Physical Hypercomputation and the Church-Turing Thesis

Oron Shagrir (shagrir@cc.huji.ac.il) and Itamar Pitowsky (itamarp@vms.huji.ac.il).
Department of Philosophy, The Hebrew University of Jerusalem, Israel.

Abstract. We describe a possible physical device that computes a function that cannot be computed by a Turing machine. The device is physical in the sense that it is compatible with General Relativity. We discuss some objections, focusing on those which deny that the device is either a computer or computes a function that is not Turing computable. Finally, we argue that the existence of the device does not refute the Church-Turing thesis, but nevertheless may be a counterexample to Gandy's thesis.

1. Introduction

A *hypercomputer* is a physical or an abstract system that computes functions that cannot be computed by a universal Turing machine. Turing (1939) was perhaps the first to introduce hypercomputers. He called them o-machines, for machines with oracles. Other examples of hypercomputers are described in Copeland and Sylvan (1999). In what follows we describe a possible *physical* device that computes a non-Turing computable function. This physical hypercomputer is a version of the devices introduced by Pitowsky (1990) and Hogarth (1992, 1994). We consider some objections to the claim that the device is indeed a physical hypercomputer. Earman and Norton (1993) discuss the physical possibility of this device in detail. Here we focus on the logical problems it raises. Finally, we argue that the existence of the device is no counterexample to the Church-Turing thesis (CTT), but nevertheless may refute Gandy's thesis.

2. Physical hypercomputation

A *physical* machine is an actual or possible system whose dynamics are in accord with the principles of current physics. Are there physical hypercomputers? In considering this question, it is essential to distinguish the analog case from the digital. The analog case is the more complex, as it requires extending Turing-computability to functions defined over real numbers. When the extended notion is in place, it turns out that many of the solutions to the equations governing physical dynamics are Turing-computable. The only known exception is given by Pour-El and Richards (1981) who constructed examples for the wave equation in which the initial data is Turing-computable, but the solution is not.

We describe here the only known physical *digital* hypercomputer. This device performs a supertask, that is, it can carry out an infinite number of steps in a finite time-span. The existence of such devices is compatible with General Relativity (Pitowsky 1990; Hogarth 1992, 1994).[1] There are solutions to Einstein's equations on which there is a time-like half-curve γ, and a point p in spacetime, such that the entire stretch of γ is included in the chronological past of p.[2] In such spacetimes we can have a hyper-machine, HM, made up of a pair of communicating standard computers, T_A and T_B. T_B moves along γ, and T_A along a future directed time-like curve that connects the beginning point q of γ with p. The time it takes T_A to travel from q to p is finite, while during that period T_B completes the infinite time trip along γ. HM operates as follows (figure 1): We feed T_A with input n. T_A sends a signal with the pertinent input n to T_B. T_B is a universal machine that mimics the computation of the n'th Turing machine operating on input n. In other words, T_B calculates the Turing-computable function f(n) that returns the output of the n'th Turing machine (operating

on input n) if this Turing machine halts, and returns no value if it does not. If T_B halts at some point, it immediately sends a signal back to T_A. If T_B never halts, it never sends a signal. Meanwhile T_A 'waits' the time it takes T_A to travel from q to p (say, a second). If T_A received a signal from T_B it prints '1'. If it received no signal from T_B it prints '0'.[3] The machine HM computes then the function h, where h(n)=1 if f(n) is defined, and h(n)=0 otherwise. But h is not Turing-computable as h characterizes the self-halting states of Turing Machines. We, therefore, described a physical machine that computes a function that is non-Turing-computable.

Let us consider three objections to our contention that HM is a physical system that computes the function h. The first objection is that HM is *not physical*. The second is that it is *not a computer*. And the third is that HM does not compute h, but another function.

Objection #1: HM is not a physical device. Its implementation is not a physical possibility. There will never be enough hardware to implement the required memory space, not even in principle.

Reply: In 1990 Pitowsky came up with a machine, based on extreme acceleration, that functions in accordance with Special Relativity. He suggested that similar set-ups could be replicated by spacetime structures in General Relativity. Malament (private communications) and later, Hogarth (1992, 1994) provided examples of such spacetime structures (e.g., anti de Sitter spacetimes). Earman and Norton (1993) discuss in detail the physical possibility of machines that perform supertasks in M-H spacetimes (M-H for Malament and Hogarth). They point out that the physical reality of M-H spacetimes, and the existence of HM are open questions that depend on the "resolution of some of the deepest foundations problems in classical general relativity, including the nature of singularities and the fate of cosmic censorship" (pp. 40-41).

We have nothing to add to Earman and Norton on these issues. Our discussion focuses on the problem of the physical realization of the requisite memory space. The problem is that in the cases T_B does not halt, HM requires an unbounded number of memory cells. It thus seems that there are not enough particles in the universe to supply the requisite memory space.

One could reply that the objection challenges standard computers as well. A standard computer for addition also requires an unbounded memory, and could also run out of memory when the two input numbers are large enough. But there is a difference. In the standard case, each single computation (a single computation being computation of one sum) has an upper bound on memory space. But in the case of HM, some single computations do not have upper bound on memory. Put differently, it is true that standard computers do not satisfy the following condition:

(I) $\Diamond \exists m \forall n C_f(n,m)$

($\Diamond$ stands for physical possibility, m for the size of memory space, n for a vector of numbers, and $C_f(n,m)$ for the single computation of f(n) with memory m).[4] Still, in the standard case each single computation can be completed. For any single computation, there is a possible physical world with enough memory that can be used for this computation. Thus a standard computer for addition is physically possible by virtue of satisfying the following condition:

(II) $\forall n \Diamond \exists m C_f(n,m)$.

But HM, the objection goes, does not satisfy even (II), and so is not physically possible.

The claim that standard computers for addition satisfy (II) could be challenged. The adequacy of the reasons for adopting (II) and not another condition could also be contended. Our reply, however, is that it may be the case that HM could

satisfy (II). The implementation of the requisite memory space could be physically

possible. There are two ways to supply the memory. One is to compress the

information in a finite space. A classic example is Moore (1990), who embeds a

universal Turing machine in the motion of a particle moving in a space, bouncing

between parabolic and linear mirrors like an ideal billiard. The infinite tape and table

of instructions can be embedded in the binary development of the coordinates of the

particle's position.[5] This implementation, however, requires the power to infinitely

differentiate between points in space. Thus the machine is not a digital device. The

other option is to realize memory in an infinite space. Reissner-Nordström spacetime

is a spatially *infinite* M-H spacetime that satisfies Einstein's field equation and the

energy conditions. The problem, however, is that we cannot use an unlimited

hardware without violating other physical necessities of the machine (Earman and

Norton, 1993: 40). But as Earman and Norton also remark, there might be other M-H

spacetimes that have both space and material enough for the proper implementation of

the HM. So HM might be a physical possibility after all.

Objection #2: HM is not a computer. It arrives at the values of *h*, but it *does not compute* them. Computation requires verification, taking verification here as the strict notion that underlies the notion of formal proof. A process is computation only if we can, at least *in principle*, keep track of it. But we certainly cannot keep track of every process in HM: we cannot keep track of a process in T_B that consist of *infinitely many steps*. HM therefore is not a computer.

Reply: We agree that we cannot keep track of the process in cases where T_B never

halts. That is, we could verify the correctness of any step in the process, but we

cannot verify all of them. But, we maintain, this strict notion of verification is not

appropriate for every mode of computation. A system can compute even if we cannot

fully keep track of its processes, verifying their correctness. A paradigm example of

such machines is analog computers, such as the differential analyzer. The nature of

these machines does not allow us to keep track of their dynamical processes. Nonetheless, we take these machines to be, and use them as, computers.

Of course, we are not denying the central role of effective computations in the verification of formal proofs. Nor are we claiming that HM computes *effectively*, or that it solves the halting problem, proves theorems, or can be used to verify theorems, when all these notions are defined in terms of effective procedures (see section 3). Our only claim is that there have always been modes of computation that have little to do with proofs and the verifications thereof. The dominant place of effective computation in current computer science and engineering should not lead us to think that it is the sole mode of computation operative. Non-discrete devices for computation have been around for hundreds of years, and analog computers have been planned, built and used since the 19th century.[6]

Nor do we deny that computation requires an explanation: if it is claimed that a device computes a function, there should be a way to *explain* what the device is doing. However, we do not believe that the only way to provide this explanation is via verification. What the device is doing can also be explained in terms of a formal or physical background theory. For instance, we do not, and often cannot, verify the function computed by analog computers. But we can use our physical theories to explain what function is being computed. Our physical theories mandate that a device with such-and-such a physical structure and initial conditions will behave according to a given set of equations, whose solutions it computes. Likewise, we have no way to verify that HM computes h. But our best physical theories can serve to explain why the machine behaves according to the function h. HM, therefore, could be used to compute h.

There is a connection between the first two objections. Objection #1 is the claim that HM is not physical, since the required memory is not physically realizable. Objection #2 can be construed as the claim that HM is not a computer, since the required memory (for a single computation of a defined value) is not finite. On this construal, objection #2 is the claim that HM does not satisfy the following condition for being a computer:

(II') $\forall n \Diamond \exists m (C_f(n,m) \,\&\, FINITE(m))$.[7]

Our reply to objection #1 is that HM might conceivably satisfy (II). Our reply to objection #2 is that (II') is not a condition for being a computer. For what could justify the additional requirement regarding memory? It is not justified by physical principles, given that the required infinite memory may be physically realizable. It is not justified by explanatory practices, given that we can explain the behavior of hyper-machines. And it is not justified by principles of verification, given that we do not apply these principles to other modes of computation.[8] We therefore see no reason why (II') should be taken as a condition on computation. We suspect, however, that the finite memory requirement has its roots in conflation of the true claim that effective computations are a means of verifying the correctness of proofs, with the false claim that the correctness of any computation must be verified.

Objection #3: HM is perhaps a computer. But it does not compute *h*. The reason is as follows: a single computation whose result is 0 exhausts the eternity of the time-like curve along T_B is moving. After this single computation, the curve, including the non-halting T_B, ceases to exist. It thus follows that HM can print the output 0 only once. After it has done so, no other value can be computed. Thus HM cannot arrive at the defined values of many of the other arguments of *h, not even in principle*.

Reply: There are two ways we can adopt to deal with this objection. One strategy would be to reject the inference from the premise that HM prints out 0 at most once to

the conclusion that HM does not compute *h*. We agree that a machine computes *h* only if it returns the halting state of an arbitrary Turing machine. But this condition can be understood in two different ways. On the weaker understanding, the machine must return the halting state of an arbitrary Turing machine. HM *passes* this test, as it can return the halting state of any one Turing machine. On the stronger understanding, the machine must be able to return, at least in principle, the halting-state of *every* Turing machine. HM *fails* this test, since after certain single computations, T_B ceases to exist. The difference between the two understandings of the condition can also be expressed in modal terms. On the weaker understanding, the relevant condition is expressed by (II), whereas on the stronger, by a stronger condition such as the following:

(III) $\forall n_1 \forall n_2 \Diamond (\exists m_1 \exists m_2 (C_f(n_1,m_1) \ \& \ C_f(n_2,m_2)))$.[9]

Which understanding is the correct remains an open question. Standard computers satisfy (III). But we can see no reason why (II) does not constitute a sufficiently strong condition on computing *h.* This being so, the condition it seems reasonable to adopt is that which least burdens the notion of computing a function, namely, (II). And on this condition, HM computes *h*.

Note that even if HM does not compute *h*, it does something that no Turing machine is able to do: it prints out the halting state of *arbitrary* Turing machine. For any Turing machine T_i, there is another Turing machine T_j that prints out the halting state of T_i (though we may not know or be able to prove that the printout is correct). But there is no single Turing machine T_j that prints out the halting state of *arbitrary* Turing machine. So HM computes something no Turing machine can compute.

There are also machines, similar to HM, which can find the truth-values of unresolved conjectures in number theory (Pitowsky, 1990). Take Goldbach's

conjecture, which asserts that every even integer larger than two is a sum of two primes. Finding the truth-value of Goldbach's conjecture, is not a non-Turing-computable function. There is a Turing machine that prints the truth-value of Goldbach's conjecture. Yet we do not know, and perhaps can never know, whether the printout is correct. But we might know the truth-value by using the following machine: we can let T_B check systematically whether the conjecture holds for each even number. If T_B finds a counterexample, it returns a signal. Thus T_A prints "Goldbach's conjecture is false" if it receives a signal, and "Goldbach's conjecture is true" if, after a second or so, it has receives no signal. Using this setup, we can also check many other conjectures whose truth-values are unknown, conjectures that may not be provable from the axioms of arithmetic. We can therefore gain knowledge – namely, knowledge of the truth-values of mathematical conjectures – that we hitherto lacked and may have had no other way of attaining.

The other strategy we could adopt in replying to objection #3 would be to take an upgraded version of HM that does compute *h*. On this strategy, we would accept the objection that (III) is a condition on computing *h*, and so that HM does not compute *h*. But we would argue that an upgraded machine does satisfy (III), and thus does compute *h*. The upgraded machine works as follows: The machine T_A can consult, as it were, an unbounded series of "oracles" T_{B1}, T_{B2}, … each of which is embedded in a different time-like curve. When T_A receives the input, it calculates, using its own clock, which oracle is available. T_A sends the signal to this oracle. The rest of the process is the same. It is clear that this machine fulfills (III). It can, in principle, compute the halting state of *every* Turing machine.

Another concern with respect to whether HM computes *h* has to do with the size of the input. If the input string is too long, T_B ceases to exist before the whole

string has been read. Thus HM can really compute only a segment of *h*. This concern can be dealt with in various ways. The string can be condensed to a rational value, in which case the machine would no longer be digital. Or, the upgraded machine can be used. T_A would then calculate not only which T_B is available, but also which T_B would be available long enough to read the input. Or, we could use HM, but adjust (by calculation) the route of T_A from q to p to fit the size of the input. This would enable us to be sure that T_B will have completed its mission when T_A arrives at p.

HM (or the upgraded version of HM) computes Σ_1 and $\prod_1$ functions. While it cannot compute other functions, higher in the arithmetical hierarchy, more complex machines are able to do so. For any function in the hierarchy, there is a machine that computes this function. We refer the reader to Hogarth (1994) for the details as well as a discussion of the physical possibilities. But a reminder is in order. The number of functions in the hierarchy is $\aleph_0$. All these machines together, therefore, can compute only a tiny fraction of the non-Turing-computable functions.

3. The Church-Turing thesis

The Church-Turing thesis (CTT) is the claim that our informal intuitions about what is *effectively* computable are captured by the well-defined notion of Turing-computability. CTT states that a number-theoretic function is effectively computable just in case it is Turing-computable. Versions of this thesis were published independently by both Turing and Church in 1936. It is often said that CTT cannot be a theorem, since we cannot apply mathematical proofs to informal notions. Turing remarked in 1936 that "all arguments [in favor of CTT] are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory

mathematically" (p. 135). But CTT can be refuted by finding a function that is effectively computable but not Turing-computable. One might suppose that our HM is a counterexample to CTT, but, we will show, this is not the case (section 3.1). Yet, HM may be a counterexample to Gandy's thesis that the functions computed by discrete deterministic mechanical devices are Turing-computable (section 3.2).

3.1 Is HC a counterexample to The Church-Turing thesis?

It has been pointed out in the past that CTT is compatible with hypercomputation.[10] An effectively computable function is a number theoretic function whose values can be calculated by means of an *effective procedure*, whereas, an effective procedure is a term of the art in logic and mathematics. M is an effective procedure just in case:

1. M is set out in terms of a finite number of exact instructions (each instruction being expressed by means of a finite number of symbols);
2. M will, if carried out without error, always produce the desired result in a finite number of steps;
3. M can (in practice or in principle) be carried out by a human being unaided by any machinery save paper and pencil;
4. M demands no insight or ingenuity on the part of the human being carrying it out. (Copeland, 1996).

On this conception, 'effective computation' (i.e., calculation by means of effective procedures) encompasses a wide, and an important, class of computations, but not necessarily all computations. It is perfectly consistent with CTT to have a device that does not compute by means of effective procedures, but does compute by other means. And it is also consistent with CTT that the functions computed by this device are not Turing-computable. Claims that hypercomputations are inconsistent with CTT rest on misconceptions of CTT. One misconception is that CTT is a broad thesis that refers to functions computed by *all* possible devices; another, that CTT is a physical thesis that refers to the functions computed by *physical* devices.[11] Yet none of the

hyper-machines described in the literature computes by means of effective procedures. In particular, HM computes h by means of infinitely many steps, and thus violates Copeland's second condition.

One might nonetheless attempt to argue that HM is a counterexample to CTT. In following a finite and fixed procedure, and completing the computation in a finite span of time, it might be claimed, HM effectively computes h, and therefore constitutes an example of a (physical) device that effectively computes a non-Turing-computable function. We agree that HM effectively computes in the sense that it follows a finite and fixed routine. But it does not effectively compute in the sense that the notion is understood in the context of CTT. In this context, effective computation is limited to processes that consist of finitely many steps (in a finite span of time). But since HM does not satisfy this constraint, it does not refute CTT.

In the context of CTT, effective computation is tied with the notions of decidability, provability, and, in particular, with the notion of formal proof.[12] That the relation "x is a proof of y", where x is a finite sequence of formulas and y is a formula, is *decidable*, is absolutely fundamental to these systems. A system in which this relation is not decidable is not a formal system for generating theorems. The role of effective computation, in this context, is to explicate the notion of decidability. When we say, in the context of proof theory, that the relation "x is a proof of y" is decidable, we mean that it can be determined, via effective computation, whether a sequence x of formulas is or is not a proof of a formula y. Or, equivalently, that the characterization function of the relation "x is a proof of y" is effectively computable.

In the 1930's, however, this notion of effective computability could no longer remain informal. The negative results concerning the scope of formal systems called for a more precise characterization. Gödel's incompleteness theorems (Gödel, 1931)

apply to a specific set of formal systems: those that include the Principia Mathematica. So "in 1931, one could be uncertain whether Gödel's incompleteness theorem applies to systems quite remote in their details from Principia Mathematica" (Kleene 1988: 43). The possibility that there might be a complete and consistent formal system for arithmetic in which the relation "x is a proof of y" is decidable had not been ruled out. Also important was the Entscheidungsproblem, formulated by Hilbert and Ackermann in 1928, which concerns the decidability of first-order predicate logic, that is, the question of whether there is an effective procedure by means of which the provability of an expression of first-order predicate logic can be determined.[13] In 1931, logicians realized that it was unlikely that there is such an effective procedure. But then again, to complete the proof that there is no suitable effective procedure, one had yet to precisely characterize what is effectively computable.

Church (1936a) and Turing (1936), as well as Kleene (1936) and Post (1936), aimed to provide a precise characterization of (the extension of) *this* notion of effective computability. There was, at that point of time, a pressing need to finalize the major limitative results that had emerged in logic and proof theory. These negative results call for a full and precise characterization of the class of effectively computable functions. And the works of 1936 provided requisite mathematical characterizations, CTT being the assertion that the characterizations are exhaustive. Of course, after 1936, it was discovered that a wide class of abstract automata and physical systems also compute by means of effective procedures. But these important developments should not mislead us into thinking that 'effective computation' refers to all possible computation or even to all physical computation. The claim that there

are systems, abstract or physical, that compute by other means, is compatible with CTT.

One may still wonder why should we limit a (terminating) process of effective computation to finitely many steps. Where do this and other constraints on effective computation come from? There is as yet no decisive answer to this question. It may well be that even Church and Turing differed on these points. The interpretation of Church and Turing that is dominant today, however, is that "both Church and Turing had in mind calculation by an abstract human being using some mechanical aids (such as paper and pencil). The word 'abstract' indicates that the argument makes no appeal to the existence of practical limits on time and space" (Gandy 1980: 123-124).[14] And since effective computations are constrained by what an abstract human can and cannot do, all terminating effective computations are limited to finitely many steps.

While Church (1936a) was less explicit about the connection between effective computation and human computers, Turing (1936) often referred to computation in terms of human computation. In addition, Turing analyzed human computability in his argument for CTT.[15] Turing observed that the process of human calculating with paper and pencil proceeds by discrete steps, producing a finite (but unbounded) sequence of symbols from a finite alphabet. At each step the action is local and consists at most of a possible change of the symbols in some of the squares, a possible change of the observed squares and a possible change in the calculator's state of mind. The constraints on these operations are:

[1] The number of symbols which may be printed [in a cell] is finite...
[2] The squares whose symbols are changed are always "observed" squares...
[3] Each new observed square is within [a bound] L of an immediately observed square
      (Turing 1936: 135-136).

Each such action is determined by the calculator's "states of mind" and the symbolic configurations the computer observes. The specific constraints on the determining factors are:

[4] There is a bound B to the number of symbols or squares the computer can observe at
        one moment…
[5] The number of states of mind which need be taken into account is finite (Turing 1936: 136).

The constraints are virtually self-evident. Thus, for example, [5] should not be understood as the claim that the number of states of mind in general is finite. It merely asserts that the number of states of mind germane to the calculation is finite. As Turing noted, this follows from the fact that we could "avoid introducing the 'state of mind' by considering a more physical and definite counterpart of it" (1936: 139), such as a finite list of written instructions.

After formulating the constraints, Turing provided an outline of a proof for the theorem stating that any number-theoretic function that can be calculated by a computer that satisfies conditions [1]-[5] is Turing computable. A comprehensive proof appears in Gandy (1980). But the idea is straightforward. Conditions [1]-[5] ensure that each computation step involves a change in one bounded part of the relevant symbolic configuration, so that the number of types of elementary steps is bounded and simple. It is then left to show that each such type of elementary step can be mimicked, perhaps by a series of steps, by a Turing machine.

Thus Turing's argument can be structured as follows:

1. A quasi-empirical thesis: A human computer satisfies conditions [1]-[5].
2. Theorem: Any number-theoretic function that can be calculated by a computer that satisfies conditions [1]-[5] is Turing computable.
3. Conclusion – CTT (Turing's version): Any number-theoretic function that can be calculated by a human computer is Turing computable.

Turing's argument is not a proof,[16] but is a very powerful argument in favor of CTT.

In light of this, *we conclude that it is very likely that CTT is true*.

We close the section with two remarks. First, the human notion of effective computation does not imply that non-human devices cannot compute effectively. The point, rather, is that effective computation is constrained by what human computers can and cannot do. Machines that are not bound by these constraints do not compute effectively, and machines that satisfy the constraints do compute effectively. Second, the notion of human computation does not refer to all cognitive or brain processes, but just to our everyday processes of calculation. Thus not only is CTT compatible with hypercomputation, it is also compatible with the claim that the computational powers of some cognitive or brain processes exceed the ones of a universal Turing machine.[17]

3.2 Gandy's thesis

Claims about the computational powers of *physical* machines have been put forward. They generally assert that the functions computed by physical devices can be computed by a universal Turing Machine. A well-known example is the thesis advanced by Robin Gandy (1980).[18] We first review Gandy's thesis, then explain why we believe it might be false.

Gandy's thesis states that any number-theoretic function that can be computed by a discrete deterministic mechanical device is Turing-computable. The thesis can be seen as an extension of Turing's analysis, which basically targets human computers, to discrete deterministic mechanical processes in general.[19] Furthermore, Gandy's argument for his thesis has precisely the same structure as Turing's argument. That is, Gandy contends that every discrete deterministic physical device must satisfy a set of constraints, then proves that the behavior of a device satisfying these constraints can be simulated by a Turing Machine. Gandy's argument can be summarized as follows:

1. "**Thesis P.** *A discrete deterministic mechanical device satisfies principles I-IV below.*" (1980: 126)
2. "**Theorem.** *What can be calculated by a device satisfying principles I-IV is computable.*" (1980: 126)
3. **Gandy's thesis.** What can be calculated by a discrete deterministic mechanical device is [Turing] computable.

Principles I-IV are roughly these:

(I)   The form of description: Any discrete deterministic mechanical device M can be described as <S,F> where S is a structural set of state-descriptions and F is a structural operation from S to S. Thus if S_0 describes the initial state, then $F(S_0), F(F(S_0)),\dots$ describes the subsequent states of M.

(II)   The Principle of Limitation of Hierarchy: The set-theoretic rank of the states is bounded.

(III)   The Principle of Unique Reassembly: Any state can be assembled from parts of bounded size.

(IV)   Local causation: Parts from which F(x) can be reassembled depend only on bounded parts of x.

We cannot discuss the fine points of Gandy's difficult paper here (we refer the reader to Sieg and Byrnes (1999) for a detailed but simpler presentation of Gandy's argument). However, the intuitive idea is this: A discrete deterministic mechanical device is a pair <S,F> where S is a potentially infinite set of states and F is a state-transition operation from S to S. Each state consists of atomic parts, whose number is unbounded. Yet the number of *kinds* of atomic parts of which all the states consist is finite, and these parts can be differentiated. As such, the devices "are in a loose sense, digital computers" (p. 126). Each state-transition F of the machine is a result of an operation on arbitrarily many bounded parts. The operations are deterministic in the sense that any subsequent behavior "is uniquely determined once a complete description of its initial state is given." (p. 126). The device is mechanical in the sense that it is a physical machine that could, in principle, be built. As such, it must satisfy two minimal physical requirements: "that there is a lower bound on the linear
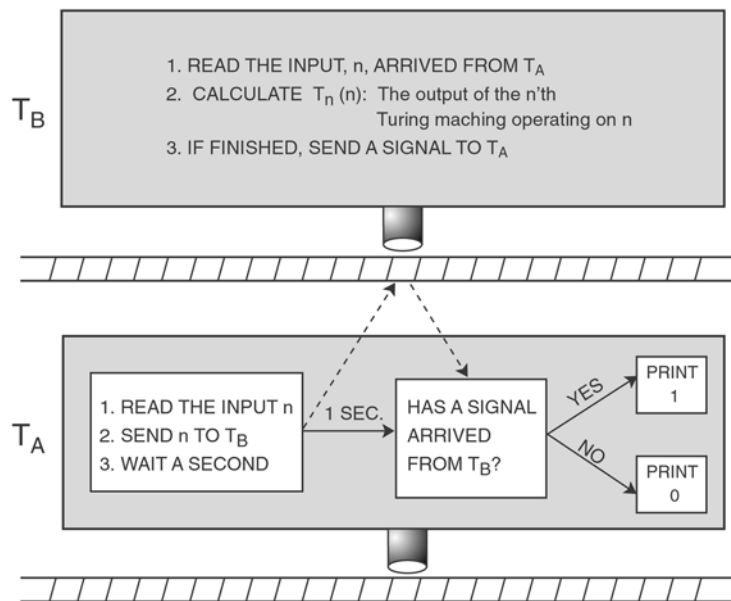
dimensions of every atomic part of the device and that there is an upper bound (the velocity of light) on the speed of propagation of changes." (p. 126).

The justification of these principles does not appeal to intuitions about human computation, but rests on physical maxims. In particular, principle IV is justified by the two minimal physical constraints just mentioned. These constraints imply that each atom in the machine can affect and be affected by a finite number of other atoms in a given state transition. Since the propagation of information is bounded, an atom can transmit and receive information only in its bounded neighborhood (in bounded time). And since there is a lower bound on the size of the atoms, the number of atoms in this bounded neighborhood is finite. It thus follows that F(x) consists of bounded, though perhaps overlapping, parts of x.

But it now seems that Gandy's thesis might be false, as HM seems to be a counterexample. HM is physical, in the sense that its dynamics satisfy the principles of General Relativity. HM is mechanical, in the sense that it satisfies the physical constraints concerning the size of the atoms and the speed of signal propagation. HM is deterministic, since the pertinent principles of General Relativity are deterministic, and HM is discrete, since it is just a pair of communicating digital machines. Finally, HM computes h. Its processes are the conjunction of the computations of T_A and of T_B, and it arrives at the values of the function h in a finite span of time (relative to A, where the inputs and output are defined). So HM is a discrete deterministic mechanical device that calculates a non-Turing-computable function.

If Gandy's thesis is false, and given that Gandy's argument is valid, we must ask which premise of his argument is false. HM is no counterexample to *Thesis P*, since HM apparently satisfies principles I-IV.[20] What about the second premise, Gandy's Theorem? In proving it, Gandy implicitly assumes that a discrete

deterministic mechanical device accomplishes a finite number of state-transitions in finite time. This means that what Gandy really proves is that whatever can be calculated *in a finite number of steps* by a device that satisfies principles I-IV is Turing-computable. But what grounds are there for the assumption that the calculation must terminate within a finite number of steps? It is not grounded in physical principles, as General Relativity is consistent with mechanical devices (e.g., HM) that go through infinitely many steps in a finite span of time, and Gandy himself assumes Relativity and not Newtonian Mechanics (see note 20). Nor does it seem to be grounded in Gandy's notion of discrete devices, given that he equates discrete with digital, and that HM consists of two communicating digital devices. The assumption cannot also be rooted in our conception of computation, given that HM is a computing device (see above, section 2, objection #2). It thus seems that Gandy is really arguing for the thesis that the functions computed by discrete deterministic mechanical devices *in a finite number of steps* are Turing-computable.[21]

Figure 1: A hypercomputer for the self-halting problem

T_B

1. READ THE INPUT, n, ARRIVED FROM T_A
2. CALCULATE $T_n(n)$: The output of the n'th
   Turing maching operating on n
3. IF FINISHED, SEND A SIGNAL TO T_A

T_A

1. READ THE INPUT n
2. SEND n TO T_B
3. WAIT A SECOND

1 SEC.

HAS A SIGNAL ARRIVED FROM T_B?

YES → PRINT 1

NO → PRINT 0

Notes

[1] The idea of logically possible machines that perform supertasks is an old one. For a comprehensive discussion of supertasks, see Laraudogoitia (1999). For a detailed discussion of (accelerating) Turing-machines that perform supertasks, see Copeland (2002).

[2] For the terminology used here, see Wald (1984), chapter 8.

[3] We assume that T_B's head always remains at the same point in space, and only its memory tape is in motion. So the time of signal propagation from T_B to T_A is constant even in those cases where the memory tape is unbounded.

[4] Here, and in the other conditions, we are assuming that $f(n)$ is defined. (I) may be too strong formulation, as it states that a memory *token* m supports all single computations. The weaker condition, $\Diamond \exists m \forall n \Diamond C_f(n,m)$, may be preferable, as it signifies that a memory of *type* m supports all single computations.

[5] See Pitowsky (1996) for a detailed exposition of this embedding.

[6] For a survey, see Goldstine (1972), especially chapter 10. In fact, Turing himself was engaged in the mid-1930's in a project for constructing an analog machine for the calculation of the Riemann Zeta-function (Hodges, 140-142).

[7] We are assuming that there are memories of infinite size m. Also note that although the memory used in the computation is infinite, at each point in the computation, the machine uses only a finite segment of it.

[8] In fact, the rejection of HM on the basis of epistemic considerations smacks of circularity. On what grounds do we attribute to standard computers the capacity to compute addition? We have practically no way to verify the addition of very long strings, but assume that the verification is a physical possibility. But it is unclear then how we can reject a physically possible machine on the basis of epistemic considerations, given that these considerations are themselves grounded in physical possibilities.

[9] These modal versions of the two readings are simplified, and cover only the memory issue, whereas the condition proper also addresses the existence of the machine as a whole. In addition, (III) does not fully express the generality of the stronger understanding, which would entail generalizing it into a scheme that ranges over arbitrarily many numbers. And it should be kept in mind that the quantifiers only range over numbers whose values, $f(n)$, are defined .

[10] See for example Earman (1986: 123-126), Pitowsky (1990), Copeland (1996) and Shagrir (1997).

[11] The former misconception is reflected in the following statement: "The Church-Turing thesis makes a bold claim about the theoretical limits to computation" (Cleland, 1993: 283). And consider two striking examples of the latter misconception: "Extensive efforts have been made to prove the Church-Turing thesis, which suggests that all realizable dynamical and physical systems cannot be more powerful than classical models of computation." (Siegelmann, 1995: 545). "But is [CTT] true? Well, it will be if there is at least one physically possible Turing machine and if there are no physically possible non-Turing computers." (Hogarth, 1994: 134). For more examples, see Copeland (1996).

[12] For comprehensive historical surveys, see Shapiro (1983), Gandy (1988), and Sieg (1994).

[13] This formulation is found in Church (1936b: 41). The original formulation, in Hilbert and Ackermann (1928, p. 73), refers to the validity of expressions, not provability. But Gödel's completeness theorem (published in 1930) demonstrates that the formulations are equivalent.

[14] See also Gandy (1988), Sieg (1994), Copeland (1996) and Shagrir (1997).

[15] The reference here is to the first argument in section 9 of Turing's 1936 paper (pp. 135-138).

[16] Gandy (1988) in fact maintains that the argument is an outline of a proof, construing the conclusion as a theorem.

[17] Roger Penrose (1994) notes that his claim that some acts of the conscious brain are not Turing-computable is compaticle with CTT. We argue, in addition, that the processes underlying these acts, even if they are not Turing-computable, can be computations.

[18] A similar thesis is advanced by Wolfarm (1985).

[19] The point is emphasized in Sieg and Byrnes (1999).

[20] Gandy (1980: 145) predicts a counterexample to Principle (IV) in Newotonian mechanics where the the speed of signal propagation may be unbounded. Gandy however insists and assumes that the speed of signal propagation is bounded, apparently assuming Special Relativity. HM, however, is no counter

example to Principle IV because it is consistent with the assumption that the speed of signal propagation is bounded.

References:

Church A. 1936a: 'An Unsolvable Problem of Elementary Number Theory'. *American Journal of Mathematics* 58: 345-363. Reprinted in Davis 1965: 88-107.

Church A. 1936b: 'A Note on the Entscheidungsproblem'. *Journal of Symbolic Logic* 1: 40-41. Reprinted in Davis 1965: 108-115.

Cleland C. 1993: 'Is the Church-Turing Thesis True?' *Minds and Machines* 3: 283-312.

Copeland B.J. 1996: 'The Church-Turing Thesis'. In Perry J. and Zalta E. (eds.): *Stanford Encyclopedia of Philosophy* [http://plato.stanford.edu].

Copeland B.J. 2002: 'Accelerating Turing Machines'. *Minds and Machines* 12: 281-301.

Copeland B.J. and Sylvan R. 1999: 'Beyond the Universal Turing Machine'. *Australasian Journal of Philosophy* 77: 46-67.

Davis M. (ed.) 1965: *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. New York: Raven.

Earman J. 1986: *A Primer of Determinism*. Holland: Reidel.

Earman J. and Norton J.D. 1993: 'Forever is A Day: Supertasks in Pitowsky and Malament-Hogarth Spacetimes'. *Philosophy of Science* 60: 22-42.

Gandy R.O. 1980: 'Church's Thesis and Principles of Mechanisms'. In Barwise J., Keisler J.J. and Kunen K. (eds.): *The Kleene Symposium*. Amsterdam: North-Holland: 123-145.

Gandy R.O. 1988: 'The Confluence of Ideas in 1936'. In Herken 1988: 55-111.

Gödel K. 1931: 'On Formally Undecidable Propositions of Principia Mathematica and Related Systems I'. *Monatshefte für Mathematik und Physik* 38: 173-198. Translated and Reprinted in Davis 1965: 5-38.

Goldstine H.H. 1972: *The Computer from Pascal to Von Neumann*. Priceton: Princeton University Press.

Herken R. (ed.) 1988: *The Universal Turing Machine A Half-Century Survey*. Oxford: Oxford University Press

Hilbert D. and Ackermann W. 1928: *Grundzuge der Theoretischen Logic*. Berlin: Springer-Verlag.

Hodges A. 1983: *Alan Turing: The Enigma*. New York: Touchstone, Simon and Schuster.

Hogarth M.L. 1992: 'Does General Relativity Allow an Observer to View an Eternity in a Finite Time?'. *Foundations of Physics Letters* 5: 173-181.

Hogarth M.L. 1994: Non-Turing Computers and Non-Turing Computability. *Proceedings of the Philosophy of Science Association (PSA)* 1: 126-138.

Kleene S.C. 1936: 'General Recursive Functions of Natural Numbers'. *Mathematische Annalen* 112: 727-742. Reprinted in Davis 1965: 236-253.

Kleene S.C. 1988: 'Turing's Analysis of Computability, and Major Applications of It'. In Herken 1988: 17-54.

Laraudogoitia J.P. 1999: 'Supertasks'. In Perry J. and Zalta E. (eds.): *Stanford Encyclopedia of Philosophy* [http://plato.stanford.edu].

Moore C. 1990: 'Unpredictablity and Undecidability in Dynamical Systems'. *Physical Review Letters* 64: 2354-2357.

Penrose R. 1994: *Shadows of the Mind*. New York and Oxford: Oxford University Press.

Pitowsky I. 1990: 'The Physical Church Thesis and Physical Computational Complexity'. *Iyyun* 39: 81-99.

Pitowsky I. 1996: 'Laplace's Demon Consults an Oracle: The Computational Complexity of Prediction'. *Studies in History and Philosophy of Physics* 27: 161-180.

Post E.L. 1936: 'Finitary Combinatory Processes – Formulation I'. *Journal of Symbolic Logic* 1: 103-105. Reprinted in Davis 1965: 288-291.

Pour-El M.B. and Richards I. (1981): 'The Wave Equation with Computable Initial Data such that its Unique Solution is not Computable'. *Advances in Mathematics* 39: 215-239.

Shagrir O. 1997: 'Two Dogmas of Computationalism'. *Minds and Machines* 7: 321-344.

Shapiro S. 1983: 'Remarks on the Development of Computability'. *History and Philosophy of Logic* 4: 203-220.

Sieg W. 1994: 'Mechanical Procedures and Mathematical Experience'. In George A. (ed.): *Mathematics and Mind*. Oxford: Oxford University Press.

Sieg W. and Byrnes J. 1999: 'An Abstract Model for Parallel Computations: Gandy's Thesis'. *The Monist* 82: 150-164.

Siegelmann H.T. 1995: 'Computation Beyond Turing Limit'. *Science* 268: 545-548.

Turing A.M. 1936: 'On Computable Numbers, with an Application to the Entscheidungsproblem'. *Proceedings of the London Mathematical Society* (2) 42: 230-265. A correction in 43 (1937): 544-546. Reprinted in Davis 1965: 115-154.

Turing A.M. 1939: 'Systems of Logic Based on Ordinals'. *Proceedings of the London Mathematical Society* (2) 45: 161-228. Reprinted in Davis 1965: 154-222

Wald R.M. 1984: *General Relativity*. Chicago: University of Chicago Press.

Wolfarm S. 1985: 'Undecidability and Intractability in Theoretical Physics'. *Physical Review Letters* 54: 735-738.