doi:10.1093/comjnl/bxh000

Are There New Models of Computation? Reply to Wegner and Eberbach*

Paul Cockshott[1] and Greg Michaelson[2]

[1]*Department of Computing Science, University of Glasgow*
17 Lilybank Gds, Glasgow, G12 8QQ
[2]*School of Mathematical and Computer Sciences, Heriot-Watt University*
Riccarton, EH14 4AS
Email: G.Michaelson@hw.ac.uk

Wegner and Eberbach [Wegner, P. and Eberbach, E. (2004) New Models of Computation. Computer Journal, 47, 4–9.] have argued that there are fundamental limitations to Turing Machines as a foundation of computability and that these can be overcome by so-called superTuring models such as interaction machines, the πcalculus and the $-calculus. In this paper we contest the Wegner and Eberbach claims.

1. INTRODUCTION

The concept of a Turing machine[1] founded Computer Science as we know it today. Turing's insightful characterisation of Hilbert's Entscheidungsproblem formalised computation in the abstract, and enabled its very practical realisation in the digital computers that underpin contemporary society. In a Kuhnian sense[2], the Turing machine (TM) has been the dominant paradigm for Computer Science for 70 years: Ekdahl[3] likens an attack on it to a "challenge to the second law of thermodynamics".

The roots of Turing's work lie in debates about the notion of computability in the pre-computer age. Just before the Second World War, in an outstanding period of serendipity, Turing, Church[4] and Kleene[5] all developed independent notions of computability (i.e. TMs, the λ-calculus and recursive function theory) which were quickly demonstrated to be formally equivalent. These seminal results form the basis for the Church-Turing Thesis that all notions of computability will be equivalent. Until now, the Church-Turing Thesis has remained unshaken. In particular, it has long been known that the Thesis holds for the von Neumann architecture of everyday digital computers, enabling the application of a profound body of theory to real-world computing.

The classic conception of computability envisages an all embracing space of problems. Within this space it is usual to distinguish those problems with solutions which are effectively computable from those which are not effectively computable. A central concern of these pre-computer Mathematical Logicians was to formalise precisely this concept of effective computation. For Church, this is a matter of *definition*, explicitly identifying effective calculability with recursive or lambda-definable functions over the positive integers. Church states that:

> If this interpretation or some similar one is not allowed it is difficult to see how the notion of an algorithm can be given any exact meaning at all.([4] p356)

Turing subsequently outlined a proof of the equivalence of his notion of "computability" with Church's "effective calculability".

All three formulations are based on systems of rules for manipulating symbols with procedures for their application. However, a fundamental distinction of Turing's approach is that he identifies a human-independent mechanism to embody his procedure, by explicit analogy with a human being:

> We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions... ([1] Section 1).

In a 1939 paper discussing the unity of these different approaches, Turing is explicit about the mechanical nature of effective calculation:

*A short version of this paper appeared as Michaelson, G. and Cockshott, P. (2006) Constraints on Hypercomputation. Beckmann, A., Berger, U., Lowe, B., and Tucker, J. V. (eds.), *Logical Approaches to Computational Barriers: Second Conference on Computability in Europe, CiE 2006, Swansea, UK*, June/July, pp. 378–387, LNCS 3988, Springer.

> A function is said to be "effectively calculable" if its values can be found by some purely mechanical process ... We may take this statement literally, understanding by a purely mechanical process one which may be carried out by a machine. It is possible to give a mathematical description, in a certain normal form, of the structures of these machines. ([6] p166).

In a late paper he makes the same point with reference to digital computers:

> The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. ([7]Section 4).

In contrast, for Church and Kleene the presence of a human mathematician that applies the rules seems to be implicit. Nonetheless, it is clear that the ability to give an explicit procedure for applying rules to symbols and to physically realise these procedures, whether in a machine or in a mathematician with pencil and paper, was central to all three conceptions of effectiveness. The crucial corollary is that a computation which is not physically realisable is not effective. We will develop these themes in more detail below.

2. HOW MIGHT THE TM PARADIGM BE DISPLACED?

2.1. Expressive power, meaning and effective computation

Much of our argument is concerned with properties of formal systems for characterising computations. In particular, we shall often refer to the expressive power of a system. It is important to clarify the differences between what can be expressed in a system, whether or not such expressions are meaningful, and if they are meaningful whether or not they may be calculated effectively.

For example, Russell's paradox formalises in predicate calculus the set of all sets which are not members of themselves. The formula is self-contradictory so one might say that it is meaningless. However, the formula translates into an apparent algorithm for an equivalent file system directory paradox which seeks to construct the directory of directories which do not have links to themselves:

```
create empty directory of directories D
REPEAT
 D' := D
 FOR each directory d reachable from
     the filing system root
  IF d does not have a link to itself AND
     d is not in D THEN
   add d to D
  ELSE
   IF d is in D THEN
    remove d from D
UNTIL D=D' i.e. D hasn't changed
```

This "algorithm" will, after the first pass, repeatedly add D to itself and then remove it from itself. It does not depend on an infinite memory or an infinite number of processes. It is clearly implementable in an arbitrary shell script and so appears to be meaningful in so far as it does describe a computation, but the computation never terminates and so is not effective.

Hayes and Jones[8] have argued that there is a clear distinction between expressive power and implementability. In specification notations like Z and VDM(Vienna Definition Method), it is possible to specify computations that are not effective, in particular those involving the unbounded traversal of infinite domains. Hence in general, they argue, specifications are not necessarily executable.

We note here that such specification notations are based essentially on set theoretic predicate calculus, typically embodying recursive function theory and are of equivalent expressive power to TMs or λ-calculus. Furthermore, infinitary specifications that characterise semi-decidable decision procedures may be realisable as TMs, which may not halt and so may or may not consume infinite resources.

As we shall see, many of the claims made by Wegner and Eberbach depend on the ability of particular systems to express unbounded traversal of infinite domains. However, the ability to express such requirements does not mean that they are meaningful or effectively calculable.

2.2. TM overview

We do not intend here to give thorough accounts of computability theory or of TMs: for more detail see Davis's classic text [9]. Rather, we will now summarise salient points to which we return below.

A TM may be thought of as a machine with a hardwired set of instructions and a record of its current state. It has a read/write head which can inspect and modify a linear tape, inscribed with symbols, and of indeterminate length in both directions. Each instruction is a rewrite rule which specifies that for a given state St_{old} and input symbol Sy_{old}, some new state St_{new} is to be entered, the old symbol is to be changed to a new symbol Sy_{new} and the tape is to be moved one symbol in either a left or right direction Dir:

$$(St_{old}, Sy_{old}) \rightarrow (St_{new}, Sy_{new}, Dir)$$

Conventionally, such instructions are encoded as quintuplet of the form:

$$(St_{old}, Sy_{old}, St_{new}, Sy_{new}, Dir)$$

A TM is started in an initial state over some designated first symbol on the tape. It then repeatedly finds an

instruction corresponding to the corresponding state and symbol, changes the state and symbol, and moves the tape in the designated direction. If a terminating state is reached then the TM halts. If no matching instruction can be found then the TM fails.

Note that the tape for a TM that halts is finite in length, as terminating execution takes a finite number of steps which can only access at most that number of sequential tape positions. However, because the termination of arbitrary TMs over arbitrary initial tapes is undecidable, as discussed below, the required length of tape for a given computation is, in general, not knowable in advance. Hence a TM tape is unbounded, rather than infinite as it is often termed.

2.3. Equivalence, reduction and the universal TM

One of the engaging features of Turing's elaboration of TMs is the deployment of simple encodings, reductions and constructions in establishing their properties, especially to demonstrate the equivalence of apparently more complex TM formulations with the original conception. In particular, it is straightforward to demonstrate that:

- a TM machine with multiple tapes can be reduced to an equivalent TM with a single tape;
- a non-deterministic TM (i.e. a TM where several quintuplets may apply for a given state and input symbol) can be reduced to an equivalent deterministic TM.

Turing showed that it is possible to formalise TMs themselves as TMs; that is, TM notation may be used as its own meta-language. Thus, a Universal Turing Machine (UTM) is a TM which given a tape describing a second TM and its initial tape, will perform the same effective computation as that second TM on that tape. Suppose we write $TM_i(T_j)$ to mean TM i started on tape j. Suppose $+$ is tape concatenation[1]. Then:

$$UTM(TM_i + T_j) = TM_i(T_j)$$

Turing suggested that such self-encoding was a strong indication that TMs captured a most general sense of computation. Subsequently, this has become accepted as one hallmark for any theory that is claimed to characterise effective computation.

2.4. Decision procedures, undecidability and canonical exemplars

A fundamental motivation for developing TMs was for use in characterising problems as solvable or unsolvable. If an instance of a problem can be expressed as a TM and tape, then the TM is said to encode the corresponding decision procedure. If that TM will terminate with a solution after a finite number of execution steps then the problem is said to be decidable. If it is not possible to construct such a TM then the problem is said to be undecidable. If a TM can be constructed but it cannot be determined by computable means whether or not it will terminate then the problem is said to be semi-decidable.

In the Church/Kleene formulation, decidable problems are those characterised by general recursive functions which always halt, where semi-decidable problems are those characterised by partial recursive functions which may not halt. Note that partial recursive functions have equivalent TMs but these TMs may not embody effective computations: an effective computation is known to halt but the TM for a partial recursive function may not.

Given that the execution of any TM may be characterised through a UTM and an encoding of that TM, it seems reasonable to speculate that it might be possible to elaborate a decision procedure to determine whether or not an arbitrary TM would terminate when started over an arbitrary tape. In an elegant self-referential construction, Turing showed that positing a general TM to decide termination is inherently contradictory. Hence, in proving that the Halting Problem is undecidable, Turing established a fundamental limitation to TMs, and to effective computation.

The Halting Problem has become a canonical exemplar of undecidability: if any other problem is reducible to an instance of the Halting Problem then it must be undecidable. Thus, it may be shown that there is no decision procedure to determine if two arbitrary TMs are equivalent in the sense of performing the same effective computation.

The technique of reducing problems to a canonical exemplar is also central to complexity theory, another heir of Turing's work. The complexity of an algorithm may be characterised in terms of the time and space behaviour of the TM class to which it belongs. In particular, an algorithm is said to be P if it belongs to the class of TMs which will compute a solution in polynomial time. Alternatively, an algorithm is said to be NP if it belongs to the class of TMs which will compute a solution in polynomial time given an oracle.

Oracles are essentially entities which can somehow correctly generate some fundamental property of a problem. It is important to note that oracles are not effectively calculable. In[6], Turing defines an oracle as "an unspecified means of solving number-theoretic problems"(p172). He goes on to say that: "We shall not go any further into the nature of this oracle apart from saying that it cannot be a machine.". Turing defines an o-machine as TM augmented with an oracle and briefly outlines a proof that o-machines cannot solve their own Halting Problem.

It is an open question whether the classes of P and NP algorithms are equivalent. However, Cook[10] established that it is possible to identify NP Complete

[1] which introduces appropriate separation symbols

(NPC) algorithms for which the construction of an equivalent P algorithm would imply that all NP algorithms have equivalent P algorithms. In particular, he proved that determining the satisfiability of sets of boolean equations (SAT) is a canonical NPC algorithm, and that any algorithm which can be reduced to an instance of SAT is also NP. From our perspective, if P and NP were equivalent then some oracles could be reduced to TMs i.e some problems which were thought to involve non-effectively computable steps would become amenable to effective computation. Where previously an oracle was required to enable the TM to complete the computation in polynomial time, it would now be possible to replace it with a polynomial time TM.

2.5. Displacing the Church-Turing Thesis

We are now in a position to discuss what might be required to overthrow the Church-Turing Thesis, that is to show conclusively that some new characterisation of computability is more powerful than all those already known, in particular TMs. We must be clear from the outset that we regard challenges to Church-Turing (C-T) as not only perfectly legitimate but highly desirable. In general, substantive challenges to established theories force their acolytes to re-examine basic tenets that until then have been complacently unquestioned, in turn reinvigorating those theories where they are not overthrown. In particular, the elaboration of new systems that transcend all known models of computation would have truly revolutionary implications for computing theory and practise, in principle enabling us to approach fundamental problems currently thought to be unassailable.

However, this places a high onus on these mounting such challenges to provide convincing evidence for their claims. Computer Science, if not an oxymoron, is presumably a species of normal science [2] which slowly accretes new self-validating knowledges in its own terms, without worrying too much about apparently minor discrepancies. Thus a successful challenge must present new results that are so bothersome that even the most pig-headed proponents of the old order, amongst whom we count ourselves in this case, are forced of necessity to reconsider their stance. So what might such results look like?

In general, a demonstration that a new system is more powerful than a C-T system involves showing that while all terms of some C-T system can be reduced to terms of the new system, there are terms of the new system which cannot be reduced to terms of that C-T system i.e the new system has greater expressive power than that C-T system and hence of any C-T system [3]. More concretely, we think that requirements for a new system to conclusively transcend C-T are, in increasing order of strength:

(i) demonstration that some problem known to be semi-decidable in a C-T system is decidable in the new system;
(ii) demonstration that some problem known to be undecidable in a C-T system is semi-decidable in the new system;
(iii) demonstration that some problem known to be undecidable in a C-T system is decidable in the new system;
(iv) characterisations of classes of problems corresponding to (i)-(iii);
(v) canonical exemplars for classes of problems corresponding to (i)-(iii).

Above all, we require that the new system actually encompasses effective computation; that is, that it can be physically realised in some concrete machine, be it an analytic engine or a human with a slate and chalk. While we are not unduly troubled by systems that require unbounded resources such as an unlimited TM tape, we reject systems whose material realisations conflict with the laws of physics, or which require actualised infinities as steps in the calculation process.

Note that in the following discussion of Wegner and Eberbach's claims, we do not systematically refer to these requirements. However, we will return to them in our summary in the final Conclusion section.

3. PHYSICAL REALISM AND COMPUTATION

Turing marks what Bachelard and Althusser[11] termed an Epistemological break in the history of the sciences. At once the problematic of Hilbertian rationalism is abandoned [12] and at the same time the Platonic view of mathematics is displaced by a materialist view.

A key point about the Universal Computers proposed by Turing is that they are material apparatuses which operate by finite means. Turing assumes that the computable numbers are those that are computable by finite machines, and initially justifies this only by saying that the memory of a human computer is necessarily limited. By itself this is not entirely germane, since the human mathematician has paper as an aide memoir and the tape of the TM is explicitly introduced as an analogy with the squared paper of the mathematician.

> Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that

[3] It would be truly astonishing if a new system were demonstrated whose terms could not be reduced to those of a C-T system i.e. the new system and C-T systems had incomparable expressive power.

> the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares.([1] section 9)

However Turing is careful to construct his machine descriptions in such a way as to ensure that the machine operates entirely by finite means and uses no techniques that are physically implausible. His basic proposition remained that :"computable numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means."

Turing thus rules out any consideration that computation by infinite means is a serious proposition. In this he follows Aristotle ([13] Book III, chap 6) in allowing potential but not actual infinities. If infinite computation were to be allowed, then the limitations introduced by the Turing machine would not apply. However, a number of proposals for superTuring computation rest on the appeal of the infinite.

Copeland[14] proposes the idea of accelerating Turing machines whose operation rate increases exponentially so that if the first operation were performed in a microsecond, the next would be done in $\frac{1}{2}\mu s$, the third in $\frac{1}{4}\mu s$, etc. The result would be that within a finite interval it would be able to perform an infinite number of steps. This obviously evades Turing's stipulation that computable numbers must be calculable by finite means, but at the same time evades all possibility of physical realisation. A computing machine must transfer information between its component parts in order to perform an operation. If the time for each operation is repeatedly halved, then one soon reaches the point at which signals travelling at the speed of light have insufficient time to propagate from one part to another within an operation step. Beyond this speed the machine could not function. Hamkins[15] discusses what could be computed on Turing machines if they were allowed to operate for an infinite time, but Turing ruled this out with obvious good reason.

A theme proposed by some advocates of Super-Turing computation is the use of analogue computation over real numbers. For a review see Copeland[16]. Copeland sees these as being possible if one has available a continuously variable physical quantity which he calls 'charge'. Since he himself recognises that electric charge is discrete, it is not clear how this notional charge is ever to be physically realised. Indeed the idea of being able to physically represent real numbers is highly questionable in view of the quantum of action h. This poses fundamental and finite limits on the accuracy with which a physical system can approximate real numbers. We will illustrate this with a concrete example of a proposed super-Turing analogue computer and show how the uncertainty principle will vitiate its action. Our example is the machine proposed by Bournez and Cosnard [17]. The basic concept of this machine is to use two real valued variables corresponding to the x, y coordinates of photons passing through plane P in Figure 1. Drawing on the work of Moore and Koiran [18], they propose that successive digits of the binary expansion of these real valued coordinates could then be used to emulate the left and right parts of a Turing machine tape. Bournez and Cosnard argue that the machine could in addition be used to simulate of a class two stack automata whose computational power might exceed those of TMs.

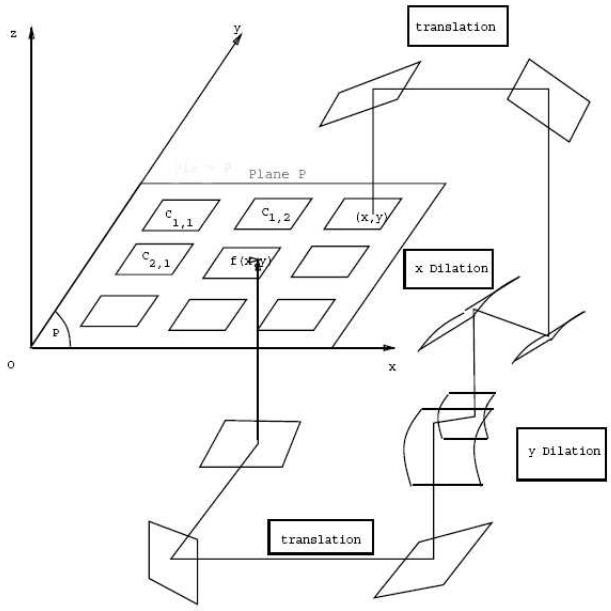

FIGURE 1. An analogue computer proposed by Bournez and Cosnard. Reproduced from [17].

Bournez and Cosnard show that if one had available iterated real valued functional systems based on piecewise affine transforms, such analogue automata could be implemented. They then propose that a system of mirrors as shown in Figure 1 would be able to implement these piecewise functions: multiplication by reals could be implemented by pairs of parabolic mirrors, and translation by arrangements of planar ones.

The problem with this design is that it assumes that photons act like billiard balls but completely ignores diffraction effects.

Any optical system has an irreducible diffraction limited circle of confusion that is inversely proportional to its aperture and proportional to its focal length and the wavelength of the photons used. The angle to the first off-center diffraction peak $\Delta\Theta$ is given by

$$sin(\Delta\Theta) = \frac{\lambda}{A} \tag{1}$$

where A is the aperture and λ the wavelength. This is a particular case of the uncertainty principle. By constraining the position of the photon to be within the

aperture, we induce an uncertainty in its momentum within the plane of the aperture.

To see what this implies, let us give some plausible dimensions to the machine. Assume the aperture of the mirrors in their system was 25mm and the path length of a single pass through the system from the first mirror shown in Figure 1 back to Plane P is 500mm. Further assume that we use light with $\lambda = 0.5\mu$. This would give us a circle of confusion with a a radius $\Delta_{f(x,y)Mirror} \approx 10\mu$. If plane P had edges of 100mm, then the optical path could resolve about 5000 distinct points in each direction as possible values for `f(x, y)`: about 12 bits accuracy.

The dispersion $\Delta_{f(x,y)Mirror}$ accounts only for the first pass through the apparatus. Let us look at the uncertainty in x, y to start out with.

It is necessary to specify x, y to greater accuracy than `f(x,y)` so that $\Delta_{x,y} < \Delta_{f(x,y)}$. Let us use a mask with a circular hole to constrain the incoming photons to be within a radius of less than 10μ. Clearly this constraint on the position of the photons amounts to an initial aperture with a diameter of, let us say 10μ, whose diffraction cone would have a $\Delta\Theta \approx 0.05$ radians. After passing through the full optical path, the uncertainty in position due to the input aperture would then be $\approx$ 25mm. Call this $\Delta_{f(x,y)Mask(\Delta_{x,y})}$. It is clear that in this case $\Delta_{f(x,y)Mask(\Delta_{x,y})} >> \Delta_{f(x,y)Mirror}$.

The more precisely we constrain x, y the more uncertain will be the result `f(x,y)`. Mirror Machines of this class will achieve a limiting accuracy for a single pass when $\Delta_{f(x,y)Mirror} + \Delta_{f(x,y)Mask(\Delta_{x,y})} \leq \Delta_{x,y}$. From simple geometry this will come about when the ratio $\Delta_{x,y}/L \approx \lambda/\Delta_{x,y}$ so

$$\Delta_{x,y} \approx \sqrt{L\lambda} \tag{2}$$

where L is the optical path length of the computation. For the size of machine which we have assumed above, this implies $\Delta_{x,y} = 500\mu$. Its accuracy of representation of the reals is thus less than 8 bits(= - $\log_2(\frac{500\mu}{100mm})$), hardly competitive with existing digital computers.

Remember too that for a Mirror Machine, the evaluation of `f(x,y)` corresponds to a single step of a Turing Machine program. As the number of steps n grows so does the traversed optical path nL, and by (2), the optimal initial aperture setting $\Delta_{x,y}$ will grow as $\sqrt{n}$. Each fourfold increase in the execution length of the program, will reduce by 1 bit the accuracy to which the machine can be set to compute.

To improve the angular accuracy of an optical machine we have to increase its aperture. To add 1 bit to the accuracy of the machine would require us to double all its linear dimensions, which would tend to increase its mass by a factor of 8. The mass required to perform a calculation to n bits of accuracy thus varies as 2^{3n} for the mirror machine. For a von-Nuemann machine implemented on silicon, the mass of the store grows linearly with the bit accuracy of computation n and the mass of the arithmetic unit grows as $n \log n$. It thus follows that whilst the Mirror Machine might be competitive with a digital computer for certain low accuracy computations, for computations requiring high accuracy and/or a large number of steps it will be out-performed by the digital machine. This indeed, was the fundamental historical reason that analogue computers lost out to digital machines. Turing foresaw in 1947 that this was likely to happen:

> That the machine is digital however has a more subtle significance. It means firstly that numbers can be represented by strings of digits that can be as long as one wishes. One can therefore work to any desired degree of accuracy. This accuracy is not obtained by more careful machining of parts, control of temperature variations, and such means, but by a slight increase in the amount of equipment in the machine. To double the number of significant figures, would involve increasing the amount of the equipment by a factor definitely less than two, and would also have some effect in increasing the time taken over each job. This is in sharp contrast with analogue machines, and continuous variable machines such as the differential analyser, where each additional decimal digit required necessitates a complete redesign of the machine, and an increase in the cost by as much as a factor of 10. [19]

In the case of the Mirror Machine, the limitation on its accuracy is set by the wave/particle duality of photons. Analogue computing systems based on particles with real rest masses would be subject to analogous limits, due in this case to de Broglie waves.

Thus :

(i) Any physically build-able analogue memory will only approximate the reals.
(ii) Analogue storage of reals, will for high precision work, always be outperformed in terms of device economy by digital storage.
(iii) Physically build-able analogue computers can not rely upon the availability of exact stored real numbers to outperform Digital Computers.

Proposals to incorporate the full mathematical abstraction of real numbers into computing devices so as to allow them to outperform Turing machines are physically implausible. Such proposed machines are mathematical abstractions rather than practical proposals.

For Turing, Computers were not mere mathematical abstractions but concrete possibilities, which he pursued in his designs for the Bombe at Bletchley and ACE(Automatic Computing Engine) at the National Physical Laboratory[19, 20]. Computing Science has undergone explosive progress in the period since Turing's original paper which has synthesised abstract computational theory with research into possible physical implementations of digital computers.

Enormous sustained research has developed successively more effective generations of such machines. The development of smaller and faster computers has been a constant struggle towards the limits of what is physically possible. One can not say in advance how small or how fast we will eventually be able to build computers, but we do know that any effective computer we build will be bound by the laws of physics. Landauer[21, 22] has examined the fundamental thermodynamic limitations of computation, and Deutsch[23] and Feynman[24, 25] opened up the new field of quantum computing research. An important result obtained by Deutsch was that although quantum machines could potentially have higher parallelism than classical ones, they would not extend effective computation beyond the recursive functions.

Kieu [26] recently challenged Deutsch's result, with a proposal to solve Diophantine equations by means of a quantum process. The basic idea is to prepare a quantum system with a set of possible states mapped to natural numbers and a Hamiltonian which encodes the Diophantine equation. The system would be allowed to adiabatically evolve to a ground state corresponding to a solution to the equation. This proposal, however, remains controversial. Critical responses include [27, 28, 29] and replies to these criticisms [30, 31]. Points made by critics are:

(i) There is no guarantee that one can distinguish between a system that has settled into a true ground state or a 'decoy' false minimum.
(ii) One would need arbitrary high precision in energy measurements.
(iii) The evolution time needed to reach the ground state is unknown and in principle Turing uncomputable. One thus reproduces the halting problem in a different form.

Claude and Pavlov [32, 33] have proposed a model of hypercomputation similar in style to that of Kieu. Their work focuses on the 'Infinite Merchant's Problem', which they argue is equivalent to the Halting Problem. Whilst their work is interesting, it is not yet clear whether they have an effective way of translating an encoding of a TM into an instance of the Infinite Merchant Problem that could be solved by their proposals.

To conclude, appeals to infinity in new models of computation seem to run up against fundamental physical limits. In the discussion below we explore how Wegner and Eberbach's proposed super-Turing models may also be subject to similar limitations.

4. WEGNER AND EBERBACH'S SUPERTURING COMPUTERS

Wegner and Eberbach[34] assert that there are fundamental limitations to the paradigmatic conception of computation which are overcome by more recent "superTuring" approaches. We will now summarise their core arguments before exploring them in greater detail.

Wegner and Eberbach draw heavily on the idea of an algorithm as an essentially closed activity. That is, while the TM realising an algorithm may manipulate an unbounded memory, the initial memory configuration is pre-given and may only be changed by the action of the machine itself. Furthermore, an effective computation may only consume a finite amount of the unbounded memory and of time, the implication being that an algorithm must terminate to be effective.

They say that the TM model is too weak to describe the Internet, evolution or robotics. For the Internet, web clients initiate interactions with servers without any knowledge of the server history. The Internet as a dynamic system of inputs and outputs, parallel processes and communication nodes is outside the realm of a static, sequential TM. Furthermore, TMs cannot capture evolution because the solutions and algorithms are changed in each generation and the solution search is an infinite process. This does not depend on a finite or infinite search space. Because evolutionary algorithms are probabilistic the search may take an infinite number of steps over a finite domain. Finally, robots interact with non-computable environments which are more complex than the robots themselves.

Wegner and Eberbach claim that there is a class of superTuring computations (sTC) which are a superset of TM computations. That is sTC includes computations which are not realisable by a TM. A superTuring computer is "any system or device which can carry out superTuring computation".

Most significantly, Wegner and Eberbach say that it is not possible to describe all computations by algorithms. Thus they do not accept the classic equation of algorithms and effective computations.

They go on to argue that there are three known systems which are capable of sTC: interaction machines (IM), the π-calculus and the \$-calculus. They give discursive presentations of these systems and explore why they transcend the TM.

5. INTERACTION MACHINES

Wegner and Eberbach refer to Interaction Machines as a class of computer that is more powerful than the Turing Machine. The latter, they claim, is restricted by requiring all its inputs to appear on the tape prior to the start of computation. Interaction machines on the contrary can perform input output operations to the environment in which they are situated. The difference between Turing Machines and Interaction Machines, they claim, corresponds to the technology shift from mainframes to workstations. Interaction Machines, whose canonical model is the Persistent Turing Machine(PTM) of Goldin [35], are not limited to

a pre-given finite input tape, but can handle potentially infinite input streams.

This argument was originally advanced by Wegner in a previous publication[36], some of whose main arguments have been criticised by Ekdahl[3]. Rather than rehearse Ekdahl's critique we shall focus on some additional weaknesses of Wegner and Eberbach's claims.

5.1. Turing's own views

As is well known, Turing's contribution to computer science did not stop with the Turing Machine. Besides his work on cryptography, he played a seminal role in the establishment of Artificial Intelligence research. His Turing Test for machine intelligence is probably as well known as his original proposal for the Universal Computer. He proposed in a very readable paper[7], that a computer could be considered intelligent if it could fool a human observer into thinking they were interacting with another human being. It is clear that his putative intelligent machine would be an Interaction Machine in Wegner's sense. Rather than being cut off from the environment and working on a fixed tape, it receives typed input and sends printed output to a person.

Turing did not, however, find it necessary to introduce a fundamental new class of computing machine for this gedanken experiment. He is quite specific that the machine to be used is a digital computer and goes on to explain just what he means by such a machine:

> The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail. We may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. He has also an unlimited supply of paper on which he does his calculations. He may also do his multiplications and additions on a 'desk machine', but this is not important. ([7] page 436)

This is of course a paraphrase of his description of the computing machine in his 1936 paper[1] where he explicitly models his machine on a person doing manual calculations. The states of the machine correspond to the finite number of states of mind of the human mathematician and the tape corresponds to the squared paper he uses. It is clear that Turing is talking about the same general category of machine in 1950[7] as he had in 1936[1]. With the practical experience of work on the Manchester Mark 1 and the ACE behind him, he speaks in more general terms of the computer as being composed of (i) store, (ii) executive unit, and (iii) control, and says that the store locations are addressable rather than being purely sequential. He says he is concerned with discrete state machines, and that a special property of such digital computers was their universality:

> This special property of digital computers, that they can mimic any discrete state machine, is described by saying that they are universal machines. The existence of machines with this property has the important consequence that, considerations of speed apart, it is unnecessary to design various new machines to do various computing processes. They can all be done with one digital computer, suitably programmed for each case. It will be seen that as a consequence of this all digital computers are in a sense equivalent.([7] p442)

This is clearly a recapitulation of the argument in section 6 of his 1936 paper[1] where he introduced the idea of the Universal Computer. Turing argued that such machines were capable of learning and that with a suitable small generalised learning program and enough teaching, then the computer would attain artificial intelligence.

5.2. Equivalence of Interaction Machines and Turing Machines

It seems that Turing considered that the class of machines which he had introduced in 1936[1] had all of the properties that Wegner and Goldin were later to claim for their Interaction Machines and Persistent Turing Machines. He may of course have been mistaken, but we think that Turing's confidence was well founded. It can be demonstrated that Turing Machines are powerful enough for the task.

Consider first a digital computer interacting in the manner foreseen by Turing in his 1950 paper[7], with teletype input/output. The teletypes of Turing's day had the capability of capturing keystrokes to paper tape, or of accepting paper tape input instead of keystrokes. Suppose then we have a computer initialised with a simple learning program following which it acquires more sophisticated behaviour as a result of being 'taught'. As the computer is taught we record every keystroke onto paper tape.

Suppose we initialise a second identical computer with the same program and, at the end of the first computer's period of interaction, we give to the second machine as an input the tape on which we have recorded the all the data fed to the first machine. With the input channel of the second machine connected to the tape reader it then evolves through the same set of states and produce the same outputs as the original machine did. The difference between interactive input from a teletype and tape input as used by Turing in 1936 is essentially trivial. By a simple recording mechanism one can emulate the behaviour of an interactive machine

on another tape-input computer. This has been a widely used and practical test procedure.

In his 1950 paper[7] Turing clearly assumes that his computer has a persistent or long term memory. Wegner and Goldin Persistent TMs allow a machine to retain data on a tape so that it can be used in subsequent computations. They claim that the idea of a persistent store was absent in TMs. However, persistence only became an issue because the combination of volatile semi-conductor store and first generation operating systems imposed on programmers a sharp pragmatic distinction between persistent disk store and volatile random access memory[37, 38]. This distinction had not been present in a first generation of magnetic core based von-Neumann machines, and was not included in the basic computational models of Turing and von-Neumann.

A small modification to the program of a conventional TM will transform it into a PTM. Like Goldin we will assume a 3 tape TM, M_1, with one tape T_1 purely for input, one tape T_2 purely for output and one tape T_3 used for working calculations. We assume that tapes T_1, T_2 are unidirectional, T_3 is bidirectional. Such a machine can be emulated on Turing's original proposal by a suitable interleaving scheme on its single tape.

M_1 has a distinguished start state S_0 and a halt state S_h. On being set to work it either goes into some non-terminating computation or eventually outputs a distinguished termination symbol τ to T_2, branches to state S_h and stops. We assume that all branches to S_h are from a state that outputs τ. Once τ has been output, the sequence of characters on T_2 up to to τ are the number computed by the machine.

We now construct a new machine M_2 from M_1 as follows: replace all branches to S_h with branches to S_0. From here it will start reading in further characters from T_1 and may again evolve to a state where it outputs a further τ on T_2.

Machine M_2 now behaves as one of Goldin's PTMs. It has available to it the persisting results of previous computation on T_3 and these results will condition subsequent computations. It is still a classic TM, but a non-terminating one. It follows that PTMs, and thus Interaction Machines of which they are the canonical example, are a sub-class of TM programs and do not represent a new model of computation.

5.3. Thermodynamic considerations

For Wegner and Eberbach, there is a fundamental difference between starting a TM over a given tape that is only changed by that TM, and where additional input comes from the environment. This alleged distinction may be further explored using an algorithmic information theoretic argument.

Chaitin introduced algorithmic information theory [39] according to which the entropy of a binary number x is bounded by the number of bits of the shortest TM program that will output x. From this standpoint there is a pragmatic difference between an isolated TM and one that can accept input from the environment.

A modern computer is initially built with a startup ROM and a blank disk drive. The ROM typically contains a BIOS, but can be replaced by any other program that will fit on the chip. Let us suppose, realistically, that the ROM chip contains 2^{19} bits. Suppose that instead of a BIOS, we put in a ROM that performs some predefined algorithm, possibly using disk I/O, and in the process of computation outputs a stream of characters from the serial port. Chaitin's result indicates that if the program performs no input operations from the keyboard, mouse, CD etc, the entropy of the information on disk plus the information output on the serial line could not exceed $2^{19}+1$, where the additional 1 bit encodes whether the initial blank state of the disk was a 1 or a 0. If the disk was much bigger than the boot chip, it could, for example, never really be randomised by the boot chip.

A practical BIOS chip will attempt to read input from keyboards, communications lines, or CD, starting a process that allows the disk to gain entropy from external sources. We know from experience that disks become cluttered and entropic over time. The external world acts as an entropy source causing the disk entropy to rise in conformance with thermodynamic laws.

Thus, we can formulate the interaction process within Chaitin's framework which is in turn grounded in TM theory. This again implies that Interaction Machines are not a new class of computer.

6. π-CALCULUS

Wegner and Eberbach give the π-calculus as another of their three superTuring computational models. The π-calculus is not a model of computation in the same sense as the TM: there is a difference in level. The TM is a specification of a material apparatus that is buildable. Calculi are rules for the manipulation of strings of symbols and these rules will not do any calculations unless there is some material apparatus to interpret them. Leave a book on the λ-calculus on a shelf along with a sheet of paper containing a formula in the λ-calculus and nothing will happen. Bring along a mathematician, give them the book and the formula and, given enough paper and pencil, the ensemble can compute. Alternatively, feed the suitably expressed λ-calculus formula into a computer with a Lisp interpreter and it will evaluate.

One has to ask if there exists any possible physical apparatus that can implement the π-calculus and, if there does, is a conventional computer such an apparatus. Since it is possible to write a conventional computer program that will apply the formal term re-write rules of the π-calculus to strings of characters representing terms in the calculus then it would appear that the π-calculus can have no greater computational

power than the von Neumann computer on which the program runs. The language Pict[40] is an example of this. Since it is also possible to implement the λ-calculus in the π-calculus[41] one can conclude that the π-calculus is just one more of the growing family of computational models that are Turing Machine equivalent.

A possible source of confusion is the terminology used to describe the π-calculus - channels, processes, evolution - which imply that one is talking about physically separate but communicating entities evolving in space/time. The π-calculus is intended to be used as a language to describe communicating physically mobile computing machines such as cell phones and their underlying communications networks. As a result there is always a tension between what is strictly laid down as the rules of a calculus and the rather less specific physical system that is suggested by the language used to talk about the calculus.

One has to be very careful before accepting that the existence of the π-calculus as a formal system implies a physically realisable distributed computing apparatus.

Consider two of the primitives: synchronisation and mobile channels. We will argue that each of these faces limits to their physical implementation that prohibits the construction of a super-Turing computational engine based on the the calculus.

6.1. Difficulties in implementing π-calculus synchronisation

It is not clear that π-calculus synchronisation is, in its general sense, physically realistic. First of all, it seems to imply the instantaneous transmission of information, that is faster than light communication, if the processes are physically separated.

Furthermore, if the processors are in relative motion, relativity theory shows that there can be no unambiguous synchronisation shared by the different moving processes. It thus follows that the processors can not be physically mobile if they are to be synchronised with at least 3 way synchronisation (see [42] pp 25-26).

Suppose we have the following pi calculus terms

$$\alpha \equiv (\bar{a}v.Q) + (by.R[y]) \tag{3}$$

$$\beta \equiv (\bar{b}z.S) + (ax.T[x]) \tag{4}$$

In the above α and β are processes. The process α tries to either output the value v on channel a or to read from channel b into the variable y. The $+$ operator means non deterministic composition, so $A + B$ means that either A occurs or B occurs but not both. The notation $\bar{a}v$ means output v to a, whilst av would mean input from a into v. If α succeeds in doing an output on channel a it then evolves into the abstract process Q; if alternatively, it succeeds in doing an input from b into y, then it evolves into the process $R[y]$ which uses the value y in some further computation.

We can place the two processes in parallel by using the $|$ operator for parallel process composition to form $\alpha|\beta$, which expands to:

$$(\bar{a}v.Q) + (by.R[y])|(\bar{b}z.S) + (ax.T[x]) \tag{5}$$

This should now evolve to

$$(Q|T[v])\mathbf{or}(S|R[z]) \tag{6}$$

where either Q runs in parallel with $T[v]$ after the communication on channel a or where S runs in parallel with $R[z]$ after the value z was transfered along channel b from process β to process α. The key ideas here are: processes, channels, synchronisation, parallel and non-deterministic composition.

Suppose further that we attempt to implement the synchronisation by a standard 3 wire handshake wherein each channel is represented by:

rqs request to send
ack acknowledge
d data

the protocol is :

Put:

(i) place data on **d** and assert **rqs** then wait for **ack**
(ii) on getting **ack**, negate **rqs** and remove data from **d**
(iii) wait for **ack** to be negated

Get:

(i) wait for **rqs**, then latch **data**
(ii) assert **ack**
(iii) wait for **rqs** to be negated
(iv) negate **ack**

The two processes are shown in Figure 2, with lines representing the wires used to implement the channel. But instead of wires one could think of these as being radio signals each at a different frequency. Let us consider how the time evolution of the processes might proceed. We will indicate times as $t_0, t_1, ...$

t_o process α places data v on line $\mathbf{d}_a$ and asserts $\mathbf{rqs}_a$
t_1 process β places data z on line $\mathbf{d}_b$ and asserts $\mathbf{rqs}_b$
t_2 process β gets the $\mathbf{rqs}_a$
t_3 process α gets the $\mathbf{rqs}_b$
What happens now?

At time t_2 process β has attempted to send on channel b and has got a request to send from channel a, so which of these should it act on?

If it responds to the **rqs** on a with an acknowledge it will be committing itself to evolve to $T[x]$ but it also has an outstanding **rqs** on b. Suppose it commits to $T[x]$ by sending $\mathbf{ack}_a$, then it can ignore any further communications on channel b. This is fine for process β considered in isolation, but poses

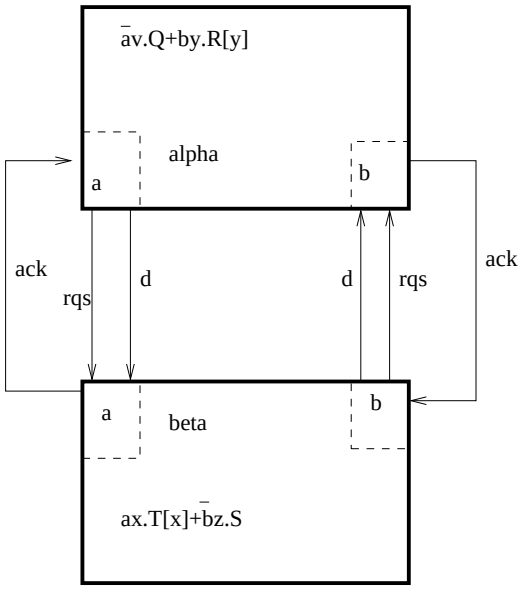

FIGURE 2. Two paradoxical processes.

problems for the other process. Since we are talking about a general implementation strategy, one has to assume that process α will follow the same rule. Thus after getting the request to send on channel *b,* it too will acknowledge, which means that it will commit to continuation $R[y]$. The consequence is that we have evolved to $T[x]|R[y]$, but this is not a permitted transition according to the π-calculus.

Suppose instead that β does not send an $\mathbf{ack}_a$ but instead gives priority to its outstanding request to send on channel *a*. In this case we have to assume that process α will likewise postpone transmission of $\mathbf{ack}_b$, since α is the mirror image of β. It follows that neither process will ever get an **ack**, so they will deadlock.

This was not just a reflection of an inadequacy of the two stage handshake protocol. Since the two processes are identical mirror images of one another, any deterministic rule by which process β commits to communication on one of the channels must cause α to commit to the other channel and hence synchronisation must fail. The argument from the processes α, β is a variant of the Liar Paradox, but it is not a paradox within the π-calculus itself. It only emerges as a paradox once you introduce the constraints of relativity theory prohibiting the instantaneous propagation of information. Nor does abandoning determinism help. If the commitment process is non-deterministic, then on some occasions synchronisation will succeed, but on other occasions the evolution of both processes will follow the same rule, in which case synchronisation will fail.

The arbitration problem is not insoluble. Suppose there was a global arbitration machine. Each process attempting a guarded non-deterministic fork could inform the arbiter of the channel names and direction of communication being tried. Then with knowledge of all outstanding requests for reads and writes, it would probably be possible for the arbiter to apply the reduction rules of the calculus to resolve the synchronisation. However the use of the arbiter machine as a tie breaker removes the parallelism that we want from a distributed version of the calculus, returning us to a sequential centralised evaluation of a key part of the calculus. A worse loss of parallelism is entailed by broadcast protocols such as Asynchronous Byzantine Agreement[43].

In conclusion it is not possible to build a reliable mechanism that will implement in a parallel distributed fashion any arbitrary composition of π-calculus processes.

More tractable systems intellectually derived from the π-calculus can be devised. The $A\pi$ calculus[44] has no synchronisation, and its processes terminate as soon as they output a message. This should be implementable if restrictive. The Java library for Grid computing $J\pi$[45] does away with non-deterministic guards on output but retains the ability to transmit channels over channels. This provides a practical tool for the parallelisation of algorithms in a manner analogous to MPI(Message Passing Interface)[46], PVM(Parallel Virtual Machine)[47, 48] or *IceT*[49]. Parallelisation speeds up sequential code, but it does not allow you to solve problems that are TM uncomputable.

6.2. Difficulties in implementing channels

As with synchronisation, it is not clear how channels may be implemented in terms of physical law. If one has a physical network of processors which do not move physically then one can connect them by wires or fibre optic cables, but these

(i) only allow fixed point to point communication
(ii) are limited in terms of the number of physical wires that can be fed into any given processing unit. Indeed, the number of wires that can be connected to a given chip has always been one of the main limitations on computer chips and remains a substantial technical challenge

Taking into account (i), it is clear that a system using fixed point to point communication can only emulate a system with dynamically created communications channels by using multiplexing and forwarding of messages. From the point of view of computational models it implies that one would have to emulate the π-calculus on the sort of parallel fixed link network that can be described by CSP. Let us refer to a π-calculus system emulated on a fixed link network as Π_{csp}.

If we do not assume wires or optical fibres but instead assume a broadcast network using radio waves like GSM, then one can have physically mobile

processes, but at the expense of removing simultaneous overlapping communication. A system like GSM relies on time multiplexing the radio packets so that in fact only a small finite number of the processes can send messages at any one instant - one process per frequency slot. Of course, GSM relies on base stations to forward packets along fixed point links, but a system like Aloha used basically similar techniques without the base stations.

It is evident that the π-calculus can be used to reason about the behaviour of, and protocols for, phones and other computing devices using radio networks: such problems were a motivation for its design. It would be reasonable to accept that the behaviour of any physical, wireless-linked, computer network can be described in the π-calculus.

However it does not follow from this that there can exist a physically constructable wireless network whose evolution will *directly* emulate that of an arbitrary term in the π-calculus. Because of the exponential decay of signal strength with distance and the finite bandwidth of the radio channel, there are limits to the number of mobile agents that can simultaneously engage in direct communication. One can allow *indirect* communication by partitioning both the bandwidth and the machines, setting aside one group of machines to act as relays and dividing frequency slots between those used by the relay machines and mobile machines. But relying on indirect communication would amount to emulating the π-calculus behaviour in Π_{csp} style. Since the number of directly communicating mobile processes that can operate without relays is modest, and since such a finite network of mobile processes could itself be emulated Π_{csp} style on a finite fixed link network, the computational power of physically realisable systems programed in π-calculus will not exceed that of a formalism like CSP which would render it equivalent to other well known computational models including Turing machines.

6.3. Wegner and Eberbach's argument

Wegner and Eberbach's argument for the super-Turing capacity of the π-calculus rests on there being an implied infinity of channels and an implied infinity of processes. Taking into account the restrictions on physical communications channels the implied infinity could only be realised if one had an actual infinity of fixed link computers. At this point we are in the same situation as the Turing machine tape - a finite but unbounded resource. For any actual calculation a finite resource is used, but the size of this is not specified in advance. Wegner and Eberbach then interpret 'as many times as is needed' in the definition of replication in the calculus as meaning an actual infinity of replication. From this they deduce that the calculus could implement infinite arrays of cellular automata for which they cite Garzon [50] to the effect that the calculi are more powerful than TMs.

We undercut this argument at two points:

(i) We have shown that the synchronisation primitive of the calculus is not physically realistic. The modelling of cellular automata in the calculus rests on this primitive.
(ii) The assumption of an infinite number of processes implies an infinity of mobile channels, which are also unimplementable.

We therefore conclude that whilst the π-calculus can be practically implemented on a single computer, infinite distributed implementations of the sort that Wegner and Eberbach rely upon for their argument cannot be implemented.

It is important to emphasise that, just as we are untroubled by an unknown but bounded TM tape, we have no concerns about the deployment of an unknown but bounded number of processes in the π-calculus. However, Wegner and Eberbach are unclear as to whether they mean this or a completed infinity of processes, which we think physically impossible.

7. $-CALCULUS

7.1. Introduction

Eberbach's $-calculus ("cost" calculus)[51, 52] is based on a process algebra extended with cost operators. The $-calculus draws heavily on the π-calculus and on interaction machines: thus the critiques of these above also apply to the $-calculus. However, Wegner and Eberbach claim that: "The unique feature of the $-calculus is that it provides a support for problem solving by incrementally searching for solutions and using cost to direct its search."(p7). Furthermore: "The $-calculus allows in a natural way to express evolution."(p7). Given the important role of evolution as one of the domains that Wegner and Eberbach claim transcends the Church-Turing paradigm, let us first examine Eberbach's argument that evolutionary computing is not algorithmic before considering the $-calculus itself in more detail.

7.2. Evolution and Effective Computation

In [53], Eberbach characterises an evolutionary algorithm (EA) as involving the repeated selection from a population under some selection operation for reproduction that introduces variations through mutation and crossover. When some selection threshold is reached, the final population is deemed to be optimal. We agree with this characterisation but note that thus far it corresponds to Kleene's unbounded minimisation i.e. general recursion. Indeed, subsequently Eberbach states that every EA can be encoded as a TM whose tape is the initial population.

Eberbach goes on to elaborate a hierarchy of EAs and corresponding TMs. The Universal Evolutionary Algorithm (UEA) consists of all possible pairs of EA TMs and initial population tapes. An Evolutionary Turing Machine (ETM) is a potentially infinite sequence of pairs of EA TMs and population tapes, where each pair was evolved in one generation from the previous pair subject to a common fitness (selection) operator.

Eberbach claims that evolution is an infinite process because the fitness operator is part of the TM and evolves along with it. This seems unremarkable: it is well understood that a Universal TM may execute a TM that modifies its own encoding. Hence, the TM's termination condition may change and may never be achieved.

Eberbach then makes the apparently stronger claim that EAs are a superset of all algorithms but this is either unremarkable or misleading. EAs are generalised unbounded minimisation and so are expressible as general recursion or TMs. Given that any effective computation can be captured as a TM or through general recursion, it seems plausible that any effective computation can be evolved. However, it is not at all clear how one would define a selection operator to decide if a required effective computation had indeed been achieved. As noted above, the equivalence of TM is undecidable so even if one could specify the required effective computation as another TM there is no effective method for proving that an arbitrary evolved TM is equivalent to the specification.

Furthermore, for EAs to be a superset of all algorithms there must be something in the set of EAs which is not itself an algorithm. However, Eberbach says that all EAs can themselves be encoded as TMs so all EAs must be algorithms. Thus, it seems more likely that the set of algorithms is at least as big as the set of all EAs.

Finally, Eberbach introduces the Universal Evolutionary Turing Machine (UETM) which takes an ETM as its initial tape. He states plausible theorems that the UETM halting problem is unsolvable by the UTM, and that the UTEM cannot solve its own halting problem. Eberbach speculates that ETM can be understood as a special case of the Persistent Turing Machine. Thus the ETM must answer the critique of Persistent Turing Machines made above.

Eberbach goes to to enunciate two more theorems. First of all he claims that the UTM halting problem is solvable by the ETM using the "infinity principle" where "Fitness is defined to reach optimum for halting instances.". As noted previously, we do not think it possible to construct an effective computation for such a fitness function. He then claims that the UTM halting problem is solvable by ETM using evolution through a TM augmented with an oracle. This argument again seems plausible though curious given that Eberbach and Wegner say that the $-calculus does not gain its power from an oracle (p7). However, as discussed above, if an ETM is a TM with an oracle then ETMs are not effectively computable and in general are not materially realisable.

7.3. $-calculus and Expressiveness

In [51], Eberbach explores the expressiveness of the $-calculus. First of all, he shows that the λ-calculus and the π-calculus may both be simulated in the $-calculus. He claims that: "the λ-calculus is a subclass of the $-calculus, because of the one-to-one correspondence between reductions in λ-terms and in their corresponding $-calculus terms."(p5) and that: "π-calculus could be claimed to be a subclass of the $-calculus, because each operator of π-calculus is simulated by a corresponding operator(s) from $-calculus."(p5) We dispute the claimed subclass relationship but note that this is not expressed in (1) below and do not consider this further.

Next, citing Milner's proof that λ-calculus may be simulated by π calculus[54], and drawing implicitly on the C-T equivalence of λ-calculus and TMs, he enunciates a hierarchy:

(1) $TM \subseteq \pi C \subseteq \$C$

We note the use of $\subseteq$ rather than $\subset$ in $\pi C \subseteq \$C$. Certainly, Eberbach does not substantiate a strong hierarchy ($\subset$) at this stage by demonstrating a $-calculus term that cannot be expressed as a π-calculus term.

Next, Eberbach says that Sequential Interaction Machines (SIMs) have less expressive power than Multi-stream Interaction Machines (MIMs), and cites Wegner's claim[55] that π-calculus has only the same expressive power as SIMs, giving the additional hierarchy:

(2) $\pi C \subseteq SIM \subset MIM$

Finally, he sketches how MIMs may be simulated by $-calculus (MIM $\subseteq$ $C), which combined with (1) and (2) gives:

(3) $TM \subseteq \pi C \subseteq SIM \subset MIM \subseteq \$C$

Note that (3) greatly strengthens the $-calculus's position in the hierarchy relative to C-T systems: (1) says that πC $\subseteq$ $C but (3) now implies that πC $\subset$ $C through transitivity of $\subseteq$ and $\subset$. Thus, the $-calculus term which is not expressible in π-calculus may come from an instance of MIM, but this is not displayed.

7.4. $-calculus and costs

As noted above, the $-calculus[52] is characterised by the integration of process algebra with cost functions derived from von Neumann/Morgenstern utility theory. As well as sequential and parallel composition, and inter-$-expression communication, cost choice and mutating send constructs are also

provided. Rather than having a unitary expression form, the $-calculus distinguishes between composite (interruptible) expressions, and simple (contract) expressions which are considered to be executed in one atomic indivisible step. While cost choices are made in composite expressions and costs communicated in simple expressions, they may be defined and evaluated in both layers.

Composition and choice are over countably infinite sets of $-expressions, which Eberbach claims as one locus of the $-calculus's increased expressive power. However, λ-calculus is also capable of expressing dynamically changing, arbitrary width and depth nesting of functions; the Y fixed-point finder being a classic example. $-calculus is also supposed to support true parallelism; the implications for effective computation have been discussed above.

Costs are asserted as a central locus of the $-calculus's strength. While it does not prescribe a base set of cost functions, crisp (i.e. algorithmic), probabilistic and fuzzy functions have all been developed. Users may also define their own cost functions. However, it is not clear how costs actually extend the $-calculus's expressive power. Without costs, in particular mutating send, the $-calculus is reminiscent of a higher-order process algebra, which as Eberbach notes[51] is no more powerful than a first order system. To add power to the $-calculus, user defined costs must be crafted in some formalism other than the $-calculus itself, or built from base cost functions, where the other formalism or the base functions are themselves more powerful than anything expressible by a either a C-T system or cost-less $-calculus. Either way, cost choice over any one bounded set of cost values and mutation over a bounded set of $-expressions are both C-T. Choice over an infinite set of cost values seems deeply problematic, where even an approximation ultimately involves the expansion of all possible execution traces of the invoking program.

Finally, the operational semantics for $-calculus are defined "in a traditional way for process algebras"(section 3) using inference rules and a labelled transition system (LTS), where the LTS always looks for a least cost action. However, inference rules and LTS are no more powerful than C-T systems so either the $-calculus is a C-T system or the semantics does not capture the full expressive power of the $-calculus. In the latter case, it is not clear how the meaning of $-calculus programs may actually be formalised or implemented.

8. CONCLUSION

8.1. Unbounded tapes and actualised infinities

It might be thought that we are guilty of special pleading in allowing TMs an unbounded tape but criticising accelerating TMs, π calculus and cellular automata for requiring infinite resources to accomplish tasks in a finite time. However, the explicit distinction between *infinite* and *unbounded* is fundamental to our understanding of effective computability.

The amount of tape required for an arbitrary TM starting over an arbitrary tape, is, of course, necessarily unknown and in that sense is unbounded at the outset of the computation. Furthermore, in principle, some computations would consume more tape than can be met by the sustained output of the even most rapacious pan-galactic memory manufacturer, chewing their way through space to feed the gaping maw of a truly universal computer. Nonetheless, if the computation is effective then, by definition, the TM must terminate having performed a finite number of state transitions, visiting at most one tape position on each, and hence consuming a potentially unrealisable but actually finite amount of tape.

In contrast, several hypercomputation systems appear to necessarily require an *actualised infinity* of physical resource, to perform computations deemed non-effective by the classic theory. Thus, where in practise effective computations on TMs may eventually run out of physical resource, such hypercomputations schemes seem in principle unable to allocate the requisite infinite resources from the outset.

Note that we have nowhere referred to a TM tape as infinite. In particular, in his 1936 paper, Turing says simply that "The machine is supplied with a tape", without specifying that the tape be infinite. However, there has long been a popular perception that TMs require explicitly infinite tapes. This may have arisen from confusion with Post's formulation of computability[56], where a *symbol-space* is explicitly defined as a "two-way infinite sequence of spaces or boxes". Davis[57] says that Post's work was independent of Turing's.

It is germane that Kleene, in his 1952 account[58] of Turing machines, refers to a TM being: "... supplied with a linear *tape*, (potentially) infinite in both directions..."(p357). However, eight years later Davis[59] refers to: "...a linear tape, assumed to be *infinite* in both directions..."(p3) without further qualification. This might be an interesting avenue for further historical research.

8.2. Summary

In Section 2.5, we enunciated the criteria that we think must be met for the Church-Turing thesis to be displaced. In general, we require a demonstration that all terms in C-T systems should have equivalent terms in the new system but there should be terms in the new system which do not have equivalents in C-T systems. In particular, the new system should be able to solve decision problems that are semi-decidable or undecidable in C-T systems. Finally, we require that a new system be physically realisable. We think that, under these criteria, Wegner and Eberbach's claims that

Interaction Machines, the π-calculus and the $-calculus are super-Turing are not adequately substantiated.

First of all, Wegner and Eberbach do not present a concrete instance of terms in any of these three systems which do not have equivalents in C-T systems. Secondly, they do not identify decision problems which are decidable or semi decidable in any of these systems but semi-decidable or undecidable respectively in C-T systems. Finally, they do not explain how an arbitrary term of any of these three systems may be embodied in a physical realisation.

Wegner and Eberbach make bold claims in their paper. But extraordinary claims require extraordinary evidence. The work of Turing has served as a foundation for computability theory for 70 years. To displace it would have required them to bring forward very strong evidence. We have discussed the criteria by which such claims could be assessed, and we have discussed the systems that they have exhibited as potentially surpassing the Turing Machine model of computation. We consider that in all three cases, Interaction Machines, the π Calculus and the $ Calculus, we have shown their claims to be invalid.

ACKNOWLEDGEMENTS

We wish to thank Eugene Eberbach for providing us with copies of his papers.

Greg Michaelson wishes to thank the School of Information Technology at the University of Technology, Sydney for their hospitality during the writing of this paper.

We wish to thank Lewis Mackenzie, Johannes Courtal, and Oliver Penrose for their assistance in checking the validity of our physical arguments.

REFERENCES

[1] Turing, A. M. (1936) On Computable Numbers, With an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **42**, 230–65.

[2] Kuhn, T. (1970) *The Structure of Scientific Revolutions*. University of Chicago Press.

[3] Ekdahl, B. (1999) Interactive Computing does not supersede Church's thesis. *Proc. Computer Science*, 17th Int. Conf. San Diego, pp. 261–265, Association of Management and the International Association of Management,.

[4] Church, A. (1936) An unsolvable problem of elementary number theory. *American Journal of Mathematics*, **58**, 345–363.

[5] Kleene, S. C. (1935) General recursive functions of natural numbers. *American Journal of Mathematics*, **57**, 727–742.

[6] Turing, A. M. (1939) Systems of logic based on the ordinals. *Proceedings of the London Mathematical Society*, **45**, 161–228.

[7] Turing, A. M. (1950) Computing Machinery and Intelligence. *Mind*, **LIX**, 433–460.

[8] Hayes, I. and Jones, C. (1989) Specifications are not (necessarily) executable. *Software Engineering Journal*, **4**, 330–338.

[9] Davis, M. (1958) *Computability and Undecidability*. McGraw Hill.

[10] Cook, S. (1971) The Complexity of Theorem Proving Procedures. *Proceedings of Third Annual ACM Symposium on Theory of Computing*, May, p. 151..158, ACM.

[11] Althusser, L. and Balibar, É. (1970) *Reading Capital*. New Left Books.

[12] Lassegue, J. (2002) Turing, entre formel et forme ; remarques sur la convergence des perspectives morphologiques. *Intellectica*, **2002/2**, 185–198.

[13] Hussey, E. (ed.) (1983) *Aristotle's Physics*. Clarendon.

[14] Copeland, J. (2002) Accelerated Turing Machines. *Minds and Machines*, **12**, 281–301.

[15] Hamkins, J. D. (2002) Infinite Time Turing Machines. *Minds and Machines*, **12**, 521–539.

[16] Copeland, J. and Sylvan, R. (1999) Beyond the universal Turing machine. *Australasian Journal of Philosophy*, **77**, 46..66.

[17] Bournez, O. and Cosnard, M. (1995) On the computational power and super-Turing capabilities of dynamical systems. Tech. Rep. 95-30, Ecole Normal Supérior de Lyons.

[18] Koiran, P. and Moore, C. (1999) Closed-form analytic maps in one and two dimensions can simulate Turing machines. *Theoretical Computer Science*, **210**, 217–223.

[19] Turing, A. M. (2004) Lecture on the Automatic Computing Engine, 1947 . Copeland, B. J. (ed.), *The Essential Turing*, OUP.

[20] Hodges, A. (1983) *Alan Turing the enigma*. Touchstone.

[21] Landauer, R. (1961) Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, **5**, 183–91.

[22] Landauer, R. (1992) Information is Physical. *Physics and Computation, 1992. PhysComp'92., Workshop on*, pp. 1–4.

[23] Deutsch, D. (1985) Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, **400**, 97–117.

[24] Feynman, R. P. (2002) There's plenty of room at the bottom. Hey, A. (ed.), *Feynman and Computing*, Westview Press.

[25] Feynman, R. P. (1999) Simulating physics with computers. Hey, A. (ed.), *Feynman and computation: exploring the limits of computers*, pp. 133–153, Perseus Books.

[26] Kieu, T. D. (2003) Quantum algorithm for Hilbert's tenth problem. *International Journal of Theoretical Physics*, **42**, 1461 – 1478.

[27] Smith, W. D. (2006) Three counterexamples refuting Kieu's plan for "quantum adiabatic hypercomputation" and some uncomputable quantum mechanical tasks. *J.Applied Mathematics and Computation*, **187**, 184–193.

[28] Hodges, A. (2005) Can quantum computing solve classically unsolvable problems? Tech. Rep. quant-ph/0512248, arXiv.org.

[29] Davis, M. (2006) The Church-Turing Thesis: Consensus and Opposition. Beckmann, A., Berger, U., Lowe, B., and Tucker, J. V. (eds.), *Logical Approaches to Computational Barriers: Second Conference on Computability in Europe, CiE 2006, Swansea, UK*, June/July, pp. 125–132, no. 3988 in LNCS, Springer.

[30] Kieu, T. D. (2006) Reply to Andrew Hodges. Tech. Rep. quant-ph/0602214, arXiv.org.

[31] Kieu, T. D. (2006) On the identification of the ground state based on occupation probabilities:. Tech. Rep. quant-ph/0602145, arXiv.org.

[32] Adamyan, V., Calude, C., and Pavlov, B. (2003) Transcending the Limits of Turing Computability. *Arxiv preprint quant-ph/0304128*.

[33] Calude, C. and Pavlov, B. (2002) Coins, Quantum Measurements, and Turing's Barrier. *Quantum Information Processing*, **1**, 107–127.

[34] Wegner, P. and Eberbach, E. (2004) New Models of Computation. *Computer Journal*, **47**, 4–9.

[35] Goldin, D., Smolka, S., Attie, P., and Sonderegger, E. (2004) Turing machines, transition systems, and interaction. *Information and Computation*, **194**, 101–128.

[36] Wegner, P. (1997) Why interaction is more powerful than algorithms. *Communications of the ACM*, **40**, 80–91.

[37] Cockshott, P., Atkinson, M., Chisholm, K., Bailey, P., and Morrison, R. (1984) POMS- a persistent object management system. *Software Practice and Experience*, **14**, 49..71.

[38] Cockshott, P. (1982) *Orthogonal Persistence*. Ph.D. thesis, University of Edinburgh Dept of Computer Science.

[39] Chaitin, G. (1987) *Information, Randomness and Incompleteness*. World Scientific.

[40] Turner, D. and Pierce, B. (2000) Pict: A programming language based on the π-calculus. Plotkin, G., Stirling, C., and Tofte, M. (eds.), *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pp. 455–494, MIT Press.

[41] Milner, R. (1993) Elements of interaction: Turing award lecture. *Communications of the ACM*, **36**, 78–89.

[42] Einstein, A. (1920) *Relativity*. Methuen and Company.

[43] Bracha, G. and Toueg, S. (1983) Asynchronous consensus and byzantine protocol in a faulty environment. Tech. Rep. TR-83-559, CS Dept., Cornell University, Ithaca,NY 14853.

[44] Sangiori, D. and Walker, D. (2001) *The π-calculus*. Cambridge University Press.

[45] Yarmolenko, V., Cockshott, P., Borland, E., and Mackenzie, L. (2004) Jπ INTERFACE: A Java Implementation of the π-Calculus for Grid Computing. *Proceedings Middleware Grid Conference*, October.

[46] Walker, D. and Dongarra, J. (1996) MPI: A Standard Message Passing Interface. *Supercomputer*, **12**, 56–68.

[47] Ferrari, A. (1998) JPVM: network parallel computing in Java. *Concurrency Practice and Experience*, **10**, 985–992.

[48] Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1995) *PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing*. MIT Press Cambridge, MA, USA.

[49] Gray, P. and Sunderam, V. (1997) IceT: distributed computing and Java. *Concurrency - Practice and Experience*, **9**, 1161–1167.

[50] Garzon, M. (1995) *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*. EATCS, Springer-Verlag.

[51] Eberbach, E. (2000) Expressiveness of $-calculus: What matters? Klopotek, M., Michalewicz, M., and Wierzchon, S. T. (eds.), *Advances in Soft Computing,*, pp. 145–157, Physica-Verlag.

[52] Eberbach, E. (2001) $-calculus bounded rationality = process algebra + anytime algorithms. J.C.Misra (ed.), *Applicable Mathematics: Its Perspectives and Challenges*, pp. 213–220, Narosa Publishing House.

[53] Eberbach, E. (2002) On Expressiveness of Evolutionary Computation: Is EC Algorithmic? *Proc. 2002 World Congress on Computational Intelligence WCCI'2002*, pp. 564–569.

[54] Milner, R. (1992) The polyadic π-calculus: A tutorial. Bauer, F. L. and Brauer, W. (eds.), *Logic and Algebra of Specification*, Springer-Verlag.

[55] Wegner, P. (1997) Interactive software technology. Tucker, A. B. (ed.), *The Computer Science and Engineering Handbook*, CRC Press.

[56] Post, E. L. (1936) Finite Combinatory Processes. Formulation 1. *Journal of Symbolic Logic*, **1**, 103–105.

[57] Davis, M. (1965) *The Undecidable*. Raven Press.

[58] Kleene, S. (1952) *Introduction to Metamathematics*. North-Holland.

[59] Davis, M. (1958) *Computability and Unsolvability*. McGraw-Hill.