What The Church-Turing Thesis Does Not Say

Abstract: In 1967, Kleene formulated the Church-Turing Thesis (CTT). It is one of the most fundamental theses as far as computation is concerned. More often than not, the CTT is misinterpreted as making a statement concerning the powers of the human mind, human intelligence, or a modern computer. These "variants" are not, as many claim, equivalent to the original thesis. In fact, most of them make claims that are stronger than the actual one and make overt assumptions rendering these variants uninteresting. We reject such claims and argue that neither Turing nor Church intended to characterise their theses as such. In 1980 Robin Gandy forwarded his Thesis M and established that any function computable by discrete dynamic systems is also computable by Turing Machines. Some of the flawed readings are closer to Thesis M than they are to the CTT.

Computation is one of the most inestimable gifts of philosophy and logic. One can trace its origins to the seventeenth century in the work of Gottfried Leibniz. Computation becomes more prominent in the nineteenth and twentieth century. Formulated in the late twentieth century, the Church-Turing Thesis is one of the most important theses in Computation. To understand the thesis, its intended characterisation, and the context in which it was written, it is essential to look at it through the lens of history.

In 1900, the German mathematician David Hilbert, published a list of 23 problems which were yet to be solved. He selected 10 problems to present at the International Congress of Mathematicians held on August 8, 1900 in Sorbonne. They were later translated to English and published in the Bulletin of the American Mathematical Society in 1902. The tenth problem proposed by Hilbert seeks a procedure (*verfahren*) for deciding (*entscheidung*) whether the Diophantine equations are solvable or not, and not just the existence of such a process. Hilbert did not use the word "algorithm" in his formulation of the problem. However, the term "algorithms" has been used before that time, e.g. the algorithm for deciding if a number is prime. We believe this is worth noting because it emphasises Hilbert's desire for a *mechanical* method. Later in 1928, Hilbert and

Ackermann explicitly formulated "*Das Entscheidungsproblem*" (The Decision Problem), in the context of "*engere Funktionenkalkül*" (function calculus).

> "Is there an effective procedure which, given a set of axioms and a mathematical proposition, decides whether it is or is not provable from the axioms?" (Hilbert & Ackermann, 1928)

Hilbert did not believe in the idea of an "unsolvable problem" until 1930 (Hodges, 1983). In first-order logic, the decision problem seeks a procedure to decide the provability of a statement (from the axioms and rules of the logic).

It is important to understand Hilbert's philosophy of mathematics to understand Turing's intention in (Turing, 1936) which was to provide an answer to Hilbert's question. Hilbert wanted to build a logical foundation for mathematics. He wanted to show that mathematics follows from a finite system of axioms and a finite set of rules as well as the provability of such a system. To do so, he developed the epsilon calculus which enabled him to give a proof of consistency. He was concerned about whether such methods can be applied to human mathematicians.

In May 1936, in his paper *"On Computable Numbers, With An Application To The Entscheidungsproblem,"* Alan M. Turing proposed the concept of computability by an abstract model of computation. Earlier, in April 1936, American Mathematician Alonzo Church had independently published the proof of the undecidability of a problem using his formal system of Lambda Calculus (Church, 1936). The approaches of Turing and Church were very different. Turing reduced Hilbert's Entscheidungsproblem to the "halting problem" for his model of computation, namely, the "Logical Computing Machines" (LCMs) and proved its unsolvability. He showed

> "...that there can be no general process for determining whether a given formula **U** of the functional calculus **K** is provable, i.e. that there can be no machine which, supplied with any one **U** of these formulae, will eventually say whether **U** is provable." (Turing, 1936)

Turing had stated his thesis in many of his papers. For example,

> " LCMs [i.e. Turing machines] can do anything that could be described as "rule of thumb" or "purely mechanical" (Turing, 1992)

and

> "It is possible to invent a single machine which can be used to compute any computable sequence [or functions [1] "naturally regarded as computable"]" (Turing, 1936)

Following we present a version of the Turing Thesis which we take to be apt.

Turing's Thesis (TT): *Anything that can be stipulated as purely mechanical procedures is computable by a Turing Machine.*

Church arrived at the same conclusion as Turing, but using a different approach. He

> "define(s) the notion ... of an effectively calculable function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a lambda-definable function of positive integers)" (Church, 1936)

Church's thesis is thus formulated as follows.

[1] Turing talks about numbers, but here functions and numbers can be treated as the same. "No attempt has yet been made to show that the "computable" numbers include all numbers which would naturally be regarded as computable."

Church's Thesis (CT): *What is effectively calculable is computable.*

The following formulation of the Church-Turing Thesis is a standard one.

Church-Turing Thesis (CTT): *Any function that is intuitively computable is computable by some Turing machine.*

There is a wide spread disagreement regarding the interpretation of the Church Turing Thesis. Many philosophers have debated on what the intentions of Turing and Church were when they presented their theses in (Turing, 1936) and (Church, 1936) respectively and in their other works. The term "Church-Turing Thesis" was coined later in 1967 by Kleene, who happened to be a student of Church.

> "So Turing's and Church's theses are equivalent. We shall usually refer to them both as Church's thesis, or in connection with that one of its ... versions which deals with 'Turing machines' as the Church-Turing thesis." (Kleene, 1967)

The Church-Turing thesis is often formulated in ways which fail to express the intentions of the authors, or misstate the thesis completely, or over specify the range of the applicability of the thesis. Some philosophers like B.J. Copeland assert that

> "Church and Turing claimed only that a universal Turing machine can match the behavior of any human mathematician working with paper and pencil in accordance with an algorithmic method—a considerably weaker claim that certainly does not rule out the possibility of hypermachines.' " (Copeland & Proudfoot, 1999).

Paul and Patricia Churchland state (Churchland & Churchland, 1990) that Turing's

> "results entail something remarkable, namely that a standard digital computer, given only the right program, a large enough memory and sufficient time, can compute any rule-governed input-output function. That is, it can display any systematic pattern of responses to the environment whatsoever."

Surprisingly, some have interpreted the Church-Turing Thesis as making a claim about the powers of the human mind. For example,

> “Can the operations of the brain be simulated on a digital computer? ... The answer seems to me ... demonstrably ‘Yes’ ... That is, naturally interpreted, the question means: Is there some description of the brain such that under that description you could do a computational simulation of the operations of the brain. But given Church's thesis that anything that can be given a precise enough characterization as a set of steps can be simulated on a digital computer, it follows trivially that the question has an affirmative answer” (Searle, 1992).

The authors above do not, however, restrict "any functions", "any computer", "any rule-governed input-output function" to Turing's notion. What we aim to show is that without such restrictions, the Church-Turing Thesis, does not hold, and so we must reject, or at least question, such interpretations.

In (Turing, 1936) Turing presents his abstract model of computation – “Computing machines”. This abstract machine is a computer “in the process of computing a real number to a machine which is only capable of … .” The word “computer” is not used in the general

sense used today. It refers to the “man in the process”. Following the lead of Robin Gandy, we will call them “computors”. We will present a brief and simplified version of the computing machine model, now called Turing Machines. The model consists of a machine with a fixed bound on the number of conditions which is supplied with a “tape” divided into “squares” each bearing a “symbol”. At any given time, the machine is aware of the symbol on the square “in the machine”. The machine can “scan” and “print” symbols, and move the tape left and right, one square at a time. The action of the machine is determined by the state it is in, say **q** and the symbol it reads in, say **s**. For the computor, to ensure that he can distinguish a square from the other, at a glance, the number of symbols in the square needs to be finite. As Sieg remarks in (Seig, 2008),

> "If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent", and the computor could not distinguish at a glance between symbols that are "sufficiently close"... Arabic numerals like 178 or 99999999 as one symbol; then it is not possible for the computor to determine at one glance [“immediately recognizable”] whether or not 9889995496789998769 is identical with 98899954967899998769."

Thus, the computor can perform simple and mechanical tasks like read and write a symbol and can move left or right. To make it deterministic, the machine has a “transition” for all relevant combinations of states and symbols mapping to a state and task to perform. The computor has a “state of mind”. After reading a symbol, the updated “state of mind”, and the following actions, are uniquely determined. Turing also points out that,

> "I think it is reasonable to suppose they can only be squares whose distance from the closest of the immediately previously observed square does not exceed a certain fixed amount."

Turing thus intended to characterize calculation by an abstract human being provided with pencil and eraser, and paper. Turing emphasizes this repeatedly in many of his works.

> "A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine." (Turing, 1948)

The notion of a computor is central to, and a starting point of, his analysis. Moreover, the computor is involved in some simple and mechanical steps which do not require any understanding of the significance of the symbols or the machine itself. Turing's analysis in (Turing, 1936) does not, for example, appeal to the use the intelligence or cognitive power of the "human computer". Instead, there are a few restrictions and limitations being imposed on the computer qua tape reader. The restrictions are as follows[2]:

(B1) The number of symbols a computor can immediately recognise has a fixed and finite bound.

(B2) The number of states of mind that needs to be taken into account has a fixed finite bound.

(B1) and (B2) bounds the possible combinations of states of mind and relevant symbols for the computor. This leads to the following to locality conditions:

(L1) The altered symbol(s) are those that are relevant to the computor.

(L2) The new symbol(s) written by the computor are bounded by a distance from the originally observed symbol(s).

[2] These conditions are from (Seig, 2008). Please consult (Seig, 2008) for details.

Thus, Turing did not have the notions of a "digital machine" as found in (Churchland & Churchland, 1990), or "functions of the brain" as claimed by (Searle, 1992) in mind when characterising his computors. Instead, his focus is on the *mechanical* nature of the tasks that the computor is expected to perform. The entire procedure carried out by this computer consists of tasks that cannot be simplified further by any mechanical method. This is the most important theme in the construction of the abstract human. Robin Gandy, in "Principles for mechanisms" states

> "What we can say is that Turing outlined a proof of the following:
>
> Theorem T: What can be calculated by an *abstract human being* working in a *routine* way is computable." (Gandy, 1980) [my emphasis]

Gandy gave an axiomatised characterisation of the computors and showed that any machines abiding the boundedness and locality conditions can be reduced to a Turing Machine. The additional claim that Turing wanted to analyse the workings of a computing machine, or an analogue computer, or any such physical device is thus simply not true. However, it is often claimed that Turing's intention was to analyse a computing machine. Such errors of interpretation must be rectified in order to understand the true essence of Turing's thesis. Hodges, in (Hodges, 2006) remarks,

> "Gandy criticised Newman [1955] for saying that Turing embarked on 'analysing the general notion of a computing machine'. "

The same reasoning and criticism also applies to some of the authors cited at the onset, especially those who maintain that Turing thesis is applicable to modern computers.

In (Gandy, 1980), we see the Thesis M, "What can be calculated by a *machine* [my emphasis] is computable" which is quite distant from the Church-Turing thesis. Gandy explicitly states in this paper that by machines he means no "abstract human" but rather in the traditional sense where machines are discrete dynamic systems. Although Turing's analysis is valid only for a computor with the above mentioned restrictions, Gandy's analysis is applicable for any machines. This difference in quite remarkable. Recall, a human cannot distinguish complicated symbol sequences at a glance. Moreover, a human can do only one mechanical task at a time. For example, write one symbol at a time. However, there are physical machines that are capable of doing certain things that Turing computors cannot. These are as follows.

(Q1) A machine can perform two tasks at the same time.

(Q2) A machine can exist in multiple "states of mind" at a given time.

It is thus important to distinguish Thesis M from the Church-Turing Thesis. Some of the flawed readings and interpretations mentioned in the opening paragraphs of this article, is closer to Thesis M than they are to the CTT.

It is essential to note, then, that an abstract human being, able to imitate the workings of what is now called Turing Machines, is in no way a characterization of the human intelligence. In fact, Turing claimed quite the opposite. He remarks in (Turing, 1948)

> "If the untrained infant's mind is to become an intelligent one, it must acquire both discipline and initiative. So far we have been considering only discipline… But discipline is certainly not enough in itself to produce intelligence. That which is required in addition we call initiative… Our task

is to discover the nature of this residue as it occurs in man, and to try and copy it in machines."

While Turing states that machines could be taught to have "initiative", this is in no way related to his thesis proposed in 1936.

Although Turing does not refer to any automated machines in (Turing, 1936), a machine that simply automatizes the *mechanical* tasks of the computor, is in effect the same as a Turing's computor. The machine has a scanner which can scan one symbol at a time, a printer that prints a symbol on the tape cell, and equipment to aid the moving of the tape. Here we move the tape left and right instead of moving the scanner and printer (i.e. the tape head) itself. We will assume that an infinite paper supply is available. This machine works exactly like a computer. It has machinery to record its current "state of mind" and a transition function that maps states of mind and scanned symbols uniquely to a new state of mind and action. Formally, such a machine can be defined as follows. A Turing Machine T is a five tuple $< \Sigma, \Gamma, \delta, Q, q_0, q_a, q_r >$ where Q is the set of states, Σ is the input alphabet (i.e. the alphabet of the input symbols), Γ is the tape alphabet($\Sigma \subseteq \Gamma$), $\delta: (\Sigma\ X\ Q) \rightarrow$ (Γ X Q X {L, R}) is the transition function, q_0 is the start state, q_a and q_r are the accept and reject states respectively reaching which the Turing Machine halts. $\{q_0, q_a, q_r\} \subseteq Q$. L and R denote the actions of moving left and R right respectively. Of course, by definition, such a machine will be subjected to the restrictions of the computor. Since Turing's analysis of the computor can so easily be applied on the aforementioned automated machine, many mistakenly think that Turing is making his claim about *all* physical machines. In (Turing, 1939), Turing gave his own version of the Church-Turing thesis in terms of "some purely mechanical process… one which could be carried out by a machine." This raises two

questions. Is Turing referring to "Turing machines"? Did he take Church's review (Church, 1937a) an acceptable one? There is no reference for this in Turing's later works on the same subject. In his review, Church states

> "To define effectiveness as computability by an arbitrary machine, subject to restrictions of finiteness, would seem an adequate representation of the ordinary notion, and if this is done the need for a working hypothesis disappears."

Although one might attempt to justify a yes or no answer to the first question, I hope I have shown why it is impossible for us to answer the second question with any degree of certainty.

In 1936, Emil Post proposed an extremely simple model of computation in his paper "Finite combinatory processes—formulation 1". He conjectured that his model was "logically equivalent to recursiveness" and hence Turing computors. Indeed, later, it was proved so. Post's model consists of a "symbol space" with a "two-way infinite sequence of spaces or boxes." Each box can be in one of the following conditions: "marked" or "unmarked". When a box is "marked", it has a single vertical stroke otherwise it is empty. Initially, a finitely-many boxes are marked. A "worker" who works according to a *fixed* and *finite* "set of directions" starting from the first instruction on the list and proceed sequentially. He must work in one box at any given time. The "operations" or "basic acts" performed by the worker are determining if a box is marked or unmarked, marking a (unmarked) box, erasing the mark from a (marked) box, moving to the box immediately to left or right of the box he is currently working in. There is also a Stop instruction that marks the end of work for the post-worker. It is essential to note all the worker needs to do

is follow the list of instructions blindly without having any understanding of it. This shows the simplicity of the Turing computor. Both, the post worker and the Turing computor are humans with the natural restrictions imposed by the boundedness and the locality conditions.

Turing's model of computation is, in a sense, the most powerful model of computation. The modern computer uses a variant of the von Neumann architecture, named after John von Neumann, which was developed in 1945, and is very close to Turing's Universal Machine. Since 1936, many models of computation have been proposed, for example, register machines, multi-tape Turing machines (a Turing machine with more than one tape), etc. However, all of them are Turing equivalent, i.e., the models are not more powerful than Turing Machines. Turing's proposal in 1936 is thus not only simple and elegant, but, in a very important way, is foundational to the very notion of computation. This often leads to misunderstanding the CTT and misconstruing Turing.

It is true that we see a difference in Turing's opinions before and after the war. It seems Turing revised some of his views on intelligence, learning and the human mind. Hodges states that this change was due to the events at Hut 8 in Bletchley while solving the problem of the Naval Enigma,

> "My guess is that there was a turning point in about 1941. After a bitter struggle to break U-boat Enigma, Turing could then taste triumph. Machines turned and people carried out mechanical methods unthinkingly, with amazing and unforeseen results. ... [I] suggest that it was at this period that [Turing] abandoned the idea that moments of intuition corresponded to uncomputable operations. Instead, he decided, the scope of the

computable encompassed ... quite enough to include all that human brains did, however creative or original" (Hodges, 1977)

However, these differences still fail to justify the far fetched variants of the CTT penned by Churchland and the like.

For these reasons, we reject the unfounded, unjustified and far fetched claims mentioned at the onset. The Church-Turing Thesis *does not* make any claims about the limitations of the human intelligence, human mind or its structure.

Bibliography

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

Church, A. (1937). Review of [Turing 1936]. *Journal of Symbolic Logic, 2*(1), 42-43.

Church, A. (1937a). Review of Turing [1936]. *Journal of Symbolic Logic, 2*, 42.

Churchland, P. M., & Churchland, P. S. (1990). Could a Machine Think? *Scientific American, 262*(Jan), 26-31.

Copeland, B. J., & Proudfoot, D. (1999). Alan Turing's Forgotten Ideas in Computer Science. *Scientific American, 253*(4), 98-103.

Gandy, R. (1980). Principles for mechanisms. 124.

Hilbert, D., & Ackermann, W. (1928). *Foundations of Theoretical Logic (Grundzüge der theoretischen Logik).*

Hilbert, D., & Ackermann, W. (1928). *Grundzüge der Theoretischen Logik.* Springer.

Hodges, A. (1977). *Turing: A Natural Philspoher.* London: Phoenix.

Hodges, A. (1983). *Alan Turing: The Enigma.* New York: Simon and Schuster.

Hodges, A. (2006). Did Church and Turing Have a Thesis about Machines?

Kleene, S. C. (1967). *Mathematical Logic.* New York: Wiley.

Searle, J. (1992). *The Rediscovery of the Mind.* Cambridge, Mass: MIT Press.

Seig, W. (2008). On Computability. In *Handbook of the Philsophy of Science. Philsophy of Mathematics* (pp. 525-623).

Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (2)*, *42*, pp. 230-265.

Turing, A. M. (1939). Systems of Logic Based on Ordinals. *Proceedings of London Mathematical Society, 2*(45), 161-228.

Turing, A. M. (1948). *Intelligent Machinery.* National Physical Laboratory Report. Retrieved from http://www.AlanTuring.net/intelligent_machinery

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*(49), 433-460.

Turing, A. M. (1992). Intelligent Machinery. In D. C. Ince, *Collected Works of A. M. Turing — Mechanical Intelligence.* (pp. 107-127). North Holland.