ARE THERE NEW MODELS OF COMPUTATION: REPLY TO WEGNER AND EBERBACH

PAUL COCKSHOTT AND GREG MICHAELSON

ABSTRACT. Wegner and Eberbach[Weg04b] 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 Weger and Eberbach claims.

1. INTRODUCTION

The concept of a Turing machine[Tur36a] 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[Kuh62], the Turing machine (TM) has been the dominant paradigm for Computer Science for 70 years: Ekdahl[Ek99] 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[Chu36] and Kleene[Kle35] 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, the von Neumann architecture of everyday digital computers has long been known to satisfy the Thesis, 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.(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... ([Tur36a] Section 1).

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

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. ([Tur39] 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. ([Tur50]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[Hay89] have argued that there is a clear distinction between expressive power and implementability. In specification notations like Z and VDM, 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 necesarily executable.

We note here such that 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 [Dav58]. 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 desigated 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. 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 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.

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 inherentaly 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 reducable 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 guess some fundamental property of a problem. It is important to note that oracles are not effectively calculable. In[Tur39], 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[Cook71] established that it is possible to identify NP Complete (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.

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 accolytes 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 [Kuh62] which slowly accretes new self-validating knowledges in its own terms, without worrying too much about apparently minor discrepancies. Thus a successfull 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 [1]. More concretely, we think that requirements for a new system to conclusively transcend C-T are, in increasing order of strength:

(1) demonstration that some problem known to be semi-decidable in a C-T system is decidable in the new system;
(2) demonstration that some problem known to be undecidable in a C-T system is semi-decidable in the new system;
(3) demonstration that some problem known to be undecidable in a C-T system is decidable in the new system;
(4) characterisations of classes of problems corresponding to 1-3;
(5) canonical exemplars for classes of problems corresponding to 1-3.

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

[1]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 disjoint expressive power.

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. Wegner and Eberbach's SuperTuring Computers

Wegner and Eberbach[Weg04b] 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 iteself. 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 evolutiuonary algorithms are probabalistic the search may take an infrinite 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 compuations 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.

4. Physical Realism and Computation

Turing marks what Bachelard and Althusser[Bal78] termed an Epistemological break in the history of the sciences. At once the problematic of Hilbertian rationalism is abandoned [Las02] 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 the computation is carried out on onedimensional paper, i.e. on a tape divided into squares.([Tur36a] section 9)

However he 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."

He thus rules out any consideration that computation by inifinite means is a serious proposition. In this Turing follows Aristotle ([Ari83] Book III, chap 6) in allowing potential but not actual infinities. If inifinite computation were to be allowed, then the limitations introduced by the Turing machine would not apply. Most proposals for superTuring computation rest on the appeal of the infinite.

Copeland[Cop02] proposes the idea of accelerating Turing machines whose operation rate increases exponentially so that if the first operation were perfomed 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 inifinite 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 traveling 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[Ham00] 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.

Another theme of those advocating Super-Turing computation is the use of analogue computation over real numbers. For a review see [Cop99]. 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 appropriately clunky mechanical example, in keeping with the mechanical nature of the Turing Machine. Representations of reals using electric charge would encounter similar obstacles since h constrains the accuracy of all complementary measurements.

Suppose we want to use analoge encoding, representing real numbers directly as spatial separation, materialising the mathematical abstraction of the real number line using an apparattus such as that shown in Figure 1. The intention would be to encode a real number x so that it could be used by some Super Turing analogue computational process. If x is to be used as a computational parameter, we must first set up the the distance x and then measure it during the process of the computation. The position of the slider acts as an analogue memory holding a real number. This raises two questions:

(1) How precisely can we, in principle at least, measure the distance x?

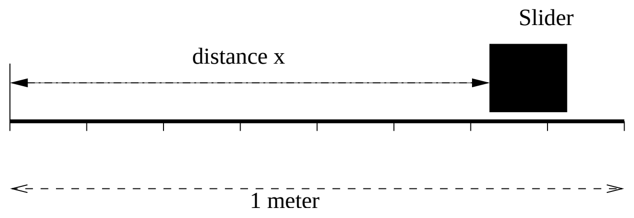

FIGURE 1. An analogue representation of a real number using a physical version of the real number line.

(2) How stable is such an analogue memory. For long can it store the information.

Because of Heisenberg's equation

$$\Delta p \Delta x = \frac{h}{4\pi}$$

there is a tradeoff between the accuracy to which x can be represented as distance and the period for which x can be stored. If the mass of the slider is m the uncertainty in its velocity is given by $\Delta v = \frac{h}{4\pi m \Delta x}$. This implies that the persistence time of the analogue store T_p is constrained such that

$$T_p \approx \frac{\Delta x}{\Delta v} = \frac{4\pi m \Delta x^2}{h}$$

It is clear that for a practical device, T_p must be chosen exceed the time taken to measure the distance x. This in turn, must be greater than $\frac{2x}{c}$, since any distance measurement will involve at least one reflection off the slider. If $T_p < \frac{2x}{c}$ then the computer would be unable to read the real. We thus have the constraint

$$\frac{2x}{c} < \frac{4\pi m \Delta x^2}{h}$$

which implies

$$\Delta x^2 > \frac{2xh}{4\pi mc}$$

For a 1 kilo slider adjustable over a range of 1 meter which allows the representation of reals in the range 0..1 as $x \pm \Delta x$, we find that $\Delta x > 5.93 \times 10^{-22}$meters.

This corresponds to a real number stored with about 70 bits of precision.

To add an additional bit of precision to our real number we would have to quadruple the mass of the slider. We find therefore, that the analogue encoding of the reals requires a mass which grows with $\mathbf{O}e^{2b}$ where b is the number of bits of precision to which the reals are stored. Contrast this to a digital computer like a Turing Machine where the mass required to record a real number grows linearly with the number of bits - each requiring an additional cell on the tape. Turing himself pointed this out in 1947:

> 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. [Tur47]

Thus :

(1) Any physically buildable analogue memory will only approximate the reals.
(2) Analogue storage of reals, will for high precision work, always be outperformed in terms of device economy by digital storage.
(3) Physically buildable analogue computers can not rely upon the availablility 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. One also has to ask just how one is to program a machine requiring infinite precision reals in the first place. How, by finite means, is one to specify the infinitely precise analogue numbers that are needed?

For Turing, Universal Computers were not mere mathematical abstractions but concrete possibilities, which he pursued in his designs for the Bombe at Bletchley and ACE at the National Physical Laboratory[Tur47, Hod83]. Computing Science has undergone great progress in the period since Turing's original paper. This progress has synthesised abstract computational theory with research into possible physical implementations of digital computers. An enormous 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[Lan61, Lan91, Lan00] has examined the fundamental thermodynamic limitations of computation, and Deutsch[Deu85] and Feynman[Fey59, Fey82] 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.

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 having to have all its inputs 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 Interaction Machines and Turing Machines, they claim, corresponds to the technology shift from mainframes to workstations. Interaction Machines, whose cannonical model is the Persistent Turing Machine(PTM) of Goldin [Gol01], 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[Weg97a], some of whose main arguments have been criticised by Ekdahl[Ek99]. Rather than rehearse his arguments 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[Tur50], 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 recieves 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. (Turing 1950, page 436)

This is of course a paraphrase of his description of the computing machine in his 1936 paper 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 as he had in 1936. With the practical experience of work on the Manchester Mk1 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.(Turing 1950, page 442)

This is clearly a recapitulation of the argument in section 6 of his 1936 paper 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 the class of machines which he had introduced in 1936 had all of the properties that Wegner and Goldin were later to claim for to 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 forseen by Turing in his 1950 paper, 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 is 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 working life (for all computers operate only for a finite time), 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 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 this misrepresents the role played by the idea of persistence in the history of computing. 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[Coc82, Coc84]. 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-Neuman.

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 it's 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_2now 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 PTM's, and thus Interaction Machines of which they are the cannonical example, are a sub-class of TM programs and do not represent a new model of computation.

5.3. **Thermodynamic considerations.** Chaitin introduced algorithmic information theory [Cha99] 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 attempts 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.

To the extent that the distinction being made by W&E between TM's and Interaction Machines parallels this information theoretic point it has some practical validity.

6. π-Calculus

Wegner and Eberbach give the π-alculus as one 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 apparattus 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 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 is such an apparattus, is a conventional computer such an apparattus. 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 computer runs. The language Pict[Tur00] is an example of this. Since it is also possible to implement the λ-calculus in the π-calculus[Mi93] 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 language 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.

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

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.**

Is π-calculus synchronisation in its general sense physically realistic?

Does it not imply the instantaneous transmission of information - faster than light communication if the processes are physically separated?

Further, if the processors are in relative motion, relativity theory shows that there can be no unambigous 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 [Ein20] pp 25-26).

Suppose we have the following pi calculus terms

$$\alpha \equiv (\bar{a}v.Q) + (by.R[y]) \tag{1}$$

$$\beta \equiv (\bar{b}z.S) + (ax.T[x]) \tag{2}$$

In the above α and β are processes. The process αtries 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 thus: $\alpha|\beta$, which expands to:

$$(\bar{a}v.Q) + (by.R[y])|(\bar{b}z.S) + (ax.T[x]) \tag{3}$$

This should now evolve to

$$(Q|T[v])\mathbf{or}(S|R[z]) \tag{4}$$

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 βto 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:

(1) place data on **d** and assert rqs then wait for **ack**
(2) on getting **ack**, negate **rqs** and remove data from **d**
(3) wait for **ack** to be negated

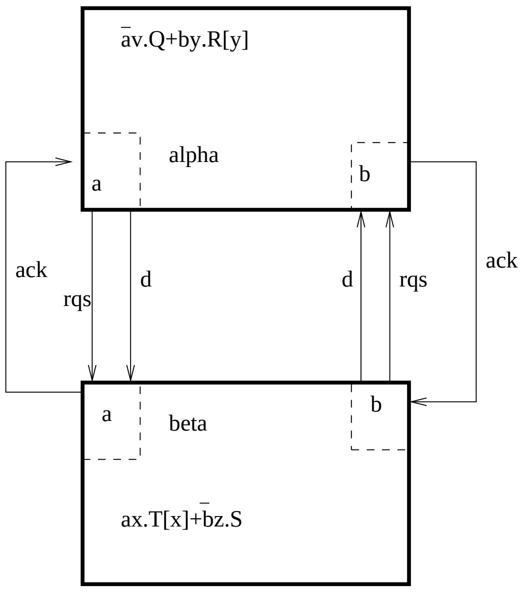

FIGURE 2. Two paradoxical processes.

Get:

(1) wait for **rqs**, then latch **data**
(2) assert **ack**
(3) wait for **rqs** to be negated
(4) 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, which of these should it act on?

If it responds to the **rqs** on a with and acknowledge it will be commiting 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 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 permited 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, what will happen then?

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.

It might be thought that this was just a reflection of an inadequacy of the two stage handshake protocol, but it is not. 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 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[Bra83].

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[Wal01] has no synchronisation, and it's processes terminate as soon as they output a message. This should be implementable if restrictive. The Java library for Grid computing $J\pi$[Yar04] does away with non-deterministic guards on output but retains the ability to transmit channels over channels. This provides a practical tool for the prallelisation of algorithms in a manner analogous to MPI[MPI94], PVM[Fer98, Ge94] or IceT[Gra97]. Parallelisation speeds up sequential code, it does not allow you to solve problems that are TM uncomputable.

6.2. **Difficulties in implementing channels.** How are channels to be implemented in terms of physical law?

If one has a physical network of processors which do not move physically then you can connect them by wires or fibre optic cables, but these only

(1) allow fixed point to point communication
(2) are limited in terms of the number of physical wires that can be fed into any given processing unit. This has indeed always been one of the main limitations on computer chips - how many wires can you connect to a given chip -this has been a technical challenge

Taking into account (1), 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 any phyisical, wireless-linked, computer network can be described in the π-calculus.

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 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'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 acount 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.W&E then interprets 'as many times as is needed' in the definition of replication in the calculus as meaning an actual infinity of replication. From this he deduces that the calculus could implement infinite arrays of cellular automata for which he cites Garzon [Gar95] to the effect that they are more powerful than TMs.

We undercut this argument at two points:

(1) 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.
(2) 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 W&E rely upon for their argument can not be implemented.

7. $-CALCULUS

7.1. **Introduction.** Eberbach's $-calculus ("cost" calculus)[Ebe00, Ebe01] 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 [Ebe02], 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 threshhold 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 potentialy 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 [Ebe00], 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[Mil92], and drawing implictly on the C-T equivalence of λ-calculus and TMs, he enuciates a hierarchy:

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

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[Weg97b] that π-calculus has only the same expressive power as SIMs, giving the additional hierarchy:

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

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

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

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[Ebe01] 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 exectued in one atomic indivisble step. While cost choices are made in composite expresssions 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 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), probabalisitic 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[Ebe00] 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. If the $-calculus is more powerful than C-T systems but is definable using inference rules and a LTS, that implies that the latter also have more expressive power than C-T systems.

8. Conclusion

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 almost 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 assesed, 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.

The Church-Turing thesis remains, for now, secure.

Acknowledgements

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.

References

[Ari83] Aristotle, Aristotle's Physics, books III and IV, trans Edward Hussey, Clarendon, Oxford, 1983.

[Bal78] E Balibar, From Bachelard to Althusser: The Concept of" Epistemological Break, - Economy and Society,Vol. 7, No. 3, pages 207-237, 1978

[Bra83] G. Bracha and S. Toueg Asynchronous consensus and Byzantine protocol in faulty environment TR-83-559, CS Dept., Cornell University, Ithaca, NY 14853, 1983.

[Cha99] G. Chaitin, Information and Randomness: A survey of Algorithmic Information Theory, in G. Chaitin, *The Unknowable,* Springer, Signapore, 1999.

[Chu36] A. Church, An Unsolvable Problem of Elementary Number Theory, American Journal of Mathematics, Vol. 58, No. 2, pages 345-363, 1936.

[Coc82] P. Cockshott, Orthogonal Persistence, PhD Thesis, University of Edinburgh, 1982.

[Coc84] P. Cockshott, M. Atkinson, K. Chisholm, P. Bailey, R. Morrison. POMS - a persistent object management system, Software Practice and Experience, Vol. 14, No 1, pages 49- 71, 1984.

[Cook71] S. Cook, The Complexity of Theorem Proving Procedures, Proceedings of Third Annual ACM Symposium on Theory of Computing, pp 151-158, May 1971.

[Cop99] J. Copeland and R. Sylvan, Australasian Journal of Philosophy, vol. 77 (1999), pp. 46-66.

[Cop02] J. Copeland, Accelerated Turing Machines, Minds and Machines, Vol. 12, pages 281-301, 2002.

[Dav58] M. Davis, Computability and Undecidability, McGraw-Hill, 1958.

[Deu85] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, *Proceedings of the Royal Society of London* A 400, pp. 97-117, 1985.

[Ebe00] E. Eberbach, Expressiveness of $-Calculus: What Matters?, in: Advances in Soft Computing, in: (eds. M.Klopotek, M.Michalewicz, S.T.Wierzchon) Adances in Soft Computing, Proc. of the 9th Intern. Symp. on Intelligent Information Systems IIS'2000, Bystra, Poland, Physica-Verlag, 2000, 145-157.

[Ebe01] E. Eberbach, $-Calculus Bounded Rationality = Process Algebra + Anytime Algorithms, in: (ed.J.C.Misra) Applicable Mathematics: Its Perspectives and Challenges, Narosa Publishing House, New Delhi, Mumbai, Calcutta, 2001, 213-220.

[Ebe02] E. Eberbach, On Expressiveness of Evolutionary Computation: Is EC Algorithmic?, Proc. 2002 World Congress on Computational Intelligence WCCI'2002, Honolulu, HI, 2002, 564-569.

[Ein20] A. Einstein, Relativity, Methuen and Company, London, 1920.

[Ek99] B. Ekdahl, Interactive Computing does not supercede Church's thesis, Association of Management and the International Association of Management, 17th Int. Conf. San Diego, Proc. Computer Science, Vol 17, No. 2, Part B, pages 261-265,1999.

[Fer98] A. Ferrari, JPVM: Network Parallel Computing in Java, ACM 1998 Workshop on Java for High Performance Network Computing, 1998.

[Fey59] R. Feynman, There's plenty of Room at the Bottom, lecture to the American Physical Society, 1959, reprinted in in Hey, Antony, ed. *Feynmann and Computing* Westview Press, Oxford, 2002.

[Fey82] R. Feynman, Simulating Physics with Computers, *International Journal Theoretical Physics*, Vol 21, Nos. 6/7, (1982), reprinted in in Hey, Antony, ed. *Feynmann and Computing* Westview Press, Oxford, 2002.

[FLP] M.Fischer, N. Lynch, M. Paterson, Impossibility of distributed consensus with one faulty process, Proc. 2nd ACM SIGACT-SIGMOD Symposium on principles of database systems, 1982.

[Gar95] M. Garzon, Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks, An EATCS series, Springer-Verlag, Berlin, Heidelberg, New York, 1995.

[Ge94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press, 1994.

[Gol01] D. Goldin, S. Smolka and P.Wegner, Turing Machines, Transition Systems, and Interaction, in Proc. 8th Int. Workshop on Expressiveness in Concurrency, Aalborg, Denmark, 2001.

[Gra97] P. Gray, V. Sunderam, IceT Distributed Computing and Java, Concurrency, Practice and Experience, Vol. 9, No. 11, Nov. 1997.

[Ham00] J. Hamkins and A. Lewis, Infinite time Turing machines, The Journal of Symbolic Logic, Vol. 65(2), pages 567-604, 2000.

[Hay89] P. Hayes and C. B. Jones, Specications are not (necessarily) executable, Software Enmgineering Journal, Vol. 4, No. 6, pp330-338, 1989.

[Hod83] A, Hodges, Alan Turing: The Enigma, Simon and Schuster, New York, 1983.

[Kle35] S. C. Kleene, General Recursive Functions of Natural Numbers, American Journal of Matematics, Vol. 57, pages153-173 & 219-244, 1935.

[Kuh62] T. Kuhn, The Structure of Scientific Revolutions, University of Chicago, 1962.

[Lan61] R. Landauer, 'Irreversibility and Heat Generation in the Computing Process', *IBM Journal of Research and Development* vol. 5, No. 3, pp. 183–91, 1961.

[Lan91] R. Landauer, 'Information is Physical', *Physics Today*, May 1991.

[Lan00] R. Landauer, Information is Inevitably Physical, in Hey, Antony, ed. *Feynmann and Computing* Westview Press, Oxford, (2002).

[Las02] J. Lassgue, Turing, entre formel et forme ; remarques sur la convergence des perspectives morphologiques, Intellectica, Vol 2002/2, No. 35, pages 185-198, 2002.

[Mil92] R. Milner, The Polyadic π-calculus: A Tutorial, in F. L. Bauer, W. Brauer (eds.), Logic and Algebra of Specification, Springer-Verlag, 2003-246, 1992.

[Mi93] R. Milner, Elements of Interaction: Turing Award Lecture, Com. ACM, Vol 36, No. 1, pages 78-90, 1993.

[MPI94] F. MPI, MPI: A Message-Passing Interface. Technical Report. UMI Order Number: CSE-94-013., Oregon Graduate Institute School of Science & Engineering, 1994.

[Tur36a] A. M. Turing, On Computable Numbers with an Application to the Entschiedungsproblem, Proc. London Mathematical Soc., Ser. 2 , Vol 42, pages 230-265, 1936-7.

[Tur39] A. M. Turing, Systems of Logic Based on Ordinals, Proc. London Mathematical Soc., Ser. 2 , Vol 45, pages 161-228, 1939.

[Tur47] A. M. Turing, Lecture on the Automatic Computing Engine (1947), in B. J. Copeland (ed), The Essential Turing, Clarendon, Oxford, 2004.

[Tur50] A. M. Turing, Computing Machinery and Intelligence, Mind, Vol. LIX, No. 236, pages , 1950.

[Tur00] D. Turner, B. Pierce,"Pict: A programming language based on the picalculus." . In G. Plotkin, C. Stirling and M. Tofte, editors, Proof, Language and Interaction: Essays in Honour of Robin Milner, pages 455–494. MIT Press, 2000.

[Wal01] D. Sangiori and D. Walker, The π-calculus, Cambridge University Press, 2001.

[Weg97a] P. Wegner, Why Interaction is more Powerful than Algorithms, Communications of the ACM, Vol 40, No. 5, pages 80-91, 1997.

[Weg97b] P. Wegner, Interactive Software Technology, in: A. B. Tucker Jr (ed), The Computer Science and Engineering Handbook, CRC Press, pp2440-2463, 1997.

[Weg04b] P. Wegner, and E. Eberbach, New Models of Computation, Computer Journal, Vol 47, No. 1, pages 4-9, 2004.

[Yar04] V. Yarmolenko, P. Cockshott, E. Borland, L. Mackenzie, P. Graham, Jπ INTERFACE: A Java Implementation of the π-Calculus for Grid Computing, Proceedings Middleware Grid Conference, Toronto-Canada (Oct 2004)