Philosophia Scientiæ
Travaux d'histoire et de philosophie des sciences

16-3 | 2012
Alan Turing

Cleland on Church's Thesis and the Limits of Computation

Clayton Peterson and François Lepage

Electronic version
URL: http://journals.openedition.org/philosophiascientiae/772
DOI: 10.4000/philosophiascientiae.772
ISSN: 1775-4283

Publisher
Éditions Kimé

Printed version
Date of publication: 1 November 2012
Number of pages: 69-85
ISBN: 978-2-84174-603-3
ISSN: 1281-2463

Electronic reference
Clayton Peterson and François Lepage, « Cleland on Church's Thesis and the Limits of Computation », *Philosophia Scientiæ* [Online], 16-3 | 2012, Online since 01 November 2015, connection on 04 May 2019. URL : http://journals.openedition.org/philosophiascientiae/772 ; DOI : 10.4000/philosophiascientiae.772

Cleland on Church's Thesis and the Limits of Computation

Clayton Peterson
Université de Montréal (Canada)
François Lepage
Université de Montréal (Canada)

Résumé : Cet article se veut une critique de la thèse défendue par [Cleland 1993], laquelle soutient que la thèse de Church doit être rejetée puisque les limites du calcul dépendent de la structure physique du monde. Dans un premier temps, nous offrons un (très) bref aperçu de la thèse de Church puis nous présentons l'argument de Cleland. Par la suite, nous proposons une analyse critique de son argument, ce qui nous amènera à faire quelques distinctions conceptuelles par rapport aux notions qui concernent la calculabilité. Finalement, nous montrons que les limites du calcul ne sont pas physiques mais bien logiques. En résumé, notre argument est que les limites du calcul sont déterminées en partie par le fait qu'une procédure effective doit pouvoir être décrite de manière finie.

Abstract: Even though Church's thesis is widely accepted among mathematicians, it is nonetheless controversial. In this paper, we argue against the position of [Cleland 1993], which defends that Church's thesis must be rejected because the limits of computation depend upon the physical structure of the world. We first give a brief overview of Church's thesis and then we present Cleland's argument. We then propose a critical analysis of Cleland's argument, which will involve some conceptual distinctions regarding the notion of computability, and finally we will show that the limits of computation are not physical but logical. In short, we argue that computation is limited by the fact that an effective procedure must be described in a finite way.

Philosophia Scientiæ, 16 (3), 2012, 69–85.

1 Church's thesis and Turing machines

Church's thesis emerged in reaction to the work done by various mathematicians in the 1930's. In trying to develop a reliable method to solve mathematical problems, individuals such as Church, Gödel, Kleene and Turing, which respectively proposed the lambda calculus, the Recursive functions, the Formal systems and the Turing machines, proposed decidable methods to compute functions [Immerman 2008, 1]. A function $f : X \longrightarrow Y$ is an abstract relation between two sets X and Y (where X is the domain and Y the image of f) which associates at most one member of Y to a member of X. Put differently, a function f is a set of ordered pairs $< x_i, y_j >$, with $x_i \in X$ and $y_j \in Y$, and can be represented as such: $f = \{< x_i, y_j >: x_i \in X \wedge y_j \in Y\}$. A function can be *total*, meaning that each member of f's domain will be paired with a member of its image, or it can be *partial*, meaning that only members of a proper subset of f's domain will be associated with a member of its image.

The notion of 'computing' a function is an intuitive one. A function $f(x)$ is said to be computable if there is an effective procedure that could provide one the value of $f(x)$ from the value of x. The basic idea that lies behind the notion of an 'effective procedure' is that it must be a list of instructions that enables one to compute $f(x)$ step-by-step when starting from x in a finite number of steps. This list of instructions must be non ambiguous and there must be only one rule to apply at each step of the computation. A function $f(x)$ is thus said to be *effectively computable* if there is such a procedure that could (in principle) enable one to obtain $f(x)$'s value for each argument x member of f's domain [Boolos 2007, 23].

In order to give a formal representation of the notion of 'effective computability', Alan Turing [Turing 1936] proposed an ideal machine which is now known as a Turing machine. The idea was to obtain an effective procedure that could enable one to compute some functions from $\mathbb{N}$ to $\mathbb{N}$. A Turing machine is composed of a finite number of states

$$\{q_1, \ldots, q_n\}$$

and can do four operations, namely 'go left', 'go right', 'write S_0' or 'write S_1':

$$L, R, S_0, S_1.$$

It is possible to picture a Turing machine as being a small box with four wheels that moves on a tape which is separated in a denumerable number of squares [Boolos 2007, 25]. A Turing machine can be represented by its (finite) list of instructions

$$q_0 S_0 S_1 q_1, \ldots, q_n S_1 S_1 q_n,$$

where each quadruple is read "if the machine is in state q_i and reads S_j, then do S_k and be in the state q_l. Since a Turing machine is a finite list of non

ambiguous instructions that enables one to compute step-by-step the value of a function by applying one rule at each step, it follows that a Turing machine is an effective procedure to compute some functions $f : \mathbb{N} \longrightarrow \mathbb{N}$.

Even though a function can be viewed as an abstract relation between two sets, the work done in the 1930's was not meant to apply to all type of functions but only to arithmetical functions, that is functions which pair natural numbers. Every definition of computation that has been proposed by the aforementioned mathematicians has been proved to be equivalent to one another [Immerman 2008, 3]. While Church proved that the class of lambda-computable functions is equivalent to the class of recursive functions, Turing showed that there is an equivalence between Turing- and recursive-computability, which proved the equivalence between the three definitions of computation [Copeland 2002, 4]. Seeing that these definitions of computation are equivalent, Church proposed that they are an adequate representation of our intuitive notion of an effective computation. Thus, Church's thesis, also known as the 'Church-Turing thesis'—since Turing proposed that every function that is effectively computable is Turing-computable—, means that a function is effectively computable if and only if it pertains to the class of recursive functions. In other words, Church's thesis means that a function is effectively computable if and only if it is Turing-computable. [1]

2 Cleland's argument

Even though Church's thesis is widely accepted by mathematicians—there is no widely accepted counter example—, it is nonetheless controversial. Among those who argue against it lies Carol E. Cleland. In her article of 1993 [Cleland 1993], she proposes an argument against Church's thesis. The rationale behind Cleland's argument, which is actually the central point that we want to discuss in this paper and has been maintained throughout her career [see Cleland 1995, 2001, 2002], is that computation's limits are physical rather than logical:

> ... the limits to computation are not to be found in abstract Turing machine theory; for the limits to computation are causal, as opposed to logical. [Cleland 1993, 286]

Since its first formulation, Church's thesis has been interpreted in three different ways [Cleland 1993, 285], namely:

1. a number theoretic function is effectively computable if and only if it is Turing-computable;

1. Church's thesis is intimately linked to the halting problem. Since it can be proved that the halting function is not Turing-computable [Boolos 2007, 40], it follows that if one is able to find an effective procedure that can compute the halting function, then Church's thesis is false.

2. any function is effectively computable if and only if it is Turing-computable;
3. any effective procedure, including mental or physical phenomena, can be reduced to the effective procedure of a Turing machine.

In order to argue against these three interpretations of Church's thesis, Cleland's strategy is to define a notion of effective procedure which is superior to Turing machines. Indeed, she wants to show that there is a kind of effective procedure—the mundane procedure—which could in principle enable one to compute functions which are not Turing-computable. Her aim is not to show that Church's thesis is false but rather to show that since computation's limits are physical, there is a strong possibility that the set of Turing-computable functions is a proper subset of the set of functions which are 'effectively computable'.

Since mundane procedures can be applied to physical processes, it follows that they can produce empirical results, unlike Turing machines which can only give theoretical or abstract results. Therefore, mundane procedures are superior to Turing machine's procedure since they can "generate a causal process when followed [Cleland 1993, 286]". While a mundane procedure does not differ from the procedure of a Turing machine regarding its instructions—since both consist of a list of instructions which must be done in a temporal order [Cleland 1993, 288, 291]—the distinction between the two resides in the *effectiveness* of a mundane procedure. Indeed, Cleland considers a mundane procedure to be effective "if following it always results in a certain kind of outcome" [Cleland 1993, 291]. However, the effectiveness of a Turing machine's procedure does not depend upon the outcome but only upon the formulation of its instructions.Thus, a mundane procedure differs from a Turing machine's procedure since it will only be effective if it gives us, by means of causal processes, a result in the physical world.

This notion of 'effectiveness' implies that an effective mundane procedure must take into account the physical structure of the world in order to achieve its goal, since the list of (mundane) instructions needs to order actions that will make a causal chain and result in a certain causal process:

> In general, the effectiveness of a mundane procedure for a given end depends upon what kinds of causal processes would be generated, were the procedure to be followed. [Cleland 1993, 294]

From this conception follows that an effective mundane procedure is not effective for all possible scenarios. The effectiveness of a mundane procedure depends upon the physical context: instructions that aim at making a fire would not be effective under water! In short, effective mundane procedures differ from Turing machine's procedures since they require causality in order to achieve their goal.

With that being said, Cleland applies the notion of mundane procedure to the three aforementioned interpretations of Church's thesis. Since a Turing machine's procedure does not have any causal dimension, it follows that the third interpretation is obviously wrong since a physical process cannot be obtained by a Turing machine [Cleland 1993, 301]. [2] In short, Cleland's reasoning is that:

> [...] the actions performed by a physical system are different from those performed by a Turing machine. Unlike physical actions, Turing machine actions are purely formal and have no causal consequences. As a consequence, no Turing machine can realize the causal activity of a physical system. But some of the things that are done by physical system require physical activity. [Cleland 1993, 303]

Thus, she rightly rejects the third interpretation of Church's thesis since there are things that can be done by effective mundane procedures, but cannot be done by any Turing machine [Cleland 1993, 304].

After having rejected the third interpretation, Cleland turns to the second and the first ones. In order to reject the second interpretation, which concerns any type of function, she points out that a Turing machine cannot compute the identity function $i : \mathbb{R} \longrightarrow \mathbb{R}$ [Cleland 1993, 306]. Obviously, a Turing machine is not able to compute a function from $\mathbb{R}$ to $\mathbb{R}$ since there is no finite representation of the real numbers. Considering that a real number can be represented by a denumerable sequence of natural numbers, it follows that a Turing machine would not even be able to read the argument of the function, and thus would never be able to finish its first step.

However, and this is where the argument is seriously going astray, Cleland argues that a mundane procedure would be able to do so. If we suppose a 'possible world' where space and time are continuous, then a particle that moves in this continuum could be taken as a function which computes $i(x)$ since it would be pairing each point of the time with a point in the space [Cleland 1993, 306]. In other words,

> [...] if we let the places and times concerned stand for the appropriate real numbers, then an object moving at unit speed can be interpreted as computing $i(x)$. [Cleland 1993, 307]

Therefore, insofar

> [...] as it is possible for an effective mundane procedure to initiate a genuinely continuous causal process, it is possible for an

2. Let us note that, to the best of our knowledge, no decent mathematician nor any philosopher of the 1930's has supported that all physical or mental phenomena could be reproduced by a Turing machine—or any analytical function for that matter.

> effective mundane procedure to compute a function that could be computed by no Turing machine. [Cleland 1993, 308]

Finally, taking into account that a physical process can do more than a Turing machine, Cleland concludes that there

> [...] is no conceptual basis for even the modest supposition that no effective mundane procedure could compute a so called uncomputable number-theoretic function. [Cleland 1993, 310]

To sum up, the rationale behind Cleland's argument in order to reject Church's thesis is that the limits of computation are not logical but causal [Cleland 1993, 309]. Since a mundane procedure is superior to a Turing machine's procedure, it follows that there is not any good reason that could convince one that a mundane procedure could not compute more than a Turing machine. A mundane procedure is characterized by causality and this enables it to mirror a more complex structure than a Turing machine could do. Considering that it is possible to imagine a mundane procedure which could—in principle—accomplish more than a Turing machine's procedure, Cleland concludes that we have a good reason to doubt Church's thesis, and thus "shift the burden of the proof" to its defenders.

3 Critical analysis

In short, Cleland's strategy is to show the 'possibility' of Church's thesis falsity. In order to do so, she argues that the limits of computation are physical. Since Turing machines cannot generate causal processes but causal processes can simulate Turing machines, she concludes that it is perfectly rational to assume that mundane machines are, in principle, able to compute more things than Turing machines could. Rather than proving that Church's thesis is false, she "shifts the burden of the proof" to those in favor of it. But this strategy is bogus: Church's thesis is not provable! On the one hand, the notion of 'effective computation' is an intuitive one and can be interpreted in many different ways. Thus, it is not possible to prove that every thing which is effectively computable is Turing-computable. However, it has been shown that all notions of effective computability developed in the 1930's are equivalent. While it is not a proof of Church's thesis, it is clearly an argument in its favor. [3] On the other hand, there are $2^{\aleph_0}$ functions from $\mathbb{N}$ to $\mathbb{N}$. How could we be able to prove that each function $f \subset \mathbb{N} \times \mathbb{N}$ which is computable—that is for which there is an effective procedure—is Turing-computable? We only know what an 'effective procedure' means if we accept Church's thesis! The only thing we can do with Church's thesis is to prove its falsity and show that there is at

3. We will return on that point in section 3.4.

least one function which is effectively computable but not Turing-computable. Therefore, "shifting the burden of the proof" of Church's thesis does not really mean anything.

For the sake of the argument, let us assume that "shifting the burden of the proof" is an acceptable strategy. The question that needs to be answered is this one: does Cleland's argument give us good reasons, that is rational reasons, in order to doubt Church's thesis? In our view, the answer is no. In order to show that, our analysis will be aiming at two points. First, we will show that Cleland's reasoning is seriously flawed. Before criticizing her main argument regarding the computation of $i(x) : \mathbb{R} \longrightarrow \mathbb{R}$ within a physical structure isomorphic to the formal structure of the real numbers, we will briefly criticize her argument regarding the third interpretation of Church's thesis, even though this is not the central point of our critic. The core of our critic regards the argument against the second interpretation of Church's thesis, which is used by Cleland in order to show the 'possibility' of the first interpretation's falsity. After showing that this argument relies upon a false premise, we will show that her analysis relies upon an incomprehension of some notions. This will lead us to make some conceptual clarifications regarding both notions of 'computability' and 'effectiveness'. In the second part of the critic, we will discuss the core of Cleland's argument, namely the thesis such that the limits of effective computation are physical rather than formal.

3.1 The third argument

Before analyzing the core of Cleland's reasoning, we thought that it would be appropriate to note some points regarding the argument she proposes against the third interpretation of Church's thesis. In order to show that the third interpretation of Church's thesis is false—that is any effective procedure, including mental or physical phenomena, can be reduced to the effective procedure of a Turing machine—, Cleland argues that since a physical phenomenon requires causality, it follows that a Turing machine, which lacks causality, cannot reproduce a physical phenomenon.

> Insofar as the action-kinds specified by a Turing machine program are purely formal, they have no causal consequences and, hence, cannot, when performed, generate a causal process. The upshot is that there are no Turing machines which are effective for making Hollandaise sauce. [Cleland 1993, 301]

This is true enough: a Turing machine, which is an abstract machine—by opposition to a real/concrete machine—, cannot act in the world. She makes this point because she argues that it is widely accepted that Church's thesis applies to a class of functions wider than the class of number theoretic functions.

> The thesis is frequently taken by computer scientists to represent an ultimate limit to *all* computation and, hence, has been construed as applying to functions in general, as opposed to just the number theoretic functions. [Cleland 1993, 284]

However, while the first part of that affirmation is true, the second is misleading. What is true is that there are many functions that can be reduced to number theoretic functions. Let us take an extreme example: the brain. If we assume that the basic mechanisms of the brain are discrete, then the brain's functioning can be viewed as inputs, outputs and discrete states, which can be simulated by a Turing machine (trivial since it is finite).

Now, regarding the argument of causality, let us suppose that there is a mundane machine that follows a mundane procedure in order to make Hollandaise sauce. Suppose that there are compartments in which we can put the ingredients and so on and so forth. We push a button and then there is a finite sequence of operations which gives one Hollandaise sauce. Nobody is arguing that this is a Turing machine: this is a food processor! However, even if the mundane machine cannot be simulated by a Turing machine, its processor can.

Thesis. A mundane machine is a Turing machine which is plugged into a food processor.

It would be interesting to have a clear counterexample that contradicts this thesis.

3.2 The computability of $i(x)$

Cleland's main argument in order to show the possibility of Church's thesis falsity regards the computability of $i(x) : \mathbb{R} \longrightarrow \mathbb{R}$. In order to show that some mundane machines could in principle be able to compute more things than Turing machine could, she tries to show that a mundane machine would, in an adequate physical structure, be able to compute $i(x)$.

> Let us imagine a possible world in which space and time are both genuinely continuous, i.e., a world in which any interval of space and time, no matter how small, has an uncountably infinite number of places and times; it is worth noting that many people believe that the actual world is such a world. Let us further suppose that motion is a continuous process, more specifically, a process of occupying different places at different times in such a way that every intervening place along the path of motion is (at least) passed through; in other words, moving objects do not jump over (to speak metaphorically) intervening places. Thus, a moving object

> can be thought of as pairing places with times. But the times and places concerned are, by assumption, uncountably infinite in number. As we have seen, no Turing machine could mirror such a pairing. This certainly suggests the possibility that there are functions which can be computed by physical systems which can't be computed by Turing machines. [Cleland 1993, 306]

The first thing to say is that this argument relies upon a false premise. Indeed, in such a world, $\mathbb{R}$ would be well ordered by '$\leq$', which is mathematically false! The idea that an object would be able to pass through each point of the continuum in the order—without "jumping over" any point—is absurd. This is Zeno's argument all over again! It would be impossible that an object moving at unit speed in a continuum pass through $[0, 1]$ going from one point to another respecting the order $\leq$. Even if one accepts the axiom of choice and the existence of a well ordering of $[0, 1]$, such an order would have nothing to do with $\leq$. On a continuous interval such as $[0, 1]$, every x_i has neither an immediate predecessor nor an immediate successor. The same applies if we assume that time is also continuous: the object would not be able to pass through every point of the continuum. Even if we accept that a continuous interval of time is isomorphic with a continuous interval of linear space, this is not an argument in favor of the existence of an enumeration of the space interval's points (nor of the time interval's moments). There is no possible world such as Cleland describes it. The history of mathematics shows us that the notion of a 'genuine continuum' is illusory. All notions of continuum which are consistent are those of Cauchy, Dedekind, Bolzano, Weierstrass, Borel, etc., which rely on computable approximations. For example, $f(x)$ is continuous at x_0 means that $\forall \epsilon > 0$, $\exists \delta > 0$ such that $|x - x_0| < \delta$ implies that $|f(x) - f(x_0)| < \epsilon$. A genuine continuum is an illusion. Even if most (classical) physicists consider that space, time, mass, etc., are continuous, every calculation they do in $\mathbb{R}$ is an approximation, and thus is Turing-computable. Computing π's value for the n^{th} decimal can be done by a Turing machine.

But even without this remark Cleland's objection is flawed. Indeed, there seems to be a confusion between the *definition* of a function and its *computation*. At first, she says that:

> [...] an actual computation of a function must mirror the detailed formal structure of the function: Each and every element in the function's domain must become matched to the appropriate element in its range. [Cleland 1993, 305]

Then, she continues and says that since

> [...] no Turing machine can [match to all of the values in the domain their assigned values in the range], no Turing machine is capable of precisely mirroring the detailed formal structure of

> a function from the reals onto the reals. As a consequence, no Turing machine can, strictly speaking, compute even the identity function $i(x)$ which assigns to each real number x the value x itself. [Cleland 1993, 306]

Finally, she concludes that a mundane machine could, unlike any Turing machine and in an appropriate physical structure, compute the function $i(x)$.

> What is required for the computation of such a function is a way of achieving the requisite assignment of values to arguments, and such a way would be provided by a process of motion such as that described above. Indeed, if we let the places and times concerned stand for the appropriate real numbers, then an object moving at unit speed can be interpreted as computing $i(x)$. [Cleland 1993, 306–307]

In these quotes, Cleland is confusing the *computation* of a function with its *definition*. As we know, a function can be defined analytically (e.g., $f(x) = 3x + 2$) or by stating the set of its ordered pairs (e.g., $f = \{< 3, 7 >, < 5, 2 >, < 7, 8 >\}$). However, in general, *pairing* the numbers is not equivalent to *computing* them. While it is possible to define a function by *paring* a number x with a number y, the computation of a function is the procedure by which we obtain y from x. Saying that $f(7) = 49$ for $f(x) = x^2$ is not the computation of f. One can know that $f(7) = 49$ without being able to compute it! The computation of a function is done by an effective procedure, which is a description of the way of pairing. When the function is partial and the set of ordered pairs is finite, an effective procedure can be trivially constructed from the set of ordered pairs (the definition of the function). However, while such a definition provides one (trivially) with a mean to effectively compute a function in finite cases, it is important to note the distinction between the *definition* of a function and its *computation*. Even if an effective procedure can be constructed from a finite set of ordered pairs, it makes no sense in an infinite case, in particular for a function $f : \mathbb{N} \longrightarrow \mathbb{N}$. A set of ordered pairs cannot be a definition for a function which has a denumerable (or non denumerable) domain. The description of a set of ordered pairs is finite. Thus, it would not provide one with the means to compute the function for any member of its domain. It would only be possible to obtain an effective procedure that would enable one to compute the function over a finite subset of its domain, namely the subset represented by the finite set of ordered pairs. In short, computing a function is not equivalent to "matching each and every element in the function's domain to the appropriate element in its image". This is neither a way to define nor to compute a function in infinite cases. The distinction between the *computation* of a function and its *definition* is a fundamental one. A physical process by which it is possible to pair numbers is not a way to compute a function. In the same manner that the graph of a function is not its computation—it is only a way to represent it—, constructing a set of ordered pairs is not either.

Moreover, she does not seem to grasp the meaning of the concept of 'effectively computable'. On the one hand, she makes a non necessary distinction between the effectiveness of a Turing machine and the effectiveness of a mundane machine. Basically, Cleland argues that the effectiveness of a mundane procedure is characterized by its outcome. Indeed, she says that "the effectiveness of a mundane procedure depends upon the kind of causal process that would be generated" (section 2). But what Cleland is referring to is the *efficiency* of the procedure rather than its *effectiveness*. A procedure is said to be *effective* when it can be described in an non ambiguous way, meaning that the list of instructions must be finite and that one must be able to know what to do at each step of the computation. However, a procedure is *efficient* if it gives one the desired result. With that being said, there is no reason to assume that the effectiveness of a mundane procedure depends upon its potential outcome. The effectiveness of a procedure depends upon the clarity and the non ambiguity of its instructions. As a Turing machine, a mundane procedure can be *effective* without being *efficient*. A Turing machine which computes $f(x) = x + 1$ is effective and efficient for computing that function over a finite subset of $\mathbb{N}$, but it is not efficient for computing $g(x) = 5x^3$. However, it is still effective nonetheless. The same applies to mundane procedures: a mundane procedure that makes Hollandaise sauce is effective and efficient for doing so. However, while it is not efficient in order to make a fire, it will still be effective, in the sense that its list of instructions will be well defined. The procedure of a Turing machine does not differ from the procedure of a mundane machine: in both cases, if we want a result, we will need to *apply* the procedure. While the *effectiveness* of a procedure depends upon the clarity and the non ambiguity of its list of instructions, the *efficiency* of a procedure depends upon the outcome.

On the other hand, she does not seem to understand the concept of 'computability' that lies behind Church's thesis. Indeed, Church's thesis concerns functions such that the members of f's domain and image are described in a finite and non ambiguous way, meaning that the members can be distinguished unequivocally. A function that answer this condition is computable if and only if it can be described by a non ambiguous procedure—an algorithm—which enables one to transform the description of a member of the domain into the description of a member of the image within a finite number of steps. While these two conditions may not be sufficient, it only matters that they are both necessary conditions for computability.[4] The point is that Cleland herself admits that $\mathbb{R}$ does not satisfy the first condition. A function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is not computable since there is no finite representation of the real numbers. So the question is: what does Cleland mean when she affirms that $i(x) : \mathbb{R} \longrightarrow \mathbb{R}$ could be *computed* by a mundane procedure in an appropriate physical struc-

4. Each function $f : X \longrightarrow Y$ that satisfies these two conditions is Turing-computable by gödelisation.

ture? The answer to that question seems to be 'nothing specific'. Cleland's conception of computation is naive and unrealistic: she argues that mundane procedures could compute more functions than Turing machines because "computing a function requires that each and every argument of the function be matched to its assigned value [Cleland 1993, 307]." There are two things going on here. First, as we said earlier, constructing a set of ordered pairs is a way to *define* a function, not to *compute* it. Second, with such an understanding of computability, neither $i(x)$ nor any function $f \subset \mathbb{N} \times \mathbb{N}$ could be computed! Insofar as it is impossible to "match each and every argument of the function to its assigned value" for functions in the natural or the real numbers, it is impossible to *compute* these functions in Cleland's sense. To match each and every element of the domain to its assigned value would be an *actual* computation of the function. Thus, such an understanding of 'computability' is not suitable for total functions. [5]

3.3 The argument against Church's thesis

The rationale of Cleland's reasoning is that computation's limits depend upon the physical structure of the world. In order to argue against the 'real' Church's thesis, which corresponds to the first interpretation, [6] she uses the aforementioned example regarding $i(x)$ to make her point. Yet, her argument is never totally and explicitly mentioned. Anyhow, assuming that she is talking about functions into $\mathbb{N}$, the following quote is the less obscure formulation of her argument:

> Whether a particular function can be in fact computed depends, however, upon the detailed causal structure of our world, i.e., upon whether there exist causal processes having the requisite formal structure. It is very likely that not all functions can be computed in our world. It is also possible that some functions that cannot be computed in our world can, nonetheless, be computed in other causally possible worlds, i.e., worlds whose causal structure differs in important ways from our world. Such functions could be said to be in theory, but not in fact, computable. [Cleland 1993, 309]

The argument, if we understand her correctly, can be rephrased as following:

5. Let us also point out that since there can only be finite number of functions that are *actually* computed, it would imply that there are (considerably) more Turing-computable functions.

6. A function $f : \mathbb{N} \longrightarrow \mathbb{N}$ is effectively computable if and only if it pertains to the class of recursive functions.

P1 The limits of computation depend upon the physical structure of the world.
P2 A function can be *in fact* computed if the world has the appropriate physical structure.
P3 A function can be *in theory* computed if it is possible to imagine a world that could mirror its formal structure.

∴ There are functions that are *in theory* computable but are not Turing-computable.

Even with Cleland's understanding of 'computability', it makes no sense to say that a total function is *in fact* computable. Thus, the distinction between *in fact* and *in theory* is unnecessary: to say that a function is computable means that it has the potential of being actually computed over a finite interval of the domain. The problem with this argument is that the only example that Cleland proposes regards a function into the reals. But as we saw, the notion of computability does not even apply to the real numbers, nor does Cleland's understanding of it. So, does this argument show the possibility of Church's thesis falsity? Notwithstanding the fact that since Church's thesis is not provable, it follows that it is per se possible that it is false, Cleland's argument does not give us a good reason to doubt it. We would like to have a clear counterexample which contradicts Church's thesis.

3.4 The limits of computation

Notwithstanding the aforementioned analysis of Cleland's argument, it is important to bring forward an actual argument in favor of Church's thesis. Contra Cleland, the limits of computation are not to be found within the physical structure of our world. The limits of computation are logical, which makes Church's thesis highly plausible. The limits of computation only depends upon the procedure, and this can be shown even within Cleland's own argument.

The argument she proposes in order to doubt Church's thesis' plausibility relies upon a distinction between *in theory* and *in fact* computability. An example of a function which is *in theory* computable but not *in fact* computable would be $f(x) = 2^{x!}$.[7] For the sake of the argument, we will assume that the distinction she makes is an adequate one. In short, she argues that it is possible to imagine a world in which there are some functions that are *in theory* computable but not Turing-computable. But what does it mean for a function to be *in theory* computable? It means that there can be an effective mundane procedure which is not efficient in our world but could be efficient in another

7. Note that this function is however Turing-computable since it can be defined recursively.

world whose physical structure differs from ours. The question is: since a function can be *in theory* computable without being *in fact* computable, is it correct to assume that computability depends upon the physical structure of our world? No, because a function can be computable without being *in fact* computable. Therefore, Cleland's conception of computability does not depend upon the efficiency of the procedure but rather depends upon its effectiveness (thus the non necessary distinction between mundane procedures and Turing machine procedures). In other words, the physical structure of the world only matters for the *in fact* computability, and so the physical structure of the world is not relevant for Cleland's notion of *in theory* computability. Cleland's notion of 'computability' does not depend upon the physical structure of the world because it does not depend upon the efficiency of a procedure. A function can be computable even if there is no result in the world. Thus, even in the case of mundane procedures, the notion of 'computability' does not depend upon causality but depends only upon the effectiveness of the procedure. Computability, at the end, depends upon the effectiveness of the procedure, and not its efficiency.

At this point, we tried to show that Cleland's argument is flawed. Now, we want to show that Church's thesis is plausible and that computation's limits are logical. In short, our argument relies upon the following thesis:

Thesis. If a function is computable, then its computation process can be described.

Note that the converse is not true. For example, the halting function can be described but is not, according to Church's thesis, computable. As we said earlier, the two following criteria are necessary conditions to computability.

1. The members of f's domain and image can be described in a finite and non ambiguous way, meaning that the members can be distinguished unequivocally.
2. The function f can be described by a non ambiguous procedure—an algorithm—which enables one to transform the description of a member of the domain into the description of a member of the image within a finite number of steps.

In other words, it makes no sense to say that a function is computable if it cannot be described! If we accept this thesis, it follows that Church's thesis is highly plausible. Let us have a little thought experiment in order to illustrate this point.

As we saw earlier, even in the case of mundane procedures, computability relies upon the effectiveness of the procedure, meaning that a function is mundane-computable if and only if there is a finite and non ambiguous mundane procedure that could enable one to compute that function in some

possible world. With such a definition of computability, it is easy to think of an abstract mundane machine that is able to follow a mundane procedure. This conception of an abstract mundane machine is perfectly consistent with Cleland's conception of '*in theory* computability'. As a Turing machine can informally be represented by a little box on four wheels which prints or erases 'strokes' on a denumerable tape, let us imagine a mundane machine as being a little human in an infinite space, an infinite time, with infinite resources and which does not make any mistake. This machine has three states, it can:

q_0 : stop
q_1 : receive information
q_2 : act.

For the sake of the example, let us see the actions of the machine like phenomena: the machine causes and perceives phenomena. Since each phenomenon can be described, it follows that there is a denumerable number of phenomena. Indeed, a description is a finite combination of words, which are finite combinations of letters. So even if we accept a denumerable alphabet, there will still be only a denumerable number of finite descriptions.[8] Thus, if we accept that there can only be a denumerable number of phenomena that can be described, it follows that our machine will be able to causes and perceives only a denumerable number of phenomena:

$$S_0, \ldots, S_i, \ldots$$

No matter how many possible worlds there can be, we can assume that each mundane machine can be described. Moreover, we do not have to take into account the physical structure of these worlds because '*in theory* computability' only requires the effectiveness of the mundane procedure—and not its efficiency. It is possible to represent a mundane machine by its list of instructions. Thus, let us represent a mundane machine by a list of quadruples

$$q_i S_j S_n q_m.$$

with $0 \leq i; m \leq 2$ and $j; n \in \mathbb{N}$. Since a mundane machine can be represented by its list of instruction, it follows that there are as many mundane machines as there are lists of instructions. If we accept that a non ambiguous list of instruction is finite, it follows that there is a denumerable number of mundane machines.

8. This is not controversial: it would not be possible to know every letter of a non denumerable alphabet and it does not make sense to speak of a 'denumerable' description of a phenomenon.

Proposition 1. There is a denumerable number of mundane machines with three states.

Proof. There is a denumerable number of possible instructions. Indeed, there is a denumerable number of quadruples since there is 3 possible q_i and $\aleph_0$ possible S_j, and so there is $3 \cdot \aleph_0 \cdot \aleph_0 \cdot 3 = \aleph_0$ possible quadruples. Since the list of instruction is finite, it follows that there are ${\aleph_0}^m$ lists of length m, and so there are ${\aleph_0}^1 + \cdots + {\aleph_0}^m + \cdots = \aleph_0$ possible lists. □

Proposition 2. There is a denumerable number of mundane machines with $\aleph_0$ states.

Proof. Even if we suppose that a mundane machine can have a denumerable number of states, there will not be more mundane machines. There will be $\aleph_0 \cdot \aleph_0 \cdot \aleph_0 \cdot \aleph_0 = \aleph_0$ possible quadruples. By the same reasoning, there will be ${\aleph_0}^1 + \cdots + {\aleph_0}^m + \cdots = \aleph_0$ possible lists. □

Therefore, if we accept that it must be possible to describe an effective procedure, it follows that there can only be a denumerable number of mundane machines. Let f_T and f_M be respectively the class of Turing- and mundane-computable functions. We know that both cardinality of f_T and f_M are $\aleph_0$. Cleland assumes that everything which is Turing-computable is mundane-computable and that there are functions which are mundane-computable but not Turing-computable, that is $f_T \subset f_M$. The interesting point is that even though Cleland assumes that mundane machines can compute functions that Turing machines cannot, mundane machines cannot compute *more* functions. The cardinality of the set of functions from $\mathbb{N}$ to $\mathbb{N}$ is $2^{\aleph_0}$. The upshot is that as soon as we accept the two aforementioned conditions for computability, it follows that the cardinality of the set of computable functions is at most $\aleph_0$.

4 Conclusion

So what are the limits of computation? The computability of a function depends upon the possibility of describing in a finite and non ambiguous way the algorithm by which we can go from x's to $f(x)$'s value. The limits of computation are to be found within the formal structure of mathematics, and this ultimately depends upon the manner in which we describe it. Thus, computation is limited by our understanding of mathematics and the way we can describe them. These are two formal limits, and this is why Cleland's thesis must be rejected. The main point we tried to make within this paper was that the limits of computation do not depend upon the physical structure of our world: computation's limits are logical. This can be illustrated by the fact that in order for a procedure to be effective, it must be possible to describe

it in a finite an non ambiguous way. So here lies the limit of computation: the cardinality of the class of computable functions cannot be higher than $\aleph_0$ because the finite description of an effective procedure is a necessary condition for computability. Even though we cannot prove Church's thesis, this gives us a good reason in order to accept it.

Bibliography

Boolos, George

2007 *Computability and logic*, New York: Cambridge University Press, 5th ed.

Cleland, Carol

1993 Is the Church-Turing thesis true?, *Minds and machines*, 3(3), 283–312.

1995 Effective procedures and computable functions, *Minds and machines*, 5(1), 9–23.

2001 Recipes, algorithms, and programs, *Minds and machines*, 11(2), 219–237.

2002 On effective procedures, *Minds and machines*, 12(2), 159–179.

Copeland, Jack

2002 The Church-Turing thesis, *Stanford Encyclopedia of Philosophy*, http://plato.stanford.edu.

Immerman, Neil

2008 Computability and complexity, in *Stanford Encyclopedia of Philosophy*, edited by Zalta, Edward N.

Turing, Alan

1936 On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, 2(42), 230–265.