DOI:10.1145/3198448

In its original form, the Church-Turing thesis concerned computation as Alan Turing and Alonzo Church used the term in 1936—*human* computation.

BY B. JACK COPELAND AND ORON SHAGRIR

The Church-Turing Thesis: Logical Limit or Breachable Barrier?

THE CHURCH-TURING THESIS (CTT) underlies tantalizing open questions concerning the fundamental place of computing in the physical universe. For example, is every physical system computable? Is the universe essentially computational in nature? What are the implications for computer science of recent speculation about physical uncomputability? Does CTT place a fundamental logical limit on what can be computed, a computational "barrier" that cannot be broken, no matter how far and in what multitude of ways computers develop? Or could new types of hardware, based perhaps on quantum or relativistic phenomena, lead to radically new computing paradigms that do breach the Church-Turing barrier, in which the uncomputable becomes computable, in an *upgraded* sense of "computable"? Before addressing these questions, we first look back to the 1930s to consider how Alonzo Church and Alan Turing formulated, and sought to justify, their versions of CTT. With this necessary history under our belts, we then turn to today's dramatically more powerful versions of CTT.

History of the Thesis

Turing stated what we will call "Turing's thesis" in various places and with varying degrees of rigor. The following formulation is one of his most accessible.

Turing's thesis. "L.C.M.s [logical computing machines, Turing's expression for Turing machines] can do anything that could be described as ... 'purely mechanical'."[38]

Turing also formulated his thesis in terms of numbers. For example, he said, "It is my contention that these operations [the operations of an L.C.M.] include all those which are used in the computation of a number."[36] and "[T]he 'computable numbers' include all numbers which would naturally be regarded as computable."[36]

Church (who, like Turing, was working on the German mathematician David Hilbert's *Entscheidungsproblem*) advanced "Church's thesis," which he expressed in terms of definability in his lambda calculus.

Church's thesis. "We now define the notion ... of an effectively calculable

» key insights

- **The term "Church-Turing thesis" is used today for numerous theses that diverge significantly from the one Alonzo Church and Alan Turing conceived in 1936.**
- **The range of algorithmic processes studied in modern computer science far transcends the range of processes a "human computer" could possibly carry out.**
- **There are at least three forms of the "physical Church-Turing thesis"—modest, bold, and super-bold—though, at the present stage of physical inquiry, it is unknown whether any of them is true.**

IMAGE BY NASA, ESA, AND THE HUBBLE HERITAGE TEAM (STSCI/AURA)

Is everything in the physical universe computable? Hubble Space Telescope view of the Pillars of Creation in the Eagle Nebula.

function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a λ-definable function of positive integers)."[5]

Church chose to call this a definition. American mathematician Emil Post, on the other hand, referred to Church's thesis as a "working hypothesis" and criticized Church for masking it in the guise of a definition.[33]

Upon learning of Church's "definition," Turing quickly proved that λ-definability and his own concept of computability (over positive integers) are equivalent. Church's thesis and Turing's thesis are thus equivalent, if attention is restricted to functions of positive integers. (Turing's thesis, more general than Church's, also encompassed computable real numbers.) However, it is important for a computer scientist to appreciate that despite this extensional equivalence, Turing's thesis and Church's thesis have distinct meanings and so are different theses, since they are not intensionally equivalent. A leading difference in their meanings is that Church's thesis contains no reference to computing machinery, whereas Turing's thesis is expressed in terms of the "Turing machine," as Church dubbed it in his 1937 review of Turing's paper.

It is now widely understood that Turing introduced his machines with the intention of providing an idealized

description of a certain human activity—numerical computation; in Turing's day computation was carried out by rote workers called "computers," or, sometimes, "computors"; see, for example, Turing.[37] The Church-Turing thesis is about computation as the term was used in 1936—human computation. Church's term "effectively calculable function" was intended to refer to functions that are calculable by an idealized human computer; and, likewise, Turing's phrase "numbers which would naturally be regarded as computable" was intended to refer to those numbers that could be churned out, digit by digit, by an idealized human computer working ceaselessly.

Here, then, is our formulation of the historical version of the Church-Turing thesis, as informed by Turing's proof of the equivalence of his and Church's theses:

CTT-Original (CTT-O). Every function that can be computed by the idealized human computer, which is to say, can be effectively computed, is Turing-computable.

Some mathematical logicians view CTT-O as subject ultimately to either mathematical proof or mathematical refutation, like open mathematical conjectures, as in the Riemann hypothesis, while others regard CTT-O as not amenable to mathematical proof but supported by philosophical arguments and an accumulation of mathematical evidence. Few logicians today follow Church in regarding CTT-O as a definition. We subscribe to Turing's view of the status of CTT-O, as we outline later.

In computer science today, algorithms and effective procedures are, of course, associated not primarily with humans but with machines. (Note, while some expositors might distinguish between the terms "algorithm" and "effective procedure," we use the terms interchangeably.) Many computer science textbooks formulate the Church-Turing thesis without mentioning human computers at all; examples include the well-known books by Hopcroft and Ullman[24] and Lewis and Papadimitriou.[29] This is despite the fact that the concept of human computation was at the heart of both Turing's and Church's analysis of computation.

We discuss several important modern forms of the Church-Turing thesis, each going far beyond CTT-O. First, we look more closely at the algorithmic form of thesis, as stated to a first approximation by Lewis and Papadimitriou[29]: "[W]e take the Turing machine to be a precise formal equivalent of the intuitive notion of 'algorithm'."

What Is an Algorithm?

The range of algorithmic processes studied in modern computer science far transcends the range of processes a Turing machine is able to carry out. The Turing machine is restricted to, say, changing at most one bounded part at each sequential step of a computation. As Yuri Gurevich pointed out, the concept of an algorithm keeps evolving: "We have now parallel, interactive, distributed, real-time, analog, hybrid, quantum, etc. algorithms."[22] There are enzymatic algorithms, bacterial foraging algorithms, slime-mold algorithms, and more. The Turing machine is incapable of performing the atomic steps of algorithms carried out by, say, an enzymatic system (such as selective enzyme binding) or a slime mold (such as pseudopod extension). The Turing machine is similarly unable to duplicate (as opposed to simulate) John Conway's Game of Life, where—unlike a Turing machine—every cell updates simultaneously.

A thesis aiming to limit the scope of algorithmic computability to Turing computability should thus not state that every possible algorithmic process can be performed by a Turing machine. The way to express the thesis is to say the extensional input-output function $\iota\alpha$ associated with an algorithm α is always Turing-computable; $\iota\alpha$ is simply the extensional mapping of α's inputs to α's outputs. The algorithm the Turing machine uses to compute $\iota\alpha$ might be very different from α itself. A question then naturally arises: If an algorithmic process need not be one a Turing machine can carry out, save in the weak sense just mentioned, then where do the boundaries of this concept lie? What indeed is an algorithm?

The dominant view in computer science is that, ontologically speaking, algorithms are abstract entities; however, there is debate about what abstract entities algorithms are. Gurevich defined the concept in terms of abstract-state machines, Yiannis Moschovakis in terms of abstract recursion, and Noson Yanofsky in terms of equivalence classes of programs, while Moshe Vardi has speculated that an algorithm is both abstract-state machine and recursor. It is also debated whether an algorithm must be physically implementable. Moschovakis and Vasilis Paschalis (among others) adopt a concept of algorithm "so wide as to admit 'non-implementable' algorithms,"[30] while other approaches do impose a requirement of physical implementability, even if only a very mild one. David Harel, for instance, writes: [A]ny algorithmic problem for which we can find an algorithm that can be programmed in some programming language, any language, running on some computer, any computer, even one that has not been built yet but can be built ... is also solvable by a Turing machine. This statement is one version of the so-called Church/Turing thesis."[23]

Steering between these debates—and following Harel's suggestion that the algorithms of interest to computer science are always expressible in programming languages—we arrive at the following program-oriented formulation of the algorithmic thesis:

CTT-Algorithm (CTT-A). Every algorithm can be expressed by means of a program in some (not necessarily currently existing) Turing-equivalent programming language.

There is an option to narrow CTT-A by adding "physically implementable" before "program," although in our view this would be to lump together two distinct computational issues that are better treated separately.

The evolving nature and open-endedness of the concept of an algorithm is matched by a corresponding open-endedness in the concept of a programming language. But this open-endedness notwithstanding, CTT-A requires that all algorithms be bounded by Turing computability.

Later in this article we examine complexity-theoretic and physical versions of the Church-Turing thesis but first turn to the question of the justification of the theses introduced so far. Are CTT-O and CTT-A correct?

What Justifies the Church-Turing Thesis?

Stephen Kleene—who coined the term "Church-Turing thesis"—catalogued four types of argument for CTT-O: First,

the argument from non-refutation points out the thesis has never been refuted, despite sustained (and ongoing) attempts to find a counterexample (such as the attempts by László Kalmár and, more recently, by Doukas Kapantais). Second, the argument from confluence points to the fact that the various characterizations of computability, while differing in their approaches and formal details, turn out to encompass the very same class of computable functions. Four such characterizations were presented (independently) in 1936 and immediately proved to be *extensionally* equivalent: Turing computability, Church's λ-definability, Kleene's recursive functions, and Post's finitary combinatory processes.

Third is an argument usually referred to nowadays as "Turing's analysis." Turing called it simply argument "I," stating five very general and intuitive constraints—or axioms—the human computer may be assumed to satisfy: "The behavior of the computer at any moment is determined by the symbols which he is observing, and his 'state of mind' at that moment"; "[*T*]here is a bound *B* to the number of symbols or squares which the computer can observe at one moment"; "[E]ach of the new observed squares is within *L* squares of an immediately previously observed square"; "[I]n a simple operation not more than one symbol is altered"; and "[T]he number of states of mind which need be taken into account is finite." Turing noted that reference to the computer's states of mind can be avoided by talking instead about configurations of symbols, these being "a more definite and physical counterpart" of states of mind.[36]

The second part of Turing's argument I is a demonstration that each function computed by any human computer subject to these constraints is also computable by a Turing machine; it is not difficult to see that each of the computer's steps can be mimicked by the Turing machine, either in a single step or by means of a series of steps. In short, Turing's five axioms entail CTT-O. (Turing's axiomatic approach to computability was in fact foreshadowed by Kurt Gödel in a suggestion to Church a year or so earlier.[15] Some more recent axiomatic approaches to computability proceed differently; for example, Erwin Engeler employs the Schönfinkel-Curry idea of "combinators" in order to axiomatize the concept of an algorithmic function.)

The Turing machine is restricted to, say, changing at most one bounded part at each sequential step of a computation.

Fourth in this catalog of considerations supporting CTT-O are arguments from first-order logic. They are typified by a 1936 argument of Church's and by Turing's argument II, from Section 9 of Turing's 1936 paper. In 2013, Saul Kripke[28] presented a reconstruction of Turing's argument II, which goes as follows: Computation is a special form of mathematical deduction; and every mathematical deduction—and therefore every computation—can be formalized as a valid deduction in the language of first-order predicate logic with identity (a step Kripke referred to as "Hilbert's thesis"); following Gödel's completeness theorem, each computation is thus formalized by a provable formula of first-order logic; and every computation can therefore be carried out by the universal Turing machine. This last step regarding the universal Turing machine is secured by a theorem proved by Turing: Every provable formula of first-order logic can be proved by the universal Turing machine.

The third and fourth of these arguments provide justification for CTT-O but not for CTT-A. As Robin Gandy[20] pointed out, the third argument—Turing's I—contains "crucial steps ... where he [Turing] appeals to the fact that the calculation is being carried out by a human being."[20] For example, Turing assumed "a human being can only write one symbol at a time," and Gandy noted this assumption cannot be carried over to a parallel machine that "prints an arbitrary number of symbols simultaneously."[20] In Conway's Game of Life, for instance, there is no upper bound on the number of cells that make up the grid, yet the symbols in all the cells are updated simultaneously. Likewise, the fourth argument (Turing's II) involves the claim that computation is a special form of formal proof, but the notion of proof is intrinsically related to what a human mathematician—and not some oracle—can prove.

It is thus perhaps not too surprising that the third and fourth arguments in this catalog seldom if ever appear in logic and computer science textbooks. The two arguments that are always given for the Church-Turing thesis (in, for example, Lewis and Papadimitriou[29]) are

confluence and non-refutation. Yet both those arguments are merely inductive, whereas the third and fourth arguments are deductive in nature.

However, a number of attempts have sought to extend Turing's axiomatic analysis to machine computation; for example, Gandy[20] broadened Turing's analysis in such a way that parallel computation is included, while Dershowitz and Gurevich[16] gave a more general analysis in terms of abstract state machines. We return to the topic of extending the analysis to machine computation later in this article but first address the important question of whether CTT-O is mathematically provable.

Is the Thesis Mathematically Provable?

It used to be thought by mathematical logicians and others that CTT-O is not amenable to formal proof, since it is not a mathematically precise statement. This is because it pairs an informal concept—a "vague intuitive notion," Church called it[5]—with a precise concept. However, Elliott Mendelson gave a powerful critique of this general argument; and today the view that CTT-O is formally provable seems to be gaining acceptance; see, for example, Dershowitz and Gurevich.[16] Inspired by Gandy,[20] Wilfried Sieg[35] stated that a tightened form of Turing's argument I proves the thesis; and Kripke[28] entertained the same claim for Turing's argument II.

Turing's own view was that, on the contrary, his thesis is not susceptible to mathematical proof. He thought his arguments I and II, and indeed "[a]ll arguments which can be given" for the thesis, are "fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically."[36] Hilbert's thesis is another example of a proposition that can be justified only by appeal to intuition, and so Kripke's[28] tightened form of argument II, far from proving CTT-O, merely deduced it from another thesis that is also not amenable to mathematical proof.

Much the same can be said about argument I. If axioms 1–5 are formulated in precise mathematical terms, then it is certainly provable from them that computation is bounded by Turing computability; this is probably what Gandy[20] meant when he said Turing's argument I proves a "theorem." But the real issue

Turing's own view was that, on the contrary, his thesis is not susceptible to mathematical proof.

is whether these axioms completely capture the concept of a computational or algorithmic process, and, so far as we see, no one has ever given a rigorous mathematical justification of that claim. The axioms may be supported by informal arguments, but the whole edifice then falls short of mathematical proof. This is most apparent when the informal arguments offered for the axioms invoke limitations in the cognitive capacities of human computers, as we point out elsewhere.[13] A justification of the second axiom may, for instance, refer to the limitations of human observation. The axioms most certainly lie beyond the scope of mathematical demonstration if their truth depends on contingent human limitations. Turing himself cheerfully appealed to cognitive limitations in the course of his analysis, saying, for example, "[J]ustification lies in the fact that the human memory is necessarily limited."[36]

In summary, our answer to "Is CTT-O mathematically provable?" is: Turing thought not and we have found no reason to disagree with him. The various historical arguments seem more than sufficient to establish CTT-O, but these arguments do indeed fall short of mathematical proof.

We next address complexity theoretic forms of the Church-Turing thesis, then turn to the question of whether CTT-A is justified in the context of physically realistic computations.

Complexity: The Extended Church-Turing Thesis

It is striking that the Turing machine holds a central place not only in computability theory but also in complexity theory, where it is viewed as a universal model for complexity classes.

In complexity theory, the time complexities of any two general and reasonable models of computation are assumed to be polynomially related. But what counts as "reasonable"? Aharonov and Vazirani[1] glossover "reasonable" as "physically realizable in principle"; see also Bernstein and Vazirani.[3] If a computational problem's time complexity is t in some (general and reasonable) model, then its time complexity is assumed to be poly(t) in the single-tape Turing machine model; see also Goldreich.[21] This assumption has different names in the literature; Goldreich[21] called it the

Cobham-Edmonds thesis, while Yao[40] introduced the term "Extended Church-Turing thesis." The thesis is of interest only if $\mathtt{P} \neq \mathtt{NP}$, since otherwise it is trivial.

Quantum-computation researchers also use a variant of this thesis, as expressed in terms of probabilistic Turing machines. Bernstein and Vazirani[3] said: "[C]omputational complexity theory rests upon a modern strengthening of [the Church-Turing] thesis, which asserts that any 'reasonable' model of computation can be efficiently simulated on a probabilistic Turing machine."[3]

Aharonov and Vazirani[1] give the following formulation of this assumption, naming it the "Extended Church-Turing thesis"—though it is not quite the same as Yao's earlier thesis of the same name, which did not refer to probabilistic Turing machines:

CTT-Extended (CTT-E). "[A]ny reasonable computational model can be simulated efficiently by the standard model of classical computation, namely, a probabilistic Turing machine."[1]

As is well known in computer science, Peter Shor's quantum algorithm for prime factorization is a potential counterexample to CTT-E; the algorithm runs on a quantum computer in polynomial time and is much faster than the most-efficient known "classical" algorithm for the task. But the counterexample is controversial. Some computer scientists think the quantum computer invoked is not a physically reasonable model of computation, while others think accommodating these results might require further modifications to complexity theory.

We turn now to extensions of the Church-Turing thesis into physics.

Physical Computability

The issue of whether every aspect of the physical world is Turing-computable was broached by several authors in the 1960s and 1970s, and the topic rose to prominence in the mid-1980s.

In 1985, Stephen Wolfram formulated a thesis he described as "a physical form of the Church-Turing hypothesis," saying, "[U]niversal computers are as powerful in their computational capacities as any physically realizable system can be, so that they can simulate any physical system."[39] In the same year, David Deutsch, who laid the foundations of quantum computation, independently stated a similar thesis, describing it as "the physical version of the Church-Turing principle."[17] The thesis is now known as the Church-Turing-Deutsch thesis and the Church-Turing-Deutsch-Wolfram thesis.

Church-Turing-Deutsch-Wolfram thesis (CTDW). Every finite physical system can be simulated to any specified degree of accuracy by a universal Turing machine.

Deutsch pointed out that if "simulated" is understood as "perfectly simulated," then the thesis is falsified by continuous classical systems, since such classical systems necessarily involve uncomputable real numbers, and went on to introduce the concept of a universal quantum computer, saying such a computer is "capable of perfectly simulating every finite, realizable physical system." Other physical formulations were advanced by Lenore Blum et al., John Earman, Itamar Pitowsky, Marian Pour-El, and Ian Richards, among others.

We next formulate a strong version of the physical Church-Turing thesis we call the "total physical computability thesis." (We consider some weaker versions later in the article.) By "physical system" we mean any system whose behavior is in accordance with the actual laws of physics, including non-actual and idealized systems.

Total physical computability thesis (CTT-P). Every physical aspect of the behavior of any physical system can be calculated (to any specified degree of accuracy) by a universal Turing machine.

As with CTT-E, there is also a probabilistic version of CTT-P, formulated in terms of a probabilistic Turing machine.

Arguably, the phrase "physical version of the Church-Turing thesis" is an inappropriate name for this and related theses, since CTT-O concerns a form of effective or algorithmic activity and asserts the activity is always bounded by Turing computability, while CTT-P and CTDW, on the other hand, entail that the activity of every physical system is bounded by Turing computability; the system's activity need not be algorithmic/effective at all. Nevertheless, in our "CTT-" nomenclature, we follow the Deutsch-Wolfram tradition throughout this article.

Is CTT-P true? Not if physical systems include systems capable of producing unboundedly many digits of a random binary sequence; Church showed such sequences are uncomputable, as we discussed elsewhere.[8] Moreover, speculation that there may be deterministic physical processes whose behavior cannot be calculated by the universal Turing machine stretches back over several decades; for a review, see Copeland.[9] In 1981, Pour-El and Richards[34] showed that a system evolving from computable initial conditions in accordance with the familiar three-dimensional wave equation is capable of exhibiting behavior that falsifies CTT-P; even today, however, it is an open question whether these initial conditions are physically possible. Earlier papers, from the 1960s, by Bruno Scarpellini, Arthur Komar, and Georg Kreisel, in effect questioned CTT-P, with Kreisel stating: "There is no evidence that even present-day quantum theory is a mechanistic, i.e., recursive theory in the sense that a recursively described system has recursive behavior."[27] Other potential counterexamples to CTT-P have been described by a number of authors, including what are called "relativistic" machines. First introduced by Pitowsky,[32] they will be examined in the section called "Relativistic Computation."

CTT-P and Quantum Mechanics

There are a number of theoretical countermodels to CTT-P arising from quantum mechanics. For example, in 1964, Komar[26] raised "the issue of the macroscopic distinguishability of quantum states," asserting there is no effective procedure "for determining whether two arbitrarily given physical states can be superposed to show interference effects." In 2012, Eisert et al.[19] showed "[T]he very natural physical problem of determining whether certain outcome sequences cannot occur in repeated quantum measurements is undecidable, even though the same problem for classical measurements is readily decidable." This is an example of a problem that refers unboundedly to the future but not to any specific time. Other typical physical problems take the same form; Pitowsky gave as examples "Is the solar system stable?" and "Is the motion of a given system, in a known initial state, periodic?"

Cubitt et al.[14] described another such undecidability result in a 2015 *Nature* article, outlining their proof that "[T]he

spectral gap problem is algorithmically undecidable: There cannot exist any algorithm that, given a description of the local interactions, determines whether the resultant model is gapped or gapless." Cubitt et al. also said this is the "first undecidability result for a major physics problem that people would really try to solve."

The spectral gap, an important determinant of a material's properties, refers to the energy spectrum immediately above the ground-energy level of a quantum many-body system, assuming a well-defined least-energy level of the system exists; the system is said to be "gapless" if this spectrum is continuous and "gapped" if there is a well-defined next-least energy level. The spectral gap problem for a quantum many-body system is the problem of determining whether the system is gapped or gapless, given the finite matrices (at most three) describing the local interactions of the system.

In their proof, Cubitt et al.[14] encoded the halting problem in the spectral gap problem, showing the latter is at least as hard as the former. The proof involves an infinite family of two-dimensional lattices of atoms. But they pointed out their result also applies to finite systems whose size increases, saying, "Not only can the lattice size at which the system switches from gapless to gapped be arbitrarily large, the threshold at which this transition occurs is uncomputable." Their proof offers an interesting countermodel to CTT-P, involving a physically relevant example of a finite system of increasing size. There exists no effective method for extrapolating the system's future behavior from (complete descriptions of) its current and past states.

It is debatable whether any of these quantum models correspond to real-world quantum systems. Cubitt et al.[14] admitted the model invoked in their proof is highly artificial, saying, "Whether the results can be extended to more natural models is yet to be determined." There is also the question of whether the spectral gap problem becomes computable when only local Hilbert spaces of realistically low dimensionality are considered. Nevertheless, these results are certainly suggestive: CTT-P cannot be taken for granted, even in a finite quantum universe.

Summarizing the current situation with respect to CTT-P, we can say, although theoretical countermodels in which CTT-P is false have been described, there is at present—so far as we know—not a shred of evidence that CTT-P is false in the actual universe. Yet it would seem most premature to assert that CTT-P is true.

Weaker Physical Computability Theses

Piccinini[31] has distinguished between two different types of physical versions of the Church-Turing thesis, both commonly found in the literature, describing them as "bold" and "modest" versions of the thesis, respectively. The bold and modest versions are weaker than our "super-bold" version just discussed (CTT-P). Bold versions of the thesis state, roughly, that "Any physical process can be simulated by some Turing machine."[31] The Church-Turing-Deutsch-Wolfram thesis (CTDW) is an example, though Piccinini emphasized that the bold versions proposed by different researchers are often "logically independent of one another" and that, unlike the different formulations of CTT-O, which exhibit confluence, the different bold formulations in fact exhibit "lack of confluence."[31]

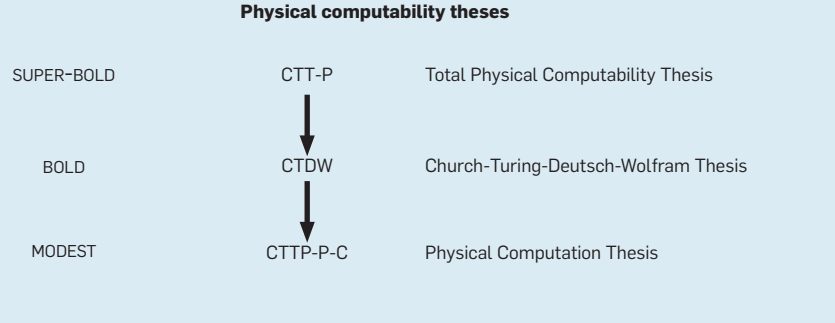
CTDW and other bold forms are too weak to rule out the uncomputability scenarios described by Cubitt et al.[14] and by Eisert et al.[19] This is because the physical processes involved in these scenarios may, so far as we know, be Turing-computable; it is possible that each process can be simulated by a Turing machine, to any required degree of accuracy, and yet the answers to certain physical questions about the processes are, in general, uncomputable. The situation is similar in the case of the universal Turing machine itself. The machine's behavior (consisting of the physical actions of the read/write head) is always Turing-computable since it is produced by the Turing machine's program, yet the answers to some questions about the behavior (such as whether or not the machine halts given certain inputs) are not computable.

Nevertheless, bold forms (such as CTDW) are interesting empirical hypotheses in their own right and the world might confute them. For instance, CTDW fails in the wave-equation countermodel due to Pour-El and Richards[34] where the mapping between the wave equation's "inputs" and "outputs" is not a Turing-computable (real) function; although, as noted earlier, the physicality of this countermodel can readily be challenged. We discuss some other potential countermodels later in the article, but turn first to what Piccinini termed "modest" versions of the thesis.

Modest versions maintain in essence that every physical computing process is Turing-computable; for two detailed formulations, see Gandy[20] and Copeland.[8] Even if CTT-P and CTDW are in general false, the behavior of the subset of physical systems that are appropriately described as computing systems may nevertheless be bounded by Turing-computability. An illustration of the difference between modest versions on the one hand and CTT-P and CTDW on the other is given by the fact that the wave-equation example is not a countermodel to the modest thesis, assuming, as seems reasonable, that the physical dynamics described by the equation do not constitute a computing process.

Here, we formulate a modest version of the physical Church-Turing thesis we call the "Physical Computation" thesis, then turn to the question of whether it is true.

Relationships between the three physical computability theses: CTT-P, CTDW, and CTT-P-C.



Physical Computation Thesis

This form of the thesis maintains that physical computation is bounded by Turing-computability.

Physical computation thesis (CTT-P-C). Every function computed by any physical computing system is Turing-computable.

Is CTT-P-C true? As with the stronger physical computability theses, it seems too early to say. CTT-P-C could be false only if CTT-P and CTDW turn out to be false, since each of them entails CTT-P-C (see the figure here, which outlines the relationships among CTT-P, CTDW, and CTT-P-C). If all physical computation is effective in the 1930s sense of Turing and Church, then CTT-P-C is certainly true. If, however, the door is open to a broadened sense of computation, where physical computation is not necessarily effective in the sense of being bounded by Turing-computability, then CTT-P-C makes a substantive claim.

There is, in fact, heated debate among computer scientists and philosophers about what counts as physical computation. Moreover, a number of attempts have sought to describe a broadened sense of computation in which computation is not bounded by Turing-computability; see, for example, Copeland.[6] Computing machines that compute "beyond the Turing limit" are known collectively as "hypercomputers," a term introduced in Copeland and Proudfoot.[11] Some of the most thought-provoking examples of notional machines that compute in the broad sense are called "supertask" machines. These "Zeno computers" squeeze infinitely many computational steps into a finite span of time. Examples include accelerating machines,[7,12] shrinking machines, and the intriguing relativistic computers described in the next section.

Notional machines all constitute rather theoretical countermodels to CTT-P-C, so long as it is agreed that they compute in a broadened sense, but none has been shown to be physically realistic, although, as we explain, relativistic computers come close. In short, the truth or falsity of CTT-P-C remains unsettled.

Relativistic Computation

Relativistic machines operate in spacetime structures with the property that the entire endless lifetime of one component of the machine is included in the finite chronological past of another component, called "the observer." The first component could thus carry out an infinite computation (such as calculating every digit of π) in what is, from the observer's point of view, a finite timespan of, say, one hour. (Such machines are in accord with Einstein's general theory of relativity, hence the term "relativistic.") Examples of relativistic computation have been detailed by Pitowsky, Mark Hogarth, and Istvan Németi.

In this section we outline a relativistic machine *RM* consisting of a pair of communicating Turing machines, T_E and T_O, in relative motion. T_E is a universal machine, and T_O is the observer. *RM* is able to compute the halting function, in a broad sense of computation. Speaking of computation here seems appropriate, since *RM* consists of nothing but two communicating Turing machines.

Here is how *RM* works. When the input (m,n), asking whether the m^{th} Turing machine (in some enumeration of the Turing machines) halts or not when started on input n, enters T_O, T_O first prints 0 (meaning "never halts") in its designated output cell and then transmits (m,n) to T_E. T_E simulates the computation performed by the m^{th} Turing machine when started on input n and sends a signal back to T_O if and only if the simulation terminates. If T_O receives a signal from T_E, T_O deletes the 0 it previously wrote in its output cell and writes 1 in its place (meaning "halts"). After one hour, T_O's output cell shows 1 if the m^{th} Turing machine halts on input n and shows 0 if the m^{th} machine does not halt on n.

The most physically realistic version of this setup to date is due to Németi and his collaborators in Budapest. T_E, an ordinary computer, remains on Earth, while the observer T_O travels toward and enters a slowly rotating Kerr black hole. T_O approaches the outer event horizon, a bubble-like hypersurface surrounding the black hole. Németi theorized that the closer T_O gets to the event horizon, the faster T_E's clock runs relative to T_O due to Einsteinian gravitational time dilation, and this speeding up continues with no upper limit. T_O motion proceeds until, relative to a time t on T_O clock, the entire span of T_E's computing is over. If any signal was emitted by T_E, the signal will have been received by T_O before time t. So T_O will fall into the black hole with 1 in its output cell if T_E halted and 0 if T_E never halted. Fortunately, T_O can escape annihilation if its trajectory is carefully chosen in advance, says Németi; the rotational forces of the Kerr hole counterbalance the gravitational forces that would otherwise "spaghettify" T_O. T_O thus emerges unscathed from the hole and goes on to use the computed value of the halting function in further computations.

Németi and colleagues emphasize their machine is physical in the sense it is "not in conflict with presently accepted scientific principles" and, in particular, "the principles of quantum mechanics are not violated."[2] They suggest humans might "even build" a relativistic computer "sometime in the future."[2] This is, of course, highly controversial. However, our point is that Németi's theoretical countermodel, which counters not only CTT-P-C but also CTT-P and CTDW, helps underscore that the "physical version of the Church-Turing thesis" is quite independent of CTT-O, since the countermodel stands whether or not CTT-O is endorsed. We next reconsider CTT-A.

CTT-A and Computation in the Broad

The continuing expansion of the concept of an algorithm is akin to the extension of the concept of number from integers to signed integers to rational, real, and complex numbers. Even the concept of human computation underwent an expansion; before 1936, computation was conceived of in terms of total functions, and it was Kleene in 1938 who explicitly extended the conception to also cover partial functions.

Gurevich argued in 2012 that formal methods cannot capture the algorithm concept in its full generality due to the concept's open-ended nature; at best, formal methods provide treatments of "strata of algorithms" that "have matured enough to support rigorous definitions."[22] An important question for computer science is whether CTT-A is a reasonable constraint on the growth of new strata. Perhaps not. In 1982, Jon Doyle[18] suggested equilibrating systems with discrete spectra (such as molecules and other quantum many-body systems) illustrate a concept of effectiveness that is broader than the

classical concept, saying, "[E]quilibrating can be so easily, reproducibly, and mindlessly accomplished" that we may "take the operation of equilibrating as an effective one," even if "the functions computable in principle given Turing's operations and equilibrating include non-recursive functions."

Over the years, there have been several departures from Turing's 1936 analysis, as the needs of computer science led to a broadening of the algorithm concept. For example, Turing's fourth axiom, which bounds the number of parts of a system that can be changed simultaneously, became irrelevant when the algorithm concept broadened to cover parallel computations. The future computational landscape might conceivably include more extensive revisions of the concept, if, for example, physicists were to discover that hardware effective in Doyle's extended sense is a realistic possibility.

If such hardware were to be developed—hardware in which operations are effective in the sense of being "easily, reproducibly, and mindlessly accomplished" but not bounded by Turing computability—then would the appropriate response by computer scientists be to free the algorithm concept from CTT-A? Or should CTT-A remain as a constraint on algorithms, with instead two different species of computation being recognized, called, say, algorithmic computation and non-algorithmic computation? Not much rides on a word, but we note we prefer "effective computation" for computation that is bounded by Turing computability and "neo-effective computation" for computation that is effective in Doyle's sense and *not* bounded by Turing computability, with "neo" indicating a new concept related to an older one.

The numerous examples of notional "hypercomputers" (see Copeland[9] for a review) prompt similar questions. Interestingly, a study of the expanding literature about the concept of an infinite-time Turing machine, introduced by Joel Hamkins and Andy Lewis in 2000, shows that a number of computer scientists are prepared to describe the infinite-time machine as computing the halting function. Perhaps this indicates the concept of computation is already in the process of bifurcating into "effective" and "neo-effective" computation.

Conclusion

In the computational literature the term "Church-Turing thesis" is applied to a variety of different propositions usually not equivalent to the original thesis—CTT-O; some even go far beyond anything either Church or Turing wrote. Several but not all are fundamental assumptions of computer science. Others (such as the various physical computability theses we have discussed) are important in the philosophy of computing and the philosophy of physics but are highly contentious; indeed, the label "Church-Turing thesis" should not mislead computer scientists or anyone else into thinking they are established fact or even that Church or Turing endorsed them. C

References

1. Aharonov, D. and Vazirani, U.V. Is quantum mechanics falsifiable? A computational perspective on the foundations of quantum mechanics. Chapter in *Computability: Gödel, Turing, Church and Beyond*, B.J. Copeland, C.J. Posy, and O. Shagrir, Eds. MIT Press, Cambridge, MA, 2013.
2. Andréka, H., Németi, I., and Németi, P. General relativistic hypercomputing and foundation of mathematics. *Natural Computing 8*, 3 (Sept. 2009), 499–516.
3. Bernstein, E. and Vazirani, U. Quantum complexity theory. *SIAM Journal on Computing 26*, 5 (Oct. 1997), 1411–1473.
4. Castelvecchi, D. Paradox at the heart of mathematics makes physics problem unanswerable. *Nature 528* (Dec. 9, 2015), 207.
5. Church, A. An unsolvable problem of elementary number theory. *American Journal of Mathematics 58*, 2 (Apr. 1936), 345–363.
6. Copeland, B.J. The broad conception of computation. *American Behavioral Scientist 40*, 6 (May 1997), 690–716.
7. Copeland, B.J. Even Turing machines can compute uncomputable functions. Chapter in *Unconventional Models of Computation*, C. Calude, J. Casti, and M. Dinneen, Eds. Springer, Singapore, 1998.
8. Copeland, B.J. Narrow versus wide mechanism: Including a re-examination of Turing's views on the mind-machine issue. *The Journal of Philosophy 97*, 1 (Jan. 2000), 5–32.
9. Copeland, B.J. Hypercomputation. *Minds and Machines 12*, 4 (Nov. 2002), 461–502.
10. Copeland, B.J. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, Plus the Secrets of Enigma.* Oxford University Press, Oxford, U.K., 2004.
11. Copeland, B.J. and Proudfoot, D. Alan Turing's forgotten ideas in computer science. *Scientific American 280*, 4 (Apr. 1999), 98–103.
12. Copeland, B.J. and Shagrir, O. Do accelerating Turing machines compute the uncomputable? *Minds and Machines 21*, 2 (May 2011), 221–239.
13. Copeland, B.J. and Shagrir, O. Turing versus Gödel on computability and the mind. Chapter in *Computability: Gödel, Turing, Church, and Beyond*, B.J. Copeland, C.J. Posy, and O. Shagrir, Eds. MIT Press, Cambridge, MA, 2013.
14. Cubitt, T.S., Perez-Garcia, D., and Wolf, M.M. Undecidability of the spectral gap. *Nature 528*, 7581 (Dec. 2015), 207–211.
15. Davis, M. Why Gödel didn't have Church's thesis. *Information and Control 54*, 1-2 (July 1982), 3–24.
16. Dershowitz, N. and Gurevich, Y. A natural axiomatization of computability and proof of Church's thesis. *Bulletin of Symbolic Logic 14*, 3 (Sept. 2008), 299–350.
17. Deutsch, D. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 400*, 1818 (July 1985), 97–117.
18. Doyle, J. What is Church's thesis? An outline. *Minds and Machines 12*, 4 (Nov. 2002), 519–520.
19. Eisert, J., Müller, M.P., and Gogolin, C. Quantum measurement occurrence is undecidable. *Physical Review Letters 108*, 26 (June 2012), 1–5.
20. Gandy, R.O. Church's thesis and principles for mechanisms. In *Proceedings of the Kleene Symposium*, J. Barwise, H.J. Keisler, and K. Kunen, Eds. (Madison, WI, June 1978). North-Holland, Amsterdam, Netherlands, 1980.
21. Goldreich, O. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, New York, 2008.
22. Gurevich, Y. What is an algorithm? In *Proceedings of the 38th Conference on Current Trends in the Theory and Practice of Computer Science* (Špindleův Mlýn, Czech Republic, Jan. 21–27), M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, and G. Turán, Eds. Springer, Berlin, Heidelberg, Germany, 2012.
23. Harel, D. *Algorithmics: The Spirit of Computing, Second Edition.* Addison-Wesley, Reading, MA, 1992.
24. Hopcroft, J.E. and Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.
25. Kleene, S.C. *Introduction to Metamathematics.* Van Nostrand, New York, 1952.
26. Komar, A. Undecidability of macroscopically distinguishable states in quantum field theory. *Physical Review 133*, 2B (Jan. 1964), 542–544.
27. Kreisel, G. Mathematical logic: What has it done for the philosophy of mathematics? Chapter in *Bertrand Russell: Philosopher of the Century*, R. Schoenman, Ed. Allen and Unwin, London, U.K., 1967.
28. Kripke, S.A. Another approach: The Church-Turing 'thesis' as a special corollary of Gödel's completeness theorem. Chapter in *Computability: Gödel, Turing, Church, and Beyond*, B.J. Copeland, C.J. Posy, and O. Shagrir, Eds. MIT Press, Cambridge, MA, 2013.
29. Lewis, H.R. and Papadimitriou, C.H. *Elements of the Theory of Computation.* Prentice Hall, Upper Saddle River, NJ, 1981.
30. Moschovakis, Y.N. and Paschalis, V. Elementary algorithms and their implementations. Chapter in *New Computational Paradigms: Changing Conceptions of What Is Computable*, S.B. Cooper, B. Lowe, and A. Sorbi, Eds. Springer, New York, 2008.
31. Piccinini, G. The physical Church-Turing thesis: Modest or bold? *The British Journal for the Philosophy of Science 62*, 4 (Aug. 2011), 733–769.
32. Pitowsky, I. The physical Church thesis and physical computational complexity. *Iyyun 39*, 1 (Jan. 1990), 81–99.
33. Post, E.L. Finite combinatory processes: Formulation I. *The Journal of Symbolic Logic 1*, 3 (Sept. 1936), 103–105.
34. Pour-El, M.B. and Richards, I.J. The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics 39*, 3 (Mar. 1981), 215–239.
35. Sieg, W. Mechanical procedures and mathematical experience. Chapter in *Mathematics and Mind*, A. George, Ed. Oxford University Press, New York, 1994.
36. Turing, A.M. On computable numbers, with an application to the *Entscheidungsproblem* (1936); in Copeland.[10]
37. Turing, A.M. *Lecture on the Automatic Computing Engine* (1947); in Copeland.[10]
38. Turing, A.M. *Intelligent Machinery* (1948); in Copeland.[10]
39. Wolfram, S. Undecidability and intractability in theoretical physics. *Physical Review Letters 54*, 8 (Feb. 1985), 735–738.
40. Yao, A.C.C. Classical physics and the Church-Turing thesis. *Journal of the ACM 50*, 1 (Jan. 2003), 100–105.

B. Jack Copeland (jack.copeland@canterbury.ac.nz) is Distinguished Professor of Philosophy at the University of Canterbury in Christchurch, New Zealand, and Director of the Turing Archive for the History of Computing, also at the University of Canterbury.

Oron Shagrir (oron.shagrir@gmail.com) is Schulman Professor of Philosophy and Cognitive Science at the Hebrew University of Jerusalem, Jerusalem, Israel.