Toward a Formal Philosophy of Hypercomputation

SELMER BRINGSJORD[1,2] and MICHAEL ZENZEN[2]
[1]*Department of Philosophy, Psychology & Cognitive Science and* [2]*Department of Computer Science, Rensselaer Polytechnic Institute (RPI), Troy, NY 12180, USA; E-mail: selmer@rpi.edu zenzem@rpi.edu*

Abstract. Does what guides a pastry chef stand on par, from the standpoint of contemporary computer science, with what guides a supercomputer? Did Betty Crocker, when telling us how to bake a cake, provide an effective procedure, in the sense of 'effective' used in computer science? According to Cleland, the answer in both cases is "Yes". One consequence of Cleland's affirmative answer is supposed to be that hypercomputation is, to use her phrase, "theoretically viable". Unfortunately, though we applaud Cleland's "gadfly philosophizing" (as, in fact, seminal), we believe that unless such a *modus operandi* is married to formal philosophy, nothing conclusive will be produced (as evidenced by the problems plaguing Cleland's work that we uncover). Herein, we attempt to pull off not the complete marriage for hypercomputation, but perhaps at least the beginning of a courtship that others can subsequently help along.

Key words: algorithms, classical and constructivist mathematics, computationalism, effective procedures, hypercomputation

1. Introduction

First-rate philosophy can be profoundly irritating, especially to non-philosophers. Socrates showed us that, and thereby inaugurated a tradition Cleland, perhaps unwittingly, follows to the letter. She starts with innocent inquiry, pushes and probes, makes some seemingly innocuous inferences ... and boom! — suddenly she has shown that what scientists and engineers take for granted *shouldn't* be taken for granted. Does what guides a pastry chef stand on par, from the standpoint of contemporary computer science, with what guides a supercomputer? Did Betty Crocker, when telling us how to bake a cake, provide an effective procedure, in the sense of 'effective' used in computer science? According to Cleland, the answer in both cases is "Yes". One consequence of Cleland's affirmative answer is supposed to be that hypercomputation is, to use her phrase, "theoretically viable". Unfortunately, though we applaud Cleland's "gadfly philosophizing" (as, in fact, seminal), we believe that unless such a *modus operandi* is married to formal philosophy, nothing conclusive will be produced (as evidenced by problems we uncover). Herein, we attempt to pull off not the complete marriage for hypercomputation, but perhaps at least the beginning of a courtship that others can subsequently help along.

The plan of the paper is as follows. In Section 2, we provide a primer on hypercomputation — without which, from our standpoint, philosophizing about this phenomenon is guaranteed to be unproductive. In Section 3 we introduce a

Minds and Machines **12:** 241–258, 2002.

simple scheme that allows the key issues to be set out in clearer fashion than they are in Cleland's work to this point. In Section 4 we share some of the worries we have about Cleland's position from the perspective of formal philosophy. In Section 5, we present encapsulated arguments for our own positions on one of the — arguably *the* — key issues identified in Section 2. Finally, in Section 6, we discuss some of the points Cleland has made in personal communication about an earlier draft of this paper. It will turn out that there are probably rock-bottom differences between Cleland and us — differences that are rarely brought out in the open when computation and the mind are discussed.

2. A Primer on Hypercomputation

The story (at least the contemporary version) begins with Turing, who in his dissertation (Turing, 1938, 1939) pondered the possibility of so-called *oracle machines*. These machines are architecturally identical to Turing machines, but are assumed to be augmented with an oracle which, upon being consulted about a Turing machine m and input i, returns a correct verdict as to whether m halts on i. Oracle machines are part of the canon of computer science today.[1] For example, here's a quote from a recently updated classic textbook on computability and uncomputability:

> Once one gets used to the fact that there are explicit problems, such as the halting problem, that have no algorithmic solution, one is led to consider questions such as the following: Suppose we were given a "black box" or, as one says, an *oracle*, which can tell us whether a given Turing machine with given input eventually halts. Then it is natural to consider a kind of program that is allowed to ask questions of our oracle and to use the answers in its further computation ... (Davis et al., 1994, p. 197).

How do Davis et al. transform this figurative scheme into a mathematically respectable one? To answer this question, note that instead of Turing machines, Davis et al. use an equivalent programming language $\mathcal{L}$, the programs of which are composed of lists of statements with optional labels. $\mathcal{L}$ allows for three types of statements: adding one to a variable V ($V \leftarrow V + 1$), subtracting one from a variable V ($V \leftarrow V - 1$), and moving by a conditional to a line labeled with L in a program (IF $V \neq 0$ GOTO L). With just these three statements it's possible to write a program that computes every Turing-computable function. Traditionally, to make it easier to see this, "macros" $V \leftarrow V'$ and GOTO L are allowed. The first macro moves the contents of variable V' to variable V; the second is an unconditional branch that moves the active line to the one with label L; both macros can be easily decomposed into a program written with only the three fundamental statements. (Readers new to this material are encouraged to carry out the decomposition.) As an example of an excruciatingly simple program in $\mathcal{L}$, consider a program that

computes the function $f(x_1, x_2) = x_1 + x_2{}^2$:

$Y \leftarrow X_1$
$Z \leftarrow X_2$
$[B]$ IF $Z \neq 0$ GOTO A
GOTO E
$[A]$ $Z \leftarrow Z - 1$
$Y \leftarrow Y + 1$
GOTO B

At this point we're in position to see how Davis et al. formalize oracles. The trick is simply to allow a new statement (an *oracle statement*) of the form

$$V \leftarrow O(V)$$

into the syntax of $\mathcal{L}$: "We now let G be some partial function on **N** [the natural numbers] with values in **N**, and we shall think of G as an oracle" (Davis et al., 1994, p. 198). So if the value of variable V is m before an oracle statement is encountered, when the statement is then reached, the value of V changes to $G(m)$ (assuming that G is defined for this argument). As should be plain, *there is absolutely no sense in which G is computed*. G is just a placeholder for what at this point is nothing less than magic. In connection, specifically, with the halting problem, where $m_1, m_2, \ldots$ enumerates all Turing machines, the function

$$h(m, n) = \begin{cases} 1 & \text{if } m_m \text{ halts with input } n \\ 0 & \text{otherwise} \end{cases}$$

can be "solved" by a program in $\mathcal{L}$ in which an encoding u of m and n is given as an argument to G.

Unfortunately, this leaves the *nature* of oracles completely out of the picture. They are, to say it again, simply magical; all that matters is that they return verdicts that can be used by Turing machines (and their equivalents). The idea so far, logico-mathematically speaking, is to set up a scheme for investigating *relative* computability; and in such a scheme, how oracles do their thing is wholly irrelevant. For Cleland and the two of us, and other philosophers interested in hypercomputation, this situation is unacceptable. What is an oracle? How does it pull off these amazing feats? Fortunately, there are answers: A number of logico-mathematical devices have been specified to explain how an oracle can accomplish its amazing work. In fact, just as there are an infinite number of mathematical devices equivalent to Turing machines (machines running programs from the language $\mathcal{L}$ visited above, Register machines, the λ-calculus, abaci, ...; these are all discussed in the context of an attempt to define computation in Bringsjord (1994)), there are an infinite number of devices beyond the Turing Limit. As you might also guess, a small proper subset of these devices dominate the literature. In fact, three kinds of hypercomputational devices — analog chaotic neural nets, trial-and-error machines, and Zeus machines — are generally featured in the literature. In

the interests of reaching a wider audience, we discuss only the latter two devices here.[3]

Trial-and-error machines have their roots in a paper by Hilary Putnam (1965), and one by Mark Gold (1965); both appeared in the same rather famous volume and issue of the *Journal of Symbolic Logic*. So what *are* trial-and-error machines? Well, they have the architecture of Turing machines (read/write heads, tapes, a fixed and finite number of internal states), but produce output "in the limit" rather than giving one particular output and then halting. Here is a trial-and-error machine $\mathcal{M}$ that solves the halting problem. Take some arbitrary Turing machine m with input u; let $n^{m,u}$ be the Gödel number of the pair m, u; place $n^{m,u}$ on $\mathcal{M}$'s tape. Now have $\mathcal{M}$ print 0 immediately (recall the function h, defined above), and then have it simulate the operation of m on u. If $\mathcal{M}$ halts during the simulation, have it proceed to erase 0 in favor of 1, and then have it stop for good. It's as easy as that.[4]

Zeus machines (or "Weyl Machines" from Weyl (1949); see also Bertrand Russell's (1936) discussion of the possibility of his embodying such devices) are based on the character Zeus, described by Boolos and Jeffrey (1989). Zeus is a superhuman creature who can enumerate **N** *in a finite amount of time*, in one second, in fact. He pulls this off by giving the first entry, 0, in $\frac{1}{2}$ second, the second entry, 1, in $\frac{1}{4}$ second, the third entry in $\frac{1}{8}$ second, the fourth in $\frac{1}{16}$ second, ..., so that, indeed, when a second is done he has completely enumerated the natural numbers. Obviously, it's easy to (formalize and then) adapt this scheme so as to produce a Zeus machine that can solve the halting problem: just imagine a machine which, when simulating an arbitrary Turing machine m operating on input u, does each step faster and faster ... (There are countably many Turing machines, and those that don't halt are trapped in an unending sequence of the same cardinality as **N**.) If, during this simulation, the Zeus machine finds that m halts on u, then a 1 is returned; otherwise 0 is given.

With this primer now digested, we're ready to identify the central issues arising from hypercomputation.

3. Toward Identifying the Central Issues

Cleland tells us that "Effective Procedures and Causal Processes" is "a defense of the theoretical viability of the concept of hypercomputation" (p. 1). She tells us in the next sentence that the "received view" is that hypercomputation is not possible. But what does this mean? What is Cleland's objective, exactly? One (unlikely) possibility is that she means to show that hypercomputation is *logically* possible. Another is that she intends to show that hypercomputation is *physically* possible. Or perhaps she means to maintain that hypercomputation is somehow "humanly possible." Yet another option, seemingly consistent with her opening prose, is to understand her main thesis to be that hypercomputation is logically physically possible. There are in fact any number of defensible construals that could be laid on the table, and Cleland's writings provide insufficient clues as to which to pick.[5]

3.1. NEEDED OPERATORS AND PREDICATES

In order to clarify the situation it's necessary to have on hand at least the bulk of the relevant operators and predicates. Accordingly, please see Table I. The leftmost column in that table lists the three main modal operators tacitly invoked by Cleland's discussion of hypercomputation. The column "Computation Predicates" is based on the fact that computational information processing can be divided into three categories: processing that can be carried out by machines having less power than Turing machines (designated by the predicate F, which is intended to suggest finite state automata); processing that can be carried out by machines with the power of Turing machines and their equivalents (T); and, finally, processing that can be carried out by machines more powerful than Turing machines (O), for example, trial-and-error machines.

Table I. Relevant Operators and Predicates

Operators	**Computation predicates**	**Engineeringish predicates**	**Partitioned domain**
$\Diamond$: logically possibly	F: FSA computation	$\mathcal{H}^u$: unconsciously harnessable	$m_1, m_2, \ldots$: mentations
$\Diamond_p$: physically possibly	T: Turing machine computation	$\mathcal{H}^c$: consciously harnessable	$c_1, c_2, \ldots$: computations
$\Diamond_h$: humanly possibly	O: "Oracle" computation	$\mathcal{A}$: actualizable	$p_1, p_2, \ldots$: persons

The rightmost column, "Partitioned Domain," simply presents the core of a sorted calculus allowing us to conveniently refer to mentations, computations, and persons. Alternatively, we could introduce predicates to range over these sub-domains.

The column "Engineeringish Predicates" is a bit more tricky, and is at the heart of things. Consider a finite state automaton F_1 designed to process strings over the alphabet $A = \{a_1, a_2\}$; specifically, F_1 is designed to accept the language L_1 composed of those strings having three consecutive a_1's. Let c denote a computation carried out by F_1. Obviously, we could build a physical device to incarnate F_1; this device could, say, be a toy railroad system whose track is divided into squares upon which either of the characters a_1 or a_2 can be written before F_1 begins to operate. We thus say that F_1 is *actualizable*; abbreviated: $\mathcal{A}F_1$. What about the two other predicate letters in the column in question? Well, suppose that we have actualized F_1, and suppose as well that we would like to investigate whether $a_1a_1a_2a_2a_1a_1a_1 \in L_1$. We say that F_1 can be *harnessed* because we can use the automaton to divine the answer (which is of course that the string is a member of L_1). More specifically, we say in this case that F_1 can be *consciously* harnessed, because a human can formulate a plan that he or she deliberately follows in order to secure the answer. On the other hand, sometimes the solution to a problem just "pops into one's head" through a process that isn't accessible to conscious thought. In this case we say that the process is *unconsciously* harnessable. Overall, what we verified in the case of F_1, where $O_1/O_2/\ldots/O_n$ means that any one O_i of the operators can be selected, is that

(1) $\Diamond_h/\Diamond_p/\Diamond\exists c(Fc \wedge \mathcal{A}c \wedge \mathcal{H}^c c)$.

3.2. WHAT ABOUT CLELAND'S OPERATOR?

In personal communication, Cleland has informed us that we are right that something like "logically physically possible" is her target. But only something *like* this operator: Cleland says she has in mind a different operator:

> My real target is causal possibility, which is to be distinguished from physical possibility (which is traditionally construed in terms of the actual physical laws of our world). A world in which objects (with real valued mass) travel faster than light is physically impossible (assuming Einstein is right) but nonetheless causally possible (since Newton might have been right and his laws represent bona fide causal laws too). Thus, although the idea of "logically physically possible" is close, it is not quite right since some physical laws (functional laws) are not causal laws. (Cleland, personal communication, May 28, 2001)

This leaves us quite puzzled. The examples Cleland gives here are perfectly at home under "logically physically possible". Let w_α be the actual world. That which is logically possible at w_α is that which is true at worlds accessible from w_α; that is, $\Diamond A$ at w_α iff A is true at some w such that $w_\alpha R w$. If we restrict the accessibility relation in some way based on the laws of nature at w_α, the standard route, then $\Diamond_p A$ at w_α iff A is true at some w such that $w_\alpha R' w$. To say

$$\Diamond\Diamond_p A',$$

where A' is some "Newtonian proposition", then seems to capture Cleland's Einstein/Newton example very nicely. Intuitively put, a Newtonian proposition is true at some world physically accessible from some world logically accessible from w_α. We simply don't understand in the least Cleland's distinction between physical, functional, and causal laws. The distinction doesn't seem to relate to the example she gives, which, as we point out, is easily handled by standard logics. One would think that the notion of a causal law that is not a physical law would require that her operator "causally possible" not be reducible as it apparently is, given her example. The bottom line is that we doubt very much that the standard formalisms for logical and physical possibility can't be easily used to capture what Cleland has in mind. And if we're wrong, then, with all due respect, we doubt very much that what Cleland has in mind is clear and rigorous enough to worry about (see the final section of this paper).

3.3. THE CENTRAL ISSUES

There are myriad propositions of great interest that can be expressed on the basis of Table 1, all of which will need to be investigated in a truly mature formal philosophy of hypercomputation. From our standpoint, what is interesting, and

ultimately profitable, is to reflect upon patterns created by the various permutations of the machinery encapsulated in Table 1, and to then head to full-blown logics to capture these patterns, and their descendants. These new logics would of course inherit patterns from established intensional logics. For example, it's because we know that $\Diamond_h\phi \rightarrow \Diamond_p\phi$ and $\Diamond_p\phi \rightarrow \Diamond\phi$ that we can assert (1). These new logics would also inherit much from physics, which is revealed by Section 5.2. Here's a small fragment of the theses that would presumably be investigated in a mature philosophy of hypercomputation:

(2) $\Diamond/\Diamond_p\exists c Oc$

(3) $\Diamond/\Diamond_p/\Diamond_h\exists c(\mathcal{A}c \wedge Oc)$

(4) $\Diamond/\Diamond_p/\Diamond_h\exists c(\mathcal{H}^c/\mathcal{H}^u c \wedge Oc)$

(5) $\forall c(\mathcal{A}c \rightarrow \mathcal{H}^c/\mathcal{H}^u c)$

There are many other interesting propositions, but perhaps the most important question is whether it's humanly possible to build a consciously harnessable hypercomputer, that is, whether this proposition is true:

(6) $\Diamond_h/\Diamond_p/\Diamond\exists c(Oc \wedge \mathcal{A}c \wedge \mathcal{H}^c c)$.

3.4. "STRONG" AI, "WEAK" AI, AND THE SUPERMIND DOCTRINE

The operators and predicates we've allowed ourselves also allow reference to mentations and persons, and this allows propositions that serve to encapsulate "Weak" AI, "Strong" AI (= computationalism), and the supermind view we subscribe to, and explain and defend in our forthcoming book *Superminds*. "Strong" AI can be identified with the proposition that

(SAI) $\forall m\exists c(Tc \wedge m = c)$.

"Weak" AI can be be identified with the proposition that

(WAI) $\forall m\exists c(Tc \wedge m \approx c)$,

where '$\approx$' stands for the relation of *simulation*. Roughly put, the idea is that all mentations can be perfectly simulated by Turing computation.[6]

Finally, as to our supermind view, it can be encapsulated by the following four propositions. The basic idea behind these propositions, as indicated by Figure 1 (wherein the circle represents superminds) is that human persons comprise at least three "parts:" one part that engages in information processing at and below the Turing Limit, one part that engages in such processing above the Turing Limit, and one part that cannot be expressed in any third-person scheme whatsoever. This last part includes such things as subjective awareness and qualia.

(SUPER1) $\exists m\exists c(m = c \wedge Tc)$

(SUPER2) $\exists m\exists c(m = c \wedge Fc)$

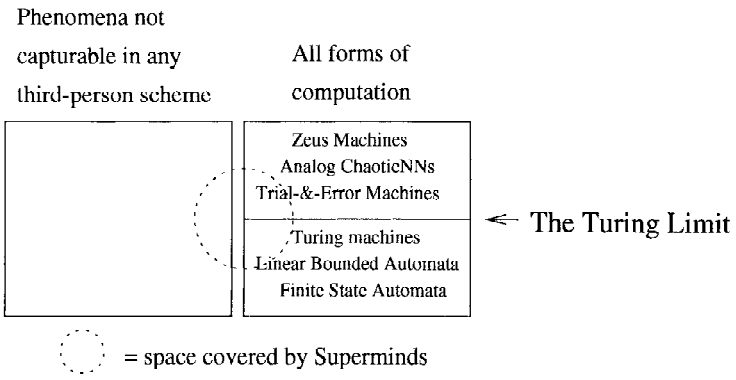

Figure 1. Superminds include parts of three spaces

(SUPER3) $\exists m \exists c(m = c \wedge Oc)$

(SUPER4) $\exists m \neg \exists c m = c$

The kernel of the supermind view is based upon well-known material. Computation at the level of Turing machines and below is in large part set out in every comprehensive textbook on computability theory (e.g., Lewis and Papadimitriou, 1981). The supermind view says that human persons can perform feats at this level. Information processing *above* the Turing Limit, as we've discussed, is also well-understood mathematically, and the supermind view includes the proposition that human persons can perform feats at this level. Finally, there are many well-known arguments for the position that human persons do things that can't be described in any symbolic scheme whatsoever (see e.g., Bringsjord, 1992; Searle, 1992; Jacquette, 1994).

4. Some Problems With Cleland's Schemes

4.1. On Cleland's account, the halting problem appears effectively computable

Cleland (1995) defines a procedure to be effective if and only two conditions hold:

(i) it meets Minsky's condition [that next step is determined by the present step] and

(ii) each of the action-kinds [the procedure] specifies invariably has (under normal conditions) a certain kind of consequence (Cleland, 1995, p. 13).

This definition can't be right, as is easy to see. For suppose that this definition is correct, and consider the trial-and-error machine $\mathcal{M}$ described in the primer we gave earlier. Obviously, $\mathcal{M}$ satisfies the first condition, because each step is determined by a prior step in in exactly the same manner that a step in an ordinary Turing machine is predetermined. Condition (ii) is also clearly satisfied by $\mathcal{M}$, because each of the action-kinds here has a consequence in a way exactly similar to that seen in ordinary Turing machines. Since (i) and (ii) are satisfied, it follows

that $\mathcal{M}$ is an effective procedure for solving the halting problem. But if we know anything rigorous about computability and uncomputability we know that there is no effective procedure for solving the halting problem. By *reductio*, then, Cleland's account must be wrong.[7] (We deal with Cleland's objections to the argument just given in Section 6.2.1.)

4.2. QUOTIDIAN PROCEDURES *are* HOPELESSLY IMPRECISE

By our lights, recipes are laughably vague, and don't deserve to taken seriously from the standpoint of formal philosophy, logic, or computer science. Cleland, in response to this view, tells us that

> [Recipes] consist of instructions. Each instruction-expression (e.g., "pat the mozzarella balls dry with absorbent kitchen paper") makes reference to an occurrence which is to be brought about (done) as opposed to undergone or merely happen. That is, each instruction designates an action, more specifically, since different chefs may apply the same instruction, an action-type (vs. token). The instructions are expressed by imperatives (e.g., "cut into slices . . . "), indicating that the follower is to perform the designated action-types. (Cleland 2001, p. 221).

But it's hard to take her seriously here. The account she gives is induced from just one recipe (for mozzarella balls). There are many recipes that don't "make reference to an occurrence which is to be brought about (done) as opposed to undergone or merely happen." For example, here is a recipe one of us follows for easy-to-make waffles; it has no "instructions expressed by imperatives".

Selmer's Brainless Waffles

- Make sure that $\frac{5}{2}$ cups of white flour along with $\frac{1}{2}$ cup of whole wheat flour are in a large bowl, along with $\frac{1}{4}$ teaspoon of salt, 4 teaspoons of baking powder, and 2 teaspoons of sugar — and that these ingredients are mixed together.
- Egg yolks are not desired; 3 whites are needed. Along with 3 cups of fat-free Lactaid 100© milk, these whites should be mixed by fork or whisk (etc.) into the dry ingredients.
- The iron (preferably one with the capacity to sound a signal when pre-set temperature settings are reached) works best when sprayed with non-stick oil, and when it's heated to the desired temperature your batter is ready to be received.

On Cleland's account, this recipe isn't a recipe, but it *is* a recipe; hence her account is defective. If we use our imagination, we can see that recipes like this can be followed without *doing* much. For example, suppose that when setting out to make some of Selmer's brainless waffles in my kitchen, you find that a container of white flour, suspended above one of my counters, is leaking. And suppose that, as luck would have it, the falling flour is being caught by a measuring cup. We're sure you can continue the story in such a way that some brainless waffles are serendipitously produced. When you serve them to someone who comes down for

breakfast, they might say that the waffles are delicious (esp. given than they are fat- and cholesterol-free), and you might reply that "Well, in following the recipe I was a bit lucky this morning".

5. Arguments Against the Conscious Harnessability of Hypercomputation

Due to space constraints, in this section we can only provide a prolegomenon to a fully developed case against the view that a hypercomputational device can be built and consciously harnessed. But we believe that this prolegomenon suffices to point the way to future research and development.[8]

5.1. THE ARGUMENT FROM INFINITY

Pessimism about physically realizing and harnessing hypercomputational devices is based, first, upon the observation that to build a hypercomputational device would be to somehow harness and compress the power of the infinite in a finitely bounded, physical artifact. Hopefully our little primer on hypercomputation makes this plain. If you recall this primer, the problem seems obvious in the case of trial-and-error and Zeus machines. In the former, to build and use them we would need to be either able to see the future (in order to see the in-the-limit verdict) or to compress an infinite period of symbol manipulation into a finite interval. In the latter, once again, to build and use means to achieve such preternatural compression. In the case of analog chaotic neural nets the same problem is there, but camouflaged. An analog chaotic neural net processes information by using irrational numbers (Siegelmann and Sontag, 1994). So think of the challenge in these terms: Suppose that you can build a hypercomputational device as long as whenever your device reaches ordinary computational state s, it carries out operation O on s *and* irrational real $r \in [0, 1]$. The operation here has in fact been neatly formalized in the form of the analog shift map (see Note 3). How do you capture and manipulate r *physically*? There is of course a sense in which every time you operate a physical artifact by in part using readings from an analog dial (the speedometer on a car, e.g.), you are "using irrational numbers." But this is a *very* weak sense of 'use.' Your Volvo may get you to work in the morning, and there may be a sense in which it or parts of it enter into into physical processes that are uncomputable (which is what people like Pour-El and Richards (1981a, b) can be read as having probably established). But to use the Volvo to attack the halting problem seems rather futile. And it's hard to imagine an antidote to this futility from *any engineer*, for *any device*.

5.2. THREE ARGUMENTS FROM PHYSICS

There are *many* powerful arguments from physics for the view that it's humanly impossible to build an artifact which engages in hypercomputation that is consciously harnessed. We give only three here, in the interests of space, and in each case we

provide only an encapsulated version. A mature, formal philosophy of hypercomputation would need to come to grips with the details in all such arguments (and hence a fully formal philosophy of hypercomputation will include some rigorous physics and philosophy of physics).

5.2.1. *The Argument From Digital Physics*

Long before Cleland's notion of a physics beyond ordinary (= Turing machine) computation, Feynman (1982) envisioned physics as an exclusively Turing-computational enterprise fueled by advancing computer technology (that enables sophisticated modeling and simulation) — yet Cleland nowhere considers (let alone casts doubt upon) the strong program of "digital physics" that Feynman inaugurated. And it isn't just Feynman's program Cleland needs to confront: it's this program *and* a pragmatic, positivistically inspired rejection of the real numbers and the continuum as relevant to physics (Ford, 1983).[9]

5.2.2. *The Argument From the Absence of Candidates*

Cleland doesn't give us any candidates for physical processes that instantiate hypercomputation. We are given only the remarks:

> Some physical processes may only have a few points at which causal intervention is physically possible, thus allowing one to specify an input and output relation but not permitting a *mechanistic account* of what goes on between input and output. An oracle would be such a process. It is important to keep in mind, however, that being *non-mechanistic* is not sufficient for being an oracle. (Cleland, 2000, p. 18; emphasis ours).

Given what she says here, what would be a likely candidate? It seems that being "non-mechanistic" is a necessary (but not sufficient) condition. Given this, and given the impressionistic description of the the type of physical process that could function as an oracle, one would think that quantum mechanical phenomena might work. Indeed, these phenomena are quintessentially those where what goes on between input and output is unavailable (or perhaps it's better to say that causal intervention yields an output). If one has hopes of finding some sort of physical process to function as an oracle, this seems to be a promising place to look. At least given today's science, one can hardly get more exotic.

But attempts to access and artifactually exploit these promising phenomena have not taken us into hypercomputing. Theorists and experimentalist working on quantum computers have long recognized that this type of machine will only give us *faster* Turing computation.[10] Now one could, of course, say that we haven't exploited all the possibilities. This is true — *if* one assumes that other candidates will arrive on the scene in the future. But the issue is plausibility — *today*. Since quantum computing has proved to be a dead end, where else should we look? If Cleland can't present us with any other promising candidates, the most reasonable position would seem to be the denial of $\Diamond_p \exists c O c$ and related propositions.

The situation is actually worse for Cleland than it appears to be. In order to see this, let's grant that physically instantiated oracles exist. We have seen that at present we have no candidates for such things. So, we need to set off in search of candidates. But what should our search procedure be? It's of course vastly improbable that arbitrary searching will prove successful, so we need help from our best (current and relevant) theories. But our best theories offer no guidance for where to look for "causal openings".

5.2.3. *The Argument From Turing Machines and Oracles as Strange Bedfellows*

As we saw earlier, Turing's original speculation involves the *concept* of a Turing machine and the *concept*, vague though it may be, or an oracle. Now for Turing the oracle is a black box, but it's supposed to accept digital input and yield digital output (as is reflected in the primer we offered earlier: recall G). But what reasons do we have to believe that such concepts can be instantiated as *physical processes*? For all we know, even if physically instantiated oracles exist, it may be that when harnessed and made to function in the service of a Turing machine, they will simply yield to the Turing machine and march according to its digital tune somewhat like a free spirited dancer who finds herself in the midst of a marching platoon. Indeed, when one tries to apply the oracle description to quantum phenomena, it's evident why they are unable to serve as oracles for us: The very properties that we wish to exploit (properties that devolve from the superposition principle and the capacity to sustain "entangled states") slip through our hands as soon as we try to exploit them. It's as if the delicate properties of quanta cannot survive the complexity and level of interaction of our quotidian world. So, the question for Cleland is: What conceivable *causal* link could there be between a Turing machine and oracle(s) that, once linked to the Turing machine, would yield hypercomputation?[11] What can it mean to hybridize some physical instantiation of a Turing and a (minimally) non-mechanistic process?

6. Dialectic — To a Point

As you know by now, having having read to this point, Cleland has kindly provided objections to an earlier draft of this paper; some of these objections have been rebutted above. But a number of more serious objections remain; we treat them in this section. As will soon be seen, this treatment quickly reveals that there are probably "rock-bottom" disagreements between us and Cleland — disagreements so fundamental that it's rather hard to see how any further substantive dialectic is possible. This paralysis isn't a bad thing; not at all. On the contrary, it's a *good* thing, because perhaps it's about time philosophy faces up to the brute fact that there *are* rock-bottom disputes lurking at the bottom of philosophy of mind and computation.

6.1. MORE ON BRAINLESS WAFFLES

About Selmer's Brainless Waffles Cleland writes:

> I must confess I found the discussion in this section to be a gross distortion of my account. First, no serious recipe book would express a recipe in the way that you have expressed "Selmer's Brainless Waffles." Second, it is clear that the expressions that you use (e.g., "make sure") are non-standard ways of communicating instructions — any competent speaker of the English language would interpret them this way. Third, despite what you say, your recipe does designate physical consequences of action, e.g., that a bowl is to end up (after activity) containing a certain amount of white flour, whole wheat flour (etc.). [Fourth] As for your claim that the recipe could be "followed" without doing anything — produced by serendipity — this trades on an ambiguity between interpreting your strangely worded "instructions" as bona fide instructions and interpreting them merely as descriptions of what actually happens. (Cleland, personal communication, May 28, 2001)

Unfortunately, this won't do at all. On Cleland's first point: So what? Nothing follows from the fact that no serious (hmm, *are* there *serious* recipe books?) recipe book contains recipes like SBW. No serious book on computer programming contains specifications of Turing machines for carrying out anything substantive[12] — but such specifications clearly count as computer programs. Regarding the second point: Again, so what? The recipe in question clearly lacks "instructions expressed as imperatives". Competent speakers of English will doubtless have all *sorts* of thoughts about what is going on here — especially if they come upon the container of leaking white flour that we describe. The third point: Again, alas, so what? Our claim is that recipes, compared with what we find in books on formal relative computability,[13] are intolerably vague, mired as they are (as we show) in the mud and fog of natural language. Cleland's fourth point, unlike the preceeding trio, is directly relevant. Sure, we *are* trading on an ambiguity; that's part of the point! Who really knows what an instruction is? As we all know, distinguishing between actions and mere happenings is notoriously difficult. We are not alone among philosophers in holding that in order to make this distinction one must rely, at least in part, on the concept of intentionality, which is bogged down in its own age-old mud and fog. Given this, it seems positively irrational to turn to recipes in order to make sense of the operation of machines. After all, can a machine have intentionality? Who knows? This question alone has given rise to fierce debates that have produced not a shred of consensus. To turn away from the mathematical accounts of information processing that underlie computer science (and therefore cognitive science and AI) in favor of the documented disagreement and confusion swirling around the notion of a recipe does not seem to us particularly wise.

6.2. ROCK-BOTTOM DIFFERENCES?

6.2.1. *Classical Versus Constructivist Mathematics*

Recall that we argued earlier via a particular type of hypercomputer (trial-and-error machines) that Cleland's definition of 'effective procedure' implies that there is an effective procedure for solving the halting problem, which is absurd. Cleland replies:

> Your characterization of a trial & error machine as satisfying the second condition of my definition trades on an ambiguity, as is revealed by your comment that each of its action-kinds has a consequence "in a way exactly similar . . . ". . . . In order to solve the halting problem, a t & e machine has to perform an utterly mysterious action, namely, take the limit of an infinite number of "guesses" and get the correct answer without actually making an infinite number of guesses. . . . But such "actions" are not "exactly similar" to what a Turing machine (let alone, a concrete machine or person) does in any significant way? How does one interpret them constructively? (Cleland, personal communication, May 28, 2001)

There are two objections here. The first is that the actions taken by trial-and-error machines don't really match those taken by ordinary Turing machines. The second objection, which moves the three of us toward a rock-bottom clash, is that trial-and-error machines cannot be interpreted on the basis of constructivist mathematics, and so are inadmissible. There are fatal problems infecting both of these objections.

The problem with the first objection is that when we say that the actions of a Turing machine are "exactly similar" to the operations of a trial-and-error machine, we are referring not to some sort of vague, global actions of the sort Cleland seems to have in mind, but to the *individual, primitive* actions of such a machine. At each step, a trial-and-error machine is carrying out actions that are in fact not only exactly similar, but — and we should have used this stronger language in the original — *exactly the same* as a Turing machine. This is even clearer in the case of the aforementioned hypercomputing Zeus machines, which Copeland (1998) has aptly called "accelerated Turing machines". At each step, a Zeus machine (based on the quadruple formalism for Turing machines) will scan a square, take a primitive action, and enter a new state. Primitive actions include all and only: moving left or right one square, and writing some symbol on the square that is being scanned (after erasing the symbol that may already be there). These are exactly the possible actions a (quadruple) Turing machine can take at each step. The only difference is that a Zeus machine can perform infinitely many such actions in a finite amount of time, by (as we explained above) taking less and less time for each action in a computation. The math needed to fully formalize such so-called "supertasks" is trivial.

This math, however, is classical, and Cleland, in her second objection, says that bona fide computation needs to be able to be understood constructively. (This is a point that Cleland made repeatedly in personal communication.) Buy why? Why

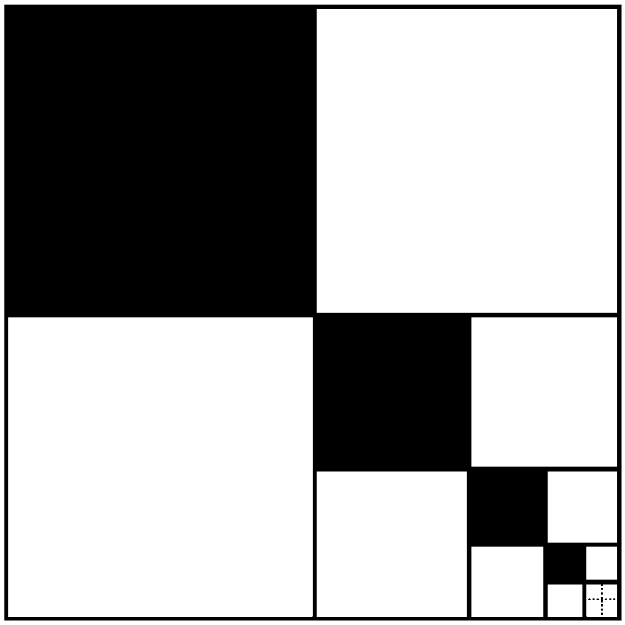

Figure 2. Picture of supertask from seventh grade math.

should one presuppose constructivist mathematics when looking at relative computability? From the standpoint of classical mathematics, there are an infinite number of theoretically viable, logically physically possible (and perhaps physically possible, *simpliciter*) hypercomputers. What does it matter that an idiosyncratic take on mathematics implies that the theoretical viability of hypercomputation is not settled?

The notion of a limit, central to elementary calculus, presupposes the coherence of supertasks by the lights of Salmon (1975) and others. Even children are frequently taught that supertasks are perfectly coherent, because they are prepared early on, in mathematics, for calculus down the road. (Perhaps the easiest way to see that constructivism is violently idiosyncratic is to ponder what math education would be like if students learned math in accordance with it. (Schechter, 2001) For example, see Figure 2, which is taken from p. 268 of Eicholz et al. (1995). Bringsjord's son, Alexander, in the sixth grade, was asked to determine the "percent pattern" of the outer square consumed by the ever-decreasing shaded squares. The pattern, obviously, starts at $\frac{1}{4}$, and then continues as $\frac{1}{16}, \frac{1}{64}, \frac{1}{256}, \ldots$. When asked what percent "in the limit" the shaded square consumes of the original square, Alexander was expected to say "Zero" — but the notion of a limit was a bit tricky for him (perhaps understandably). When asked what percentage the shaded square would "get down to" if someone could work faster and faster, and smaller and smaller, at drawing the up-down and left-right lines that make each quartet of smaller squares, Alexander said zero. It would be interesting to systematically poll students about these matters.

Cleland will doubtless stick to her guns and maintain a constructivist stance. Perhaps she would welcome a sea change in math education in order to clear the way for a thoroughgoing constructivism that would banish now-standard hypercomputers. Of course, since we have no plans to reject classical mathematics, further dialectic on this issue would seem to be otiose. At bottom, the clash may revolve around radically different conceptions of time. Whereas one of us (Bringsjord) routinely teaches supertasks as an innocent, obviously coherent, viable part of relative computability, it may be that Cleland, in keeping with constructivism, thinks we humans are immersed in time, that our thought processes are necessarily temporal, and that we can have no meaningful, coherent, non-temporal experience. On this view, the very infinitude of the natural numbers is understood in terms of pure temporal structure and any supertask, because it involves the actual infinite, cannot be coherent. We leave it to the reader to decide if this view is correct.[14]

Acknowledgements

We're greatly indebted to Carol Cleland, not only for the seminal publications we discuss herein, but also for her gracious e-debate on an earlier draft of our paper. Thanks are also do to some of our colleagues in the Rensselaer Reasoning Group (Yingrui Yang, Jim Fahey, Frank Lee, Bram van Heuveln), and to a number of clever students.

Notes

[1]We therefore find it exceedingly peculiar that at the outset of her "Effective Procedures and Causal Processes" Cleland tells us that the "received view" is that hypercomputation is not possible. Hypercomputation, as shown momentarily, is agreed to be, in many senses, "theoretically viable."

[2]Note that a conditional or unconditional branch that directs flow to a label not present in the program causes halting. In the program here, then, the label E can be read as "exit".

[3]Analog chaotic neural nets are characterized by Siegelmann and Sontag (1994). For cognoscenti, analog chaotic neural nets are allowed to have irrational numbers for coefficients. For the uninitiated, analog chaotic neural nets are perhaps best explained by the "analog shift map," explained in Siegelmann (1995), and summarized in Bringsjord (1998). Analog Turing machines with hypercomputational power are presented in Bringsjord (2001).

[4]For full exposition, along with arguments that human persons are trial-and-error machines, see Kugel (1986), a seminal paper that situates trial-and-error machines nicely within both the formal context of the Arithmetic Hierarchy and the philosophical context of whether minds are computing machines.

[5]But see Section 3.2.

[6]In personal communication, Cleland asks what "perfect simulation" amounts to. Ostensive definitions have been provided elsewhere by one of us. For example, Bringsford and Ferrucci (2000) provides a robust example of a Turing machine-level simulation of the mentation involved in producing belletristic fiction. Simulation in our sense piggybacks on the sense of simulation firmly in use in computability theory; see, e.g., Lewis and Papadimitriou (1981).

[7]In connection with this problem, it would no doubt be helpful for Cleland to carefully take account of the felt need, on the part of logicians and computer scientists, to make clear that *printing* is far from non-trivial when one is trying to be precise about information processing. For example, Ebbinghaus

et al. (1984), in their formal account of Register machines, insist on a PRINT command appearing exactly once in every Register machine program. Also, careful study of the importance of printing in such mathematical contexts is discussed by Kugel (1986). It is generally agreed that Minsky, in the old book that Cleland cites, had a rather immature grasp of some of these issues, and Minsky himself says that an entire chapter in his book isn't to be taken seriously as straight logic or mathematics.

[8]It's important to note this this prolegomenon isn't intended to be inconsistent with Cleland's views. In general, Cleland doesn't see the issue of conscious harnessability to be *the* issue. For her the crucial issue is whether hypercomputation is "causally possible". For reasons explained herein, her issue seems to us to be easily settled: Hypercomputation is obviously causally possible, in no small part because it's logically physically possible. What we are concerned with is whether or not human persons can harness hypercomputation. At least one of us (Bringsjord) believes he has established that human persons do harness hypercomputation (e.g., see the argument against Church's Thesis in Bringsjord and Ferrucci (2000), but these arguments do *not* imply that human persons *consciously* harness hypercomputation.

[9]For a good review of physics and computation see Feltovich et al. (1983). The hypothesis that there will be found a single cellular automaton rule that exactly models all of microscopic physics is explored in Fredkin (1990).

[10]Quantum computers are discussed in Feynman (1986). Brooks (1999) give a concise discussion of quantum computation and the status of current research. Deutsch (1985) considers a universal quantum computer and argues that it is compatible with the Church-Turing Principle. He shows that this computer wouldn't be able to compute non-recursive functions.

[11]Three current models for QMS are spontaneous localization, pilot-waves, and consistent histories. Physicists/philosophers agree that the principle of superposition is the basic construction rule for the formalism of QMS; they also agree that this entails "entanglement" and that this entanglement has been empirically corroborated.

Those concerned with ontology can't agree on what counts as primitive in the theory. Indeed, some have argued that QMS refutes a particularist ontology and hence as well the very basis of *any* "causal chain" accounts (See e.g., Teller, 1989).

[12]E.g., a specification of a TM that multiplies is rather complex (see e.g., Boolos and Jeffrey, 1989), and basic arithmetic is far from what we're thinking about when we use the term 'substantive.'

[13]Relative computability subsumes computability and *un*computability theory.

[14]It may be that there is another rock-bottom disagreement between us and Cleland: namely, one over the virtues of formal versus informal philosophy and science. We are deeply distrustful of accounts of computation and causation which reject what we see as hard-won progress in computer science and physics toward richer and richer formalization. Cleland, on the other hand, seems deeply distrustful of formal accounts of computation and causation.

References

Boolos, G.S. and Jeffrey, R. C. (1989), *Computability and Logic*, Cambridge: Cambridge University Press.

Bringsjord, S. (1992), *What Robots Can and Can't Be*, Dordrecht: Kluwer Academic Publishers.

Bringsjord, S. (1994), 'Computation, Among Other Things, Is Beneath Us', *Minds and Machines* 4, pp. 469–488.

Bringsjord, S. (1998), 'Philosophy and 'Super' Computation', in J. Moor and T. Bynam, eds., *The Digital Phoenix: How Computers are Changing Philosopy*, Oxford: Blackwell, pp. 231–252.

Bringsjord, S. (2001), 'In Computation, Parallel Is Nothing, Physical Everything', *Minds and Machines* 11, pp. 95–99.

Bringsjord, S. and Ferrucci, D. (2000), *Artificial Intelligence and Literary Creativity*: *Inside the Mind of Brutus, a Storytelling Machine*, Mahwah, NJ: Lawrence Erlbaum.

Brooks, M. (1999), *Quantum Computing and Communications*, Berlin: Springer.

Cleland, C. (1995), 'Effective procedures and computable functions', *Minds and Machines* 5, pp. 9–23.

Cleland, C. (2000), 'Effective Procedures and Casual Processes', *Hypercomputation Workshop*, London, England, May 24, 2000.

Cleland, C (2001), 'Recipes, Algorithms, and Programs', *Minds and Machines* 11, pp. 219–237.

Copeland, B. J. (1998), 'Even Turing Machines Can Compute Uncomputable Functions', in J. Casti, ed., *Unconventional Models of Computation*, London: Springer, pp. 150–164.

Davis, M., Sigal, R. and Weyuker, E. (1994), *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, New York, NY: Academic Press.

Deutsch, D. (1985), 'Quantum Theory, The Church-Turing Principle, and The Universal Quantum Computer', *Proceedings of the Royal Society of London, Series A* 400, pp. 87–117.

Ebbinghaus, H. D., Flum, J. and Thomas, W. (1984), *Mathematical Logic*, New York, NY: Springer.

Eicholz, R. E., O'Daffer, P. G., Charles, R. I., Young, S. I., Barnett, C. S., Clemens, S. R., Gilmer, G. F., Reeves, A., Renfro, F. L., Thompson, M. M. and Thorntoon, C. A. (1995), *Ýit Grade 7 Addison-Wesley Mathematics*, Reading, MA: Addison-Wesley.

Feltovich, P., Ford, K and Hayes, P., eds (1983), *Proceedings of a Conference on Physics and Computation, International Journal of Theoretical Physics 21(3/4), 21(6/7), 21(12)*.

Feynman, R. (1986), 'Quantum Mechanical Computers', *Foundations of Physics* 16, pp. 507–531.

Feynman, R. P. (1982), 'Simulating Physics With Computers', *International Journal of Theoretical Physics* 21, pp. 467–488.

Ford, J. (1983), 'How Random Is a Coin Toss?', *Physics Today*, pp. 40–47.

Fredkin, E. (1990), 'Digital Mechanics', *Physica D* 45, pp. 20–32.

Gold, M. (1965), 'Limiting Recursion', *Journal of Symbolic Logic* 33(1), pp. 28–47.

Jacquette, D. (1994), *Philosophy of Mind*, Englewood Cliffs, NJ: Prentice-Hall.

Kugel, P. (1986), 'Thinking May Be More Than Computing', *Cognition* 18, pp. 128–149.

Lewis, H. and Papadimitriou, C. (1981), *Elements of the Theory of Computatution*, Englewood Cliffs, NJ: Prentice-Hall.

Pour-El, M. and Richards, I. (1981a), 'A Computable Ordinary Differential Equation Which Possesses No Computable Solution', *Annals of Mathematical Logic* 17, pp. 61–90.

Pour-El, M. and Richards, I. (1981b), 'The Wave Equation Computable Initial Data Such That Its Unique Solution Is Not Computable', *Advances in Mathematics* 39, pp. 215–239.

Putnam, H. (1965), 'Trial and Error Predicates and a Solution To a Problem of Mostowski', *Journal of Symbolic Logic* 30(1), pp. 49–57.

Russell, B. (1936), 'The Limits of Empiricism', *Proceedings of the Aristotelian Society* 36, pp. 131–150.

Salmon, W. C. (1975), *Space, Time and Motion: A Philosophical Introduction*, Encino, CA: Dickenson.

Schechter, E. (2001), 'Constructivism Is Difficult', *Ýit the Mathematical Association of America Monthly* 108, pp. 50–54.

Searle, J. (1992), *The Rediscovery of the Mind*, Cambridge, MA: MIT Press.

Siegelmann, H. (1995), 'Computation Beyond the Turning Limit', *Science* 268, pp. 545–548.

Siegelmann, H and Sontag, E. (1994), 'Analog Computation Via Neural Nets', *Theoretical Computer Science* 131, pp. 331–360.

Teller, P. (1989), Relativity, Relation Holism, and The Bell Inequalities, in J. Cushing and E. McMullin, eds., 'Philosophical Consequences of Quantum Theory', Notre Dame, IN: University of Notre Dame Press.

Turing, A. (1938), *Dissertation for the PhD: 'Systems of Logic Based on Ordinals*', Princeton, NJ: Princeton University.

Turing, A. (1939), 'Systems of Logic Based on Ordinals', *Proceedings of the London Mathematical Society (Series 2)* 45, pp. 161–228.

Weyl, H. (1949), *Philosophy of Mathematics and Natural Science*, Princeton, NJ: Princeton University Press.