The Physical Church-Turing Thesis: Modest or Bold?[1]

Gualtiero Piccinini

University of Missouri – St. Louis

Email: piccininig@umsl.edu

Abstract: This paper defends a modest version of the Physical Church-Turing thesis (CT). Following an established recent trend, I distinguish between what I call Mathematical CT—the thesis supported by the original arguments for CT— and Physical CT. I then distinguish between bold formulations of Physical CT, according to which any physical process—anything doable by a physical system—is computable by a Turing machine, and modest formulations, according to which any function that is *computable* by a physical system is computable by a Turing machine. I argue that Bold Physical CT is not relevant to the epistemological concerns that motivate CT and hence not suitable as a physical analogue of Mathematical CT. The correct physical analogue of Mathematical CT is Modest Physical CT. I propose to explicate the notion of physical computability in

[1] Thanks to Darren Abramson, Dave Chalmers, Jack Copeland, Martin Davis, John Earman, Corey Maley, John Norton, Toby Ord, Michael Rabin, Oron Shagrir, Wilfried Sieg, and Mark Sprevak, among others, for helpful discussions on this topic. Thanks to Oron Shagrir and the anonymous referees for helpful comments. Material that led to this paper was presented at the 2005 Eastern APA (meeting of the Society for Machines and Mentality) and to the Department of Mathematics and Computer Science at the University of Missouri – St. Louis in February 2006. Thanks to the audiences and the APA commentator, Jack Copeland. Thanks to Jeff Dauer and James Virtel for editorial assistance. This material is based in part upon work supported by a University of Missouri – St. Louis Research Award and National Science Foundation Grant No. SES-0924527.

terms of a usability constraint, according to which for a process to count as relevant to Physical CT, it must be usable by a finite observer to obtain the desired values of a function. Finally, I suggest that proposed counterexamples to Physical CT are still far from falsifying it because they have not been shown to satisfy the usability constraint.

The Church-Turing thesis (CT) may be stated as follows:

> **CT**: Any function that is intuitively computable is computable by some Turing machine (Turing-computable for short). (Figure 1)

Or in Alan Turing's terms, CT pertains to functions that may be "naturally regarded as computable" (Turing 1936-7, p. 135).[2]

The *converse* of CT is the thesis that any Turing-computable function is intuitively computable. This is true in the sense that any competent human being (or digital computer) can carry out the computation in question for as long as she has resources such as time, writing material, etc. She can do so because she can execute the instructions that make up the program that defines the Turing machine. It is irrelevant that for most inputs and programs, she lacks sufficient resources to complete the computation (or even to read the whole input and program, if they are long enough).

[2] Turing talked about computable numbers rather than functions, but that makes no difference for present purposes.

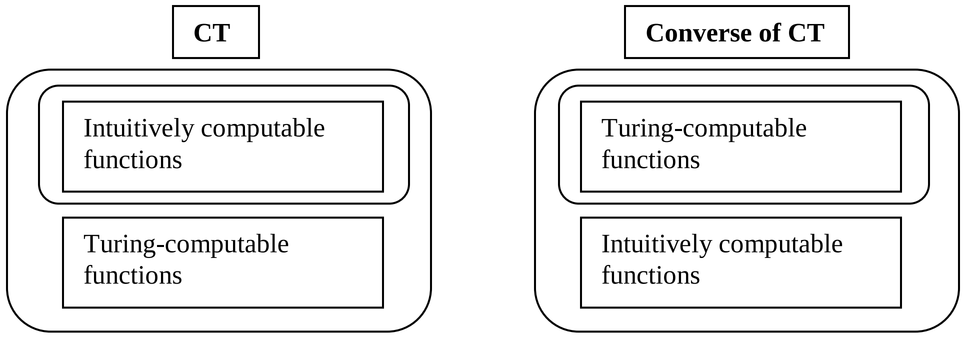
CT
Intuitively computable functions
Turing-computable functions
Converse of CT
Turing-computable functions
Intuitively computable functions

Figure 1. Venn diagrams representing The Church-Turing thesis and its converse.

While the converse of CT is relatively easy to establish, CT itself is more difficult to assess. For starters, how should we explicate "intuitively" or "naturally regarded as"? In recent decades, the literature on CT has mushroomed. While some aspects of CT have become clearer, little consensus has emerged on how to formulate and evaluate CT. This paper offers some suggestions on how to make progress.

Following an established recent trend, I will distinguish between what I call Mathematical CT, which is the thesis supported by the original arguments for CT, and Physical CT, which pertains to the computational limitations of physical processes. In addition, I will distinguish between bold formulations of Physical CT, according to which any physical process—anything doable by a physical system—is computable by a Turing machine, and modest formulations, according to which any function that is *computable* by a physical system is computable by a Turing machine.

Mathematical CT: Any function that is computable by following an effective procedure is Turing-computable. (Figure 2)

Bold Physical CT: Any physical process is Turing-computable. (Figure 3)

Modest Physical CT: Any function that is physically computable is Turing-computable. (Figure 4)

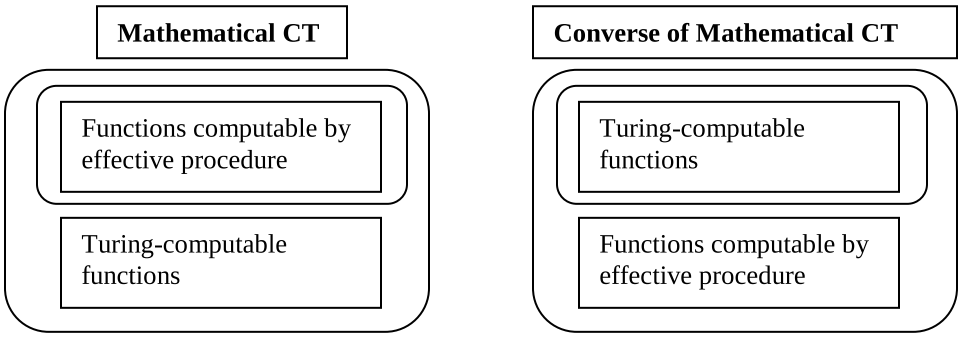
Mathematical CT
Functions computable by effective procedure
Turing-computable functions
Converse of Mathematical CT
Turing-computable functions
Functions computable by effective procedure

Figure 2. The Mathematical Church-Turing thesis and its converse.

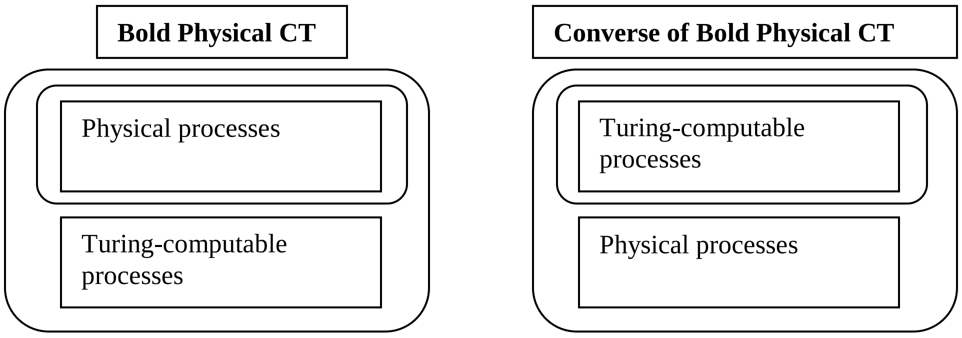
Bold Physical CT
Physical processes
Turing-computable processes
Converse of Bold Physical CT
Turing-computable processes
Physical processes

Figure 3. The Bold Physical Church-Turing thesis and its converse.

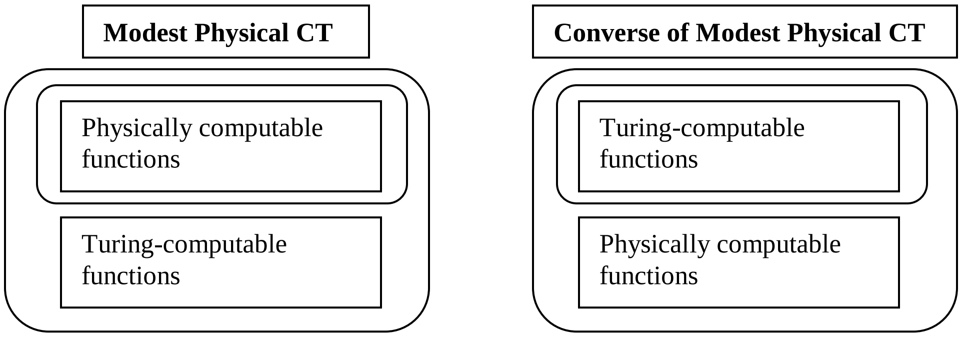
Modest Physical CT
Physically computable functions
Turing-computable functions
Converse of Modest Physical CT
Turing-computable functions
Physically computable functions

Figure 4. The Modest Physical Church-Turing thesis and its converse.

The converses of these theses are easily established on the same grounds as the converse of (generic) CT. The converse of Mathematical CT is that any Turing-computable function is computable by following an effective procedure. The converse of Bold Physical CT is that any Turing-computable process can be physically realized. Finally, the converse of Modest Physical CT is that any Turing-computable function can be physically computed. All of these are true for the same reason that the converse of generic CT is true. The computational behavior of any Turing machine is exhaustively specified by an effective procedure consisting of Turing machine instructions. Any competent human being or digital computer can follow those instructions and carry out the computations for as long as they have appropriate resources. While performing the computation, either a computing human or a digital computer physically realize the relevant Turing-computable process. Although such physical realizations may be unable to *complete* most Turing-computable processes for lack of sufficient resources, this is beside the point.

Once again, however, the specialized versions of CT—especially the physical versions—are more difficult to assess. It's not even clear which version of CT is the correct physical analogue of Mathematical CT. Accordingly, this paper has two main purposes: to determine which version of Physical CT is the physical analogue of Mathematical CT and to determine whether any version of Physical CT is plausible.

In Section 1, a few remarks about Mathematical CT will establish that the notion of computability that grounds CT is an epistemological notion. In the case of Mathematical CT, it is the notion of what can be deduced by a finite observer exploiting procedures that are executable, automatic, uniform, and reliable. More precisely, a function defined over a denumerable domain (such as the natural numbers) is *computable* just in case a finite observer can find the function's values by a procedure that is executable, automatic, uniform, and reliable.

In Section 2, I will argue that Bold Physical CT is both implausible and irrelevant to the epistemological concerns that motivate CT properly so called—Bold Physical CT is not about what a finite observer can discover about the desired values of a function. Thus, contrary to what some authors assume, Bold Physical CT is not suitable as a physical analogue of Mathematical CT.

In Section 3, I will introduce and articulate a usability constraint on physical computation: for a physical process to count as a computation (and thus for it to be relevant to Physical CT properly so called), it must be usable by a finite observer to obtain the desired values of a function using a process that is executable, automatic, uniform, and reliable.

In Section 4, I will argue that the correct physical analogue of Mathematical CT is Modest Physical CT, which is grounded on a notion of physical computation based on the usability constraint. I will also argue that current proposals for machines that falsify Physical CT are still far from doing so because they have not been shown to satisfy the usability constraint. In conclusion, Modest Physical CT is the correct physical formulation of CT and is plausible in light of current evidence.

Of course, many others have previously suggested or assumed that for a physical process to count as a genuine computation, it must be *usable*. But previous efforts in this direction have not clarified what "usable" means and have failed to persuade the many authors who remain largely indifferent to considerations of usability. In light of this, I will formulate the relevant notion of usability in terms of explicitly justified sub-constraints. These sub-constraints delimit the scope of the version of Physical CT that deserves its name, i.e., the modest version.

1. The Mathematical Church-Turing Thesis

Many authors assume that the intuitive sense of "computable" has to do with what can be computed by physical systems. This is too simplistic without some qualifications. CT was originally proposed and evaluated by a group of mathematical logicians, who were investigating the foundations of mathematics not the general properties of physical systems. They were concerned with what can be established by following certain procedures, namely, effective procedures for generating values of functions defined over effectively denumerable domains, such as the natural numbers or strings of letters from a finite alphabet. This notion of effective procedure was closely related to the notion of proof within a formal logical system (as explicated by, e.g., Church 1956, §7).

An effective procedure for evaluating a given function may be informally characterized as follows:

- *Executability*: The procedure consists of a *finite* number of *deterministic instructions* (i.e., instructions determining a unique next step in the procedure), which have *finite* and *unambiguous* specifications commanding the execution of a *finite* number of *primitive operations*.
- *Automaticity*: The procedure requires *no intuitions* about the domain (e.g., intuitions about numbers), no ingenuity, no invention, and no guesses.

- *Uniformity*: The procedure is the same for each possible argument of the function.[3]
- *Reliability*: When the procedure terminates, the procedure generates the *correct* value of the function for each argument after a *finite* number of primitive operations are performed.

To determine what can be accomplished by following effective procedures, different authors proposed and studied different formalisms, such as Turing machines, general recursive functions, and λ-definable functions. They suggested that these formalisms capture in a precise way the informal notion of what can be calculated by effective procedures. Since Alonzo Church and Alan Turing were the first to make this suggestion, Stephen Kleene dubbed it the *Church-Turing thesis*.[4]

The logicians who initially proposed and endorsed CT did not base their defense of CT on the limitations of physical systems. Their main reasons were the following (cf. Kleene 1952, § 62, § 67):

1) Lack of counterexamples. Every function known to be effectively calculable, and every operation for defining a function effectively from other functions, has been shown to be Turing-computable.
2) Failure of diagonalization. Diagonalization over Turing machines, which might be expected to yield a function that is not Turing-computable, does not lead outside the class of Turing-computable functions.

[3] This condition rules out Kálmar's "arbitrary correct methods," which may change from one argument of a function to another (Kálmar 1959).
[4] For more on the history of computability theory, including more details on the epistemological concerns that motivated it, see Davis 1982, Shapiro 1983, Gandy 1988, Sieg 1994, 1997, 2005, 2006a, Soare 1999, Piccinini 2003, Copeland 2006, Hodges 2006, and Shagrir 2006.

3) Confluence. A number of authors, working with different primitive notions, defined different mathematical formalisms and proposed that the functions definable within their formalism be identified as the class of effectively calculable functions. Some of the earliest formalisms are: general recursiveness (Gödel 1934), λ-definability (Church 1932, Kleene 1935), Turing-computability (Turing 1936-7), and reckonability (Gödel 1936). These notions have been proven to be extensionally equivalent to one another in the sense that any function that falls within one of these formally defined classes falls within all of them. Since the different formalisms have little in common and yet they all define the same class of functions, it is likely that the class of functions so defined is the intended one.[5]

4) Turing's argument. A Turing machine seems capable of reproducing any operation that a human being can perform while following an effective procedure (Turing's main argument for CT in his 1936-7 paper).

I will call these the *original arguments* for CT.

It doesn't take a detailed analysis to see that the original arguments have little to do with the general properties of physical systems. The view that CT pertains directly to what can be computed by physical systems makes no sense of the fact that most logicians and theoretical computer scientists accept CT on the grounds of one or more of the original arguments. Such a view implies that logicians are confused as to the proper justification for CT. Of course, they might be confused in many ways, but it is implausible that they are confused that badly. It is more charitable to distinguish two

[5] The argument from confluence is the most popular in computer science, where CT is often formulated as the thesis that any new formalization of the notion of effectively calculable functions will yield the same old class—that of the Turing-computable functions (e.g., Newell 1980).

versions of CT. One, *Mathematical CT*, is the version supported by the original arguments.

> **Mathematical CT**: Any function that is computable by following an effective procedure is Turing-computable.

The other is the version that pertains to what can be done or computed by physical systems. It may be called *Physical CT* (following Pitowsky 1990).

During the last two decades or so, the distinction between Mathematical and Physical CT has become fairly well established, on grounds similar to those I discussed.[6] The distinction between Mathematical and Physical CT is an important clarification, but it by no means completes the task of understanding CT. If anything, it doubles the task. We now have to understand two theses instead of just one.

There is much to say about Mathematical CT. The two largest areas of disagreement are how to further explicate Mathematical CT and whether it is rigorously provable.[7] I will not address those debates, because I wish to focus on Physical CT. I will conclude this all too brief discussion of Mathematical CT by reiterating a few points that *should*[8] be uncontroversial. First, Mathematical CT—however it ought to be further explicated—is supported by the original arguments. Second, one or more of the original arguments are good enough to establish that Mathematical CT is true. Third, Mathematical CT is formulated in terms of functions defined over effectively

[6] E.g., Gandy 1980, Earman 1986, Odifreddi 1989, Pitowsky 1990, Mundici and Sieg 1995, Sieg 2002, Shagrir 1997, Copeland 2002c, Shagrir and Pitowsky 2003, Fitz 2006, Button 2009. For examples of people who conflate or fail to distinguish explicitly between Mathematical and Physical CT, see Churchland and Churchland 1990, Cleland 1993, Hogarth 1994, Siegelmann 1995, Smith 2006a.

[7] On how to explicate Mathematical CT, see Kleene 1952, 1987, Gandy 1980, Shapiro 1981, 1993, Shagrir 2002, Sieg 2002, 2008, Copeland 2002c, Rescorla 2007, and Arkoudas 2008. On whether it's provable, see Gandy 1980, Mendelson 1990, Shapiro 1993, Sieg 1994, 1997, Folina 1998, Black 2000, and Smith 2007.

[8] For a recent though unsuccessful attempt to challenge the notion of effective procedure that grounds Mathematical CT, see Hogarth 2004. For a rebuttal, see Button 2009.

denumerable domains. Hence, it does not apply directly to functions defined over domains of larger cardinality. Fourth, the notion of computation in whose terms Mathematical CT is formulated is an epistemological notion: it is the notion of what can be established by following effective procedures defined over effectively denumerable domains.[9] Fifth, Mathematical CT does not pertain directly to what can be computed by physical systems in general.

Well, then, what should we say about what can be computed by physical systems in general?

2. The Bold Physical Church-Turing Thesis

We need a physical version of CT that we can evaluate. This requires two things. First, we need to replace Mathematical CT's appeal to effective procedures with an appeal to physical processes. Second, we need this appeal to physical processes to be relevant to the epistemological notion of computation—the generation of values of functions over strings, the solution of mathematical problems, etc.—that motivates CT in the first place.

The simplest starting point is the following:

Bold Physical CT: Any physical process is Turing-computable.

This is not very precise, but we need to start here because in the literature, Physical CT is often formulated in roughly these terms. Bold Physical CT ranges over all physical processes—implicitly, it counts all physical processes as *computations*.

[9] Cf. Gödel's remark:

> Tarski has stressed in his lecture (and I think justly) the great importance of the concept of general recursiveness (or Turing's computability). It seems to me that this importance is largely due to the fact that with this concept one has for the first time succeeded in giving an absolute definition of an interesting *epistemological* notion, i.e., one not depending on the formalism chosen (Gödel 1946, 84, emphasis added).

Sometimes, versions of (Bold) Physical CT are formulated that are more clear and precise, and hence more clearly evaluable. The following is a list of representative formulations:

(A) Any physical process can be simulated by some Turing machine (e.g., Deutsch 1985, Wolfram 1985, Pitowsky 2002).

(B) Any function over denumerable domains (such as the natural numbers) that is computable by an idealized "computing machine" that manipulates arbitrary real-valued quantities (as defined by Blum et al. 1998) is Turing-computable.[10]

(C) Any system of equations describing a physical system gives rise to computable solutions (cf. Earman 1986, Pour-El 1999). A solution is said to be computable just in case given computable real numbers as initial conditions, it returns computable real numbers as values. A real number is said to be computable just in case there is a Turing machine whose output effectively approximates it.

(D) For any physical system *S* and observable *W*, there is a Turing-computable function f: $\mathbf{N} \rightarrow \mathbf{N}$ such that for all $t \in \mathbf{N}$, $f(t)=W(t)$ (Pitowsky 1990).

Each of (A)-(D) deserves to be discussed in detail. Lacking space for that, I will limit my discussion to a few general points to the effect that none of these formulations are adequate physical analogues of Mathematical CT. I will argue that Bold Physical CT is both too strong and too indifferent to epistemological considerations to be relevant to the notion of computability that motivates CT. As a consequence, Bold Physical CT

[10] Thesis (B) is a special case. Unlike (A), (C), and (D), it does not range over all physical processes but is restricted to a certain specific "computations". Furthermore, Blum et al. do not formulate (B) as a version of Physical CT. Instead, they simply prove the opposite of (B). Nevertheless, Blum et al.'s result is often mentioned by critics of Physical CT, even though, as I will argue, Blum et al.'s "computations" are not computations in the epistemological sense that motivates CT and so should not be seen as counterexamples to Physical CT properly so called. Because of this, it is appropriate to include (B) as a version of Bold Physical CT. More on this below.

needs to be replaced by a more modest formulation of Physical CT, which pertains to what can be *computed* by physical means in a restricted sense that encompasses more than what can be accomplished by following effective procedures but does not include all physical processes. In the next section, considerations similar to those that show Bold Physical CT to be too strong will motivate a number of constraints on the notion of physical computability, which in turn will ground the formulation of Modest Physical CT.

2.1 Lack of Confluence

One of the original arguments for Mathematical CT is that seemingly disparate notions (Turing machines, general recursive functions, etc.) turn out to be extensionally equivalent. This confluence is a major strength of Mathematical CT.

By contrast, different formulations of Bold Physical CT appear to be logically independent of one another. They appeal to disparate notions whose mutual connections are less than transparent. This state of affairs demonstrates, at the very least, that clarity and consensus on Physical CT are lacking. It also raises doubts on the widespread practice of formulating putative physical formulations of CT without introducing constraints on what counts as a relevant physical process. Should we accept any of these formulations as our physical formulation of CT? Which one? On what basis should we choose one over the others? If we are to choose one, presumably we need arguments to rule out the others as irrelevant (or less likely, to show that the others reduce to our preferred one). This already shows that before we settle on a formulation of Physical CT, some philosophical work is in order.

2.2 Falsification by Cheap Counterexamples

Bold Physical CT encompasses all physical processes regardless of whether they can be used by a finite observer to generate the values of a function. As a consequence, Bold Physical CT is liable to be refuted by cheap counterexamples. By a "cheap counterexample," I mean a process that is either obviously not physically implementable or useless for generating the values of a desired function. I will now argue that by encompassing all physical processes and thereby exposing itself to cheap counterexamples, Bold Physical CT abandons the epistemological notion of computation that motivated CT in the first place. Thus, it is not a good physical analogue of Mathematical CT.

The most obvious case is formulation (B). Blum et al. (1989) set up a mathematical theory of "computation" over real-valued quantities, which they see as a fruitful extension of ordinary computability theory. They define idealized "computing machines" that perform addition, subtraction, multiplication, division, and equality testing as primitive operations on arbitrary real-valued quantities. They easily prove that such "machines" can "compute" *all* sets defined over denumerable domains by encoding their characteristic function as a real-valued constant (ibid., 405). Hence, (B) is far from true. Blum et al. do not discuss this result as a refutation of Physical CT, and yet their work is often cited by critics of Physical CT.

But it is hard to see Blum et al.'s result as relevant to what can be physically computed in any interesting sense. Their "computing machines" perform operations on arbitrary real-valued quantities: this is hardly a kind of operation that a finite observer can exploit to discover the values of a desired function (more on this below). And Blum et al.'s way of "computing" Turing-uncomputable functions relies on encoding the values of such functions as real-valued constants. While this is mathematically trivial, doing it in the physical world would require already possessing super-Turing computational powers.

More generally, formulations (A)-(D) would be falsified by a sequence generated by a random (i.e., nondeterministic) physical process. Consider a discrete random process, such as the decay of atoms from a radioactive sample, over an infinite period of time. Its output at regular time intervals is a string of digits—'0' if no atoms decay during a time interval, '1' if one or more atoms decay during a time interval.[11,12] A simple cardinality consideration shows that, with probability one, the sequence produced by our random process is not Turing-computable.

There are uncountably many infinite strings of digits. (Even more strongly, there are uncountably many infinite strings of digits with any given limiting frequency of '0's and '1's.) But there are only countably many Turing-computable infinite strings. Therefore, assuming that each infinite string (or each infinite string with a certain limiting frequency) has the same probability of occurring as a result of a random process, the probability that a random process would generate a Turing-computable string of digits is zero, whereas the probability that the string of digits is not Turing-computable is one. Thus, simply by using a random process to generate a string, there is a sense in which we would have physical means that go beyond what is Turing-computable. As Alan Turing

[11] Terminological note: I use 'digit' to denote a concrete instantiation of a letter from a finite alphabet.
[12] Later, I will discuss a difficulty with counting the output of a random process as a string of digits. Since promoters of Bold Physical CT are typically not concerned with that type of difficulty, for now I will ignore it for the sake of the argument.

(1950, 438-439) pointed out, a machine with a "random element" can do "more" than a Turing machine.

Someone might be tempted to bite the bullet and conclude that if there are genuinely random processes, then Physical CT is false (e.g., Bowie 1973, 74; Calude 2005, 10). But there are several reasons to deny that a genuine random process *P* should count as a computation.

First, there is no way to define the function f: $\mathbf{N} \rightarrow \{0,1\}$ whose values are being generated by *P* without reference to *P* itself. Of course, *f* exists in the set-theoretic sense of a set of pairs whose output values *P* happens to emit. But in this context, defining *f* means actually specifying the relationship that obtains between the arguments and values of *f*, as we do when we define, for instance, the halting function for Turing machines.[13] If *P* is genuinely random, there is no way to specify *f* without generating the values of *f* by running *P*.

Computation is interesting precisely because it finds the values of functions defined independently of the processes that compute them. If we can't find out what *f* is before we run *P*, we are not going to learn anything interesting about *f*, such as the values we desire to know for selected arguments, by running *P*. Hence, we should not count *P* as a physical computation, and we should not count Bold Physical CT, which is falsified by *P*'s existence, as a genuine version of CT.[14]

[13] The halting function is the function that returns '1' if Turing machine *t* halts on input *x*, '0' otherwise.
[14] Determining which *f* is such that its values are generated by a random process *P* is not only impossible to do ahead of running *P*. It also contains two other elements of arbitrariness that constitute disanalogies with genuine computation. First, in the case of *P*, we can define indefinitely many *f*'s depending merely on which time intervals we consider when observing *P*'s outputs. We could count *P*'s outputs per second, or millisecond, or year. Each temporal criterion will give rise to a different *f*. Second, in the case of *P*, we can define indefinitely many *f*'s depending merely on which time interval we decide to count as the first one. By contrast, genuine computing systems come with a univocal criterion for what counts as the function being computed and the beginning of the computation. For instance, in the case of Turing

A second reason against counting random processes as computations is that they are not repeatable, whereas computations must be repeatable.[15] Of course, the very same sequence of events cannot occur twice. But physical processes may be repeatable in the sense that the same sequence of states (as defined by, say, a set of dynamical equations) can occur either within the same physical system at different times or within two similar physical systems (e.g., two systems satisfying the same equations). Genuine random processes are not repeatable in this sense. Unrepeatable processes can only be observed by the lucky few who happen to be in their presence. They cannot be used by others who might be interested in learning from them. If no one is present, no one can learn from an unrepeatable process. If someone is present but makes a mistake in recording a result, that mistake can never be corrected. But again, computation matters to us (and to the logicians who created computability theory) because we can learn from it. If we can't repeat a process when we need it, it is not computation, and hence it is irrelevant to an adequate formulation of Physical CT.[16]

A third reason is that a "user" of P cannot select the value of f she wishes P to generate. Ordinary computing systems can be set so as to compute the value of a function for a desired argument. In the case of random processes, however, a "user" has no way to do this. If she wishes to obtain the nth value of f, all she can do is let P take its

---

machines, the process begins with the machine acting on a designated first square on the tape while in a designated initial state; the function being computed is the one whose arguments and values are written on the tape, respectively, at the beginning and at the end of the process.

[15] I owe this observation to Michael Rabin. I don't know whether he agrees with the rationale I offer for it.

[16] Repeatability of a process is also helpful to check that a process is performed correctly and thus that a computation is reliable (see below). This may not matter in the case of random processes, for there doesn't seem to be any useful sense in which a random process can perform incorrectly. If it can't perform incorrectly, there is no need to check for its correctness. But this observation only reinforces the conclusion that random processes are not computations, for random processes, unlike computations, cannot go wrong. For more on the possibility of miscomputation as a feature of genuine computation, see Piccinini 2007b.

course and wait until the n^{th} value is generated. If this won't happen until a million years from now, that's too bad.

Someone might object that the case of a random process that takes too long to be useful is analogous to ordinary unfeasible computation. But there is an important disanalogy. The reason why ordinary computations are unfeasible is that they take too many steps for our current technology. Although some computations require so many steps that they will always be unfeasible, that is beside the point. For in general, as long as we shorten each computational step, more and more (ordinary) computations become feasible. Not so in the case of random processes. Since there is no way to repeat them, a fortiori there is no way to repeat them using faster technology. If the function *f* whose values are produced by our random process is defined to take one year per output digit, it will take one thousand years to obtain the thousandth value of *f*. We can't do anything to obtain faster results.

The above remarks converge on the conclusion that genuine random processes are *not* computations. Unlike computations properly so called, random processes cannot be used to generate the desired values of a function or solve the desired instances of a general problem. Random processes can be *exploited* by a computation, of course—there are important computational techniques that rely on random or pseudo-random choices at some stages of a computation. If some quantum random sequences were random in the sense of algorithmic information theory, they may even raise the probability of obtaining correct solutions from computational techniques that rely on random choices (Calude 2005, 10). But no computational technique can amount to a mere sequence of random choices. So any thesis, such as Bold Physical CT, that would be falsified by a sequence generated by a random process is too strong to capture the notion of *physical* computability—the physical analogue of computability by effective procedure. Contrary to what some authors seem to assume, Bold Physical CT is too strong to be a physical analogue of Mathematical CT.

2.3 Unconstrained Appeals to Real-valued Quantities

Many current physical theories assume that nature contains real-valued quantities that vary along a continuum. These may include the velocity of objects, the coordinates that define the position of their center of gravity in the spacetime manifold, and more. If these physical theories are correct, then many properties of many entities take arbitrary real numbers as their values. Hence, systems of physical equations, whose simulations, solutions, or observables are appealed to, respectively, by (A), (C), and (D), involve arbitrary real numbers. Since (B) explicitly involves arbitrary real-valued quantities, all versions of Bold Physical CT involve, explicitly or implicitly, arbitrary real numbers.

But most real numbers in any continuous interval are Turing-uncomputable. In fact, there are only countably many Turing-computable numbers, while any continuous interval contains uncountably many real numbers. Thus, the probability that a randomly selected real-valued quantity is Turing-computable is zero. Hence, if our physical theories are correct, most transformations of the relevant physical properties are transformations of Turing-uncomputable quantities into one another.

For instance, an object's change of speed, or even its simple change of spatial location may be transformations of one Turing-uncomputable real-valued quantity into another. A transformation of one Turing-uncomputable value into another Turing-uncomputable value is certainly a Turing-uncomputable operation. Hence, it would seem that given many of our physical theories, the physical world is chock-full of operations that outstrip the power of Turing machines. If this is correct, it falsifies Bold Physical CT.

But there is no reason to believe that a finite observer can use the Turing-uncomputable operations just mentioned to compute in the epistemological sense that motivates CT in the first place—to solve problems, to generate the values of desired functions for desired arguments. While Blum et al. (1989) may find it useful to define notional "machines" that manipulate arbitrary real-valued quantities, this kind of manipulation cannot be exploited by a finite observer.

In order to measure a real-valued quantity exactly at an arbitrary time, or even in order to measure the exact value of an arbitrary digit in the expansion of a real-valued quantity, an observer needs unbounded precision in measurement. Since the value of the variable may change over time, the observer also needs unbounded precision in the timing of the measurement.[17] There is no reason to believe that such unbounded precision is available to a finite observer. Furthermore, there is no reason to believe that a finite observer can discover the exact value of a real-valued quantity, either computable or not, by means other than measurement. Finally, preparing a real-valued quantity with an arbitrary exact value also requires unbounded precision. Again, there is no reason to believe that unbounded precision in preparation is available to a finite observer.

As far as we know, finite observers are limited to preparing and measuring physical quantities to a finite degree of approximation. Thus, manipulations of arbitrary

[17] This creates a further difficulty in counting the output of a genuine random process as a string of digits. For if an output occurs too close to the transition point between two digits, it may be practically impossible to establish whether it falls within a digit or its successor.

real-valued quantities should not count as computation—at least until someone shows how a finite observer can exploit them in practice. Bold Physical CT, which is falsified by such manipulations, is not interestingly analogous to Mathematical CT.[18]

In order to put the debate over Physical CT on more fertile ground, we need to distinguish the issue of physical computability proper—the issue that pertains to the physical analogue of Mathematical CT—from other issues that connect computability and physics. Many questions about the relationship between physical processes and computability deserve to be asked. What can be computationally approximated to what degree under what circumstances?[19] What can be accomplished by performing certain operations over arbitrary real-valued quantities? Which systems of equations describing a physical system give rise to computable solutions? This is presumably what (A), (B), and (C), respectively, are after. These questions are interesting and deserve to be investigated. (I don't see that (D) is formulated well enough to address any interesting question.) Nevertheless, these questions do not properly belong in discussions of CT, because they are different from the question of what can be physically computed, which is the question that motivates CT in the first place. In the end, so-called Bold Physical CT is a cluster of more or less interesting theses relating computation and physics, none of which is a version of CT properly so called.

3. A Usability Constraint on Physical Computation

[18] The point just made does not impugn analog computation in the standard sense of Pour-El (1974). Analog computation does not manipulate exact values of arbitrary real-valued quantities but rather continuous variables. Although a continuous variable may be assumed to take any real value within a relevant interval, a successful concrete analog computation requires only the measurement of real variables with some degree of approximation. No exact values of arbitrary real-valued quantities are exploited by an analog computation, so analog computation does not falsify Bold Physical CT (cf. Rubel 1989).
[19] For a more detailed discussion of this question, see Piccinini 2007a.

The above considerations suggest that Bold Physical CT, which appeals to what can be done by physical systems, is not an adequate physical analogue of Mathematical CT. The shortcomings of Bold Physical CT are all motivated by the same underlying principle: a physical process should not count as a computation unless a finite observer can use it to generate the desired values of a given function. This principle can be formulated as an explicit constraint on physical computation:

> **Usability Constraint**: If a physical process is a computation, it can be used by a finite observer to obtain the desired values of a function.

The usability constraint is an epistemic constraint.[20] It can be made more precise by specifying what it means to be usable by a finite observer and who counts as a finite observer.

As to the latter question, one possibility is to include among finite observers human beings and any other intelligent beings of similar finitude. Under this notion of finite observer, the usability constraint is relevant to computer scientists, engineers, entrepreneurs, and consumers, whose aims are designing, building, selling, and using computers now or in the near future. If we include among observers real or hypothetical creatures at regions of spacetime that are physically inaccessible to us, the usability constraint is also relevant to physicists whose aim is investigating the computational power of physical systems anywhere in spacetime.

Another possibility is to construe "finite observer" more broadly, to include any functionally organized system whose behavior is influenced by a computation in an appropriate way. If nervous systems are computing systems, then observers in this

[20] The usability constraint is weaker than the verifiability constraint discussed, and rightly rejected, by Shagrir and Pitowsky 2003, 90-1.

extended sense are the bodies of organisms whose behavior is influenced by their own neural computations. Under this broader notion of finite observer, the usability constraint is relevant to scientists interested in the computational explanation of cognition.

The constraints I will propose may be applied to observers in either sense. Given that the special relationship between organisms and their nervous systems may require additional caveats, however, in the interest of brevity I will focus the discussion on finite observers understood under the former, more restricted construal.

As to what it means for a physical process to be usable by a finite observer, we need a physical counterpart to the notion of effective procedure that grounds Mathematical CT. As per Section 1, a finite observer can follow an effective procedure because an effective procedure is executable, automatic, uniform, and reliable. By analogy, in order to be usable by a finite observer, a physical process must be executable, automatic, uniform and reliable. These requirements ought to be qualified to take into account the differences between effective procedures and physical processes.

An *executable* physical process is a physical process that a finite observer can set in motion to generate the values of a desired function until it generates a readable result. This requires that the observer can discover which function is being computed, that the process's inputs and outputs be readable by the observer, and that the finite observer be able to construct the system that exhibits the process (more on this below). An executable physical process is also a process that, like an effective procedure, can be repeated any time a finite observer wishes to run it.

An *automatic* physical process is a physical process that runs until it produces its results if it halts, or else it continues indefinitely as long as it doesn't break down.

A *uniform* physical process is a physical process generated by a system that, given different inputs corresponding to the arguments of a function, generates outputs corresponding to the different values of the function without needing to be redesigned or modified for different inputs.

Finally, a *reliable* physical process is a physical process that generates correct results at least some of the time.

These principles follow from the usability constraint by analogy with effective procedures. They can be spelled out further in terms of the following sub-constraints, many of which address observations we made in discussing the pitfalls of Bold Physical CT:

1. *Readable Inputs and Outputs*.[21] The inputs and outputs of a computation must be readable. A quantity readable in this sense is one that can be measured to the desired degree of approximation, so as to be used as output, and either prepared or discovered by a finite observer to the desired degree of approximation, so as to be used as input. This first sub-constraint rules out exact values of arbitrary real-valued quantities as inputs or outputs of computations, because they cannot be prepared or discovered by a finite observer (as per Section 2.3). One important reason why computing technology is paradigmatically digital is that strings of digits are the only known inputs and outputs to be reliably readable without error.[22,23]

[21] Do all computations have to have inputs and outputs? The mathematical resources of computability theory can be used to define "computations" that lack inputs, outputs, or both. But the computations that are generally relevant for applications are computations with both inputs and outputs. Here I focus on the ordinary case.
[22] This point is discussed in more detail in Piccinini 2007b, especially Section 2.
[23] Someone might ask, what about analog computers? Don't they have real-valued quantities as inputs and outputs? Actually, analog computers (in the sense of Pour-el 1974) are designed to yield the values of functions of real variables, which is not quite the same as functions of real-valued quantities. What analog computers do is useful for solving certain classes of equations. Their output is still a string of digits

For an output to be readable in the intended sense, the computing system must have a recognizable halting state. As a counterexample, consider a machine that purportedly computes characteristic function *f*. Suppose that for any argument of *f*, the machine is guaranteed to output its correct value at some future time. The machine gives its output as a one or a zero on a display. Suppose further that at any time, the value on the display may change from a one to a zero or vice versa and there is no known time after which the result of the computation is definitive. It may seem that the output is readable, because one can see whether it's a zero or a one. But it is not readable in the relevant sense, because the user never knows whether what she is reading is the desired value of *f*.

2. Process-Independent Rule. In a genuine computation, the problem being solved (or equivalently, the function being computed) must be definable independently of the process of solving it (computing it). If this is not the case, a finite observer is in no position to select the values of a function that she wishes to obtain—as the usability constraint requires—because she doesn't even know what the function is.[24] A classic example of Turing-uncomputable function is the halting function for Turing machines. The problem can be stated as that of finding the values of an explicitly defined function from strings of digits to strings of digits. Any putative machine that solves the halting problem by reliably generating (measurable) values of the function for any argument (of reasonable size) satisfies this second sub-constraint (as well as the first sub-constraint, readable inputs and outputs). By contrast, as we saw above, a genuine random process

representing a real number to some degree of approximation. For more details on analog computers and their relation to digital computers, see Piccinini 2008a, Section 3.5.

[24] The "process-independent rule" sub-constraints entails that modeling one physical system using another is not enough for computing. For example, testing a model of an airplane in a wind tunnel does not constitute a computation because there is no way to define the function being putatively computed apart from performing the modeling experiments.

cannot be characterized as generating the values of an independently specified function—it fails to satisfy the second sub-constraint. That is enough to rule it out of the class of genuine computations.

3. Repeatability. For something to count as a genuine computation, it must be repeatable by any observer whenever they wish to obtain its results. In other words, any competent observer must be able to set up a physical system that undergoes the relevant sequence of state transitions. Like the other sub-constraints, repeatability must be taken with a grain of salt. A computing system may be able to perform only one computation (or part thereof) during its lifetime, but it should be possible to repeat the same computation on another system. Repeatability applies during ordinary conditions of use, whatever they may be. It does not entail that a computing system must work under all physical conditions, or that computation must be instantaneous to allow for cases in which users want a computation repeated immediately after it starts. Within ordinary conditions of use, it must be possible to repeat a process, or else the process does not count as a computation.[25]

4. Settability. An ordinary computing system is something that, within its limits, can compute any value of one or more functions. Universal computers can compute any value of any Turing-computable function until they run out of memory or time. For this to be the case, normally a user sets the system to its initial state and feeds it different arguments of the function being computed. When a new computation is needed, the system is *re*set to its initial state, where it may be given either the same or a different

[25] Repeatability may also be needed to establish reliability (see below). One standard way to check that a computer is functioning properly is to run the same computation more than once, making sure that each time the results are the same. (Of course, repeatability does not rule out all malfunctions, since the results of repeated computations could be the same but still wrong.)

input. If the system is given the same input, it should yield—barring malfunctions—the same output. Thus, resettability plus repeatability of the input entails repeatability of the output. But resettability is not required for computation; settability is enough for present purposes. If a system is not even settable, so that a user cannot choose which value of the function is going to be generated in a given case, then that system does not count as computing.

5. Physical Constructibility. If a system cannot be physically constructed, it may count as performing notional computations, but it is irrelevant to Physical CT. A system is physically constructible in the present sense just in case the relevant physical materials can be arranged to exhibit the relevant properties. The materials may well include entities that are very large, small, or distant, so long as they can be made to exhibit the relevant properties.

Suppose that firing *n* pulses of energy into a sunspot reliably and repeatedly causes *f*(*n*) pulses of radiation to be emitted, for some nontrivial *f*. The system that includes the sunspots as well as the apparatus for emitting and receiving energy pulses counts as physically constructible in the present sense. By the same token, every computing technology exploits natural properties of existing materials.

Developing computing technology involves many engineering challenges. That is why we don't yet have (computationally nontrivial) DNA or quantum computers, in spite of their apparent possibility in principle. The practical obstacles facing these technologies may be surmountable. But suppose that a putative computing technology involved obstacles that are practically insurmountable, for principled physical reasons. For instance, a putative computer might require physically unattainable speeds, more matter-energy than the universe contains, or parts smaller than the smallest physical particles. Such a technology would not be relevant to our practical interests. What can't be physically constructed can't refute Physical CT.

Someone may object as follows. Consider Turing machines, with their tape of unbounded length. If the universe contains only a finite amount of matter-energy, then tapes of unbounded length might not be physically constructible. Furthermore, it seems unlikely that a finite user would ever be able to harness an infinite amount of matter-energy. All that we can construct is, most likely, finite approximations of Turing machines, such as our ordinary digital computers. But Turing machines are the very machines in terms of which CT is formulated. If Turing's idealization is legitimate, someone might conclude, other idealizations are legitimate too. Since we accept Turing machines as relevant to CT, we should also consider other machines, regardless of whether they are physically constructible.

Arguments along these lines may be found in the literature on physical computability.[26] They miss the way in which Turing machines are relevant to CT. All that CT says is that any function that is intuitively computable is computable by some Turing machine. Computability by Turing machines is the *upper bound* on what CT deems intuitively computable. And acting as an upper bound does not require being physically constructible.

[26] See Button (2009, 775-8) for a recent discussion. According to Button, the argument just rehearsed may be resisted if there is an appropriate notion of physical possibility according to which, roughly, for any computation by a Turing machine, there is a possible world containing enough physical resources to complete the computation. Although I side with Button against the argument under discussion, Button's request for a specialized notion of physical possibility, postulating enough physical resources for any Turing machine computation, is both onerous and ad hoc. The considerations to follow in the main text provide a simpler and more principled reply.

Perhaps the objection against physical constructibility derives some allure from the frequent practice of lumping CT and its converse together and calling their conjunction ‘the Church-Turing thesis’. Such a lumping may be innocuous in other contexts, but it is misleading here. If CT and its converse are lumped together, Physical CT is taken to be the thesis that the physically computable functions are the same as the Turing-computable ones. If we take this to be Physical CT and notice that Turing machines may not be physically constructible, we might find nothing wrong in considering whether other physically non-constructible machines extend the range of the physically computable, thereby refuting Physical CT. This would be a mistake. To see that it is, it helps to keep CT and its converse separate.

When we focus on what can be physically computed, we know that any machine with a finite memory, such as our digital computers, computes less than Turing machines (for lack of a literally unbounded tape). We may then ask whether there are physical means to compute as much or more than what Turing machines can compute. But to surpass the power of Turing machines and falsify Physical CT, it is not enough to show that some *hypothetical* machine is more powerful than Turing machines. That’s easy and yet says nothing about what can be *physically computed*. What we need to show is that some machine is both more powerful than Turing machines and physically *constructible*. Given that Turing machines themselves are unlikely to be physically constructible, this is quite a tall order.

In conclusion, neither CT nor its converse entail that Turing machines are physically constructible, and this is how it should be. But anyone who maintains that some physically computable functions are not Turing computable ought to show that the systems that compute such functions are physically constructible.

6. Reliability. Demanding machines that never break down would be unrealistic, but machines that never complete their computation successfully are not worth building. To be useful, a computing system must operate correctly long enough to yield correct results at least some of the time. For this to happen, the system's components must not break too often. In addition, the system's design must be such that noise and other external disturbances are generally insufficient to interfere with the results. The history of electronic computation illustrates this point nicely.

When large electronic computers were initially proposed, some skeptics argued that their vacuum tubes would break too often for the computers to be useful. The skeptics were proven wrong, but this was no small feat. Even so, early electronic computers broke often enough that they required full-time technicians devoted to fixing them. Since then, computers have become remarkably reliable; this contributes to making them useful and accessible to a large public. The case of early computers shows that yielding a correct result even a low percentage of the time is compatible with usefulness. But we should be wary of computer designs that are guaranteed to break, self-destroy, or destroy the user (see below) before the results of the computation can be known. Those early skeptics had a point: if a machine is unreliable as a matter of principle, it is not worth building.

This list of constraints should be enough to illustrate the kind of property that is required for something to be useful for physical computing, and hence to be relevant to Physical CT. The constraints are not meant to be exhaustive; the list is open ended. For what makes a system useful for computing depends in part on the way the physical world is, and that is not a matter for philosophers of computation to settle. The important point is that we are interested in computation because of what we (finite observers) can learn from it. If we can't learn anything from a putative computation, then that process is not relevant to Physical CT.

4. The Modest Physical Church-Turing Thesis

If Physical CT is to be grounded on the epistemological notion of computation, thereby avoiding the pitfalls of Bold Physical CT, it needs to cover less than what can be *physically done*. What Physical CT needs to cover is any process of genuine *physical computation*, where physical computation is explicated in terms of our usability constraint and its sub-constraints.

This proposal turns Physical CT into the following:

> **Modest Physical CT**: Any function that is physically computable is Turing-computable.

Modest Physical CT asserts that every function defined over a denumerable domain that can be physically computed—every usable transformation of input strings into output strings in accordance with a process-independent rule defined over the strings—is Turing-computable.

Modest Physical CT has two desirable features. First, it is relevantly analogous to Mathematical CT, because it is faithful to the epistemological concerns that motivated CT in the first place. In other words, Modest Physical CT has to do with which physical processes can be used by a finite observer to generate the values of a desired function. The second desirable feature is that Modest Physical CT is open to empirical refutation.[27]

Prototypical examples of physical computation are the processes of ordinary digital computers and their components, including digital circuits. Such processes can be exhaustively described by effective procedures, which are already covered by Mathematical CT. Mathematical CT says that any function computable by an effective procedure is Turing-computable. As Turing machines can be physically approximated (or replaced by a computing human), any process that follows an effective procedure is physically computable (up to the memory and time limitations of concrete digital computers).

But physical computation in the present sense is a broader notion than computation by effective procedure. A process may count as a physical computation even if there is no effective procedure for describing the process, perhaps because there are no finite instantaneous descriptions of the internal states that constitute the process or no way to finitely and exactly specify the transition from one instantaneous description to the next. Thus, Modest Physical CT, which is formulated in terms of physical computability, is stronger than Mathematical CT, which is formulated in terms of

[27] By contrast, consider the alternative approach to Physical CT that originates with Robin Gandy (1980) and has been developed primarily by Wilfried Sieg (2002, 2006b, 2008). According to Gandy and Sieg, Physical CT pertains to what can be "computed" by certain "discrete dynamical systems". Gandy and Sieg define their discrete dynamical systems in terms of a set of assumptions. Specifically, they postulate finiteness and locality conditions, such as discrete states, discrete dynamics, a lower bound on the size of atomic components, and an upper bound on signal propagation. Gandy and Sieg prove that anything "computable" by any of their discrete dynamical systems is computable by Turing machines. Some of their assumptions—such as a lower bound on the size of atomic components and an upper bound on signal propagation—are empirically well-motivated. But other assumptions—such as discreteness of states and discreteness of dynamics—are not empirically justified. Thus, there remains the empirical question of whether there are physical systems that violate Gandy and Sieg's assumptions and are computationally more powerful than Turing machines. This is the real question of interest here, and it is not settled by Gandy and Sieg's work. This is probably why Gandy and Sieg's work has had relatively little impact on discussions of physical computability. For related discussions of Gandy and Sieg's approach see Shagrir and Pitowsky 2003 and Copeland and Shagrir 2007.

computability by effective procedure. In addition to physical processes that follow effective procedures, Modest Physical CT may cover processes that exploit properties of spacetime (as in relativistic computing), continuous dynamical processes (as in certain kinds of connectionist computing), and quantum processes (as in quantum computing).

Since Modest Physical CT is restricted by epistemologically relevant criteria, it doesn't raise the worries associated with Bold Physical CT—namely, that it's too easy to falsify and irrelevant to the epistemological notion of computability that motivates CT. It also allows us to shed light on why most computability theorists and computer scientists believe in Physical CT. To illustrate how to assess Modest Physical CT and why it is plausible, I will briefly discuss some purported counterexamples.

4.1 Hypercomputation: Genuine and Spurious

The term *hypercomputer* is often used for any system that yields the values of a Turing-uncomputable function. If what counts as yielding the values of a function is left unspecified, any of the systems discussed in Section 2, such as systems with genuinely random outputs and systems that manipulate arbitrary real-valued quantities, would count as hypercomputers. But in discussing Bold Physical CT, we saw that yielding the values of a function that is Turing-uncomputable, without further constraints, is not enough for genuine physical computation.

By analogy with the distinction between Bold Physical CT and Modest Physical CT, let us distinguish between a weak and a strong notion of hypercomputation by distinguishing genuine from spurious hypercomputers.

A *spurious* hypercomputer is a physical system that fails to satisfy at least one of the first four constraints on physical computation. Examples include processes whose inputs or outputs are arbitrary real-valued quantities (which are not readable without error) and genuine random processes (which have no rule characterizing the inputs and outputs independently of the process, and are neither repeatable nor settable). These putative hypercomputers are spurious because they cannot be used by an observer to compute arbitrary values of an independently defined function on an input chosen by the user, as ordinary computing systems can (given enough time and space). Since spurious hypercomputers are not computing systems, they are irrelevant to Modest Physical CT.

A *genuine* hypercomputer is a physical system that satisfies at least the first four constraints on physical computation. It has readable inputs and outputs, there is a rule characterizing its input-output properties that may be defined independently of the process itself, and its processes are repeatable and settable. There remains the question of whether any genuine hypercomputers are physically constructible and reliable. If they are, they refute Modest Physical CT.

Many schemes for putative hypercomputers have been proposed. In some cases, it is obvious that they are not physically constructible. For instance, infinitely accelerating Turing machines (Copeland 2002) are Turing machines that perform each computational operation in half the time as their previous operation. As a consequence, infinitely accelerating Turing machines complete an infinite number of operations (a *supertask*) within twice the time it takes them to perform their first operation. This allows infinitely accelerating Turing machines to compute functions, such as the halting function, that are Turing-uncomputable. But infinitely accelerating Turing machines are usually discussed as notional entities, without suggesting that they can be constructed. Purely notional systems, of course, do not falsify Modest Physical CT. To do that, a system must satisfy at least the fifth and sixth constraints on physical computation: it must be physically constructible and it must operate reliably.

One way to construct something like an infinitely accelerating Turing machine would be to make a computing machine that, after performing some computational operations, builds a smaller and faster copy of itself (Davies 2001). The smaller and faster copy will also perform some operations and then build a faster and smaller copy, and so on. Given appropriate assumptions, the resulting series of infinitely shrinking machines will be able to complete an infinite number of computational operations within a finite time, thereby surpassing the power of Turing machines. While infinitely shrinking machines appear to be consistent with Newtonian mechanics, Davies (2001, 672) points out that the atomic and quantum mechanical nature of matter in our universe makes infinitely shrinking machines physically impossible.

In recent years, several designs for hypercomputation have been proposed. If genuine hypercomputation turns out to be physically constructible and reliable, it may refute Modest Physical CT.

4.2 Relativistic Hypercomputers

One of the best-known proposals for a hypercomputer is due to Mark Hogarth (1994, 2004), who developed an idea of Itamar Pitowsky (1990). I will now briefly discuss Hogarth's proposal to illustrate what is required for a genuine hypercomputer to falsify Modest Physical CT.

Hogarth's relativistic hypercomputers exploit the properties of a special kind of spacetime, called *Malament-Hogarth spacetime*, which is physically possible in the sense of constituting a solution to Einstein's field equations for General Relativity. Malament-Hogarth spacetimes contain what may be called spacetime *edges*—regions containing an *infinite* time-like trajectory λ that can be circumvented by a *finite* time-like trajectory g. In other words, λ and g have a common origin—here called a *bifurcation*—and there is a spacetime point p on g such that λ, even though it is infinite, lies entirely in p's chronological past.

In addition to the operations performed by Turing machines (TMs), relativistic hypercomputers exploit five further primitive operations, described by the following instructions: (1) Position yourself at a bifurcation (where both λ and g start); (2) Launch a TM along λ while you remain on g; (3) Pass the edge (i.e., go to point p, by which time g has circumvented λ); (4) Send a signal (from λ to g); and (5) Receive a signal (coming from λ).

With the resources offered by relativistic hypercomputers, we can define a procedure for solving the halting problem for Turing machines. The halting problem asks, given a TM t and an input x, does t halt on x? Exploiting the power of relativistic hypercomputers, a procedure for solving the halting problem can be defined as follows:

1. Prepare t with input x and add to t's instructions the instruction to send a signal (from λ to g) upon halting.
2. Position yourself at a bifurcation.
3. Launch t along λ while you remain on g.
4. Pass the edge while receiving a signal coming from λ, if there is one.

In the finite time it takes an observer to travel through *g*, this procedure determines whether *t* halts on *x*. This is because by the time *g* circumvents λ, which it will do by the definition of Malament-Hogarth spacetime, the observer traveling through *g* will receive a signal if and only if *t* halts on *x*. Hence, the above procedure solves the halting problem for TMs.[28]

Are relativistic hypercomputers genuine hypercomputers? They are if they satisfy Constraints 1-4. Since they operate on strings of digits, they appear to satisfy Constraint 1 (readable inputs and outputs). This appearance hides the difficulty in transmitting *t*'s output from λ to *g* in such a way that the receiver can read it. This is a serious challenge, which I will discuss below under the rubric of reliability. Since the functions relativistic hypercomputers allegedly compute (e.g., the halting function) are defined independently of their activity, they satisfy Constraint 2 (process-independent rule). As Hogarth initially defines them, however, relativistic hypercomputers fails to satisfy Constraints 3 (repeatability). This is because relativistic hypercomputers, as they are usually defined, have access to at most one spacetime edge. If there is only one accessible edge, a relativistic hypercomputer may be run only once, on one input, in the history of the universe. This violates repeatability. This point is generally ignored in the literature on relativistic hypercomputers.

Hogarth also defines Malament-Hogarth spacetimes that contain an infinite number of edges. For our purposes, we don't need an actual infinity of edges; we only need a large number. If they are accessible one after the other, each of them may be exploited to run a distinct computation. That would give us enough resources to repeat

[28] For a more detailed treatment of what relativistic hypercomputers can compute under various conditions, see Welch 2008.

computations or compute different values of the same function, thereby satisfying Constraint 3.[29] As we shall see, however, the hypothesis that one spacetime edge can be exploited successfully by a relativistic hypercomputer is fantastic enough.

As to settability, some components of a relativistic hypercomputer get lost in the course of the computation—they are never recovered after they are launched along λ. So relativistic hypercomputers are not resettable. But at least they are settable, because each TM can be set to compute the value of a desired function for a desired input. Thus, relativistic hypercomputers satisfy Constraint 4 (Settability).

Given the above discussion, we may tentatively conclude that relativistic hypercomputers satisfy Constraints 1-4, which makes them genuine hypercomputers. It remains to be seen whether they falsify Modest Physical CT. For that, they must be physically constructible and reliable.

Constructing a relativistic hypercomputer is a highly nontrivial affair. The first question is whether our spacetime is Malament-Hogarth; the answer is currently unknown. Even if our spacetime is not Malament-Hogarth globally, it might still contain regions that have the Malament-Hogarth property locally. An example is the region surrounding a huge, slowly rotating black hole; there is evidence that our universe contains such regions (Etesi and Németi 2002). If the universe contains no Malament-Hogarth regions, Hogarth's relativistic computers are not physically constructible. But even if there are Malament-Hogarth regions in our universe, there remain considerable obstacles.

[29] Repeatability is not the purpose for which Hogarth introduces his spacetimes with infinitely many edges; I'm putting them to a different use. Cf. Shagrir and Pitowsky's discussion of their Objection #3, which they eventually handle in the same way (2003, 91-3).

In order to exploit the special properties of Malament-Hogarth spacetimes, relativistic hypercomputers need to be started at a bifurcation. So, for a user to do anything with a relativistic hypercomputer, there must be bifurcations within regions of spacetime that she can access. Otherwise, if all bifurcations are out of reach, she cannot exploit them to run relativistic hypercomputers. But notice that the huge, rotating black hole that is closest to us, which is the basis for Etesi and Németi's proposed implementation of a relativistic hypercomputer, is presumably out of our reach as well as the reach of our descendants.[30]

Additionally, in order to work with a relativistic hypercomputer, a user needs to *know* that she is at a bifurcation. She must also know how to launch a TM along the infinite trajectory λ that starts at the bifurcation while proceeding along the finite trajectory g. Finally, she must know when she has circumvented λ. Hence, for relativistic hypercomputers to be constructible, it must be possible to discover when one is in the presence of a bifurcation, how to launch machines along λ's, how to proceed along g's, and when one has circumvented λ's. I've never seen any mention of these issues in the current literature on relativistic hypercomputation. Yet an observer who happen to travel on a g path while a Turing machine happens to travel on the corresponding λ path performs a computation only if the observer can deliberately use this set up to learn the desired values of a function.

Another difficulty is that a relativistic hypercomputer requires an unbounded memory store. The reason is that in some cases, the hypercomputer gives a correct

[30] An anonymous referee objected that the interest in Modest Physical CT is simply the question of what computations the physical laws allow; whether the needed resources are too far away is irrelevant. But this is precisely what I've been arguing against. If needed resources are too far away, a finite observer (of finiteness comparable to ours) cannot use them. Any scheme requiring resources that are too far away violates the usability constraint and thus fails to refute Modest Physical CT.

output just in case its TM does not halt; to perform its infinitely many steps correctly, the TM needs an unbounded memory in which to store an unbounded number of bits of information (cf. Shagrir and Pitowsky 2003, p. 88-90). Because of this, a relativistic hypercomputer demands more than a Turing machine, which requires only finitely much memory storage to produce any given correct output. (Plus, recall that according to Modest Physical CT, Turing machines act as the upper bound to what is physically computable, which does not require that they be physically constructible.) An unbounded memory may require an unbounded amount of matter-energy. Recent evidence suggests that the universe is infinite not only in space and time, but also in matter-energy. Thus, letting our relativistic hypercomputer spread over the universe, and assuming that the universe's expansion does not keep accelerating forever, there may be enough matter-energy for a relativistic hypercomputer (Németi and Dávid 2006). It's not clear, however, how spreading a relativistic hypercomputer over the entire universe can be reconciled with its need to travel through a specific time-like trajectory λ. If we are unlucky and there is no way to harness an unbounded amount of matter-energy, there might be other ways to keep a relativistic hypercomputer running; for one thing, recent developments in quantum computing suggest that storing one bit of information does not require a fixed amount of matter-energy (Németi and Dávid 2006). In any case, building an unbounded memory is quite a challenge.

Yet more exotic obstacles to the physical constructibility of relativistic hypercomputers involve the evaporation of black holes and the eventual decay of matter, both of which are possible within a finite amount of time according to some current physical theories. Either of these possibilities might prevent the completion of a putative hypercomputation. I cannot do justice to these issues within the scope of this paper. Suffice it to say that the solutions proposed, such as sending matter into a black hole to prevent its evaporation (Németi and Dávid 2006), have not been worked out in great detail. If all of these practical obstacles could be overcome, and if our other needs were fulfilled—two very big ifs—then relativistic hypercomputers would be physically constructible.

The final constraint is reliability. One serious problem is the successful decoding of the signal, coming from λ, that carries the result of the computation. It is not easy to ensure that the signal is distinguishable from other possible signals coming from outer space. In addition, the signal is subject to amplification. If the signal has infinitely long duration, upon amplification it will destroy the receiving agent (Earman and Norton 1993). In such a case, the completion of the computation leads to the user's destruction. If the signal is finite, under Etesi and Németi's proposal, gravitational forces would shorten the signal: as the receiver approaches the edge, the signal tends to zero length. Therefore, decoding the signal will require time measurements of unbounded precision (Etesi and Németi 2002). Since such measurements are unlikely to be available and appear to violate quantum mechanical constraints, Németi and Dávid (2006) and Andréka, Németi, and Németi (2009, 508-9) have proposed alternative solutions to the reception problem. One solution involves sending a messenger from λ towards g; roughly speaking, although the messenger cannot reach g, it can come close enough to transmit the signal in a decodable form without damaging the receiver. None of the solutions appear especially easy to implement.

Even if the problem of decoding the signal can somehow be solved, two more problems remain. First, it is unlikely that a machine can operate for an infinite amount of time without breaking down and being eventually unable to repair itself.[31] Second, as Pitowsky (2009) argues, the singularities involved in relativistic hypercomputation (such as electromagnetic waves whose energies are too high to be described by known laws of physics) may indicate a limitation of General Relativity rather than the physical possibility of hypercomputation.

Relativistic hypercomputers are fascinating. They are a fruitful contribution to the debate on the foundations of physics. But at the moment, they are not even close to being technologically practical. It is extremely unlikely that relativistic hypercomputers will ever be built and used successfully. So for now and the foreseeable future, relativistic hypercomputers do not falsify Modest Physical CT.

4.3 Other Challenges to Modest Physical CT

While I lack the room for a detailed treatment of all hypercomputer designs proposed in the literature, I will briefly mention two other widely discussed classes of computing systems.

Neural networks have sometimes been proposed as computing systems that may go beyond Turing-computability.[32] This opinion is unwarranted. During the last couple of decades, there has been considerable progress in the study of the computational and complexity properties of large classes of neural networks. The relevant systems have

[31] Button has independently argued that a relativistic hypercomputer will malfunction with probability 1 and concludes that this is enough to make it useless (2009, 779).
[32] Cf.: "connectionist models … may possibly even challenge the strong construal of Church's Thesis as the claim that the class of well-defined computations is exhausted by those of Turing machines" (Smolensky 1988, 3). See also Horgan 1997, 25.

digital inputs and outputs (so as to satisfy Constraint 1) but may have, and typically do have, non-digital internal processes. If we restrict our attention to classes of connectionist systems that contain all systems with current or foreseeable practical applications, the main results are the following. Feedforward networks with finitely many processing units are computationally equivalent to Boolean circuits with finitely many gates. Recurrent networks with finitely many units are equivalent to finite state automata. Networks with unbounded tapes or with an unbounded number of units are equivalent to Turing machines.[33]

Neural networks more powerful than Turing machines may be defined, however, by exploiting the expressive power of real numbers. The best-known networks of this kind are Analog Recurrent Neural Networks (ARNNs) (Siegelmann 1999). ARNNs should not be confused with analog computers in the traditional sense (Pour-El 1974, Rubel 1993, Mills 2008). Whereas analog computers manipulate real variables without relying on the exact value of arbitrary real-valued quantities, ARNNs manipulate strings of digits by (possibly) relying on the exact value of arbitrary real-valued quantities. Specifically, the weights connecting individual processing units within ARNNs can take exact values of arbitrary real numbers, including values that are Turing-uncomputable. When their weights are Turing-uncomputable, ARNNs can go beyond the power of Turing-machines: they can compute any function over binary strings. The very features that make some ARNNs more powerful than Turing machines, however, also prevent them from being built and operated reliably. The reasons are roughly the same that

[33] For some classical results and discussions, see McCulloch and Pitts 1943, Kleene 1956, Minsky 1967, Minsky and Papert 1988. For more recent results, reviews, and discussions, see Hartley and Szu 1987, Hong 1988, Franklin and Garzon 1990, van der Velde 1993, Siu et al 1995, Sima and Orponen 2003. For a general account of digital computation in neural networks, see Piccinini 2008b.

militate against Blum et al.’s “computations”: first, hypercomputational ARNNs require unboundedly precise weights, and second, such weights are not Turing computable (Davis 2004, Schonbein 2005, Siegelmann 1999, 148).

Quantum computing has also been invoked as a possible source of hypercomputation. Quantum computing is the manipulation of qubits (or more generally, qudits) in accordance with the laws of quantum mechanics. Qubits are variables that, like bits, can be prepared or measured in one or two states, 0 and 1. Unlike bits, qubits can (i) take states that are a superposition of 0 and 1 and (ii) become entangled with each other during a computation. A surprising feature of quantum computing is that it allows computing certain functions much faster than any known classical computation (Shor 1994). But while mainstream quantum computing may be more efficient than classical computing, it does not allow computing any functions beyond those computable by Turing machines (Nielsen and Chuang 2000).

Some authors have questioned whether the mainstream quantum computing paradigm is general enough and, if not, whether some aspects of quantum mechanics may be exploited to design a quantum hypercomputer (Nielsen 1997, Calude and Pavlov 2002). The most prominent proposal for a quantum hypercomputer is by Tien Kieu (2002, 2003, 2004, 2005). He argues that an appropriately constructed quantum system can decide whether an arbitrary Diophantine equation has an integral solution—a problem which is known to be unsolvable by Turing machines. Kieu’s method involves encoding a specific instance of the problem in an appropriate Hamiltonian, which represents the total energy of a quantum system. Kieu shows that such a system can dynamically evolve over time into an energy ground state that encodes the desired solution. Unfortunately, Kieu's scheme does not appear to be workable. For one thing, it requires infinite precision in setting up and maintaining the system (Hodges 2005). For another thing, Kieu does not provide a successful criterion for knowing when the system has evolved into the solution state, and the problem of determining when the solution state is reached is unsolvable by Turing machines (Smith 2006b, Hagar and Korolev 2007, Pitowsky 2007). Thus, operating Kieu's proposed hypercomputer would require already possessing hypercomputational powers.

In conclusion, the most prominent candidate hypercomputers proposed so far have not been shown to be physically constructible and reliable.[34] For the time being, Modest Physical CT remains plausible. It may well be that, for all practical purposes, any function that is physically computable is Turing-computable.

5. Conclusion

In the literature on computation in physical systems, there is growing concern with whether various proposals for physical computation lead to physically usable processes (e.g., Fitz 2006, Németi and Dávid 2006, Ord 2006, Smith 2006a, Beggs and Tucker 2007, Button 2009). In this paper, I have urged a more explicit, careful, and systematic treatment of this problem and the related problem of formulating an adequate version of Physical CT.

In formulating Physical CT, we should mind the reason computability is interesting in the first place—it's the epistemological notion of which antecedently

[34] For some related skepticism about these and other hypercomputation proposals, see Cotogno 2003, Davis 2006, Galton 2006, Potgieter 2006. For a mistake in Cotogno's paper (which is irrelevant here), see Welsch 2004 and Ord and Kieu 2005.

defined problems, defined over denumerable domains, can be solved, either by effective procedures (Mathematical CT) or by physical means (Physical CT).

Mathematical CT does not rule out the possibility that we construct a genuine hypercomputer. It merely rules out the possibility that a hypercomputer computes Turing-uncomputable functions by following an effective procedure.

If we formulate physical versions of Physical CT too strongly—what I called Bold Physical CT—we face two related problems. First, it becomes relatively easy to falsify CT, but the putative counterexamples have no apparent practical utility. Second, and more importantly, these versions of CT are irrelevant to the epistemological notion of computability that motivated CT in the first place. They change the subject.

There is nothing wrong with changing the subject if you are interested in something else. There are legitimate and interesting questions pertaining to which aspects of which physical systems can be computationally approximated to what degree, which systems of equations give rise to computable solutions, and more. These questions are not the same as whether Physical CT (properly so called) is true, but they do deserve to be investigated in their own right.

Physical CT should be formulated modestly, using a notion of physical computation that is broader than that of computation by effective procedure but considerably narrower than the general notion of a physical process. Such a notion should satisfy a set of usability constraints: For a process to count as a genuine physical computing system, it need not follow an effective procedure, but it must still be usable by a finite observer to solve problems of a certain kind. That is, it must generate readable outputs from readable inputs according to a fixed rule that links the inputs to the outputs without making reference to the process itself. It must also be repeatable, settable, constructible, and reliable.

It is important to understand the exact scope of Modest Physical CT. Modest Physical CT does not entail that everything physical is a computing system. It only says that *if* something physical is a computing system, *then* the functions it computes are Turing-computable.

Modest Physical CT is true if and only if genuine hypercomputers are impossible in the sense that they do not satisfy our usability constraints. Whether any hypercomputer does satisfy our usability constraints remains an open empirical question, as does Modest Physical CT. As yet, however, there is no hard evidence against it. Instead, there are good reasons to believe Modest Physical CT. All computing systems that have been physically built, or are in the process of being built, only compute functions that are Turing-computable.

References

Andréka, H., I. Németi, and P. Németi (2009). "General Relativistic Hypercomputing and Foundation of Mathematics." *Natural Computing* **8**: 499-516.

Arkoudas, K. (2008). "Computation, Hypercomputation, and Physical Science." *Journal of Applied Logic* **6**: 461-475.

Beggs, E. J. and J. V. Tucker (2007). "Can Newtonian Systems, Bounded in Space, Time, Mass and Energy Compute all Functions?" *Theoretical Computer Science* **371**: 4–19.

Black, R. (2000). "Proving Church's Thesis." *Philosophia Mathematica* **8**: 244-258.

Blum, L., F. Cucker, et al. (1998). *Complexity and Real Computation*. New York, Springer.

Bowie, G. L. (1973). "An Argument Against Church's Thesis." *The Journal of Philosophy* **70**: 66-76.

Button, T. (2009). "SAD Computers and Two Versions of the Church-Turing Thesis." *British Journal for the Philosophy of Science* **60**: 765-792.

Calude, C. S. (2005). “Algorithmic Randomness, Quantum Physics, and Incompleteness,” in *Proceedings of the Conference "Machines, Computations and Universality" (MCU 2004)*, M. Margenstern (ed.), Berlin: Springer, pp. 1-17.

Calude, C. S., and B. Pavlov (2002). “Coins, Quantum Measurements, and Turing's Barrier,” *Quantum Information Processing* **1**(1-2): 107-127.

Church, A. (1932). "A Set of Postulates for the Foundation of Logic." *Annals of Mathematics* **33**: 346-366.

Cleland, C. E. (1993). "Is the Church-Turing Thesis True?" *Minds and Machines* **3**: 283-312.

Churchland, P. M. and P. S. Churchland (1990). "Could a Machine Think?" *Scientific American* **CCLXII**: 26-31.

Copeland, B. J. (2000). "Narrow Versus Wide Mechanism: Including a Re-Examination of Turing's Views on the Mind-Machine Issue." *The Journal of Philosophy* **XCVI**(1): 5-32.

Copeland, B. J. (2002a). "Hypercomputation." *Minds and Machines* **12**: 461-502.

Copeland, B. J. (2002b). "Accelerating Turing Machines." *Minds and Machines* **12**: 281-301.

Copeland, B. J. (2002c). The Church-Turing Thesis. *The Stanford Encyclopedia of Philosophy (Fall 2002 Edition)*. E. N. Zalta. URL = <http://plato.stanford.edu/archives/fall2002/entries/church-turing/>.

Copeland, B. J. (2006). Turing's Thesis. *Church's Thesis after 70 Years*. A. Olszewski, J. Wolenski and R. Janusz. Heusenstamm, Ontos**:** 147-174.

Copeland, J. and O. Shagrir (2007). "Physical Computation: How General are Gandy's Principles for Mechanisms?" *Minds and Machines* **17**(2): 217 - 231.

Cotogno, P. (2003). "Hypercomputation and the Physical Church-Turing Thesis." *British Journal for the Philosophy of Science* **54**: 181-223.

Davies, E. B. (2001). "Building Infinite Machines." *British Journal for the Philosophy of Science* **52**(4): 671-682.

Davis, M. (1982). "Why Gödel Didn't Have Church's Thesis." *Information and Control* **54**: 3-24.

Davis, M. (2004). The Myth of Hypercomputation. *Alan Turing: Life and Legacy of a Great Thinker*. C. Teuscher. Berlin, Springer**:** 195-211.

Davis, M. (2006). The Church-Turing Thesis: Consensus and Opposition. *Logical Approaches to Computational Barriers: Second Conference on Computability in Europe, CiE 2006, Swansea, UK, July 2006, Proceedings*. A. Beckmann, U. Berger, B. Löwe and J. V. Tucker. Berlin, Springer-Verlag**:** 125-132.

Deutsch, D. (1985). "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer." *Proceedings of the Royal Society of London A* **400**: 97-117.

Earman, J. (1986). *A Primer on Determinism*. Dordrecht, D. Reidel.

Earman, J. and J. Norton (1993). "Forever is a Day: Supertasks in Pitowsky and Malament-Hogarth Spacetimes." *Philosophy of Science* **60**: 22-42.

Etesi, G. and I. Németi (2002). "Non-Turing Computations via Malament-Hogarth Spacetimes." *International Journal of Theoretical Physics* **41**: 342-370.

Fitz, H. (2006). Church's Thesis and Physical Computation. *Church's Thesis after 70 Years*. A. Olszewski, J. Wolenski and R. Janusz. Heusenstamm, Ontos**:** 175-219.

Folina, J. (1998). "Church's Thesis: Prelude to a Proof." *Philosophia Mathematica* **6**: 302-323.

Franklin, S. and M. Garzon (1990). Neural Computability. *Progress in Neural Networks*. O. Omidvar. Norwood, NJ, Ablex**:** 127-145.

Galton, A. (2006). "The Church–Turing Thesis: Still Valid After All These Years?" *Applied Mathematics and Computation* **178**: 93-102.

Gandy, R. (1980). Church's Thesis and Principles for Mechanism. *The Kleene Symposium*. J. Barwise, H. J. Keisler and K. Kuhnen. Amsterdam, North-Holland**:** 123-148.

Gandy, R. (1988). The Confluence of Ideas in 1936. *The Universal Machine: A Half-Century Survey*. R. Herken. New York, Oxford University Press**:** 55-111.

Gödel, K. (1934/1965). On Undecidable Propositions of Formal Mathematical Systems. *The Undecidable*. M. Davis. Ewlett, NY, Raven**:** 41-71.

Gödel, K. (1936). "Über die Lange von Beweisen." *Ergebnisse eines mathematischen Kolloquiums* **7**: 23-24.

Gödel, K. (1946/2004). Remarks Before the Princeton Bicentennial Conference on Problems in Mathematics. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. M. Davis. Dover, Mineola**:** 84-88.

Hagar, A. and A. Korolev (2007). "Quantum Hypercomputation--Hype or Computation?" *Philosophy of Science* **74**(3): 347-363.

Hamkins, J. D. (2002). "Infinite Time Turing Machines." *Minds and Machines* **12**: 521-539.

Hartley, R. and H. Szu (1987). *A Comparison of the Computational Power of Neural Network Models*. Proceedings of the IEEE First International Conference on Neural Networks, San Diego, IEEE Press.

Hodges, A. (2005). “Can quantum computing solve classically unsolvable problems?” http://arxiv.org/pdf/quant-ph/0512248.

Hodges, A. (2006). Did Church and Turing Have a Thesis About Machines? *Church's Thesis after 70 Years*. A. Olszewski, J. Wolenski and R. Janusz. Heusenstamm, Ontos**:** 242-252.

Hogarth, M. (1994). "Non-Turing Computers and Non-Turing Computability." *PSA 1994*: 126-138.

Hogarth, M. L. (2004). "Deciding Arithmetic Using SAD Computers." *British Journal for the Philosophy of Science* **55**: 681-691.

Horgan, T. (1997). “Connectionism and the Philosophical Foundations of Cognitive Science.” *Metaphilosophy* **28**: 1-30.

Hong, J. (1988). "On Connectionist Models." *Communications on Pure and Applied Mathematics* **XLI**: 1039-1050.

Kálmar, L. (1959). An Argument Against the Plausibility of Church's Thesis. *Constructivity in Mathematics*. A. Heyting. Amsterdam, North-Holland**:** 72-80.

Kieu, T. D. (2002). "Quantum Hypercomputation." *Minds and Machines* **12**: 541-561.

Kieu, T. D. (2003). “Computing the Noncomputable,” *Contemporary Physics*, 44: 51-71.

Kieu, T. D. (2004). “A Reformulation of Hilbert's Tenth Problem through Quantum Mechanics,” *Proceedings of the Royal Society A*, **460**(2045): 1535-1545.

Kieu, T. D. (2005). “An Anatomy of a Quantum Adiabatic Algorithm that Transcends the Turing Computability,” *International Journal of Quantum Information* **3**(1): 177-183.

Kleene, S. C. (1935). "A Theory of Positive Integers in Formal Logic." *American Journal of Mathematics* **57**: 153 - 173 and 219 - 244.

Kleene, S. C. (1952). *Introduction to Metamathematics*. Princeton, Van Nostrand.

Kleene, S. C. (1987). "Reflections on Church's Thesis." *Notre Dame Journal of Formal Logic* **28**(4): 490-498.

Kleene, S. C. (1952). *Introduction to Metamathematics*. Princeton, Van Nostrand.

McCulloch, W. S. and W. H. Pitts (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics* **7**: 115-133.

Mendelson, E. (1990). "Second Thoughts about Church's Thesis and Mathematical Proofs." *The Journal of Philosophy* **88**: 225-233.

Mills, J. W., (2008). "The Nature of the Extended Analog Computer," *Physica D*: *Nonlinear Phenomena*, **237**(9): 1235-1256.

Minsky, M. (1967). *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ, Prentice-Hall.

Minsky, M. L. and S. A. Papert (1988). *Perceptrons: An Introduction to Computational Geometry*.

Mundici and W. Sieg (1995). "Paper Machines." *Philosophia Mathematica* **3**(3): 5-30.

Németi, I. and G. Dávid (2006). "Relativistic Computers and the Turing Barrier." *Journal of Applied Mathematics and Computation* **178**(1): 118-142.

Newell, A. (1980). "Physical Symbol Systems." *Cognitive Science* **4**: 135-183.

Nielsen, M. A. (1997). "Computable Functions, Quantum Measurements, and Quantum Dynamics," *Physical Review Letters*, **79**(15): 2915-2918.

Nielsen, M. A., and Chuang, I. L. (2000). Quantum Computation and Quantum Information. Cambridge, Cambridge University Press.

Odifreddi, P. (1989). *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers*. Amsterdam, North-Holland.

Ord, T. (2006). "The Many Forms of Hypercomputation." *Applied Mathematics and Computation* **178**: 143-153.

Ord, T. and T. D. Kieu (2005). "The Diagonal Method and Hypercomputation." *British Journal for the Philosophy of Science* **56**: 147-156.

Piccinini, G. (2003). "Alan Turing and the Mathematical Objection." *Minds and Machines* **13**(1): 23-48.

Piccinini, G. (2007a). "Computational Modeling vs. Computational Explanation: Is Everything a Turing Machine, and Does It Matter to the Philosophy of Mind?" *Australasian Journal of Philosophy* **85**(1): 93-115.

Piccinini, G. (2007b). "Computing Mechanisms." *Philosophy of Science* **74**(4): 501-526.

Piccinini, G. (2008a). "Computers." *Pacific Philosophical Quarterly* **89**(1): 32-73.

Piccinini, G. (2008b). "Some Neural Networks Compute, Others Don't." *Neural Networks*.

Pitowsky, I. (1990). "The Physical Church Thesis and Physical Computational Complexity." *Iyyun* **39**: 81-99.

Pitowsky, I. (2002). "Quantum Speed-Up of Computations." *Philosophy of Science* **69**: S168-S177.

Pitowsky, I. (2007). From Logic to Physics: How the Meaning of Computation Changed Over Time. *Computation and Logic in the Real World, Proceedings of the Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007*. S. B. Cooper, B. Lowe and A. Sorbi. Lecture Notes in Computer Science 4497**:** 621-631.

Pour-El, M. B. (1974). "Abstract Computability and Its Relation to the General Purpose Analog Computer (Some Connections Between Logic, Differential Equations and Analog Computers)." *Transactions of the American Mathematical Society* **199**: 1-28.

Potgieter, P. H. (2006). "Zeno Machines and Hypercomputation." *Theoretical Computer Science* **358**: 23-33.

Rescorla, M. (2007). "Church's Thesis and the Conceptual Analysis of Computability." *Notre Dame Journal of Formal Logic* **2**: 253-280.

Rubel, L. A. (1989). "Digital Simulation of Analog Computation and Church's Thesis," *Journal of Symbolic Logic,* **54**(3): 1011-1017.

Rubel, L. A. (1993). "The Extended Analog Computer," *Advances in Applied Mathematics* **14**(1): 39-50.

Schoenbein, W. (2005). "Cognition and the Power of Continuous Dynamical Systems." *Minds and Machines* **15**(57-71).

Shagrir, O. (1997). "Two Dogmas of Computationalism." *Minds and Machines* **7**(3): 321-344.

Shagrir, O. (2002). "Effective Computation by Humans and Machines." *Minds and Machines* **12**: 221-240.

Shagrir, O. (2006). Gödel on Turing on Computability. *Church's Thesis after 70 Years*. A. Olszewski, J. Wolenski and R. Janusz. Heusenstamm, Ontos**:** 393-419.

Shagrir, O. and I. Pitowsky (2003). "Physical Hypercomputation and the Church-Turing Thesis." *Minds and Machines* **13**(1): 87-101.

Shapiro, S. (1981). "Understanding Church's Thesis." *Journal of Philosophical Logic* **10**: 353-365.

Shapiro, S. (1993). "Understanding Church's Thesis, Again." *Acta Analytica* **11**: 59-77.

Sieg, W. (1994). Mechanical Procedures and Mathematical Experience. *Mathematics and Mind*. A. George. New York, Oxford University Press**:** 71-117.

Sieg, W. (1997). "Step by Recursive Step: Church's Analysis of Effective Calculability." *Bulletin of Symbolic Logic* **3**(2): 154-180.

Sieg, W. (2002). Calculations by Man and Machine: Conceptual Analysis. *Reflections on the Foundations of Mathematics (Essays in Honor of Solomon Feferman)*. W. Sieg, R. Sommer and C. Talcott, Association for Symbolic Logic. **15:** 390-409.

Sieg, W. (2005). "Only Two Letters: The Correspondence between Herbrand and Gödel." *Bulletin of Symbolic Logic* **11**(2): 172-184.

Sieg, W. (2006a). "Gödel on Computability." *Philosophia Mathematica* **14**: 189-207.

Sieg, W. (2006b). Computability Theory. *Handbook of the Philosophy of Science: Philosophy of Mathematics*. A. Irvine, Elsevier.

Sieg, W. (2008). "Church Without Dogma: Axioms for Computability." In *New Computational Paradigms: Changing Conceptions of What is Computable*, S. B. Cooper, B. Löwe, and A. Sorbi (Eds.). Springer, New York: 139-152.

Siegelmann, H. T. (1999). *Neural Networks and Analog Computation: Beyond the Turing Limit*. Boston, MA, Birkhäuser.

Šíma, J. and P. Orponen (2003). "General-purpose Computation with Neural Networks: A Survey of Complexity Theoretic Results." *Neural Computation* **15**: 2727-2778.

Siu, K.-Y., V. Roychowdhury, et al. (1995). *Discrete Neural Computation: a Theoretical Foundation*. Englewood Cliffs, NJ, Prentice Hall.

Smith, W. D. (2006a). "Church's Thesis Meets the *N*-body Problem." *Applied Mathematics and Computation* **178**: 154–183.
Smith, W. D. (2006b). “Three Counterexamples Refuting Kieu’s Plan for Quantum Adiabatic Hypercomputation; and Some Uncomputable Quantum Mechanical Tasks.” *Applied Mathematics and Computation* **178**(1): 184-193.
Smith, P. (2007). *Gödel's Proofs*. Cambridge, Cambridge University Press.
Smolensky, P. (1988). "On the Proper Treatment of Connectionism." *Behavioral and Brain Sciences* **11**: 1-23.
Soare, R. (1999). The History and Concept of Computability. *Handbook of Computability Theory*. E. R. Griffor. New York, Elsevier**:** 3-36.
Stannett, M. (1990). "X-Machines and the Halting Problem: Building a Super-Turing Machine." *Formal Aspects of Computing* **2**: 331-341.
Turing, A. M. (1936-7 [1965]). On computable numbers, with an application to the Entscheidungsproblem. *The Undecidable*. M. Davis. Ewlett, Raven**:** 116-154.
Turing, A. M. (1950). "Computing Machinery and Intelligence." *Mind* **59**: 433-460.
Welch, P. D. (2004). "On the Possibility, or Otherwise, of Hypercomputation." *British Journal for the Philosophy of Science* **55**: 739-746.
Welch, P. D. (2008). “The Extent of Computation in Malament–Hogarth Spacetimes,” *British Journal for the Philosophy of Science* **59**: 659–74.
Wolfram, S. (1985). "Undecidability and Intractability in Theoretical Physics." *Physical Review Letters* **54**: 735-738.
van der Velde, F. (1993). "Is the Brain an Effective Turing Machine of a Finite-state Machine?" *Psychological Research* **55**: 71-79.