Facing the Computer. Some Techniques to understand Technique.

1. Introduction.

In a very short time the computer developed from a mathematical-logical object into an everyday physical object upon which whole societies rely. Possibly due to its significance in our society, people have a rather ambiguous attitude towards computers: on the one hand, they often get angry with their computer because it crashes or because it does something they did not ask for. They consider the computer as something which should not make mistakes, it should be perfect. On the other hand, while considered perfect, they understand it as a stupid machine – the only thing it does and must do is calculate. This ambiguity is not only apparent on the level of everyday life, but is maybe even more explicit on the level of scientific research. Since Church and Turing stated their respective theses, hundreds of research papers have been published in which attempts are made to beat the *physical* Church-Turing thesis (thesis P), and probably as many papers have been published trying to defend it.[1] This physical version of the thesis states that not only the computer, but every effectively realizable physical system can be defined in terms of Turing machines.
Inspired by Martin Heidegger's *Question concerning technique*[2], the main purpose of this paper is to propose some strategies which might lead one to a better understanding of what a computation actually is. Starting from this position, the discussion on the yes/no possibility of physically beating a Turing machine will be questioned. While this discussion is of course a very important one – both from a scientific as well as from a philosophical point of view – it will be argued here that it might also be interesting to shift attention from the question of what might (not) be computable to the question of what a computation actually is.

2. A question concerning Technique

In 1936 Church and Turing each independently proposed their respective theses. If these theses are true then it naturally follows that those decision problems not solvable by any Turing machine, or any other equivalent formulation, are non-computable problems. Since then hundreds of other decision problems have been shown to be unsolvable – problems which have not remained restricted to the domain of mathematical logic, even in theoretical physics there exist several unsolvable decision problems.[3] Despite or due to these results many authors claim that it might be possible to beat thesis P: they are convinced that there are physically realizable processes able to solve these decision problems.[4][5] Others argue that this will never be possible: the limit proven to exist by Turing is not only a theoretical but also a physically existing limit .
While a solution to the truth or falsity of thesis P would of course be interesting in itself, one must admit that the approaches both of pro and contra have not been able so far to provide for any definite answer. Motivated by this fact, the question must be raised whether it could not

[1] The physical Church-Turing thesis however, should be neatly distinguished from its original mathematical formulation.
[2] The official English translation of the title of this essay is: *Question concerning technology.* This translation is considered as a bad translation here, since Heidegger himself explicitly states in this essay that *technique* should not be considered as something *technological.*
[3] Some authors mistakenly use such results in arguing against thesis P. See *De Mol 2006b*.
[4] In *Cotogno 2003* one finds an overview of many different proposals for hypercomputation (including physical supertasks and infinite computations; interactive systems; analog computations and quantum computations), together with arguments which show that none of these is able to "beat" thesis P.
[5] Without wanting to argue against the possibility of "beating" thesis P, the author would like to add one important question: Suppose that one would have constructed a machine M for which it is claimed that it outputs a non-recursive function, how will one verify this empirically? How will we humans be able to observe this? This is impossible, since the "*thesis that M computes a recursive function is consistent with any finite chunk of data.*" (See *Shipman, 2000.*)

be more interesting to leave open this problem for a while, and focus on what this problem is actually about. When originally posed in 1936, the Church-Turing thesis had no link whatsoever to the question of how to outrun a computational process. It was about the question: what do we mean *exactly* when we say that something is computable? This question is not only significant in relation to the ongoing discussion on thesis P, but is maybe even more significant from a philosophical point of view, especially if the materialization of the subject of this question is taken into account: the computer.

The computer has become an omnipresent object in our society: the variety of applications and our dependencies on them could have hardly been imagined by Turing when he first described the abstract complement – the universal Turing machine – of this general-purpose machine. From its first use on – making calculations for the A-bomb – it was clear that these applications are not merely restricted to the field of scientific research, resulting in a "technology" that affects every man in the street.

In this respect Martin Heidegger's question concerning technique becomes an insistent one. Although this poetical text is often interpreted as a critique on the "technification" of modern society it can also be understood as an appeal to man to become aware of the "essence" of the "technique" present in society – technique being the way man perceives, acts in and with the world.

Since calculation and control of information are two typical "features" of the way man perceives, acts in and with the world nowadays, the computer can be understood as *a* physical realization of this "technique". Relating the question concerning technique, as posed by Heidegger, to computers implies that it is fundamental that man does not simply uses the computer as a mere instrument, but that he knows what he is using and how he uses it:

> Darum liegt alles daran, dass wir den Aufgang bedenken und andenkend hüten. Wie geschieht dies? Vor allem anderen so, dass wir das Wesende in der Technik erblicken, statt nur auf das Technische zu starren. Solange wir die Technik als Instrument vorstellen, bleiben wir im Willen hängen sie zu meistern. Wir treiben am Wesen der Technik forbei. (32)

Indeed, as long as we do not see that the computer is not merely an instrument but rather a realization of what technique is, its essence remains hidden and can thus form a danger:

> Das Gefährliche ist nicht die Technik. Es gibt keine Dämonie der Technik, wohl dagegen das Geheimnis ihres Wesens. Das Wesen der Technik ist [...] die Gefahr. (27-28)

The fact that the majority of mankind uses many of the possibilities of the computer, without ever knowing what is beyond the monopolized interface, is just one concrete example of this problem. Indeed, the only reason for this monopoly being possible is the fact that most of the people never get beyond their GUI.[6] The only way out is to face oneself with technique as it is.

3. Strategy I : Post's Machine

In a little booklet called "Post's Machine" the Russian mathematician Uspensky describes a "toy machine" – first described by Emil Post in 1936 – to show how one can advance abstract concepts such as "algorithm", "universal computing machine" and "programming" for school children. After having explained the inner workings of a Post machine, Uspensky gives a long exposition on how to "program" the basic operation of "+ 1" on a Post machine and shows in this way that this for us seemingly trivial operation becomes far from trivial when implemented on this machine. Indeed, instead of one step, the machine needs 23 instructions to perform this simple calculation. Furthermore, making it run one might need hundreds of steps to add 1 to a number, depending the initial condition. This example clearly shows how

[6]An existing strategy to counter this problem is proposed by the Free Software Foundation, which makes it possible not only to freely distribute the software itself but also the source code. In this way you can e.g. change the code when you want things to be done in another way – a freedom which is unthinkable for the average windows user.

that which is considered as a basic step in arithmetic becomes in itself a complex operation when performed on a medium which was constructed with the purpose of formalising the intuitive concept of a computation. In other words, in working with this machine something very fundamental about computations – and thus technique – is understood: when we perform calculations in our everyday life, this is merely *a* way to do it. That, which seems to be the most basic operation in a rather absolute way, is not what it seems to be. It was exactly this kind of understanding that was fundamental to Church's and especially Turing's argumentation with respect to their theses: in order for their arguments to work, they had to analyse the possible processes underlying our everyday way of e.g. adding two numbers.
The idea of teaching schoolchildren that the way they normally calculate is not the only way, and certainly not the way their computer calculates, together with the fact that in this process they can also learn to understand some basic problems of programming, should at least be taken into consideration in the light of Heidegger's question concerning technique: it is one way to go beyond the GUI. Especially in the light of the omnipresence of the computer in our society and its consequent economical value, the idea that people don't have a clue of what a program is or how their computer works can at least be called disquieting.[7]

4. Strategy II: How Computers Constitute New Branches of Science.

As was stated earlier, technique is the way man acts in and with the world – his means of communication. This implies that technique is the framework through which we perceive the world in a certain way. Since technique is on the one hand reflected in the objects man creates, and is on the other hand the framework through which we perceive the world, the "technical" world must in its turn influence our thinking and understanding of the world, since we are part of it. Understanding that technique – when understood as the "technical" world – is not something passive, solely made for man's needs, simply standing there being available, but rather something that influences and changes man in a fundamental way, that there is a reciprocal relation between man and that which he makes, is considered as basic to our understanding of technique. But in what way does the computer and its calculations, as an explicit materialisation of "technique", influence our thinking and understanding?
There are many examples – the computer has touched upon almost every aspect of our life, ranging from war to art – but the ones focused on here are motivated by a lecture given by Von Neumann in 1949.[8] After having argued for the necessity of building up an intuition of a given mathematical problem one is investigating, he remarks that in some cases the computer might be the only way to build up such an intuition. Indeed, it might e.g. be possible that such intuition is blocked off because we simply cannot perform enough calculations in a reasonable time. It is in this respect that the computer has changed and even founded new disciplines of science. It has become possible to build up an intuition of certain problems which would not have been possible before, and in some cases, the problem (and the intuition of it) even only became a problem, arising because of (the use of) the computer.
One of the most celebrated examples for which the development of the computer has been fundamental is fractal geometry, which is in its turn closely related with another such discipline, chaos theory. Neither of these branches of mathematics – with applications in other sciences – would have been possible without the help of the computer. Another example, intimately connected with the rise of the computer, is computational complexity theory, which asks questions about time and space complexity of decision problems and is fundamental with regards to the safety of the internet.
In mathematics there exist nowadays concurrent models for computations over the reals because of the explosive use of the computer in physics. Since physics works with the

[7] It should be noted though that no paradigm should be allowed to dominate education.
[8] See *Von Neumann 1966*.

continuum and the classical model of computation is discrete it seemed necessary to develop such models. The fact that even new mathematical theories were developed for handling the problem of scientific computing, clearly shows that the computer now plays a vital role in physics. In order to study certain problems one no longer sets up a traditional experiment. Instead one makes a model or a simulation of the problem at hand, for which one can easily change any parameter in a couple of minutes or even seconds in order to study the behaviour of a certain physical process in a more general way. An interesting example is the research done on cellular automata (CA), where one of the big names nowadays is Stephen Wolfram.
CA's are mathematical objects for which it was shown that they can calculate anything a universal Turing machine can calculate. Although being abstract formalisms, just as λ-calculus, CA have not remained restricted to the domain of mathematical logic and theoretical computer science. They were developed by Von Neumann in collaboration with Ulam, as mathematical models of self-reproducing artificial systems. Nowadays, CA's are studied in the field of theoretical biology, and are one of the frameworks of artificial life. Without going into further details here, it is important to note that this research on CA often comes down to the "simulation" of (certain aspects of) physical systems. Stephen Wolfram for instance gives several examples in his highly debated book *A new kind of science* of CA-like models which simulate e.g. the growth of plants, pigmentation on certain organisms, and fluidic phenomena. Another example is Langton's ant, which exhibits a dynamical transition from "chaotic" to periodic behaviour. Both Langton and Wolfram are convinced that it is possible to simulate "life" in simple models such as CA and Turing machines and they thus have to assume the truth of thesis P.[9]

5. A Battle between Nature and Computing Machines?

In 1986 Langton published a paper on artifical life in which he stated:[10]

> The ultimate goal of the study of artificial life would be to create 'life' in some other medium, ideally a *virtual* medium where the essence of life has been abstracted from the details of its implementation in *any* particular hardware. We would like to build models that are so life-like that they cease to be *models* of life and become *examples* of life themselves." (147)

Indeed, to Langton (and the same goes with Wolfram) it is not correct to speak of simulations of life: it is "real" life. This is comparable to the strong AI position in which it is stated that it is possible for a computer not only to simulate intelligence but to be *really* intelligent. Both hard AI and alife clearly presuppose thesis P. As was already stated in sec. 2 however, the question must be raised whether it could not be more interesting to leave open the problem of the truth or falsity of thesis P and instead focus on what this thesis is actually about. In its original form – the Church-Turing thesis – the question was not how can we create life in a formal system, or how does life outrun a formal system, but rather: what is a computation?
Both the defenders and opponents of thesis P never seem to go back to the original papers by Church and Turing from 1936 – in the best case they are reduced to the level of a non-read reference – and consequently never *really* reconsider this last question, although it is fundamental to their work. Indeed, there seems to be only one important direction taken into consideration in this ongoing discussion: how do we go (or can we never go) from nature to computations? Focus is always put on the physical processes themselves and how they can or cannot be implemented on a computer, the other side of thesis P is hardly investigated within this domain. Identifying physical processes with their simulation in CA or the opposite idea that there must be something "non-computable" about physical processes – whatever that may be – then become mere symptoms of this one-directedness.

[9] This is even very explicitly stated in *Wolfram 2002*, where it is presupposed that nature is algorithmic.
[10] See *Langton 1986*.

But why should one bother about this imbalance in the discussion on thesis P? There are two reasons. First of all it is more than significant that people are aware of what technique is. Since calculation is one of the features of modern technique, trying to use it as a way to capture nature or to differentiate it from technique (as calculation), while not focussing on it, is a typical example of how technique gets hidden away: one uses its materialisation as a way to escape or control it. In this way, one will never be able to understand it.

The second reason is in fact an application of Heidegger's thoughts on truth. Without going into more details, it is important to note that this concept is based on a kind of Escher-like effect: in focussing on one thing, in putting something in the foreground, there is always a background, constitutive for the foreground. Although you cannot focus on two things at one and the same time, it is important to be aware of the fact that there is a background. Trivial as this principle might seem, it is not. Applying this idea to the discussions on thesis P, one has to ask the following question: In focussing on physical processes, ignoring the other side of thesis P, does one not risk to exclude some very interesting philosophical and scientific questions?

To give just one example, why does one seldom ask: are there things computers can do, which are not apparent in nature? There seems to be at least one very nice example here. As every programmer knows it is very important to have a good random number generator (RNG). Nowadays, every newly constructed RNG has to pass for several tests – performing only one is not good enough. It is in this context that Marsaglia developed the software: DIEHARD, a battery of tests of randomness. In the instructions for using DIEHARD, one reads:

> I hope you will inform me of results, good or bad, of new kinds of generators you have tested, particularly deterministic generators, but also the output of physical devices. (I have found none of the latter that get past DIEHARD, and would like to learn of any that do.) Since, in my opinion, there is no true randomness, collective experience in finding sequences that depart from the theoretical ideal in a significant way can perhaps lead to better ways for finding those that do not.

Indeed, while one would expect physical RNG's to be the best of possible RNG's, none of the ones Marsaglia tested got past his tests. On the other hand, there are several deterministic generators which do pass it. While this is of course not a valid proof of the non-existence of statistical randomness in physics, it is an interesting observation, which – as noted in the above quote – might progress certain research.

6. Strategy III: Playing with formalisms.

As was shown, instead of focussing on the physical side of thesis P, it might be interesting and even fundamental to further investigate the computations themselves. Even if one feels the urge to solve thesis P, the question poses itself in what way we can ever solve this problem if we have never looked at the behaviour of computations without superimposing anything concrete on them?

Of course computations are always captured in a form: CA, Turing machines, λ-calculus,....The ones focused on here are tag systems. They were developed by Emil Post in trying to find the most abstract form of symbolic logic – and abstract they are. As was argued in *De Mol 2006a*, Post's method in arriving at a variant of the Entscheidungsproblem in 1921 was to construct more and more general forms of mathematics. In this process of developing more and more general forms he ended up – at a given moment – with his form of tag. A tag system is defined as follows. Given an alphabet A of μ symbols, e.g. A = {0, 1} and a natural number v, e.g. v = 3. With each of the letters of the alphabet there is a corresponding word over the alphabet. E.g.:

$1 \longrightarrow 1101$

$0 \longrightarrow 00$

Now given an initial sequence, depending on its first symbol, tag the word corresponding with this symbol at its tail and then remove the first v elements. This process is repeated for every new string produced, until the empty string is produced. Here is an example, applying the above given rules:

```
100101010011
    1010100111101
        01001111011101
            0111101110100
              ...............................
```

Now why should one be interested in tag systems? First of all it should be noted that they are "as complex" as CA, since there exist universal tag systems. Secondly, they seem to be good formalisms to allow for a further analysis of "computations" in the light of the above given discussion: it is hard to superimpose anything concrete on these systems since they were developed to avoid this! Moreover they are very easy to implement on a computer, and run very fast. Given the "meaninglessness" of tag systems it is possible to investigate computational systems in a less prejudiced way since there seems to be no "meaningful" interpretation at all: neither for their behaviour nor for their rules. So how should one start? The first thing to do is – to follow the words by Von Neumann – to build up an intuition of these systems. This can be done by setting up an experimental dialogue between tag systems, the physical machine they are run on, the programming language they are written in and the human wanting to understand them. You can then see what different kinds of behaviour tag systems can lead to in varying several parameters like v and μ. It is also important to work some systems out by hand, since performing the operations by yourself leads to other intuitions of the systems. But what other kind of research could one do? There are several interesting research questions here, but they cannot be discussed here.[11]

More significant here is the setting up of an experimental dialogue. If one ever wants to understand part of what technique is, one should not consider formal systems and their physical realization as something which is not part of the world – mere products of the human mind, standing at our disposal when we like it. Rather one should not forget that *as* products of the human mind, they are part of this world, and can be as physical as anything else. It is in this sense that setting up an experimental dialogue between e.g. tag systems and a human being is one of the possible strategies to understand technique. In translating my questions to my computer, and waiting for the answer from the tag systems, one can only learn that they cannot be controlled by a human mind – although they were invented by one. If I ask a question, I am never sure of what the answer will be. Often, the answer I get is more like another question, posed in language which is not mine – and one is forced into further questions.

Furthermore in performing computer experiments, one quickly learns that the machine running the code does and must make mistakes due to its physical nature. Overflows and calculation errors are "bugs" one does not expect if one never goes beyond the traditional GUI. Going back to the beginning of this text, many people are indeed rather ambiguous towards computers, scientists included. This ambiguous attitude can only be resolved in understanding that computers can and must make mistakes and be limited. After all, a computer is a physical system obeying physical laws. Doing computer experiments, can show one that a computer does not "deserve" the special status of being perfect and stupid at one and the same time: it shows us that computers are much more "worldly" than is often believed.

7. Conclusion

[11] To give just one example: is it possible to prove that 2 symbolic tag systems, with $v > 2$ are universal and/or whether there exist examples in this class with an unsolvable decision problem. This question is significant since one seems to be in need of methods, different from the usual ones.

What is technique? Is it something technical? No. Rather it is the way man perceives of, is in and with the world. Not being aware of the way you are in the world can at least be called problematic. To Heidegger, this is one of the biggest problems of modern society. We do not see in what way we are approaching the world we are part of. One way to make a start at an understanding of technique is to face oneself with one of the "features" of technique: computations which are physically realized in our world through the computer. In this paper, some strategies were proposed to allow for a better understanding of computers and computations. These were linked to the ambiguous attitude of man towards computers, more specifically linked to an existing ambiguous attitude of scientists.
But why does one need this obscurantist philosopher Heidegger here? It could have been perfectly possible to argue for the significance of a deeper awareness of what a computer/computation is without him. Indeed the mere omnipresence of the computer and our dependence on it would have been enough.
Heidegger's text is an appeal to man, written in a style that, when taken seriously, forces one to think. From this text one can begin to understand that it is far from trivial, and maybe even necessary, that one starts to have a closer look at the "technical" world. In this way, Heidegger's text, if read in a certain way, is yet another strategy to understand technique.[12]
The main reason however for including Heidegger here, is the mere fact that most of the philosophers don't have a clue of what a computer is. In the meantime they are discussing topics such as: the yes/no possibility of an intelligent computer, going beyond the Turing machine (or not), the notion of randomness and coincidence far away from RNG's,... They do this without ever having taken the trouble of going through the details of Turing's paper, let alone, following the motions of a universal machine, understanding what it is, and what it is not. Heidegger's text was included here in order to make an appeal to the philosophers *through* the words of a famous philosopher, obscurantist though he may be. As Friedrich Kittler once formulated it:

> In this way, computers are sold whose architecture is not so much defined by the state of the art but by a pre-history or firm bureaucracy that crystallises into hardware right away. And if the ideal of software [...] would ever triumph, the bureaucratisation would be perfect: The hardware, in spite of its programmability, would irrevocably be obscured under its packaging. To stop this coincidence from happening seems to be an eminent political goal. If computers are the first machines to reduce the contingency or incomputability of some, though not all futures to a finite degree, its own contingency should remain as open as possible. [...] If somebody went and wrote all the programmes hitherto running under the name of philosophy into hardware, that would be the goal itself. (*Kittler 1987,* 131)

8. Bibliography

Paolo Cotogno (2003), *Hypercomputation and the Physical Church-Turing Thesis*, in: British Journal for the Philosophy of Science, **54**, pp. 181—223.

Liesbeth De Mol (2006a), *Closing the Circle: An analysis of Emil Post's early work*, The Bulletin of Symbolic Logic, **12**, 2, 267-288.

Liesbeth De Mol (2006b), *Refocussing Unidecidability. Questioning some extensions of the notion of formal undecidability to other domains.* In: D. Aerts, B. D'Hooghe and N. Note (Eds.), Worldviews, Science and Us: Bridging Knowledge and Its Implications for our Perspectives of the World. World Scientific , Singapore, To appear in 2006.

[12] It was in fact this text that led the author from philosophy to computers and programming.

Christopher Langton, *Studying Artificial Life with Cellular Automata,* Physica D, **22**, 1-3, 120-149.

Friedrich Kittler (1987), *Hardware, das unbekannte Wesen*, in: Sybille Krämer (ed.), Wirklichkeitsvorstellungen und Neue Medien, 119—132.

Joe Shipman, *Physical Computability,* FOM list, 23 August, 2000. Available at: http://www.cs.nyu.edu/pipermail/fom/2000-August/004244.html.

Vladimir A. Uspensky (1983), *Post's machine*, Mir Publishers (Little Mathematics Liberary)", Moscow.

John Von Neumann (1966), *The General and Logical Theory of Automata*, Arthur W. Burks (ed.), University of Illinois Press, Urbana, London.

Stephen Wolfram (2002), *A new kind of science*, Champaign, Wolfram Inc.