Technical Report No. 2006-510
ACCELERATING MACHINES*

Robert Fraser and Selim G. Akl
School of Computing
Queen's University
Kingston, Ontario, Canada K7L 3N6
{robert,akl}@cs.queensu.ca

March 20, 2006

Abstract

This paper presents an overview of accelerating machines. We begin by exploring the history of the accelerating machine model and the potential power that it provides. We look at some of the problems that could be solved with an accelerating machine, and review some of the possible implementation methods that have been presented. Finally, we expose the limitations of accelerating machines and conclude by posing some problems for further research.

Keywords: Accelerating machine, hypercomputer, parallel computation, Turing machine, universality.

1 Introduction

The goal of this paper is to provide an overview of the work that has been published to date with respect to accelerating machines. The accelerating machine model is a fantastic theoretical entity; it promises to be able to solve problems that no machine available today could ever hope to do, even if it were allowed to run to the end of time. Although most of the attention paid to the topic has been fairly recent, the temporal patterning at the heart of the machine has roots thousands of years ago among the ancient Greeks.

This review begins with a look at the fundamentals of computing, beginning with the Turing machine and the Church-Turing thesis. Then we briefly examine hypercomputers, a class of machines which are capable of more than conventional Turing machines. The accelerating machine is an example of one such hypercomputer. This is followed by a review of the basic properties of accelerating machines and some of the problems that would become tractable should such machines be implemented. Just how this implementation might be carried out is presented next, along with a discussion of some physical limitations and bounds on the computational power of accelerating machines. The paper concludes by offering some questions for future research.

*This research was supported by the Natural Sciences and Engineering Research Council of Canada.

2 Turing Machines

Alan Turing, often considered the father of computing, was the first to propose a rigorous and constructive definition of "computation". He invented a computational model, today named after him, in order to address the question of decidability in mathematics [Akl07a]. A *Turing machine* is a conceptual object, intended as a *model* of a computer, that solves problems effectively in the manner that a human clerk would perform the same task using pencil and paper. The analogy is apt, since in those days, all *computers* were people employed to perform calculations by hand [Cop02b]. Turing's idea was to use his model of computation to solve a problem by executing a finite number of operations. The Turing machine can be in one of a finite number of states, registered by a control unit. In addition, the machine contains a paper tape of infinite length, divided into squares, which stores the data and results of a computation. The Turing machine is equipped with a read/write head that moves back and forth along the tape. A symbol from a finite alphabet can be written onto a square by the read/write head. The current state of the machine, and the symbol on the square at which the read/write head is positioned, fully determine the next action of the machine (that is, the new state of the machine, the new symbol, if any, written by the head, as well as the head's new position, if any, one square to the left or to the right). For a complete description of the Turing machine, see [Dav00].

As discussed in the next section, this machine is generally assumed to capture the essence of the process of "computation". On the basis of this assumption, the existence of problems that cannot be solved by the Turing machine implies the existence of formal mathematical systems that are undecidable [Tur37]. Turing also conceived of a *universal Turing machine*, that is, a machine capable of being "programmed" to perform a thorough simulation of the operations that any other Turing machine can do.

2.1 The Church-Turing Thesis

The basis of the Church-Turing thesis is that the Turing machine is essentially the definition of computability. There is some dissention as to the precise meaning of this thesis, but at the heart it states that any mathematical function that is somehow computable by some device, can be computed by a Turing machine, and conversely, if there is a mathematical function that cannot be computed by a Turing machine, then it is considered to be uncomputable by any other means. Nowadays, the thesis is stated as saying that a computational problem is solvable if and only if a finite machine (read a Turing machine) can solve the problem in a finite amount of time [Cop98a]. Some believe that there does not exist a convincing argument that any type of object could solve problems that are of a class beyond those solvable by Turing machines [Cas97, Sud06]. Because no model-independent all-encompassing definition of what it means "to compute" exists (or is even conceivable) this claim is a "thesis", that is, a conjecture, and an unprovable one at that. However, it may be *disproved*. Indeed several examples of *computable* functions, that cannot be computed by the Turing machine, are known [CP04, CS99, Deu97, EN02, Kie03, LN04, NA06, Pen90, Sie99, Sta90, WG97]. This paper is a review of one of the most intriguing vehicles that have been proposed to violate the Church-Turing thesis: the accelerating machine.

2.2 Hypercomputation

Turing did not deny that, given some theoretical but potentially physically realizable model of computation differing from the conventional, some problems considered uncomputable by the Church-Turing thesis may become tractable. In fact, he developed variants of the Turing machine that were able to compute functions considered uncomputable by the Church-Turing thesis, such as the O-machine [Tur39] (see also [CP99]). The O-machine is a Turing machine equipped with an oracle capable of telling instantly whether a queried number belongs to a certain (possibly infinite) set of numbers. The Halting problem is a classic problem where, given an arbitrary mathematical function and its input, it is required to determine in finite time whether the program that computes this function will ever halt or run on forever. Although the Halting problem is uncomputable for a conventional Turing machine [Dav00], an O-machine is capable of hypercomputation. If the Halting set of a function were a (possibly infinite) set of numbers, then the machine could determine whether a given input belongs to the set, thus solving the Halting problem. This is an example of the promised power and appeal of hypercomputation.

Today, we know that there exist models of computation possessing more power than a Turing machine and they collectively are attributed to the field known as *hypercomputation*. Stannett [Sta04] provides a categorization scheme for the different models of hypercomputation that have been proposed in the literature. They are summarized by four different basic strategies for the modification of the Turing model:

- the temporal structure of computation
- the information content of memory
- the information content of programs
- the information content of states.

Accelerating machines fall into the first category of these machines. For a review of the other models, see [Cop02b, Ord02, Sta04].

2.3 Algorithmic Complexity

The traditional method of measuring the complexity of an algorithm is to find an expression to describe the number of primitive operations that the algorithm requires [Akl97]. This is intuitive, as this description is independent of implementation and the system on which it is implemented. An alternative description would be a temporal expression, where the duration of each primitive operation is considered. *Time varying complexity* is a phenomenon where the complexity of a problem changes with time [Akl07b]. An example of such a problem is tracking a satellite that is traveling away from the Earth. As the distance between the satellite and Earth increases, the time required to communicate with the satellite increases accordingly, and thus the complexity of operations increases with the passage of time. Consider a simple operation such as an operator on Earth reacting to some input peripheral to the satellite by instructing the satellite to rotate towards the input. Simply due to transmission times, this operation takes twice as long when the satellite is twice the distance from the operator. (We note here that time varying complexity is not to be confused with the paradigm of *time varying variables*. In the latter, the value of a variable changes with time, be it constantly, exponentially, randomly, and so on [Akl06b].)

3 Accelerating Machines

The basic idea of the accelerating machine is deceptively simple: the time required by an operation at any given step of a computation is only half (or some other constant fraction) of that required to perform the same operation in the previous step. More formally, assuming that an operation takes one time unit in the first step, the total time to complete the computation may be expressed as the sum of a geometric series:

$$\sum_{i=0}^{n} \frac{1}{2^i},$$

where i is the current step and n is the number of steps in the computation. As i approaches infinity, the total time approaches 2, that is:

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 2.$$

The beauty of the accelerating machine is that this model of computation allows any number of iterations of a computational step to be performed in a finite amount of time. This is especially wonderful since this amount of time cannot be even twice the amount of time required by the first iteration! More generally, the accelerating machine can be formalized as saying that:

$$\sum_{i=0}^{\infty} t_i < t,$$

where t_i is the time required for the operation at iteration i, and t is some constant finite time. Thus, if we run a program on an accelerating machine, we are guaranteed that the computation will be finished by time t, even if the program recurses infinitely.

Turing did not take into account the amount of time that a computation takes on one of his machines, so the mathematics of Turing machines hold for accelerating machines [Ord02]. Further, there is no requirement that a particular operation should require a constant amount of time to perform in the Turing model [Sta04]. However, the accelerating machine cannot be considered a Turing machine, since the Turing machines operate on discrete time intervals. Accelerating machines do not. We now examine this point in depth as we turn to a discussion of Zeno's paradoxes.

3.1 Zeno's Paradoxes

There have been many discussions of the accelerating machine concept through history. Naturally, there are many references to such a model that are not targeted specifically at computing; the concept could even be attributed to the ancient Greeks.

The original description of the series characteristic of accelerating machines can be traced back to Zeno's paradoxes. Zeno was a philosopher, who studied under Parmenides, and developed his paradoxes about 450 BC to show that motion is impossible [Bar05]. His first paradox is usually discussed as an anecdote of a person either walking or running some set distance. Imagine that you are running a 10 kilometre course. Once you begin running, you can divide the distance that you must traverse to reach the finish into discrete intervals. Let's define these intervals as half of the distance that you have yet to travel, so we get the series 5, 2.5, 1.25, 0.625, ..., in an infinite series. Thus, for you to traverse the entire distance, you must pass through an infinite number of intervals. Since the Greeks did not accept infinities, Zeno used this to show that motion is impossible. Zeno's second paradox is similar. It consists of two racers: Achilles versus his opponent, who has been turned into a tortoise. The tortoise is given a 1km head start because he only runs half as fast as Achilles. When Achilles has reached the 1km mark, the tortoise is at the 1.5km mark; once Achilles arrives at 1.5km the tortoise is at 1.75km. Achilles's position can be described by

$$2 - \frac{1}{2^N},$$

for $N = 0, 1, 2, ...$, which will always be behind the tortoise whose position at the same time interval is

$$2 - \frac{1}{2^{N+1}}.$$

The tortoise will thus always be ahead, albeit by an amount that decreases infinitely over the intervals. Thus, Zeno concluded, Achilles can never pass the tortoise and motion is an illusion.

Zeno's paradoxes have also been used to suggest that accelerating machines, while sufficient, may not be necessary to solve problems uncomputable by Turing machines. Cleland [Cle02] argued that Zeno's paradox does not prove that we are required to perform an infinite number of distinct tasks to move somewhere, and thus by analogy we may not be required to perform an infinite number of distinct operations to solve a problem beyond the capabilities of a Turing machine. To our knowledge, no one has refuted this claim to date.

3.2 Russel-Blake-Weyl Temporal Patterning

Bertrand Russell discussed the race-course paradox in 1914 [Rus15] and explained the concept that the geometric sum solves the problem as the number of intervals approaches infinity (Ralph Blake [Bla26] drew similar conclusions in 1926). Russell elaborated in a later work [Rus36], introducing the idea that a man's skill at an operation could increase such that each time he repeated an operation he would accomplish it twice as quickly; this is a human version of an accelerating machine. The application in his discussion was that of a man enumerating the digits of π, and he emphasized that such a scenario is *logically* possible, although it is *medically* impossible.

Probably the first application of the idea to an actual machine can be attributed to Hermann Weyl [Wey27, Wey49]. He postulated a machine where the first decision for a given problem could be found within half a minute, and each subsequent decision required half of the time of the previous one. He asserted that this property would allow the traversal of all natural numbers in a finite amount of time, permitting the solution to any problem associated with them.

Discussion of the model of perpetual acceleration (Copeland dubbed it Russell-Blake-Weyl (RBW) temporal patterning [Cop02a]) often comes around to Zeus. Georg Cantor was the reason for this, as he first described sets of numbers that cannot be enumerated [Bar05]. Not even Zeus could enumerate these sets, no matter how hard or long he worked. Boolos and Jeffrey [BJ80] discuss Zeus' challenge, and propose the accelerating model as the solution for Zeus to enumerate any enumerable set. He could simply write a number on a list in half the time that he wrote the previous number to produce an infinite list. Thus, machines that operate using a model of computation based on RBW temporal patterning are sometimes referred to as Zeus machines [Cop02b].

3.3 Different Time Scales

Svozil [Svo98] explains the behavior of accelerating machines in a slightly different fashion, although the meaning is the same. He describes two different time scales:

- τ - this describes the time scale that our clocks would measure, what Svozil calls physical system time.
- t - this is a discrete time scale, which can be measured using the set of non-negative integers, describing the intrinsic time for a machine.

The accelerating machine operates by compressing the time scale t with respect to τ by a geometric progression. Given a factor $k < 1$, the time t can be measured in terms of τ by the following:

$$\tau_0 = 0,$$

$$\tau_1 = k,$$

$$\tau_{t+1} - \tau_t = k\left(\tau_t - \tau_{t-1}\right), t \geq 1,$$

$$\tau_t = \sum_{n=0}^{t} k^n - 1 = \frac{k\left(k^t - 1\right)}{k - 1}.$$

Since $k < 1$, as t goes to infinity, the time τ remains finite:

$$\tau_\infty = \frac{k}{(1-k)}.$$

The conventional example we have been using throughout the paper has been that the speed of the accelerating machine doubles with each iteration. To apply this pattern to Svozil's model, we would use $k = 1/2$, giving $\tau_1 = 1/2$ and $\tau_\infty = 1$. This is the same pattern as presented earlier, expect for the first iteration. If the speed of an accelerating machine doubles with each iteration, the total time required is 1 time unit if the initial iteration is 1/2 time unit. If the first iteration takes 1 time unit, the machine will take 2 time units in total for an infinite number of iterations.

The application of an accelerating series to a Turing machine has been fully discussed by Copeland [Cop98b, Cop02a, Cop04b], who coined the appellation 'accelerating universal Turing machine (AUTM)', which is commonly used by other authors [EW03, CP04]. Copeland has synonymously used accelerating Turing machine (also [Ste02, Sha04, OK05]) and accelerating digital machine [CS99]. There are many other names that have appeared, such as the Zeus machine of Boolos and Jeffrey [BJ80], or the Zeno squeezed oracle computer of Svozil [Svo98], or Stewart's [Ste91b] Rapidly Accelerating Computer (RAC). Stannett [Sta04] uses accelerating-time hypercomputer, accelerating Turing machine, accelerating-time machine, and convergent-time machine interchangeably.

4 Applications

One remarkable application of the accelerating machine is in the solution of the *Halting problem*: Given a mathematical function f and its argument x, it is required to determine in finite time whether the computation of $f(x)$ terminates or continues indefinitely. The accelerating machine computes $f(x)$ by performing the first operation in the calculation in one time unit, the second in one-half of a time unit, and so on. The answer to the problem is obtained in a total of at most two time units (even if the computation of $f(x)$ does not terminate). Many papers have described this application of an accelerating Turing machine [Hog94, Tip94, Cop98a, Cop02a, Ord02, Ste02, Sha04, Sta04], as it is a variant of a problem used by Turing to describe the limitations of Turing machines, known as the *Entsheidungsproblem*, or the "decision problem". It is interesting to note that the current and widely used incarnation of the Halting problem, is attributed to Martin Davis [Cop04a].

4.1 Internal and External Operation

Now, one may require that there be a means of indicating to external parties whether the (Turing machine) program for computing f indeed halted on a given input x. Conventionally, the proposed implementation provides the machine with a square on the tape that has a 0 as the initial value. This value is changed to a 1 if and when the program halts. If after two time units (double the time used by the first operation) the value in the square remains a 0, it can be concluded that the program has not halted. This model of computation is a modified Gold-Putnam machine, which has a standard, non-accelerating implementation [Cop04b]. Copeland [Cop98a] has also suggested using a hooter to blow a tone when the function halts, a testament to Turing and his colleagues who enjoyed the prospects offered by hooters in computing. This observation suggests that a distinction might be needed between the case in which a function is computable by a machine in an *internal sense*, and the case where it is computed in an *external sense* [Cop02b]. The method of solving the Halting problem presented here (using an accelerating machine) is done in the external sense, in that another machine or a person checks the result by reading the value on the predefined square after the set amount of time has passed. The accelerating machine itself is not truly solving the problem, as people conventionally think of problem solving being performed in the internal sense. The internal sense of solving a problem would be that the machine arrives at the solution without any input other than the initial arguments and presents its conclusions when finished. It has been suggested that no Turing machine can solve the Halting problem in the internal sense (see [Tur37] or [Cop98a] for a discussion). It may be relevant to point out here that the distinction between internal and external solutions is wholly artificial, whether the question is regarded philosophically or pragmatically. Indeed, it is clear that no computation is ever useful unless its results are noticed, directly or indirectly, by an active observer (in other words, all meaningful computations are essentially performed in an external sense).

4.2 Mathematical Applications

Stewart [Ste91a, Ste91b] suggests that we might use accelerating machines to prove or disprove Fermat's last theorem. Although this theorem has been proved in the meantime, the point remains that an accelerating machine could be used to prove any theorem one wishes by exhaustively testing all possible inputs in a finite amount of time (for that matter, one could prove all possible theorems in a finite amount of time).

A perpetual problem in computer science and mathematics is the expansion of π. Since π is irrational, it is impossible for a conventional Turing machine to perform the expansion. For each new integer, the Turing machine must calculate it and then transcribe it on paper. A conventional Turing machine would continue these operations for as long as it had paper and power. An accelerating Turing machine could accomplish this task in a finite time, since it would compute each digit in half of the time required to compute the one before. Of course, this Turing machine would also require an infinite amount of paper on which to write all of these numbers. Intuitively, this whole concept seems to entail a logical contradiction (one which Copeland [Cop02a] contended would render accelerating machines as logical impossibilities as well). Consider that this machine, in order to write the entire expansion of π, had to write a final decimal place to finish its task. But there is no such number since the expansion is infinite. This problem is accounted for by infinities, since the expansion of π is infinite, the machine required an infinite amount of paper to write the expansion, and there is no last digit, since an infinite number of digits were written. There remains the question of what would happen if each number were written on a single square, such that when the task of expanding π was finished one could simply check this square to learn the "final" number in the expansion. This apparent paradox is addressed in the following section on super tasks.

4.3 Super Tasks

The process of performing an infinite number of tasks is referred to as a *super task*, an example being the expansion of π just discussed [Cop02a]. There are other examples of tasks that could be performed by an accelerating machine that have spawned debate. James

Thompson [Tho54] developed the famous example of the lamp. Assume that we are given a lamp, and at each time interval the switch of the lamp is pressed. If the lamp is initially off, let's say that at the one second mark it is switched on by our accelerating machine, at 1.5 seconds it is switched off again, after 1.75 seconds it is on, and so on infinitely. The question is whether the lamp is on or off after the accelerating machine has flipped the switch an infinite number of times. Thompson used the argument to support his claim that super tasks are impossible. The lamp cannot be on, because the lamp would have to be switched off again, and it cannot be off for the same reason. He uses this to establish a contradiction since the lamp must be either on or off. However, Thompson later admitted that his argument is faulty as a result of a critique of his work by Paul Benacerraf [Ben62]. The lamp could be in either state at the end of an infinite number of steps, and this would be logically consistent with the description of the super task. Thompson's points are not worthless, however, since there remains the question of whether the state that an accelerating machine will be in after performing a super task will be consistent. Benacerraf works thoroughly and humourously to explain the fault in Thompson's claims. He calls the question of the state after the conclusion of the super task a *super-duper task*. The state at 2 seconds, using the same example as previously, has nothing to do with the infinite number of tasks that were performed prior to this moment. Benacerraf illustrates this point using Aladdin and the genie of the lamp. He provides a scenario where Aladdin instructs the genie to move towards a point located at 1 from point 0, and to occupy every point in the series $0, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, ...$ on his way. A point in this series is described by the function

$$p_i = 1 - \frac{1}{2^i}.$$

Once he has accomplished his traversal, the genie is to disappear. Now the super-duper task would be to ask what the genie's state is at point 1, but this was outside our purposes. We asked him to occupy all the points in this series, a super task, which he would accomplish without ever reaching 1. We can draw this easily using lines, as shown in Figure 1.

Why couldn't an accelerating machine perform super-duper tasks? The nature of super tasks is that there is an infinite number of operations to perform. When we apply an accelerating machine to the task, inevitably people try to attach a finiteness by questioning the state of the machine after the final operation. This is important: *there is no final operation.* The question of the lamp is moot, it is the same as asking whether infinity is even or odd. To pose such a query is nonsensical. The lamp is turned off and on an infinite number of times; from the perspective of the machine performing the switching, the task is never finished. Consider the following algorithm, one that is possible to implement on any machine in any language:

Algorithm A Simple Infinite Loop
  $i = 1$
  while $i > 0$ **do**
    $i = i + 1$
  end while
  return i. ■

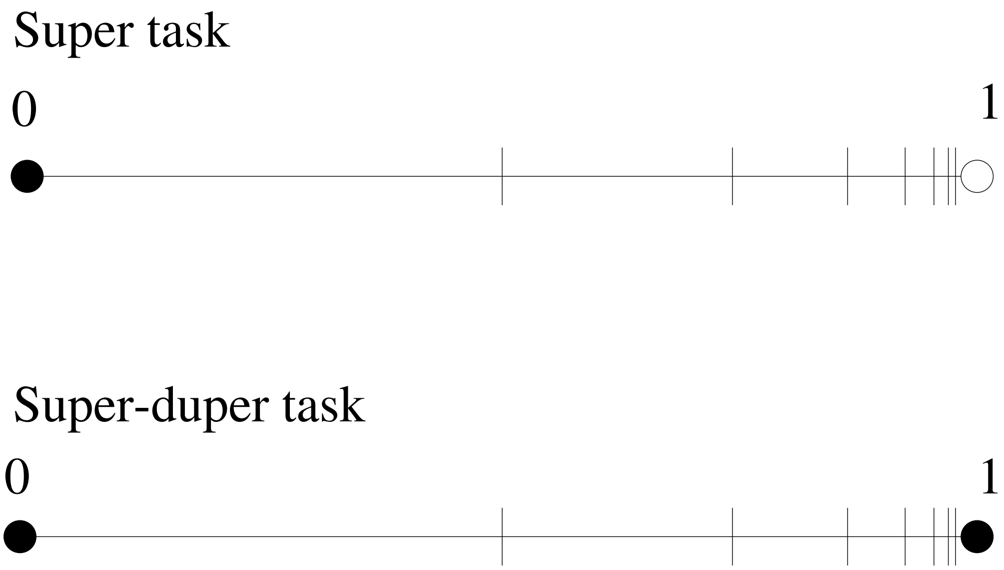

Figure 1: Aladdin's super task for the genie is to occupy all of the points on this series, which approaches 1. The super-duper task would be to occupy 1 as well.

This algorithm counts to infinity; it counts forever. A computer could never "finish" this problem. It is a perfect example of where an accelerating machine could be used to solve the halting problem. The algorithm has a halting condition (which will obviously never be met as i will never be zero or less), so we can run it on the accelerating machine to determine if it will halt. The machine will run, performing an infinite number of computations, and nothing will be returned within the time frame in which we know the machine could perform this infinite number of computations. We can conclude therefore that the program did not halt. However, for the accelerating machine, it ran forever. There is no "finishing" this task, as obviously the halting condition is never met. This is perhaps illustrative of the intended difference between internally and externally solving the problem. With such super tasks, only externally solvable problems are valid. Thus the question of the state of the machine after it has performed an infinite number of computations does not apply, as there can be no "after" infinity. We discuss this matter further while covering implementations in Section 6.

4.4 Returning to π

Copeland [Cop02a] states that no Turing machine could perform the task of expanding π, since there is no halting condition. Copeland tries to work around this problem by suggesting that a machine could be built to shut down the accelerating machine after the amount of time has passed that the machine requires to perform the expansion, but this is just avoiding the problem. To state that one needs to shut down the machine after it has carried out the super task implies that it is still doing something, that it is still performing some task.

Suppose we had an accelerating machine that is capable of expanding π. Asking what the final number that would be written is again asking for a super-duper task. The expansion of π is an infinite task, so one cannot ask this of the machine. We could ask if it halts, and it would not reply, so we would know that it does not. We could ask queries such as whether there is ever an instance of 100 consecutive 7s in the expansion, and the machine can answer that. But since there is no final digit, it is not a valid query to ask what the final digit is.

Hamkins [Ham02] uses an interesting anecdote to caution about the idiosyncratic nature of super tasks. You are fortunate enough to have an infinite number of one dollar bills, which are numbered with all of the odd integers. You meet the devil in a seedy bar, and he offers to give you two dollars for every one that you have, the only condition being that he will give you bills indexed higher than those you have. Being intrigued, you agree to the deal thinking that you have nothing to lose. You perform the transaction in an accelerating fashion, so that at time step 1 you give him your bill labelled 1, and the devil gives you those labelled 2 and 4. At time 1.5, you give him back the bill labelled with a 2, and he hands you bills bearing 6 and 8. You continue this until time 2, and you discover you are broke; hoodwinked by the devil. At iteration n of your transactions, you gave him the bill labelled with n, and by his scheme you will never see that bill again. Thus, given the ability to perform super tasks, particular care must be paid to the implementation of the algorithm, lest surprising consequences arise. It is contentious as to whether this argument is valid, however, as there is no formal proof provided in the paper. In fact, a counterproof to this argument is straightforward. At every iteration of your dealings with the devil, you gain 1 dollar. At iteration i, you could say that you have gained i dollars (technically, you still have not gained any money, because your pile is still infinite). So your cash c at iteration i is given by

$$c_i = \infty + i = \infty.$$

As i approaches infinity, your supply is clearly increasing, such that

$$c_\infty = \infty + \infty = \infty.$$

It may be true that for any iteration i, you will never see the bill labelled with i again, but it is also true that you always receive two bills in the exchange. Whether there is any proof that can support Hamkins' argument is given as an open problem. More details regarding the peculiarities arising with infinities are provided by Sorensen [Sor94], who discusses infinities in the context of decision theory.

5 Some Example Problems

In order to explore the importance of time with respect to the complexity of some problems, let us begin with an art gallery. This particular art gallery has been designed by a rather unimaginative architect, such that all of the rooms in the gallery are uniform, square, and each has four exits to similar rooms. The floor plan is illustrated in Figure 2. The art gallery is very large, and for the purposes of this example, assume that it extends infinitely in each direction. In each room is a security camera and an alarm system, which is tripped if art is damaged or stolen. The security guard has a station which contains a single monitor, on which he can observe any room in the gallery at any given time. If the alarm is tripped, he can switch to the camera in that room to see where the thief is and where he is going. Meanwhile, he can radio his colleague on the floor in the gallery, to guide him to the thief.

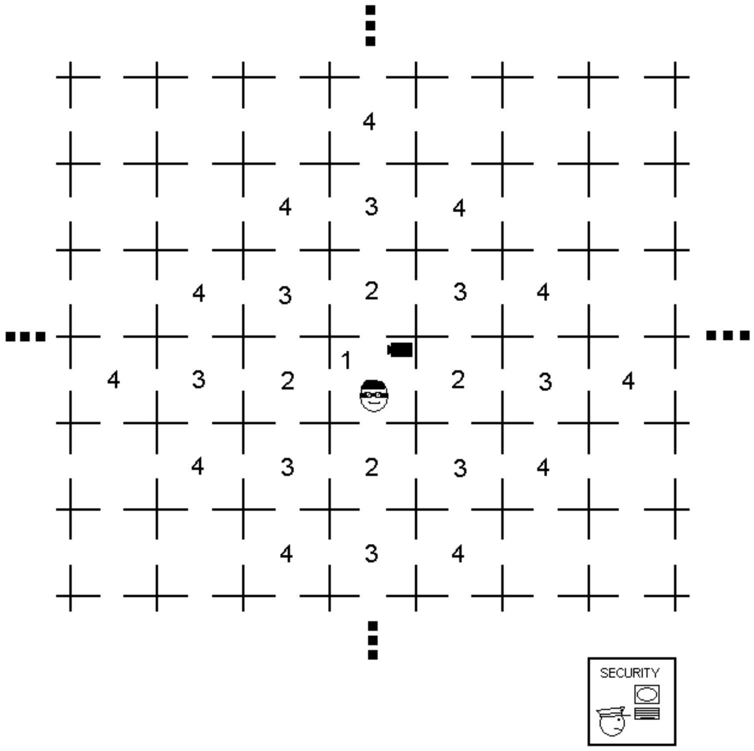

Figure 2: The layout of the art gallery is shown. The thief is in the middle of the gallery, and the security guard is shown in his office. The numbers in the rooms indicate which rooms the thief could reach with each successive time step.

Assume that the time taken for the thief to dash from one room into the next is a unit amount of time, and the time it takes the security guard to scan a room for the thief also takes the same time unit. The nature of the problem then is that for the security guard to scan N rooms of the art gallery takes N time units. If the guard is attentive and notices the alarm at time $t = 1$, then he knows the exact room that the thief is in, and the latter will be caught. If the guard does not notice until time $t = 2$, he now has 5 potential rooms that the thief may be in (he could have passed through a doorway into an adjacent room, or perhaps he remained in the initial room). At time step $t = 3$, there are now 13 rooms to be searched, and there are 25 rooms to be searched at time $t = 4$. The number of rooms to be searched at time t for this model is

$$N = t^2 + (t-1)^2.$$

Since the number of rooms is growing quadratically with t, having an accelerating machine to carry out the task could assure us of catching the thief in every instance. This machine would be designed so that it is capable of searching any given room in half the time it took to search the previous room. With such a system, the thief could have any amount of a head start, but he will be caught. The machine will have searched an infinite number of rooms in only twice the time it took to search the initial room, so we are guaranteed to have searched the room that the thief is in at some point.

We can arbitrarily increase the complexity of each step of a problem like this by adjusting the configuration of the rooms, but the guard equipped with an accelerating machine will always locate the thief. Suppose the doors are in the corners of the rooms, so that at time t the number of possible rooms the thief could be in is

$$N = (2t - 1)^2.$$

Again, this is not a problem. We could even fathom a three-dimensional configuration reminiscent of Jorge L. Borges' *Library of Babel* [Bor64]. If we had a passage in each corner of a room so that one could move from one room to any other sharing a vertex with the present room in a single time step, the possible number of rooms at time t grows only as

$$N = (2t - 1)^3,$$

and the thief can still be tracked down.

Finally, we could conceive of an accelerating thief. Each time the thief passes through a room, he learns more about the best way to escape and how to move through the art gallery. Suppose that this thief takes only half the time to pass through a room as he took to pass through the previous room. Now we have a thief who may give our accelerating machine a run for the money. Suppose our thief is in the second style of art gallery, where the passages are in the corner of each room. At step i, where $i = 1, 2, \ldots$, the time is

$$t = \frac{2^i - 1}{2^{i-1}},$$

and the number of rooms he could be at is

$$N = (2i - 1)^2.$$

Given that

$$i = \log_2\left(\frac{1}{2-t}\right) + 1,$$

when $t \geq 2$ the thief could be in any of an infinite number of rooms. Of course, to catch the thief all that is required is an accelerating machine that accelerates faster than the thief, such as one that triples in speed with every iteration. With this new machine, the time required to perform an infinite number of iterations is given by

$$\sum_{i=0}^{\infty} \frac{1}{3^i} = 1.5.$$

This pattern of increasing complexity in response to an increase in the computing ability of an accelerating machines resembles the proof of non-universality of accelerating machines, which will be revisited in Section 7 on limitations. For more examples of the kinds of problems that are well suited to accelerating machines, refer to the section on time-varying computational complexity in [Akl07b]. These include problems such as monitoring biological tissues or viruses, dealing with software viruses or spam, tracking moving objects in large volumes, addressing security issues, and modelling complex systems.

6 Implementations

In this section we examine the theoretical models that have been proposed for the actual implementation of an accelerating machine. Some believe that such machines are an impossibility [Ste91b, Cas97, Svo98]. Svozil [Svo98] presented a diagonalization argument showing that an accelerating machine would be able to compute its own Halting problem and thus be a logical impossibility, but this argument was subsequently disproved by Ord and Kieu [OK05]. There are many papers with proposed implementations, and no papers to date with a counterproof to the possibility of implementing an accelerating machine.

6.1 A Biological Implementation

Calude and Păun [CP04] discuss using biological systems to implement an accelerating machine. Their implementation involves decreasing the size of the reacting components or increasing the speed of the communication channels. They propose accomplishing these tasks using membrane computing. The membrane of a cell separates two volumes, and the compositions of the volumes usually differ. The channels across the membrane allow a certain kind of computation, and the speed of computation could be regarded as the time required for particles to traverse the membrane via these channels. The authors create a hierarchical system of membranes, which they suggest will be able to compute the Halting problem in a finite amount of time.

6.2 Appealing to the Laws of Physics

Tipler [Tip94] introduced a model that provides a possible base for the implementation of an accelerating machine, his aim being to provide examples of machine that could solve the Halting problem and thus refute the Church-Turing thesis. The model uses a system called the billiard ball computer, where parts of the machine are analogous to billiard balls such that when there is a collision between several of the balls, the energy is transferred between them according to Newtonian laws of motion. There have been works in theoretical physics demonstrating that it is possible for singularities to achieve accelerating rates of oscillation when they are arranged in particular configurations. This allows the singularity to move between other particles an infinite number of times within a finite period of time (for details, see [MM75] and [SX95]). This is possible given that the systems achieve an infinite energy by the laws of Newtonian physics. By using such singularities as the balls in the billiard ball computer, it would be possible to perform an infinite number of computations in a finite amount of time. This machine is an accelerating machine.

6.3 The Universe as Computer

There is a philosophical idea that the Universe itself is a computer [Zus70, Whe89, Dav01, Wol02, Dur04, LN04, Llo06], one that has also gained popular appeal by appearing in books and movies, such as, for examples, Isaac Asimov 's story *The Last Question* [Asi56] and Michael Crichton's *Timeline* [Cri99]; for other examples, see [HW03]. This concept is based in quantum mechanics, and holds that since every quantum particle has spin, the latter can be treated as information storage. Thus the entire universe is essentially comprised of information, and every interaction between particles changes the information stored there and could be considered a quantum computation. The universe is the only conceivable universal Turing machine, as any machine that could be simulated in the universe is in effect being simulated when it is implemented. There are different theories on the expansion rate of the universe, some hold that it will collapse again some day, some believe that it will reach an equilibrium point, and others hold that the universe is continuously expanding at an accelerating rate and will continue to do so forever (although admittedly the last theory does not hold favour with too many). If the latter is true, then the universe as a computer could be regarded as an accelerating machine.

6.4 Spacio-Temporal Patterning

Another possible implementation would be a massively parallel array. Assume that we have at our disposal an infinite array of identical processors, and that the problem we are trying to solve can be decomposed infinitely. This array can be used as an accelerating machine. Suppose that one processor is first used to perform the first iteration of the computation, and this takes one time unit. Next, the second iteration can be performed by two processors working in parallel. For many kinds of problems this can result in the iteration being performed in half the time or less. If we continue this style of division, the number of processors used is doubled, each iteration takes half the time of the previous one, and we have created an accelerating machine. We refer to this approach as *spacio-temporal patterning*.

Spacio-temporal patterning can be achieved, at least in theory, in a number of ways. For example, one may imagine that all the particles of the universe are used as a giant accelerating machine. Each particle serves as a processor. Beginning with one processor, the number of processors entering into action doubles at each step. Alternatively, one may view the accelerating machine as an entity that is permanently connected to a source of energy, and hence is constantly growing by transforming energy into computing agents. A third option would be to have processors that replicate themselves: At each step of the computation each processor makes a copy of itself, thus doubling the pool of processors available to perform that computational step.

Three remarks are in order here. First, we note that spacio-temporal patterning works (as an accelerating machine) provided that step (or iteration) i, where $i \geq 0$, can be decomposed into 2^i independent components, as assumed at the outset. Second, when indeed spacio-temporal patterning applies to the solution of a problem, this approach to implementing accelerating machines avoids the speed of light barrier: While it is true that each step is executed in half the time as the previous one, this is not due to a speeding up of the computing agents, but rather it results from the step being executed in parallel by twice as many processors as the previous step. We revisit this point in Section 7. Third, the question may be asked as to the temporal overhead involved in doubling the number of processors at each step. This point is addressed in the next section.

6.5 Parallel Universes

One way to achieve the spacio-temporal patterning configuration, while not incurring a time loss in doubling the number of processors, may be gained through the existence of an infinite number of *parallel universes* similar to ours (this is again a favourite topic in science fiction [HW03]; see, for example, Borges' *Garden of Forking Paths* [Bor64], Dick's *The Man in the High Castle* [Dic92], or Howitt's movie *Sliding Doors* [How98]). Parallel universes and quantum computation have been a topic of study for decades [Eve57, Whe89, Kak95, TW01, Teg03, Smo06]. Deutsch [Deu02] provides a detailed explanation of the operation of such a configuration (also known as the *multiverse* or the *many worlds*). Thus, we have the basis for just the configuration needed to implement spacio-temporal patterning: a quantum computer operating in parallel with others in an array, one in each of an infinite number of parallel universes. Since the processors are already *there*, each one or more in a different universe, there is no overhead involved in marshalling them to participate in the solution of the problem at hand.

6.6 Different Time Frames

Stewart [Ste91a] discussed the implementation of accelerating Turing machines as involving different time frames. The accelerating machine is able to perform an infinite amount of computations because he allows the acceleration to continue unabated by claiming that classical mechanics imposes no upper bounds on velocity. Thus the machine has its own time frame, in which an infinite number of operations may be performed in a finite amount of time for the external time frame of an observer (the machine's time frame is referred to as *real time*, and the observer's as *fake time*). This machine will appear to us to be an accelerating machine, although from the perspective of the machine it is using a constant amount of time per iteration.

One possible method of obtaining these separate time frames is through the use of Malament-Hogarth space-times [Pit90, Hog92, EN93, Hog94]. This is a curious sort of space-time where one can have local fields that will take an infinite amount of time to traverse. The proposition is that we have a machine that we start running on a problem (such as the Halting problem), and then we push it on a path carrying it through the heart of this field. We follow a path through normal space-time so that we will meet the machine on the other side of the field. From our perspective, the machine is accelerating. A toy version of this is illustrated in Figure 3. This configuration was described by Shagrir and Pitowsky [SP03] as the only known physical digital hypercomputer.

6.7 The Omega Point

Tipler [Tip94] introduced a phenomenon called the omega point (see also the discussions by Deutsch [Deu97] and Brown[Bro00]), which will be reached when the Universe collapses on itself if the Big Crunch model of the Universe is accurate. The essence of the argument for the omega point is that in order for the Big Crunch to occur, we must cross a threshold from having a Universe in the shape of a 3-sphere to that of a singularity. As the Universe approaches this threshold, it will begin to oscillate (according to this theory), distorting in

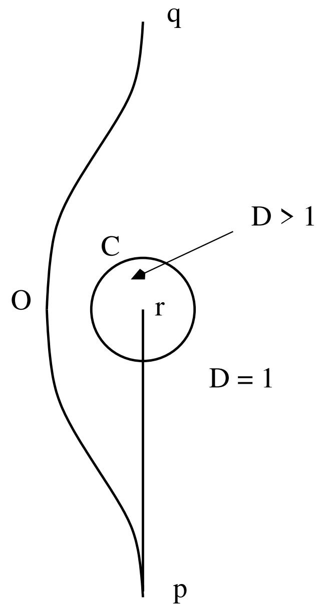

Figure 3: This is a toy example of a Malament-Hogarth space-time. The observer follows a path through space-time denoted by O, and the computer is sent along a path carrying it to point r. Their space-time paths diverge at point p. The value D denotes a sort of density of the space time field, and in the field C the value of D increases to a maximum of infinity at the point r. If the computer passes through r, it will have experienced an infinite amount of time once it meets the observer at point q on the other side of C. This figure has been adapted from [Hog94].

shape from a 3-sphere to various ellipsoids. With further compaction, the frequency of the oscillations increases, such that at the limit the frequency is infinite, thus an infinite number of oscillations will occur in a finite amount of time. Deutsch [Deu97] argues that it would be theoretically possible to construct a computer which would have access to an infinite amount of memory and be able to carry out an infinite number of computations at this point. This computer would have to be made of elementary particles and gravitational fields, as all matter would be destroyed as a result of the Crunch. Furthermore, the violence of the collapse of the Universe would be problematic for such a computer, as the oscillations will become increasingly violent, disturbing its operation. In order to address this difficulty, Deutsch requires that the computer manipulate gravity to stabilize the collapse so that its own integrity is preserved. This stabilization becomes increasingly difficult as the collapse of the Universe becomes imminent, so the computer must actually be an accelerating machine to persistently perform this stabilization. This assumes that we will have a perfect understanding of the quantum nature of particle physics and gravity so that we can manipulate gravity and the particles which comprise the memory of the computer as required. Tipler [Tip94] suggests that such a computer could be used as an accelerating machine by sending a message back in time from the end of the Universe with the required information (such as whether it halted on a given input). Here, it is taken for granted that it is possible for the message to be sent back through time, and that the computer would have sent the message had it halted. If no message is received, then after an infinite number of computations the Universe ended and the computer was destroyed. If one accepts the premise that some functions are only externally computable, then a difficulty with this approach is that something has to send the required information at the point that the Universe ends or afterwards (if there is any meaning to that). It is not clear how this would be accomplished.

7 Limitations

The limitations associated with accelerating machines are primarily those dictated by the laws of nature. In particular, there are physical properties that prohibit infinite computation. In addition, some apparent paradoxes arise when using accelerating machines. Finally, it has been shown that accelerating machines are not universal. This section examines some of these limitations in turn.

7.1 The Speed of Light

Ord [Ord02] describes a possible limitation to the accelerating machine with respect to physics. The nature of the accelerating machine is that it is capable of performing an infinite number of computations in a finite amount of time. Let us continue using the example of a Turing machine, where there is a read/write head moving along a tape to perform the computations. The machine must move the head twice as fast with every iteration in order to achieve the proposed model of acceleration without loss of generality. Before too long, the read/write head necessarily must be moving in excess of the speed of light in order to perform the calculations at this accelerating rate; this is clearly in violation of the laws of physics. Ord qualifies the speed of light criterion (we can never say something is outright impossible) by stating that it is theoretically possible to have objects moving faster than the speed of light as long as they always have been. Building a read/write head into our Turing machine that is constantly moving at a rate faster than the speed of light would be a tall order, however, since at some point it would have to be connected to something that was not moving at this speed. A work around for this problem, as proposed by Davies [Dav01], is to reduce the distance that the head is required to travel at each iteration by a half so that the head's speed can remain constant. The problem with this approach is that a point is reached where infinite spatial precision is required from the read/write head, which Ord points out is limited by the principles of quantum mechanics. Spacio-temporal patterning is an entirely different way to deal with the speed of light barrier (through, for example, doubling the number of processors at every step), as discussed in Section 6.4. Other physical limitations on accelerating machines, related to the speed of light by Einstein's famous $E = mc^2$ formula, are mass and energy.

7.2 Infinities and Infinities

Copeland [Cop02a] draws a distinction between different infinities in his work on accelerating machines. Turing defined an *effective procedure* as one where the process can be performed by working in an unintelligent but disciplined manner. The bound on this definition of an effective procedure is that there is a finite amount of work, and thus the process can be completed by performing a finite number of computations. Copeland proposes an alternate definition of an effective procedure, which is denoted with capitalization: an *Effective procedure*. The distinction is that an Effective procedure only requires that the computer use a finite amount of time to perform the computation, and this could potentially comprise an infinite number of computations.

Infinities are intuitively problematic, and yet they are at the very heart of accelerating machines. The problem of infinities is a recurring theme in the work of Weinberg [Wei92], when he discusses the infinities that arise in particle physics. He differentiates between distinct infinities, such as differentiating $\sum_{i=1}^{\infty} i$ from $\sum_{i=1}^{\infty} \frac{1}{i}$. Each of these sums are infinite, but they can be treated differently if being used to eliminate other infinities in a problem. As pointed out earlier, Cantor was the pioneer in this area; his work with infinities and proof of the hierarchy of infinities using set theory was the breakthrough that earned general acceptance of the concept of infinity [Bar05].

7.3 To Vanish or Not to Vanish

An existential problem may arise if we are able to construct an accelerating machine. If we are to build an accelerating machine that operates within our spacetime, it will disappear when it finishes its computations according to the Newtonian laws of motion [Cop02a]. This provides a solution to the problem of what an accelerating machine should do after it has performed an infinite number of operations, for what could be cleaner than a simple disappearance? Another view would be that the machine just performs a transfinite number of operations. Hamkins [Ham02] discusses this property thoroughly. Shagrir [Sha04] suggests that if the machine continues to have a physical existence, the states of the machine no longer would correspond to the states of the Turing machine. This is a convenient (and plausible) hypothesis, for it would allow for the solving of the Halting problem and such (especially through the use of a hooter), but avoids the super task specific problem concerning the final state of the machine.

7.4 The Universe as Computer Revisited

Is the Universe an accelerating machine? If one subscribes to the theory that the Universe is a computer, then the Universe has performed the maximum possible number of different operations since the Big Bang, and this is certainly a finite number although stupendously large [LN04]. On the other hand, the Universe as a computer hypothesis is not a foregone conclusion. Many people believe that there are fundamentally uncomputable processes in the Universe, such as human consciousness (see for example Descartes [Des00] and Penrose [Pen90], among others [Nag74, Pla74, Jac82]).

7.5 No Finite Computer is Universal

Finally, there is the non-universality of accelerating machines. One of the fascinations with accelerating machines, and hypercomputers in general, is the promise of unbridled computational power and the ability to solve outrageously difficult problems, or possibly any conceivable problem, as discussed in this paper. It has been hypothesized that accelerating machines may be universal, but there exists a proof that this is not the case [Akl05, Akl07a]. This result is not even restricted to accelerating Turing machines, which do not accept input from the outside world during their operation: the non-universality property holds for accelerating machines in general. A differentiation should be made clear at this point. In the context of this paper, there are two conventional notions of universality:

- The universal Turing machine. This refers to Turing's definition of computability: A function is computable by the universal Turing machine if and only if that function can be computed by an ordinary Turing machine. The universal Turing machine is not a hypercomputer, it cannot solve the Halting problem. It is fully realizable, and this concept of universality is ironclad and indisputable [Sud06].

- The universal computer. This is the notion of the universal computer that can simulate any computation that is possible on any other machine. The existence of such a universal computer is a foundational principle in computer science. This 'universality principle' (also known, as mentioned in Section 2.1, as the 'Church-Turing thesis' when the 'universal computer' in question is taken to be the Turing machine) is captured by the following quote:

  > "As far as we know, no device built in the physical universe can have any more computational power than a Turing machine. To put it more precisely, any computation that can be performed by any physical computing device can be performed by any universal computer, as long as the latter has sufficient time and memory." [Hil98]

  Unfortunately, the 'universality principle' is a fallacious concept, and the 'universal computer' is a myth, as shown in [Akl05].

The crux of the argument against the 'universality principle' rests on the requirement that the configuration of a putative 'universal computer' be specified at the outset. For example, we can specify that a machine M_1 performs k operations per time unit. Another valid specification, would be that machine M_2 *doubles* in speed with every iteration. Whatever the specifications, they must be concrete, finite, and fixed. Finiteness here refers to the number of operations that the machine can perform per step, requiring a unit of time. Finiteness does not refer to the size of the machine's memory or to the total time spent by the machine to solve a problem, both of which are allowed to be unbounded.

Once the specifications are in place, it is then easy to show that the 'universal computer' is unable to solve certain problems. Referring to the examples in the previous paragraph, machine M_1 is unable to solve a problem where $k+1$ operations are required per time unit. Similarly, machine M_2 fails if the complexity of the problem to be solved *triples* with every iteration. In general, for any given finite complexity that a computer is capable of at a given time step, it is possible to create a new problem that requires more complexity per step, *otherwise the computation is not possible.* The new problem is eminently solvable on another computer with the proper resources, but *that* computer is in turn defeated by a newer problem requiring even more resources, and so on.

It is worth repeating that this result remains true even when the purported 'universal computer' is endowed with an infinite memory, and is allowed to compute for an unlimited amount of time. Several computations are described in [Akl06a, Akl06b, Akl07a, Akl07b] that disprove the 'universality thesis'. They include computations with time-varying variables, with time-varying computational complexity, with rank-varying computational complexity, with interacting physical variables, and with global mathematical constraints.

8 Conclusions

We have shown in this paper that although there exists a significant body of research dedicated to accelerating machines, there are still many unknowns. The most pressing issue would be to determine conclusively whether an accelerating machine can actually be implemented in a useful way. We have shown that there are many proposed implementations, but none has ever been attempted. On the other hand, it has never been conclusively proven that accelerating machines cannot be built successfully.

As well, there is no clear definition that has been found for the kinds of problems that accelerating machines are capable of solving. They are clearly capable of solving tasks beyond those of a standard Turing machine, but it has also been demonstrated that they are not universal and have limitations. In addition, it has been shown that there are unusual properties of algorithms once an infinite number of iterations of a task have been performed, and this must be accounted for. Perhaps there is a new sub-class of super tasks that are tractable given an accelerating machine with some specified finite rate of acceleration. The identification of this sub-class remains an open problem.

References

[Akl97] S.G. Akl. *Parallel Computation: Models And Methods.* Prentice Hall, Upper Saddle River, New Jersey, 1997.

[Akl05] S.G. Akl. The myth of universal computation. In R. Trobec, P. Zinterhof, M. Vajteršic, and A. Uhl, editors, *Parallel Numerics, Part 2, Systems and Simulation*, pages 211–236. University of Salzburg, Salzburg, Austria and Jožef Stefan Institute, Ljubljana, Slovenia, 2005.

[Akl06a] S.G. Akl. Conventional or unconventional: Is any computer universal?. In A. Adamatzky and C. Teuscher, editors, *From Utopian to Genuine Unconventional Computers*, pages 101–136. Luniver Press, Frome, United Kingdom, 2006.

[Akl06b] S.G. Akl. Three counterexamples to dispel the myth of the universal computer. *Parallel Processing Letters*, 16(3):381–403, September 2006.

[Akl07a] S.G. Akl. Even accelerating machines are not universal. *International Journal of Unconventional Computing*, in press, 2007.

[Akl07b] S.G. Akl. Evolving computational systems. In S. Rajasekaran and J.H. Reif, editors, *Parallel Computing: Models, Algorithms, and Applications*. Taylor and Francis, CRC Press, Boca Raton, Florida, 2007.

[Asi56] I. Asimov. The Last Question. *Science Fiction Quarterly*, November 1956.

[Bar05] J.D. Barrow. *The Infinite Book - A short guide to the boundless, timeless and endless*. Johnathan Cape, Random House, London, 2005.

[Ben62] P. Benacerraf. Tasks, super-tasks, and the modern eleatics. *The Journal of Philosophy*, 59(24):765–784, 1962.

[Bla26] R. Blake. The paradox of temporal process. *Journal of Philosophy*, 23:645–654, 1926.

[BJ80] G.S. Boolos and R.C. Jeffrey. *Computability and Logic*. Cambridge University Press, Cambridge, England, 1980.

[Bor64] J.L. Borges. *Labyrinths*. New Directions Publishing, New York, 1964.

[Bro00] J. Brown. *The Quest for the Quantum Computer*. Touchstone, New York, 2000.

[CP04] C.S. Calude and Gh. Păun, Bio-steps beyond Turing. *BioSystems*, 77:175–194, 2004.

[Cas97] J.L. Casti. Computing the uncomputable. *Complexity*, 2(3):7–12, 1997.

[Cle02] C.E. Cleland. On effective procedures. *Minds and Machines*, 12:159–179, 2002.

[Cop98a] J. Copeland. Even Turing machines can compute uncomputable functions. In C. Calude, J. Casti, and M. Dinneen, editors, *Unconventional Models of Computation*, pages 150–164. Springer-Verlag, Singapore, 1998.

[Cop98b] J. Copeland. Super Turing-machines. *Complexity*, 4(1):30–32, 1998.

[Cop02a] B.J. Copeland. Accelerating Turing machines. *Minds and Machines*, 12:281–301, 2002.

[Cop02b] B.J. Copeland. Hypercomputation. *Minds and Machines*, 12:461–502, 2002.

[Cop04a] B.J. Copeland. *The Essential Turing*. Clarendon Press, Oxford, 2004.

[Cop04b] B.J. Copeland. Hypercomputation: philosophical issues. *Theoretical Computer Science*, 317:251–267, 2004.

[CP99] B.J. Copeland and D. Proudfoot, Alan Turing's forgotten ideas in computer science. *Scientific American*, 280(4):77–81, 1999.

[CS99] B.J. Copeland and R. Sylvan. Beyond the universal Turing machine. *Australasian Journal of Philosophy*, 77(1):46–67, 1999.

[Cri99] M. Crichton. *Timeline*. Knopf, New York, 1999.

[Dav00] M. Davis. *The Universal Computer: The Road from Leibniz to Turing*. W.W. Norton, New York, 2000.

[Dav01] E.B. Davies. Building infinite machines. *British Journal for Philosophy of Science*, 52:671–682, 2001.

[Des00] R. Descartes. Meditations ii,iv. In J.S. Crumley II, editor, *Problems in Mind*, pages 21–33. Mayfield Publishing Company, Toronto, 2000.

[Deu97] D. Deutsch. *The fabric of reality*. Penguin Books, London, 1997.

[Deu02] D. Deutsch. The structure of the multiverse. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 458(2028):2911–2923, 2002.

[Dic92] P.K. Dick. *The Man in the High Castle*. Vintage, New York, 1992.

[Dur04] J. Durand-Lose. Abstract geometrical computation for black hole computation. Research Report No. 2004-15, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, Lyon, France, April 2004.

[EN93] J. Earman and J. Norton. Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science*, 5:22–42, 1993.

[EW03] E. Eberbach and P. Wegner. Beyond Turing machines. *The Bulletin of the European Association for Theoretical Computer Science (EATCS Bulletin)*, 81:279–304, 2003.

[EN02] G. Etesi and I. Németi. Non-Turing computations via Malament-Hogarth spacetimes. *International Journal of Theoretical Physics*, 41(2):341–370, February 2002.

[Eve57] H. Everett. Relative state formulation of quantum mechanics. *Review of Modern Physics*, 29:454–462, 1957.

[Ham02] J.D. Hamkins. Infinite time Turing machines. *Minds and Machines*, 12:521–539, 2002.

[HW03] T. Hey and P. Walters. *The New Quantum Universe*. Cambridge University Press, Cambridge, England, 2003.

[Hil98] D. Hillis. *The Pattern on the Stone*. Basic Books, New York, 1998.

[Hog92] M.L. Hogarth. Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, 5:173–181, 1992.

[Hog94] M.L. Hogarth. Non-Turing computers and non-Turing computability. *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 1:126–138, 1994.

[How98] P. Howitt. *Sliding Doors*. http://www.imdb.com/title/tt0120148/, 1998.

[Jac82] F. Jackson. Epiphenomenal qualia. *The Philosophical Quarterly*, 32:127–136, 1982.

[Kak95] M. Kaku. *Hyperspace: A Scientific Odyssey Through Parallel Universes, Time Warps, and the Tenth Dimension*. Oxford University Press, Oxford, England, 1995.

[Kie03] T.D. Kieu, Computing the noncomputable. *Contemporary Physics*, 44(1):51–71, January-February 2003.

[Llo06] S. Lloyd. *Programming the Universe*. Knoff, New York, 2006.

[LN04] S. Lloyd and J. Ng. Black hole computers. *Scientific American*, 291(11):53–61, November 2004.

[MM75] J. Mather and R. McGehee. Solutions of the collinear four-body problem which become unbounded in finite time. *Dynamical Systems, Theory and Applications*, Lecture Notes in Physics, 38:573–597, 1975.

[Nag74] T. Nagel. What is it like to be a bat? *Philosophical Review*, 83:435–450, 1974.

[NA06] M. Nagy and S.G. Akl. Quantum measurements and universal computation. *International Journal of Unconventional Computing*, 2(1):73–88, 2006.

[OK05] T. Ord and T.D. Kieu. The diagonal method and hypercomputation. *British Journal for the Philosophy of Science*, 56:147–156, 2005.

[Ord02] T. Ord. Hypercomputation: computing more than the Turing machine. `http://arxiv.org/abs/math.LO/0209332`, 2002.

[Pen90] R. Penrose. *The Emperor's New Mind*. Oxford University Press, Oxford, 1990.

[Pit90] I. Pitowsky. The physical church thesis and physical computational complexity. *Iyyun: The Jerusalem Philosophical Quarterly*, 39:81–99, 1990.

[Pla74] A. Plantinga. *The Nature of Necessity*. Clarendon Press, Oxford, England, 1974.

[Rus15] B. Russell. *Our Knowledge of the External World as a Field for Scientific Method in Philosophy*. Open Court, Chicago, 1915.

[Rus36] B. Russell. The limits of empiricism. *Proceedings of the Aristotelian Society*, 36:131–150, 1936.

[SX95] D.G. Saari and Z.(J.) Xia. Off to infinity in finite time. *Notes of the AMS*, 42(5):538–546, 1995.

[Sha04] O. Shagrir. Super-tasks, accelerating Turing machines and uncomputability. *Theoretical Computer Science*, 317:105–114, 2004.

[SP03] O. Shagrir and I. Pitowski. Physical hypercomputation and the Church-Turing thesis. *Minds and Machines*, 13(1):87–101, 2003.

[Sie99] H.T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing limit*. Birkhäuser, Boston, 1999.

[Smo06] L. Smolin. *The Trouble with Physics*. Houghton Mifflin, New York, 2006.

[Sor94] R. Sorensen. Infinite decision theory. In J. Jordan, editor, *Gambling on God: Essays on Pascal's Wager*, pages 139–159. Rowman and Littlefield, Savage, Maryland, 1994.

[Sta90] M. Stannett. X-machines and the halting problem: Building a super-Turing machine, *Formal Aspects of Computing*, 2(4):331–341, 1990.

[Sta04] M. Stannett. Hypercomputational models. In C. Teuscher, editor, *Alan Turing: Life and Legacy of a Great Thinker*, pages 135–157. Springer-Verlag, Berlin, 2004.

[Ste02] E. Steinhart. Logically possible machines. *Minds and Machines*, 12:259–280, 2002.

[Ste91a] I. Stewart. Deciding the undecidable. *Nature*, 352:664–665, 1991.

[Ste91b] I. Stewart. The dynamics of impossible devices. *Nonlinear Science Today*, 1:8–9, 1991.

[Sud06] T.A. Sudkamp. *Languages and Machines*. Addison Wesley, New York, 2006.

[Svo98] K. Svozil. The Church-Turing thesis as a guiding principle for physics. In C. Calude, J. Casti, and M. Dinneen, editors, *Unconventional Models of Computation*, pages 371–385. Springer-Verlag, Singapore, 1998.

[Teg03] M. Tegmark. Parallel universes. *Scientific American*, 288(5):41-52, May 2003.

[TW01] M. Tegmark and J.A. Wheeler. 100 years of the quantum. *Scientific American*, 284(2):68–75, February 2001.

[Tho54] J.F. Thompson. Tasks and super-tasks. *Analysis*, 15:1–13, 1954.

[Tip94] F.J. Tipler. *The Physics of Immortality: Modern Cosmology, God and the Resurrection of the Dead*. Macmillan, London, 1995.

[Tur37] A.M. Turing. On computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society, Ser. 2*, 42:230–265, 1936, corrections Ibid., 43:544-546, 1937.

[Tur39] A.M. Turing, Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, Ser. 2, 45:161–228, 1939.

[WG97] P. Wegner and D. Goldin, Computation beyond Turing Machines. *Communications of the ACM*, 46(4):100–102, May 1997.

[Wei92] S. Weinberg. *Dreams of a final theory.* Pantheon Books, New York, 1992.

[Wey27] H. Weyl. *Philosophie der Mathematik und Naturwissenschaft.* Oldenburg, Munich, 1927.

[Wey49] H. Weyl. *Philosophy of Mathematics and Natural Science.* Princeton University Press, Princeton, New Jersey, 1949.

[Whe89] J.A. Wheeler. Information, physics, quanta: The search for links. In *Proceedings of the 3rd International Symposium on Foundations of Quantum Mechanics in Light of New Technology*, pages 354–368. Tokyo, 1989.

[Wol02] S. Wolfram. *A New Kind of Science.* Wolfram Media, Champaign, Illinois, 2002.

[Zus70] K. Zuse. *Calculating Space.* MIT Technical Translation AZT-70-164-GEMIT, Massachusetts Institute of Technology (Project MAC), Cambridge, Mass. 02139, 1970.