies, Princeton University Press, Princeton, NJ, 1956, pp. 3–41.
[33] P. Indyk, Optimal simulation of automata by neural nets, in: Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science, LNCS, vol. 900, Springer, Berlin, 1995, pp. 337–348.
[34] H. Siegelmann, E. Sontag, Computational power of neural networks, J. Comput. Syst. Sci. 50 (1) (1995) 132–150.
[35] 'Nervous system, human', in: Encyclopaedia Britannica, 2003.
[36] B. MacLennan, Field Computation: A Theoretical Framework for Massively Parallel Analog Computation, Parts I–IV, Tech. Rep. CS-90-100, Department of Computer Science, University of Tennessee, Knoxville, USA. Available from: <http://www.cs.utk.edu/~mclennan/> (1990).
[37] B. MacLennan, Transcending Turing Computability, Tech. Rep. UT-CS-01-473, Department of Computer Science, University of Tennessee, Knoxville, USA. Available from: <http://www.cs.utk.edu/~mclennan/> (2001).
[38] P. Benioff, J. Stat. Phys. 22 (1980) 563.
[39] P. Benioff, Quantum mechanical Hamiltonian models of Turing machines, J. Stat. Phys. 29 (1982) 515–546.
[40] P. Benioff, Quantum mechanical models of Turing machines that dissipate no energy, Phys. Rev. Lett. 48 (1982) 1581–1585.
[41] R. Feynman, Simulating physics with computers, Int. J. Theor. Phys. 21 (1982) 467–488.
[42] R. Feynman, Quantum mechanical computers, Found. Phys. 16 (1986) 507–531, a preliminary version appeared as Optics News, February 1985, pp. 11–20.
[43] D. Deutsch, Quantum theory, the Church–Turing principle, and the universal quantum Turing machine, Proc. Royal Soc. 400 (1985) 97–117.
[44] A. Steane, Quantum computing, Rep. Prog. Phys. 61 (1998) 117–173.
[45] D. Simon, On the power of quantum computation, in: Proceedings of the 35th Annual Symp. on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 124–134.
[46] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, in: S. Goldwasser (Ed.), Proceedings of the 35th Annual Symp. on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1994, p. 124, revised version (1995): quant-ph/9508027.
[47] M. Stannett, An Introduction to post-Newtonian and non-Turing computation, Tech. Rep. CS-91-02, Department of Computer Science, Sheffield University, UK, re-issued 2002 in revised form: <http://www.dcs.shef.ac.uk/research/resmems/> (1991).
[48] M. Stannett, Computation and hypercomputation, Mind Mach. 13 (1) (2003) 115–153.
[49] T. Kieu, Numerical simulations of a quantum algorithm for Hilbert's tenth problem, in: E. Donkor, A. Pirich, H. Brandt (Eds.), Quantum Information and Computation, Proceedings of the SPIE, vol. 1505, SPIE, Bellingham, WA, 2003, pp. 89–95.
[50] T. Kieu, Quantum algorithm for Hilbert's tenth problem., Int. J. Theoret. Phys. 42 (2003) 1451–1468.
[51] T. Kieu, Quantum adiabatic algorithm for Hilbert's tenth problem: I. The Algorithm. Available from: quant-ph/0310052.
[52] T. Kieu, Hypercomputation with quantum adiabatic processes, Theoret. Computer Sci. 317 (2004) 93–104.
[53] Y. Matiyasevich, Hilbert's Tenth Problem, MIT Press, Cambridge, MA, 1993.
[54] M. Hogarth, Does general relativity allow an observer to view an eternity in a finite time? Found. Phys. Lett. 5 (1992) 73–81.
[55] J. Earman, J. Norton, Forever is a day: supertasks in Pitowsky and Malament–Hogarth spacetimes, Philos. Sci. 5 (1993) 22–42.
[56] G. Etesi, I. Nemeti, Non-Turing computability via Malament–Hogarth Space–times, Int. J. Theor. Phys. 41 (2) (2002) 341–370.
[57] J. Earman, Bangs, Crunches, Whimpers and Shrieks—Singularities and Acausalities in Relativistic, Oxford University Press, Oxford, 1995.
[58] J. Earman, J. Norton, Infinite pains: the trouble with supertasks, in: A. Morton, S. Stich (Eds.), Benacerraf and His Critics, Blackwell, Cambridge, MA, 1996, pp. 231–261.
[59] B. Copeland, Hypercomputation, Mind Mach. 12 (2002) 461–502.

[60] J. Hamkins, A. Lewis, Infinite time Turing machines, J. Symbolic Logic 65 (2) (2000) 567–604.
[61] J. Hamkins, Infinite time Turing machines, Mind Mach. 12 (2002) 521–539.
[62] M. Burgin, Universal limit Turing machines, Notices Russ. Acad. Sci. 325 (1992) 654–658 (translated from Russian).
[63] O. Shagrir, Super-tasks, accelerating Turing machines and uncomputability, Theor. Comput. Sci. 317 (2004) 105–114.
[64] M. Kwiatkowska, M. Stannett, On Transfinite Traces, in: V. Diekert (Ed.), ASMICS Proc. Universität Stuttgart Fakultät Informatik, Bericht 4/92, 1992.
[65] C. Moore, Recursion theory on the reals and continuous-time computation, Theor. Comput. Sci. 162 (1996) 23–44.
[66] C. Shannon, Mathematical theory of the differential analyzer, J. Math. Phys. MIT 20 (1941) 337–354.
[67] H. Siegelmann, S. Fishman, Analog computation with dynamical systems, Physica D 120 (1998) 214–235.
[68] M. Pour-El, J. Richards, A computable ordinary differential equation which possesses no computable solution, Ann. Math. Logic 17 (1979) 61–90.
[69] M. Pour-El, J. Richards, The wave equation with computable initial data such that its unique solution is not computable, Adv. Math. 39 (1981) 215–239.
[70] J. Myhill, A recursive function, defined on a compact interval and having a continuous derivative that is not recursive, Mich. Math. J. 18 (1971) 97–98.
[71] A. Weir, Lebesgue Integration and Measure, Cambridge University Press, Cambridge, 1973.
[72] M. Stannett, Hypercomputation is physically irrefutable, Tech. Rep. CS-01-04, Department of Computer Science, Sheffield University, UK. Available from: <http://www.dcs.shef.ac.uk/research/resmems/> (2001).
[73] P. Cotogno, Hypercomputation and the physical Church–Turing Thesis, Brit. J. Phil. Sci. 54 (2003) 181–223.
[74] T. Ord, T. Kieu, The diagonal method and hypercomputation, Brit. J. Phil. Sci. 56 (2005) 147–156.
[75] B. Wells, Pseudorecursive varieties of semigroups–I, Int. J. Algebra Comput. 6 (1996) 457–510.
[76] B. Wells, Is there a nonrecursive decidable equational theory? Mind Mach. 12 (2002) 303–326.
[77] B. Wells, Hypercomputation by definition, Theor. Comput. Sci. 317 (2004) 191–207.
[78] A. Turing, Systems of Logic based on Ordinals, Ph.D. Thesis, Princeton University, published as [7] (1938).