Oracle Turing machines faced with the verification problem

1 Introduction

Alan Turing is widely known in logic and computer science to have devised the computing model today named "Turing machine". In computer science because the Turing machine is the theoretical model of modern computers and in logic because it is the formalization of a computable function [Turing, 1936].

Nonetheless Turing is also behind the "O-machine", a Turing machine equipped with an "oracle", namely a black box whose behaviour is not specified, which is able to provide some non computable functions results [Turing, 1939] (p. 167). From its architecture and computational power, the O-machine is not a standard model according to computability theory but a "hypercomputation" model, term that is used to denote the possibility of computing non Turing machine computable functions (non Turing-computable functions) [Copeland, 2002].

Contrary to computability theory, hypercomputation is not fully accepted within scientific and philosophical communities. Although numerous hypercomputation models have been devised from a logical point of view [Stannett, 2004], current issues are more about the physical domain. Indeed, these issues directly concern the following claim that I will call "the hypercomputation thesis": it is physically possible to build a hypercomputation model.

To prove the hypercomputation thesis, one possible strategy could be to physically build an O-machine. More precisely, it is about finding a physical theory where a hypercomputation model will be able to use some extern information from nature. Such an information could be regarded as an oracle that provide an additional element in order to go beyond Turing machines limits [Calude, 2004] [Loff and Costa, 2009].

However, there is a recurring epistemological problem about the physical construction of an O-machine [Davis, 2006] (p. 13) [Copeland, 2002] (p. 490-491). This problem called "verification problem" may be worded as follows: if we assume that we have an O-machine physically built, it would be impossible to verify that this model is able to compute a non Turing-computable function.

In my presentation I will propose an analysis of the verification problem in order to show that it does not explicitly dispute the strategy about a physical construction of an O-machine.

2 How to build an oracle Turing machine ?

In order to build an O-machine based on non computable information that comes from nature, it is necessary in the first place to locate this information within nature. The idea of finding this information from quantum randomness comes from both the standard model of quantum physics and Richard Feynman's works. In the one hand, the standard model postulates quantum randomness from the Born postulate[1]. On the other hand, Feynman concludes in his paper *Simulating Physics with Computers* that "it is impossible to represent the results of quantum mechanics with a classical universal device" [Feynman, 1982] (p. 476). More recently, Christian Calude has proposed to devise a hypercomputation model by using quantum randomness as an oracle to exceed the Turing machine's power [Calude, 2004].

Calude's strategy consists of fixing on a computer a device able to generate a string of random numbers from a quantum process. For example, the ID quantique company[2] has created a device whose name is "Quantis", which generates a string of random numbers from an elementary quantum optics process [Jennewein et al., 2000]. More specifically, photons (light particles) are sent one by one onto a semi-transparent mirror and detected. The exclusive events (reflection - transmission) are associated to "0", "1" bit values and each of them have a probability at 50% to occur. The operation of Quantis is continuously monitored to ensure immediate detection of a failure and disabling of the random bit stream.

In theory, a computer equipped with Quantis might provide an arbitrarily long string of quantum random strings. However, this computer would be considered as a hypercomputation model only if the quantum random string cannot be generated by a Turing machine, that is to say only if the string includes an infinite number of bits. In that case Quantis would be seen as an oracle able to provide non computable information from nature.

Although the physical construction of an O-machine is sufficient *prima facie* to prove the hypercomputation thesis, an epistemological problem nevertheless remains. This problem is raised in the case where we have an O-machine and may be set out as follows: even if we build an O-machine we will not be able to prove the hypercomputation thesis because it would be impossible to verify that the machine is able to compute a non Turing-computable function. I am going to analyze this problem in order to suggest a way to overcome it.

3 A possible solution to the verification problem

The verification problem has been set out by Jack Copeland in the form of a thought experiment:

> " There is an epistemological problem with the idea of hypercomputation. Suppose Laplace's genius says 'Here is a black box for solving the Turing-machine halting problem' (The problem arises

[1] The Born Postulate is the idea that a measurement of a particle will yield a result which follows probability distribution $|\psi|^2$, where ψ is the particle's wave function.

[2] **http://www.idquantique.com/**.

> no matter which non Turing-machine-computable function is considered.) Type in any integer x and the box will deliver the corresponding value of the halting function $H(x)$ or so Laplace's genius assures you. Since there is no systematic method for calculating the values of the halting function, you have no means of checking whether or not the machine is producing correct answers. Even simulating the Turing machine in question will not in general help you, because no matter how long you watch the simulation, you cannot infer that the machine will not halt from the fact that it has not yet halted" [Copeland, 2002] (p. 471).

The verification problem as set out by Copeland is particularly relevant in the case of O-machines because they are considered as black boxes whose internal behaviour is not specified. Nevertheless we can ask why the absence of a verification is a real problem about O-machines. According to Marc Gold, it is indeed impossible, exclusively from intput-output Turing machine's behavior, to check the function that is computed by the Turing machine [Gold, 1965]. Intuitively, This is due to the fact that we only have at our disposal a finite number of results, which could every time correspond to other functions. Hence why the verification problem would be an obstacle to O-machines while it seems relevant about Turing machines?

Actually, here is the real problem about verification in hypercomputation: even if identification of the computing function is impossible both in effective computation and hypercomputation, we can verify in principle, that is to say regardless of computational resources, that a standard computer provides a correct result. Since a standard computer can be studied from its theoretical model, namely the Turing machine, we can have access to its results in principle checking step by step the computation from input to output. By contrast, we cannot proceed in the same manner with an O-machine physically built because we are not able to check each computational step of a hypercomputation model due to the absence of an effective procedure. Therefore if we have an O-machine physically built, we will not able to prove the hypercomputational power of the machine.

Some authors such as Carol Cleland [Cleland, 2004] (p. 223) and Oron Shagrir [Shagrir and Pitowski, 2003] (p. 99) brought several arguments to overcome the verification problem. However no complete account have been proposed. I am going to try to resume their arguments to rebuild such an account.

The central thesis of this account is to say that computation does not presuppose verification. This thesis is based on a distinction between two types of verification:

1. The verification in principle, which disregards computational resources.

2. The verification in practice, which takes into account computational resources.

The verification problem as set out by Copeland uses a verification in principle. We have showed that such a verification could challenged O-machines because this type of verification is possible about Turing machines. Therefore, to overcome this problem we have to consider the other type of verification, namely a verification in practice.

Actually, it turns out that we regard a function as computed by a standard computer even if we are not able to verify in practice the computed results. Take a particular function as an example (the argument works no matter which function is considered). Let p the function defined by $p(n) =$ the nth decimal of the expansion of π. It is easy to see that we cannot verify in practice (due to a lack of resources) whether the 10^{12}th decimal of π recently computed by a computer is 5. Yet, although it is impossible in practice to verify that a computer correctly computes p, we would tend to say nonetheless that the computer computes this function. But where does such a trust come from? This trust arises from the fact that it is possible to use some empirical methods (e.g. probabilistic causal relations, counterfactual suppositions grounded in physical law) to claim in a plausible way that a computer computes a function even if no perfect verification is possible. Therefore, computation does not presuppose verification because in practice we claim that a computer computes although we are not able to verify this claim.

From this point a view, the verification problem should not be considered as a thought experiment as Copeland did but as an empirical hypothesis. In other words, the problem could not be solved as long as one will suppose the construction of an O-machine; on the contrary we must to have an O-machine physically build to achieve empirical tests and therefore to claim that it computes a non Turing-computable function.

To clarify this last point, take as an example the O-machine proposed by Calude. Let us recall that it uses quantum randomness to provide a random numbers string that cannot be generated by a Turing machine. In theory, the main obstacle to claim whether this hypercomputation model is able to compute a non Turing computable function is based on the true-randomness of quantum physics that comes from the Born postulate. More precisely, there are two types of random processes: true-random processes and pseudo-random processes. Pseudo-random processes generate strings of numbers from pseudo-random methods (e.g. the linear congruence method), which numbers "appear" random but that are actually provided by deterministic formulae. Hence, if quantum processes are pseudo-random processes, a Turing machine would be able to simulate them since Turing machines that use pseudo-random algorithms are equivalent to standard Turing machines [De Leeuw et al., 1956] (p. 183-212).

To solve the verification problem in an empirical way would be to increase the plausibility of the claim that the O-machine computes a non Turing-computable function. It would be to achieve tests on random strings in order to show with a high probability that they are not pseudo-random. If such tests could be achieve and that we conclude that a string is true-random, then we could claim that this string represents the results of a non Turing-computable function. However, the disadvantage is that all current pseudo-random number generators provide strings, which are in practice impossible to distinguish from true-random number strings. Nevertheless we cannot dismiss the possibility to have some day reasonable grounds to believe that a string is true-random. In the same manner, we cannot dismiss the possibility to have some day reasonable grounds to believe that an O-machine physically built is able to go beyond the Turing machine.

4 Conclusion

The verification problem from a point of view in principle could be seen as a threat to the physical construction of an O-machine. However, there is still a way to overcome this problem if we consider it from a practical point of view. Yet this solution is not entirely satisfactory: on the one hand, even if we know that the O-machine computes a non Turing-computable function we do not know what is this function; on the other hand, it is not clear that the generation of a string of numbers is the same thing as the computation of a function.

References

C.S. Calude. Algorithmic Randomness, Quantum Physics, and Incompleteness. *CDMTCS Research Report Series*, 248, 2004.

C.E. Cleland. The Concept of Computability. *Theoretical Computer Science*, 317 pp. 209-225, 2004.

J. Copeland. Hypercomputation. *Minds and Machines*, 12 pp. 461-502, 2002.

M. Davis. The Myth of Hypercomputation. *in Christof Teuscher (ed), Alan Turing : the life and legacy of a great thinker, Springer*, 2006.

K. De Leeuw, E.F. Moore, C.E. Shannon, and N. Shapiro. *Computability by P*robabilistic *M*achines. Automata Studies : Princeton University Press, 1956.

R.P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21 pp. 467-488, 1982.

M. Gold. Limiting Recursion. *The Journal of Symbolic Logic*, 30, pp. 28-48, 1965.

T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger. A Fast and Compact Quantum Random Number Generator. *Review of Scientific Instruments*, 71, pp. 1675-1680, 2000.

B. Loff and J.F. Costa. Five Views of Hypercomputation. *International Journal of Unconventional Computing*, 5, pp. 193-207, 2009.

O. Shagrir and I. Pitowski. Physical Hypercomputation and the Church-Turing Thesis. *Minds and Machines*, 13 pp. 87-101, 2003.

M. Stannett. Hypercomputational Models. In C. Teuscher, editor, *Alan Turing: Life and Legacy of a Great Thinker (2004)*. Berlin : Springer-Verlag, 2004.

A.M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1936.

A.M. Turing. Systems of Logic Based on the Ordinals. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1939.