On the possibility, or otherwise, of hypercomputation

P. D. Welch

Abstract

We claim that a recent article of P. Cotogno ([2003]) in this journal is based on an incorrect argument concerning the non-computability of diagonal functions. The point is that whilst diagonal functions are not computable by any function of the class over which they diagonalise, there is no "logical impossibility" in their being computed over a wider class. Hence this "logical impossibility" regrettably cannot be used in his argument that no hypercomputation can compute the Halting problem. This seems to lead him into a further error in his analysis of the supposed conventional status of the Infinite Time Turing machines of Hamkins and Lewis ([2000]). Theorem 1 refutes this directly.

1 The Diagonalisation Misunderstanding

2 Infinite Computation

3 Conclusion

1 The Diagonalisation Misunderstanding

Paolo Cotogno ([2003]) seeks to show that any of the current models of hypercomputation (accelerated Turing machines, Zeus machines, infinite time Turing machines (ITTM's), models derived from neural nets, quantum computation etc., etc.) ultimately do not refute the physical interpretation of the Church-Turing thesis as he describes it (in Section 3). We find the aim laudable, and for the most part he has written an admirable account of the various models, which is very useful in its succinctness. We should very much like many of his conclusions to be true.

However he seems to be looking for a way in which to show that "hypercomputation" is *logically impossible*. This seems to us to be aiming high. In his abstract he states Thesis P: "that every effectively realizable physical system can be defined by Turing machines". Reviewing the main approaches to hypercomputation he continues "even if [hypercomputations] were realizable, the Halting problem would not be affected." The latter is expanded on in the introduction proper (p142): "even if hypercomputation were realizable it would not compute recursively unsolvable objects in principle, and therefore would not falsify the Church-Turing thesis." It is this last statement we shall concentrate on, but this explains the final sentence of his abstract: "Thesis P is not essentially different from the standard Church-Turing thesis". (We shall not enter into a discussion of this sentence.) He thus has set himself the task of showing that no non-standard model of computation can compute "recursively unsolvable objects in principle."

There are two ways one might do this: the high road of showing that there is some purely general logical block to computing such unsolvable objects; or the more mundane lower one of going by rote through each particular model as it is presented and showing that, yes, yet again, this is a model that sneaks in some infinitistic methodology into its structure which thus allows for all sorts of "computation" way beyond what we can call "effective". Cotogno seems to be choosing the high road.

However, and this is very regrettable, the article is seriously flawed, resting as it does on a misunderstanding of the nature of diagonalisation.

The key definition of the paper occupies Section 2 and is that of computability and incomputability. Section 2.1 gives a definition of the notion of effective computability, and 2.2 that of the diagonal function, g, that was used by Turing to show the unsolvability of the halting problem (in one form of the problem: "does the machine whose program code is the number m halt on input m?") These are both correct. Section 2.1 also wishes to make the distinction between *practical* incomputability (where, as a matter of fact, there is no method of computing a function f) and the case where we are in possession of a proof that f cannot be computed: *logical* incomputability. Section 2.2 starts out emphasising the centrality of "logical incomputability" and this section will remind us of it. Given a class of $\mathcal{C}$ of "computable procedures", and a listing thereof, the author correctly gives the definition by "diagonalisation" of a "diagonal function" g, and then the argument by contradiction of g's incomputability by any function within that class, *tout court*: $g \notin \mathcal{C}$. (If that class is that of Turing machines we have precisely (a form of) Turing's proof of the unsolvability of the Halting problem.) However Section 2.3 gives the first warning bell for concern. The notion of *function f is* *recursive in function* γ is given in the first sentence of the next quotation.

> A function $f(x) \simeq y$ is *recursive in* another function γ when the computing process depends on γ at some point; in terms of Turing Machines this is represented by an additional tape, called an oracle, where one can look at tabulated values for γ. This has a double aspect: if γ is assumed to be a function computable by 'unspecified means' as Turing put it, then the oracle defines a notion of *relative computability*. For the sake of generality, however, γ should be an arbitrary function, and thus can be an object like the $g(x)$ of 2.2.2; in this case the oracle defines rather a notion of *relative incomputability*, as any function $f(x)$ depending on g would inherit its incomputability.

The best possible reading of the last two sentences here is that the author states a distinction between relative computability and incomputability on the grounds that γ is computable 'by unspecified means' or whether it is an 'arbitrary function'. This seems very odd, and is new to us: we can consider any arbitrary function as having been "computed by unspecified means". (Common parlance today would render for either type of γ that 'f is computable relative to γ' or 'f is (Turing-) recursive in γ.') Alarm bells ring louder on the last clause: the author appears to be saying that if $f(x)$'s computation depends on an oracle tape for g then f inherits g's incomputability. This is false. Any recursive function h is recursive in g - and such definitely do not inherit g's non-computable status. One could perhaps worm one's way out of this by saying maybe it is the case that something extra has been hidden in the notion of that occurrence of '$f(x)$ depending on g' which the author has glossed over or forgotten to explain? Well, then it is an omission, and we hope no one is depending on this account for the notion of relative computability.

The author continues:

> "Holding that $g(x)$ is computable by itself would yield one of the following alternatives:
>
> (*) one output is chosen by convention
>
> (**) both outputs are accepted
>
> The latter requires an inconsistent system; the former yields indifference to the result of the computation..."

One can only make sense of this by assuming in the first line that the author has jumped away from the notion of computation which allowed us to draw up the list $\langle\psi_i\rangle_{i\in\mathbb{N}}$ enumerating $\mathcal{C}$, to some other notion of computation, perhaps wider; but let us call it "non-standard computation" to be neutral. (Note, this must be the case since the diagonal argument proved that g is *not* computable by the standard notion: g cannot be on the list $\langle\psi_i\rangle_{i\in\mathbb{N}}$. That is, no one would ever hold $g(x)$ to be on the list, other than to derive a contradiction - or to put it another way, as many alternatives to (*) and (**) above as one wants!) But no, even that does not make sense: were $g(x)$ non-standardly-computable by some means, this only implies that this other means is telling us (on input x) whether $g(x)$ is 0 or undefined; the dichotomy of (*) and (**) is not forced upon us. The author seems to think that were g to be non-standardly-computable then the formalism of that non-standard computation would be forced into this strait-jacket of a dichotomy. This he calls the Dilemma of Section 2: were the diagonal function g to be "computable" we should be on the horns of (*) or (**). The conclusion seems to be that diagonal function(s) are impossible to compute (standardly, or non-standardly).

2 Infinite Computation

The fourth section starts on supertasks and infinite computation. Zeno machines or Rapidly accelerating computers (RAC's) (e.g., Stewart [1991]) by increasing exponentially the speed of their working can effect infinitely many operations in one second, say. Hence in bounded time they can solve all instances of the halting problem. 4.7 gets closer to the heart of the difficulty:

A standard way of coding the halting problem is introduced: the binary real number K_0 so that $K_0(n) = 1 \iff \psi_n(n)$ halts; $K(n) = 0$ otherwise. (Here we may take $\langle\psi_i\rangle_{i\in\mathbb{N}}$ as enumerating the class $\mathcal{C}$ of all Turing computable functions.) Various models then, Zeno machines, RAC's, Davies' machine (Davies [2001]) etc., then may non-standardly compute the value of $K_0(n)$ in finite time, in order to solve the halting problem. But for Cotogno this has to be an impossibility:

> We know that diagonal functions cannot be properly computed on pain of inconsistency. (4.7)

If the 'computed' here is that of 'Turing computed' then the statement is true, and a triviality (modulo Turing's theorem); if it means 'non-standard computed' then, as above we see no need for pain. The problem seems to be that the author really believes that diagonal functions cannot be computed by any means at all... He continues:

> (4.8) The critical point here is that the real K_0 is assumed as given a priori, and this is just what the logical incomputability argument rules out:...

Why does K_0 have to be given a priori? (We do not want to labour the point, since we think no real conclusion of this note stands or falls by what we shall write in this paragraph. So the reader may wish to skip it. But it does add evidence to the fact that the author believes diagonal functions are "non-computable") The following paragraph of Cotogno ([2003]) indicates why Cotogno thinks so: that we *first* have to compute the graph of the function K_0 by denumerably many steps *before* the Zeno machine or whatever gets its hands on K_0 to do its infinitely many computations in its accelerated finite time of 1 second. Only then can Zeno solve the halting problem. This is puzzling: suppose a Zeno machine is given the task of writing a 0 on its tape if ψ_n does not halt, and a 1 otherwise. The machine commences by writing a 0 to its output tape, and then simulates ψ_n; if the simulation halts after finite time, then it changes the 0 to a 1; if not then after one second the Zeno machine has the correct value of 0 still there. By using up infinitely many cells, which are initially all set to 0, on the output tape, and by a dovetailing argument Zeno can simulate all ψ_n simultaneously and have written down precisely (the characteristic function of) K_0 to the (infinite) tape after one second. This ignores all issues about how/why the ZENO, RAC, etc., machine "halts" after 1 second of real time, but after infinitely many periods of its own "intrinsic" time. Perhaps it is this non-halting, or the physically undefined nature of the Zeno machine's status at real time $t = 1$ that the author wishes to allude to when he says that Zeno cannot possibly compute K_0. This seems rather stringent: suppose $K_0(23) = 0$ and we are observing Zeno who has been given the task of calculating this value. Why insist that Zeno must *halt before* $t = 1$?; we shall know the answer after 1 second; for us the Zeno machine is a good computer of K_0. There is no "logical incomputability" to hand.

Moving on to conceptual machines that work truly transfinitely in 4.10, we have the ITTM's of Hamkins and Lewis ([2000]) run transfinitely because they have some behaviour *defined* for them at ordinal limit stages of their operation. Essentially because such machines can answer infinitely many questions, and still continue running we have, as Cotogno quotes:

> (4.10.1) The truth of any arithmetic statement is infinite-time decidable ((Hamkins & Lewis [2000]),Thm 2.1).

However he continues:

(4.10) Since some true arithmetical statement is built around diagonal functions, one should check this claim against the dilemma of Section 2: there is no reason to think of case (**) inconsistency, so the oxymoron of infinite-time decidability must rest on case (*), random choice. The relevant point is the behaviour of ITTM's at limit ordinal stages.

The author correctly gives the limit stage rules, that stated a cell of the tape had the value written on it on a final segment of stages before the limit stage λ, or, if it had alternated unboundedly often before λ 0/1, Hamkins and Lewis plumped for the value 1 as a convention. He concludes the section:

> The third rule, the value 1 assigned by convention, rather than transfinite ordinals themselves, is what makes ITTM's capable of facing logical incomputability without becoming inconsistent. Prop.4.10.1 therefore does not expand our knowledge of arithmetical truth, unfortunately, but is instructive under other respects; it shows that the omniscience involved in infinite computation has a random seed, and exposes the supertask-like assumptions that may be implicit in claims of comparable strength, in higher recursion theory, see also Welch [2000].

We do not know quite what to make of the last sentence (other than as rhetorical flourish) or how exactly the author sees 4.10.1 as being more instructive than its statement. We are not sure what "supertask-like assumptions" are cited as implicit: it seems that the ITTM model is very explicitly supertask-like: the whole idea was to create a model whose non-standard-computational capacities, unfettered by space and finite ordinal time, could be investigated. There was never any intention that this was a physical realizable supertask machine, or should be turned into one. The advantage of the ITTM model is that it comes uncluttered with nagging Thompson-lamp like doubts about what happens after the switch has been thrown to and fro infinitely often: its behavior is *defined* to be such and such at limit points in its intrinsic time. This allows us a clear view as to what is theoretically computable when we allow an infinite, or even transfinite, sequence of Turing machine-like moves. This is so, not only for such machines, but of course for many such that can be simulated on ITTM's: for example RAC's, or Hogarth's Turing machines stacked up in chunks of suitably prepared spacetimes (Hogarth [2000]). ITTM's thus provide a *mathematical* laboratory for considering some other forms of supertask computation much as original Turing machines were used to clarify the notions surrounding (standard) computability. There is now a reasonably good understanding of the mathematical behavior of ITTM's: what are the 'decidable' predicates, or equivalently the 'computable' functions that such machines produce; what the potential ordinal lengths of halting computations are, and so on and so forth.

But we return to the nub of the matter at hand. It is indeed unfortunate that this ostentatiously conventional choice of value at limit stages has led the author into thinking that we have here case (*), and that this is the diagnosis of the "problem". We make three points of increasing weight:

(i) The choice of zero would have been equally good as a convention, and led to the same class of functions; in fact there are infinitely many possible choices of a "limit rule" which again leads to the same class of functions being computed (Welch [2000a] ,Theorem 2.9).

(ii) If one does not like any kind of conventional choice such as this, one can consider a variant ITTM model of a single tape with an alphabet of 0,1, and B(lank); one then assigns a B to a cell value at any limit stage λ, where the cell value has changed unboundedly often below and up to λ ("cofinally in λ"); one can show (Welch [2001] , Section 1) that this model again infinite-time-computes the same functions as the Hamkins-Lewis model; and now a siding with a 0, or 1, value by convention is avoided. We already have the Conclusion: (*) is not the problem at all. Moreover, and conclusively:

(iii) Even if the 3-valued models of (Welch [2001]) were unavailable to us, the conventional choice of limit rule is a red herring for Cotogno: it is possible to prove the following:

Theorem 1 *There is a Hamkins-Lewis ITTM program that halts, on input 0, after the first ω many steps with the characteristic function of K_0 on its output tape* without any cell of the machine's tapes having changed its value more than once.

Conclusion: *For this particular calculation, there is no need to invoke any limit rule/convention whatsoever.*

This knock-down argument entirely sidesteps his objection. (Stronger results than Theorem 1 are provable, for example, after $\omega + \omega$ many steps we may halt having written to the output tape the *Turing jump* of K_0, - that is, the halting problem relativised to K_0, similarly without having to overwrite cell values.)

The point of the matter is not that this ITTM model cannot compute K_0, or the diagonal function g defined from an enumeration of all recursive functions - it can. Even moving away from the ITTM arena, it is not that diagonal functions are "logically incomputable" (by any means). A diagonalisation over an enumeration of all machines in a model class only yields a function incomputable by the methods of that specific model class. There is a diagonal function $\widetilde{g}$ over the class of all ITTM's (defined in exactly the same way as for regular Turing machines). It is *this* function $\widetilde{g}$ that cannot be ITTM-computed of course. No dilemma as (*) and (**) is needed to demonstrate this. The problem evanesces. We simply may build a hierarchy of "computational models" (if you will) with increasingly complex diagonal functions, each universal machine of a class capable of calculating the diagonal functions of those lower in the hierarchy. Of course this logical hierarchy says nothing about, and has nothing to do with, physical realizability etc., etc.

3 Conclusion

Our conclusion is that the high road of showing the logical impossibility of hypercomputation by use of diagonal functions fails. In fact we believe such logical proofs of its impossibility may be akin to proofs that fairies are logically impossible: damn hard to be convincing. Perhaps at best, as intimated in the introduction, we have the (hopefully not Sysiphean) task of showing each time a model of hypercomputation is enthusiatically proposed, that, when not ruled out on grounds of physical unrealizability, then it is not effective in some sense. This usually consists in showing that it involves infinitistic methods in some form or other: either infinite precision measurement; the priming of nodes of the network with infinitely long decimal real numbers; "advice" functions of a general kind; in any case allowing some way that the characteristic function of K_0 could be (potentially) coded into, or hardwired into, or accessible to, the machinery or (Siegelmann [1995]) network. For such a machine notion to calculate K_0 or rhetorically "compute beyond the Turing limit" is then no triumph but a somewhat involved subterfuge.

To date this has been the case with each notion proposed, and the idea of *effectively* (in any real sense) "computing the uncomputable" remains as much a chimera as ever.

School of Mathematics,
University of Bristol,
Bristol, BS8 1TW,
England

p.welch@bristol.ac.uk

References

Cotogno, P. [2003]: ‘Hypercomputation and the physical Church-Turing thesis’, *British Journal for the Philosophy of Science*, **54** pp. 181–223.

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

Hamkins, J.D. and Lewis, A. [2000]: ‘Infinite time Turing machines’, *Journal of Symbolic Logic*, **65**,(2) pp. 567–604.

Hogarth, M. [1994]: ’Non-Turing Computers and Non-Turing Computability’, in *eds.* D. Hull, M. Forbes, & R. M. Burian. Proceedings of the 1994 biennial meeting of the Philosophy of Science Association, 1994, **1** pp. 126-138.

Siegelmann, H. [1995]: ‘Computation beyond the Turing limit’, *Science*, **268**, pp. 545-548.

Stewart,I [1991]: ‘ The Dynamics of impossible Devices’, *Nonlinear Science Today*, **1**, pp. 8-9.

Welch, P.D. [2000]: ‘The length of infinite time Turing machine computations’ *Bulletin of the London Mathematical Society*, **32**, pp.129–136.

Welch, P.D. [2000a]: ‘Eventually infinite time Turing degrees: infinite time decidable reals’, *Journal for Symbolic Logic*, **65**(3) pp. 1193–1203.

Welch, P.D. [2001]: ‘Post’s and other problems in higher type supertasks’ In B. Löwe, W. Malzkorn, T. Räsch, (*eds.*), to appear 2004, *Complexity in Mathematics and Computer Science*, volume III of *Foundations of the Formal Sciences*, Kluwer, *Trends in Logic* series.