edição 1 /2009 - ISSN: *1984-3585*

REVISTA DIGITAL DE TECNOLOGIAS COGNITIVAS

SOME REMARKS ON THE LOGIC AND EPISTEMOLOGY OF COMPUTATION

João de Fernandes Teixeira

Universidade Federal de São Carlos, Brazil
jteixe@terra.com.br

ABSTRACT

The paper focuses on some logical and epistemological aspects of the notion of computation. The first part questions the Church-Turing thesis as a fundamental principle concerning the limits of computation and some of its consequences for Philosophy of Mind and Cognitive Science. The second part discusses one of the main presumptions of the traditional conception of computability, namely, its reliance on the absolute character of classical logic which is taken as an underlying framework.

The aim of this paper is twofold. First, I shall present some shortcomings of the standard conception of computation based on what was termed Church´s Thesis, i.e., the claim that the class of functions which can be computed by machines is identical to the class of functions that can be computed by Turing machines. Second, I shall emphatically point to a further theoretical limitation of our current, orthodox conception of computation, namely, its assumption of classical logic as the definitive background from which one could decide what machines can compute. Finally, I shall briefly examine some of the consequences of these two lines of criticism to Philosophy of Mind and to Cognitive Science. The inheritance of Church´s Thesis as well as of the assumption of classical logic as an absolute constraint to machine computation seem to have originated a myth - the myth that Turing´s article of 1936 sets forth once and for all the limits of what *any* machine can compute. Such a mythical interpretation of Turing´s conception of computation also gives rise to endless discussions concerning the existence of non-computable cognitive activities and possible asymmetries between minds and machines or human brains and machines. Examples of the inheritance of this mythical red herring are the work of John Lucas (1961) and, more recently, of Roger Penrose (1989, 1994). Furthermore, I suggest that philosophers of mind and cognitive scientists were too hasty in dispensing with the physical-symbol classical cognitivist approach to Artificial Intelligence. As we shall see, the rejection of the view that the brain´s cognitive activity can be simulated by a computing machine takes for granted the theoretical limitations we refer to, and, without offering a supporting argument, shun the possibility that the human brain might be a non-classical computing machine.

I

What reasons can we avail ourselves to question the absolute character of Church´s Thesis? Is there computation beyond Turing-machine computation? Or is Turing computation the upper bound of our theoretical resources to conceive of computation? I shall press the claim that there is more to computation than what our classical framework deploys. Moreover, I shall outline what I consider to be some serious cracks in the orthodox conception of computation.

To begin with, Turing´s view of computation presented in his paper of 1936 takes for granted that the range of computable functions coincides with what can be done by a human being acting in accordance to an algorithm. The algorithm is a mimicry of what human beings perform when they realize an effective procedure. But no one seems to have ever questioned why the limits of human computation and the limits of machine computation are to overlap. (Wittgenstein in his Philosophical Investigations sec. 1096 ascertained that : “Turing Machines: These machines are *humans* who calculate”) Such a mimic conception of computation seems to be implicitly encompassed by Church´s Thesis. The latter ascertains that the range of what is computable is identical to the range of what is computable by a Turing machine, and, in so doing, it implicitly endorses the view that no physical or notional device which could, in principle, compute beyond what human beings can compute. But we shall return to this point later on.

Turing was aware that his conception of computation quickly led to what seemed to be an insurmountable limitation, i.e., the problem of uncomputable functions. For example, his conception of computation was restricted to the set of the integers; real numbers were to be excluded as uncomputable. Moreover, there were uncomputable functions even amongst the integers, namely, his famous “halting function” which gave rise to his Halting Theorem. The recognition of such constraints seems to be the *leitmotiv* of Turing´s Ph.D. thesis, published in 1939 and yet systematically overlooked by contemporary cognitive scientists. In this thesis Turing introduced what he called “oracle-machines” or O-Machines, i.e., ordinary Turing Machines augmented with a primitive operation set to return the value of uncomputable functions on the integers. Nonetheless, Turing has never provided an explanation of how an O-machine was supposed to perform its operation: oracles were black boxes. In neither case (1936, 1939) does Turing offer a discussion of what mechanisms should occupy these black boxes.

Contemporary cognitive scientists would rather not be reminded of the quick boundaries of the orthodox conception of computation. They leave this task to the apocaliptical knights who wish to proclaim the failure of the mechanical model of mentality. Still, many of the theoretical conundrums raised by the issue of the non-algorithmicality of some human cognitive functions could be avoided if we eschew the orthodox interpretation of Church´s Thesis. I believe such a task begins to be accomplished, and that, to such a goal, emerging work on alternative conceptions of computation is of paramount importance. No less important is the idea that the traditional notion of computation starts to be shaken, both by conceptual and technological changes.

Let us consider, for instance, some alternatives to discrete, Turing-machine computation. In an influential paper, Copeland (1997) calls attention upon the recent revival of analog computing machines as well as the role of analogical representation. There are analog machines which cannot be modeled as Turing Machines. However, such analog machines can perform computations which cannot be accomplished by Turing Machines.

How do such analog machines work? To begin with, they differ of Turing Machines so long as they process analog representations. Analogical is any representation whose structure corresponds to that of which it represents. For instance, the longer a line on a map, the longer the road line it represents. Following the same strand, numerical quantities can be represented by potential difference in an electric analog computer. But the importance of analogical representation and of analog computers strikes us once they allow us to build machines which can perform computations which *cannot* be carried out by Turing machines. Copeland (1997) provides an example of such an analog computer, an idealized, *notional* machine (as are Turing machines) he labels M1.

M1 is devised to represent continuously valued physical magnitudes; so, let us suppose that M1 represents electrical charges and that any real number can be represented by some quantity of charge. M1 is a simple device, with a very simple programmable control structure. When the representation of a real number x is presented as input, M1 delivers a representation of $3x$ as output. Since x may be either a computable number or an uncomputable number, M1 computes an uncomputable function.

The action of M1 can only be *approximately* simulated by a Turing machine if for any real number *x* and for any integer *k*, some Turing machine provides the first *k* places of a decimal representation of *x* and the first *k* places of 3*x*. But even the possibility of an *approximation* between M1 and a Turing machine is highly questionable, for it presupposes a demonstration that the action of an analogue computer can always be described and simulated on a digital machine. Such a demonstration, as far as I know, has not been attained so far. So viewed, M1 is a typical example that there may exist computations which are not carried out by Turing machines. Or, in other words, the notional existence of a machine such as M1 is a counterexample to Church´s Thesis.

In addition to analogue computers, connectionism can also provide examples of computation over real numbers which break away from Church´s Thesis. For reasons of space, I shall not revise the available literature on this topic, though it is fair at least to mention some who have taken the subject-matter seriously: McClelland and Rumelhart (1986), Smolensky (1988), Garzon & Franklin (1989), Wolpert & McLennan (1993), Siegelmann & Sontag (1994), and Korb (1996).

It would be enough to refer to analogue computation and to connectionism as providing vivid examples that *machine computation* cannot be taken as synonym of *Turing-machine computation*. However, there is another line of attack to Church´s Thesis which is worth mentioning. At the outset of this section, I have emphasized that one of the main assumptions of Turing´s notion of computation is the implicit equation between the limits of human and machine computation. We may suppose that such an implicit equation is blurred once we consider quantum computation. The speed of quantum computation cannot be attained by any human being. In this sense, quantum computation breaks away from orthodox computation, *but only in this sense*: the uncomputability of the halting function does remain in quantum computation despite the increase in velocity. The same applies to all classical limitations pointed by Turing.

II

We shall now turn to a brief examination of one of the main assumptions of the orthodox conception of computation, namely, the absolute character of the standard recursion theory and of the framework provided by classical logic. Surely classical logic was the paradigm of the thirties and the forties, but from that it does not follow that classical logic should be taken as an absolute presumption when one conceives of computation nowadays. Why can´t we rethink the orthodox notion of computation in the light of non-classical logic?

The absolutist character of orthodox computation and of classical logic go hand in hand. But once we abandon such an assumption we may also discard some classical, orthodox limitations to computation which fill out our traditional, cherished textbooks. The most striking result which emerges from the rejection of classical logic as an absolutist paradigm is the possibility of devising alternatives to the Halting Theorem.

In a former paper of mine (Teixeira & Sarmento, 1997) I have shown that by using DaCosta´s paraconsistent logic C_1+ it is possible to ascertain the existence of an algorithm for the problem of non-terminating computations. Our claim for the existence of a Halting Algorithm can either be envisaged as an extra pattern of reasoning of classical logic allowed by C_1+ or as a particular application of C_1+ - an application which shows that Turing´s Halting Theorem is valid only on the assumption that human reasoning can be fully represented by classical logic.

Let us recall the statement of the Halting Theorem and its proof : Given an arbitrary Turing Machine program P and an arbitrary set of input data set I , there does not exist a single Turing Machine program that halts after a finite number of steps, and that will tell us if P will ever finish processing the input I .

Proof: Once computable sequences are enumerable, consider a_n as being the nth. computable sequence and $\phi_n(m)$ the mth. representation in a_n . Be β the sequence taking
$1-\phi_n(n)$ as its nth. representation. Once β is computable there does not exist a number k such that $1-\phi_n(n)= \phi_k(n)$ for every n. If we take n=k it follows that $1=2\phi_k(k)$. *Absurd.* Therefore, computable sequences are *not* enumerable.

A more intuitive understanding of the Halting Theorem and of its proof can be given by the following example. Let us consider a computation on a natural number *n*. If we call such a computation $C_{(n)}$ we can conceive it as providing a family of computations where there is a separate computation for each natural number, 0,1,2,3...i.e., the computations C(0),C(1), C(2),C(3)...C(*n*) are the action of some Turing Machine (TM) on the number *n*, taken as the machine input.

Suppose we have some computational procedure *A* which, when it terminates provides a demonstration that a computation such as C(*n*) does not ever stop. If in any particular case *A* itself ever comes to an end, this would provide us with a demonstration that the particular computation that it refers to does not ever stop. Furthermore, we say that *A* is sound if it does not give us wrong answers. For, if *A* were unsound, then it would erroneously assert that the computation C(*n*) does not ever terminate when in fact it does. But if this is the case, the performing of the actual computation C(*n*) would eventually lead to a refutation of *A.*

In order for *A* to apply to computations generally, we shall need a way of coding all the different computations C(*n*) so that *A* can use this coding for its action. All the possible different computations C can in fact be listed as:

C_0, C_1, C_2, C_3, C_4 ...,

and we can refer to C_q as the *q*th.computation. When such a computation is applied to a particular number *n* we shall write:

$C_0(n), C_1(n), C_2(n), C_3(n), C_4(n)$,....

This ordering can be viewed as a numerical ordering of computer programs. Moreover, this listing is *computable* i.e., there is a single computation $C_\bullet$ which gives us C_q when it is presented with *q*, or, in other words, the computation $C_\bullet$ acts on the *pair* of numbers, *q,n* (*q* followed by *n*) to give $C_q(n)$.

The procedure *A* can now be thought of as a particular computation that, when presented with the pair of numbers *q,n,* tries to ascertain that the computation $C_q(n)$ will never halt. Thus, when the computation *A terminates* we have a demonstration that $C_q(n)$ *does not halt*. Being dependent on the two numbers *q* and *n*, the computation that *A* performs can be written *A*(*q,n*), and we have:

(1) If *A*(*q,n*) stops, then $C_q(n)$ does not stop.

Now let us consider the particular statements (1) for which *q* is put equal to *n*. With *q* equal to *n*, we now have:

(2) If *A*(*n,n*) stops, then $C_n(n)$ does not stop.

We notice that $A(n,n)$ depends upon just *one* number, n, not two, so it must be one of the computations $C_0,C_1,C_2,C_3..$ (as applied to n), since this was supposed to be a listing of *all* the computations which can be performed on a single natural number n. Let us suppose that it is in fact C_k, so we have:

(3) $A(n,n)= C_k(n)$.

Now examine the particular value $n=k$. From (3) we have:

(4) $A(k,k) = C_k(k)$.

and from (2), with $n=k$

(5) If $A(k,k)$ stops, then $C_k(k)$ does not stop.

Substituting (4) in (5) we find:

(6) If $C_k(k)$ stops, then $C_k(k)$ does not stop.

From this we deduce that the computation $C_k(k)$ does *not* in fact stop, for, if it did, then it does not, according to (6). But $A(k,k)$ cannot stop either, since by (4) it is the *same* as $C_k(k)$. Therefore, our procedure A cannot ascertain that this particular computation $C_k(k)$ does not stop even though it does not.

According to such a presentation the unsolvability of Turing´s Halting Problem is derived from the second part of Cantor's diagonal slash:

(5) - If $A(k,k)$ stops, then $C_k(k)$ does not stop.

Substituting (4) in (5) we find:

(6) - If $C_k(k)$ stops, then $C_k(k)$ does not stop.

Furthermore, we saw that from this we must deduce that the computation $C_k(k)$ does *not* in fact stop. For if it did then it does not, according to (6). But $A(k,k)$ cannot stop either, since by

(4) $A(k,k)= C_k(k)$

it is the *same* as $C_k(k)$. Thus our procedure A is incapable of ascertaining that this particular computation $C_k(k)$ does not stop even though it does not. The existence of A is denied for it implies a contradiction. Since $A(k,k)= C_k(k)$ we can write:

(7) If $A(k,k)$ stops, $C_k(k)$ does not stop.

(8) If $A(k,k)$ does not stop, $C_k(k)$ does not stop.

(7) and (8) can be rewritten in the form:

(9) *A* is sound,

and

(10) *A* is not sound.

A cannot exist for it encompasses a contradiction.

Nevertheless, it is agreed on this formulation of Turing´s Halting Problem that

(11) If *A* is sound *A(k,k)* stops and $C_k(k)$ does not stop; *A(k,k)* does not stop and $C_k(k)$ stops.

Now if we consider such a formulation of Turing´s Halting Problem in the light of paraconsistent reasoning (and by contrast to classical logic) one cannot infer that

(12) If *A(k,k)* does not stop, $C_k(k)$ does not stop.

Since in the light of paraconsistent reasoning one cannot infer the truth of (12) from (9) and (10) we can ascertain that the existence of *A* is possible. Such an ontological assertion is supported by paraconsistent reasoning derived from C_1+ which, by contrast to classical logic, does not lead to trivialization in the presence of contradictions.

As it is also remarked by Copeland (1997b), the proof of the Halting Theorem proceeds by *reductio* but, in a paraconsistent setting the derivation of a contradiction is insufficient for rejecting the assumption that leads to it, namely, the assumption that *there may exist* a Turing machine capable of computing the halting function. Such a result displays a further horizon for computing theory, i.e., the possibility of developing the new emerging field of *paraconsistent computability theory*.

III

Since C_1+ is mechanisable (for it is axiomatisable) we can plausibly sustain that mental processes involved in solving the problem of terminating/non-terminating computations may result from algorithmic procedures. Once we can represent our own reasoning pointing to the existence of a Halting Algorithm as the result of a mechanical process we can shun the alleged impossibility of modeling cognition without having to appeal to some piece of "intuitive" understanding which would remain inexplicable - a cornerstone assumption which underlies the thesis that there are non-computable or non-algorithmic processes in human cognition. This mythical red herring marshaled by John Lucas (1961) and by Roger Penrose (1989, 1994) as the main plank to criticize the physical-symbol classical cognitivist approach to Artificial Intelligence cannot be sustained unless one gullibly ascertain its two major assumptions: that human minds (or brains) are *classical* logical machines and that machine computation can be equated to *Turing-machine* computation. Perhaps human minds are very powerful formal systems - systems whose strength is likely to be derivable from their intrinsic inconsistency which allows them to compute *more* than the classical systems they construct. Such a powerful and inconsistent formal system manifests in what is currently labeled *common-sense*.

REFERENCES

Copeland, B.J. (1997) - "The broad conception of Computation" - **American Behavioral Scientist**, **40**: 6 pp.690-716.

Copeland, B.J. (1997b) - "**On the relativity of computability**" (forthcoming).

Garzon, M. & Franklin, S. (1989) - "**Neural computability: II. Abstract**" - *Proceedings of the IJCNN - International Joint Conference on Neural Networks*, I, pp.631-637.

Korb,K.B. (1996)- "**Symbolicism and connectionism: AI back at a joint point**" In D.L. Dowe, K.B. Korb & J.J. Oliver (eds) - *Information, statistic and induction in science*, Singapore: World Scientific pp.247-257

Lucas, J.R. (1961) - "**Minds, machines and Gödel**" *Philosophy* **36**, pp.120-124.

Penrose, R.(1989) - **The Emperor´s New Mind** , Oxford U.K., Oxford U.P.

Penrose, R. (1994) - **Shadows of the mind: A search for the missing science of consciousness**, Oxford U.K., Oxford U.P.

Rumelhart, D.E., McClelland, J.L. & the PDP Research Group (1986) - **Parallel distributed processing: Explorations in the microstructure of cognition, Vol. I, Foundations**. Cambridge: The MIT Press.

Siegelmann, H.T. & Sontag, E.D. (1994) - "**Analog computation via neural networks" - Theoretical Computer Science**, **131** pp. 331-360.

Smolensky, P. (1988) - "**On the proper treatment of connectionism" - Behavioral and Brain Sciences**, **11** pp. 1-23.

Teixeira, J.de F. & Sarmento, G. - "**Solving Turing´s Halting Problem?**" (forthcoming)

Turing, A. M. (1936) - "**On computable numbers, with an application to the Entscheidungsproblem**" - *Proceedings of the London Mathematical Society,* **42**, pp.230-65.

Turing, A.M. (1939) - Systems of Logic Based on Ordinals. **Proceedings of the London Mathematical Society**, **45**, pp.161-228.

Wolpert, D.H. & McLennan, B.J. (1993) - **A computationally universal field computer that is purely linear** - (Technical Report 93-09-056) Santa Fe, NM: Santa Fe Institute.