Chapter 9
What is the Church-Turing Thesis?

Udi Boker and Nachum Dershowitz

[Answer:] Rosser and its inventor proved that its beta-reduction satisfies the diamond property, and Kleene (pron. clean-ee) proved that it was equivalent to his partial recursive functions. The previous result combined with a similar one with the Turing Machine, led to the Church-Turing thesis.
[Question: "What is ...?"]
—Quizbowl Tournament (2004)

Abstract We aim to put some order to the multiple interpretations of the Church-Turing Thesis and to the different approaches taken to prove or disprove it.

9.1 Introduction

The notions of *algorithm* and *computability* as formal subjects of mathematical reasoning gained prominence in the twentieth century with the advent of symbolic logic and the discovery of inherent limitations to formal deductive reasoning. The most popular formal model of computation is due to Alan Turing. The claim that computation paradigms such as Turing's capture the intuitive notion of algorithmic computation, stated in the form of a thesis, is roughly as follows:

Church-Turing Thesis: *The results of every effective computation can be attained by some Turing machine, and vice-versa.*

U. Boker (✉)
School of Computer Science, Reichman University, Herzliya, Israel
e-mail: udiboker@idc.ac.il

N. Dershowitz
School of Computer Science, Tel Aviv University, Ramat Aviv, Israel
e-mail: nachum@tau.ac.il

© Springer Nature Switzerland AG 2022

F. Ferreira et al. (eds.), *Axiomatic Thinking II*,
https://doi.org/10.1007/978-3-030-77799-9_9

Any other standard full-fledged computational paradigm—such as the recursive functions, the lambda calculus, counter machines, the random access memory (RAM) model, or most programming languages—could be substituted here for “Turing machine”. Turing’s model has the advantage, however, of appealing only to the most basic of actions, namely reading, writing, testing, and moving, as does Emil Post’s essentially identical and contemporaneous notion of computation [1]. (Other paradigms may model other aspects of computation better, such as recursion with recursive functions [2] or combinatory logic for composition [3].)

In what follows, we analyse different interpretations that this thesis has been given over the years and what it might mean to prove or disprove it.

Historically, the thesis was first formulated in the thirties. It started with a suggestion of Alonzo Church, based on the lambda calculus, in 1936: “The purpose of the present paper is to propose a definition of effective calculability” [4]. This was followed by the introduction of Turing machines, as a basic model of computability, in the same year [5]. Shortly thereafter, the claim that these two proposals both captured the intended notion was stated as a “thesis” by Stephen Kleene [6]. Church, Turing, and Kleene had their own original interpretations of the claim, which were not identical. Over time, their positions in this regard developed [7, 8]. Moreover, the understanding of the thesis, as expressed in the scientific literature, has continued to evolve.

Our intention here is not to get into the historical question of how exactly Church or Turing themselves interpreted the thesis, but rather to analyse its different interpretations today. For historical aspects of the thesis, one can consult [8, 9].

The Church-Turing Thesis itself is extensional, speaking of *what* can be effectively computed, whereas the claims for and against it are intensional, arguing about *how* a computation can be accomplished. We examine first the extensional claim, looking at what type of entities are meant to be computed.

The *Extended Church-Turing Thesis* [10], otherwise known as the *Invariance Thesis* [11], is more presumptuous. It asserts, not only that there is a Turing machine corresponding to each effective algorithm, but that there is one that is no more than polynomially slower (counting atomic actions) than any other effective means of performing that computation. We do not consider this extended thesis here and concentrate on the more fundamental computability question.

9.2 What Is Computed?

The thesis asserts that there exists some Turing machine that has the same outcomes as any particular effective calculation.

Yet, there are different views as to what it is exactly that a Turing machine computes.

What Does a Turing Machine Compute?

The input of an ordinary Turing machine is a finite tape consisting of a sequence of cells, each containing one letter taken from a finite alphabet Σ. The tape extends itself with blanks whenever the machine attempts to go beyond its current far end. The computation may also involve finitely many additional symbols, not in Σ. If and when the computation halts, the same tape contains the output—perhaps followed by trailing blanks that are not considered part of the output. By this very standard view, a Turing machine M computes a *partial function* $f_M : \Sigma^* \rightharpoonup \Sigma^*$ over *finite strings*, partial because some machines do not halt for some inputs. This is the version of Turing machine, out of many, that is most relevant to the intent of the original thesis.

All the same, different views are in place, even in Turing's classic 1936 paper. There, Turing analysed the computability of real numbers by means of his machines—using their infinite expansions to represent them:

> Some of the symbols written down will form the sequence of figures which is the decimal of the real number.... The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure. [5]

From this point of view, a machine M runs forever, generating an infinite string. If the content of every cell of the tape eventually stabilizes, then the infinite sequence of stabilized cell values is deemed its output. Accordingly, such a machine computes a *partial function* $f_M : \Sigma^* \rightharpoonup \Sigma^\omega$, where Σ^ω is the set of infinite strings (padded with infinitely many blanks, when necessary).

As each step in the run of a Turing machine can alter a symbol written in some previous step, the output of an infinite run can be viewed, not as one possibly infinite string in the limit—at the "end" of the computation, but rather as a sequence of finite strings, one per step. By this view, the machine computes a *total function* $f_M : \Sigma^* \to (\Sigma^*)^\omega$.

Another issue is what the tape contents actually represent.

The tape is often imagined to be infinite, either on one end or on both. Thus, it may be thought of as an infinite string—perhaps with only finitely many positions nonblank initially. According to this view, a finite computation of a Turing machine M is a partial function $f_M : \Sigma^\omega \rightharpoonup \Sigma^\omega$ over *infinite strings*, while an infinite computation is a partial function $f_M : \Sigma^\omega \rightharpoonup \Sigma^\omega$ or a total function $f_M : \Sigma^\omega \to (\Sigma^\omega)^\omega$, depending on whether one looks only for stable outputs or at the sequence of tapes.

Aside from the basic version of a Turing machine with a single tape, many variants of Turing machines have been considered in the literature, having multiple tapes and/or multiple heads with varying levels of ability to read and write. By some views [12], a Turing machine may be considered to compute *varying* sorts of functions over finite and infinite strings, as long as the basic model at hand preserves the atomic operations and core functionality of a Turing machine.

Nevertheless, there is a significant difference between the view that a Turing machine computes a partial function over finite strings and some of the other views that involve infinite entities. In the former, the computation producing an output is finite, whereas in the latter, the computation is expected to go on forever. Our interest here is in finite but unbounded computations alone.

What Entity Does an Effective Computation Compute?

A *quick answer* might be "a function over strings", since a Turing machine computes with strings, and we aim at comparing something to it. However, as is well known, Turing machines are computationally equivalent to many other computational models, such as the recursive functions, the lambda calculus, and modern programming languages, all of which compute functions over different domains, such as natural numbers for the recursive functions and lambda terms for the lambda calculus. Accordingly, the thesis may—and sometimes is—stated with respect to one of these formal models of computation rather than in terms of Turing machines.

Yet, how can one compare computational models that operate over different domains? The original approach of the 1930s was to consider some "natural mappings" between these domains. A more thorough approach is taken in [13], analysing how the choice of mapping between domains can influence the comparison of computational models. Following the latter approach to comparing computational power, it is shown in [14] that "Turing computability" can be formally extended from a partial unary function over strings to a partial function over an arbitrary structured countable domain by using any structure-preserving injective mapping between this countable domain and the natural numbers. It is guaranteed that regardless of the choice of such a mapping, it cannot influence the comparison result.

A *formal* answer, then, is that an effective computation is a partial function over an arbitrary structured countable domain, and that the thesis can be stated with respect to functions over natural numbers.

A different answer is that an effective computation can be any entity, as long as we find the computation effective, regardless of whether or not the domain is structured. Under this interpretation, if we agree on the existence of such an effective computation, we should either look into an appropriate method of comparing Turing machines to such effective computations, or, if we claim that they are incomparable, we must simply admit that the thesis is wrong. This approach can be further classified into two variants: (i) considering only functions, yet over domains that are uncountable or unstructured; and (ii) considering entities that are not functions.

Regarding a function over an uncountable or unstructured domain, one may argue whether it can at all be effective. The prevailing view is that it cannot—an unstructured domain is just an arbitrary set with no differentiation between its elements, so a function over it has little meaning. As for an uncountable domain, one should consider the question of whether the thesis involves a [human] agent (as elaborated on in the next section). If it does, the dominant view is that the agent's ability to distinguish between different input or output elements is limited to a countable domain [15, 16].

As for a computation that is not a function, a common example in the literature is that of *interactive computation*. It has been claimed that an interactive computation, even a sequential one, does not compute a function and is therefore not Turing computable in the above sense [17, 18].[1] There is, however, a different view of interactive

[1] Peter Wegner and Dina Goldin asserted that the Church-Turing Thesis should be interpreted with respect to functions over natural numbers [17]. Yet, if one considers it in the more general form, they claim that he/she must admit it wrong due to the actuality of interactive computations. All this

computations, under which they are comparable to Turing machines [12]. By this view, the claim of [17, 20] is a syntactic one, not much different from saying that the recursive functions are not Turing computable because they operate over numbers and not over strings. A sequential interactive computation is considered under this view as nothing more than a function over finite strings—an interactive computation I computes in each iteration a function $f_I : \Sigma^* \rightarrow \Sigma^*$, where the input is the string produced by the previous iteration, concatenated with a fresh environment input. The input string would also contain some information on the inner status of the computing device. As f should be the same function for all iterations, we may simply ask whether f is Turing computable (cf. [19, Defs. 14–15]). Turing already considered this setting in his classic 1936 paper, yet concentrated on "automatic machines", as opposed to nondeterministic machines, or possibly interactive machines, which he termed "choice machines":

> For some purposes we might use machines (choice machines or c-machines) whose motion is only partially determined by the configuration…. When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. [5]

Turning to choice machines, a natural question is what about *random* computations that do not always return the same output for the same input, and thus do not compute a function, yet are guaranteed to return the correct answer with arbitrarily high probability. A prevailing answer is that such a computation may be regarded as computing a function, with the most likely outcome as its value. See more on randomness in Sect. 9.6.2.

The Bottom Line

The dominant view is that the core question underlying the thesis concerns functions of natural numbers, or—more precisely—functions over (string) representations of numbers. Computations of different entities are to be considered, by this view, as built over numerical encodings. This makes the extensional claim clear and simple, allowing us to concentrate on the core issue of how can an effective computation be performed. By this view, we may rephrase the Church-Turing Thesis as follows:

Church-Turing Thesis: *The partial functions over the natural numbers that can be effectively computed are exactly the functions that can be computed by Turing machines.*

means is that the original thesis needs to be adapted to refer to the nature of non-function-computing uses of computation. See, for example, [19].

9.3 How Is It Computed?

There are conflicting views as to what computational means can participate in an effective computation. A first question is *who/what* makes the computation: Is it a *human being* or a *machine*? The next questions correlate between the two. If a human performs the computation, what physical entities can he/she employ? And if the computation is done by a machine, must a human be able to provide the input to the function and also understand its output?

The dominant view is that a human (or some other intelligent agent) should be able to use the computation for her own purposes. A natural phenomenon that computes some function over entities that we, qua humans, cannot figure out is possibly an interesting issue to investigate in physics, but it is somewhat off the Church-Turing track [15, 16]. (It relates more to what is sometimes called *Deutsch's Principle*, which we discuss briefly in Sect. 9.6.2.2. As opposed to the *Physical Computational Thesis* [see below], which demands that a human be able to communicate and understand the input and output of a computation, this principle relates to physics alone, its capacities and limitations, without getting into what humans might be able to derive from such computations.) The main question here is whether the human performs the computation with his/her own "bare hands" or with the possible aid of some physical machinery.

The classic minimal means for a human to use in a computation is (unlimited) paper and pencil. Turing, in his classic 1936 paper, considered these quotidian means when providing arguments in favour of Turing machines.

When allowing the usage of additional machinery, there are obviously debates over what kind of machinery should be allowed. This argument is therefore related to the intensional question of how a computation is to be performed. The more general interpretation permits arbitrary physical devices as long as their construction accords with modern physics. We elaborate on physical aspects in Sect. 9.6.

The next issue is how creative can the human participant be. Undoubtedly, no limits are put on the ingenuity and creativity of the human who *designs* a solution. On the other hand, when *performing* the computation, his or her actions should be systematic. The prevailing view is that, in the course of performing the computation, the human should follow definite, unambiguous, predefined instructions, and avoid any appeal to creativity [21]. When considering the question of whether the human mind is limited to Turing computability, the creative aspect of human activity also comes into play, a subject we will touch on later.

The Bottom Line

Both interpretations, that of only having paper and pencil and that of having arbitrary machinery at one's disposal, are widely considered in the literature. Both are referred to as the "Church-Turing Thesis" and sometimes under other names, too. We shall distinguish between the *mathematical* and *physical* theses, respectively.[2] Although

[2] There are numerous other names for the various versions of the thesis in the literature. Even the terms "mathematical" and "physical" have different connotations for different authors.

we use the adjectives "mathematical" and "physical", the theses do not speak of the limitations of mathematics and physics per se, but rather of the capabilities of humans who use those resources for performing computations.

Mathematical Computational Thesis: *The partial functions over the natural numbers whose computations can be effectively performed by a human who systematically uses only paper and pencil of finite but unlimited quantity and who has finite but unlimited time are exactly the Turing-computable numerical functions.*

Physical Computational Thesis: *The partial functions over the natural numbers whose computations can be effectively performed by a human who systematically uses any physical device of finite but unlimited resources, including finite but unlimited time, are exactly the Turing-computable numerical functions.*

We are taking a purely anthropocentric point of view: What can a human compute systematically, following predefined rules, be it with only the most basic of tools (pencil and paper), or with arbitrarily elaborate physically feasible devices? What it may be that natural systems or fabricated apparatuses can calculate for their own, internal consumption is not the issue under inquiry, to the extent that they do not contribute to the repertoire of functions available to us humans.

We elaborate on the mathematical thesis in Sect. 9.5 and on the physical thesis in Sect. 9.6.

9.4 What Can Be Proved?

To Prove or Not to Prove? This is the Philosophical Question

There is a contentious philosophical debate on whether there is a sense in trying to prove the thesis. (See, for example, the arguments in [22, 23] and the references therein.) In a nutshell, the dominant claims against a proof assert that an intuitive notion, such as effectiveness, cannot be proved equal to a formal notion, such as Turing computability. No matter what proof you provide, it is based on some basic assumptions. Therefore, the "proof" only transforms the Church-Turing Thesis to a thesis claiming that one's assumptions properly capture the essence of effective computation. On the other hand, claims in favour of provability assert that every proof takes some "first principles" as axioms, or else nothing is provable. Hence, agreeing on first principles for effective computability, on top of which a proof of the thesis is considered, is as reasonable as agreeing on first principles to logic and mathematics, on top of which every mathematical proof is considered. One can still argue about how close the principles are to the intended intuitive notions, just as one may speculate whether the axioms of ZFC properly capture the full essence of a set.

In parallel with the philosophical debate on whether or not a proof of the thesis is plausible, there is a rich literature on *arguments* for and against the thesis, as well as various *concrete proofs* of the thesis within specific formal settings. We elaborate on some of them in Sects. 9.5 and 9.6.

9.4.1 Against Provability

The view that the thesis cannot be proved harks back to Church and Kleene:

> This definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion. [4, p. 356]
>
> Since our original notion of effective calculability of a function (or of effective decidability of a predicate) is somewhat vague intuitive one, the thesis cannot be proved". [24, p. 317]

Likewise, Janet Folina claims, "There is a good deal of convincing evidence that CT [the Church-Turing thesis] is true", yet, "It is simply not possible to prove that an informal, intuitive, notion has a single precise (formal or axiomatic) articulation" [22, p. 321].

László Kalmár further claims that, not only doesn't the thesis properly capture the intuitive notion of effectiveness, but we must not try to mathematically capture it:

> There are pre-mathematical concepts which must remain pre-mathematical ones, for they cannot permit any restriction imposed by an exact mathematical definition. Among these belong, I am convinced, such concepts as that of effective calculability, or of solvability, or of provability by arbitrary correct means, the extension of which cannot cease to change during the development of Mathematics. [25, p. 79]

As for Turing, he also considered computability to be an intuitive notion that cannot be defined mathematically, though he did claim that the right viewpoint to the question is via analysing the possible processes of a computation:

> No attempt has yet been made to show that the "computable" numbers include all numbers which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is "What are the possible processes which can be carried out in computing a number?" [5, p. 249]

9.4.2 In Favour of Provability

Kurt Gödel has been reported (by Church in a letter to Kleene cited by Davis in [26]) to have thought that

> It might be possible … to state a set of axioms which would embody the generally accepted properties of [effective calculability], and to do something on that basis.

Joe Shoenfield similarly asserted that:

> It may seem that it is impossible to give a proof of Church's Thesis. However, this is not necessarily the case…. In other words, we can write down some axioms about computable functions which most people would agree are evidently true. It might be possible to prove Church's Thesis from such axioms. [27, p. 26]

The late Elliott Mendelson further claimed:

> It is completely unwarranted to say that CT is unprovable just because it states an equivalence between a vague, imprecise notion (effective computable function) and a precise mathematical notion (partial recursive function). [28, p. 232]

He asserts that a nonmathematical notion can indeed be fully matched with a mathematical one, as for example, is the case with what he calls *Peano's thesis* [28, p. 230], which provides an exact mathematical definition in terms of sets of ordered pairs for the intuitive notion of a function.

Mendelson (and Gandy [29] before him) even asserts that Turing's arguments in favour of the thesis (see Sect. 9.5) are in fact a proof for it:

> The fact that it is not a proof in ZF or some other axiomatic system is no drawback; it just shows that there is more to mathematics than appears in ZF. [28, p. 233]

Shapiro supports Mendelson's view, claiming, "We can and do prove theses a lot like CT" [23]. (For a lengthy discussion of the issues, see [30].)

Mendelson further claims that the concepts of effectiveness and Turing computability (recursiveness) are equally vague:

> The concepts and assumptions that support the notion of partial-recursive function are, in an essential way, no less vague and imprecise than the notion of effectively computable function; the former are just more familiar and are part of a respectable theory with connections to other parts of logic and mathematics.

He compares it to the case of functions and sets:

> Functions are defined in terms of sets, but the concept of set is no clearer than that of function.

In 2000, Harvey Friedman predicted that

> There will be [in the 21st century] an unexpected striking discovery that any model of computation satisfying certain remarkably weak conditions must stay within the recursive sets and functions, thus providing a dramatic "proof" of Church's Thesis. [31]

9.4.3 What Does It Mean to Disprove the Thesis?

As with the positive direction of proving the thesis, there are also philosophical debates on whether there is a sense in trying to disprove it. If we deny the ability to formalize an intuitive notion, neither can we prove that something is included in it.

Yet, the possibly more common view is that there is a significant difference between the essence of proving and disproving the thesis. Indeed, once an effective computation scheme that is stronger than Turing machines is introduced, we might agree on its effectiveness and recognize the falsehood of the thesis.

Saul Kripke, for example, writes,

> That the recursive or Turing-computable functions are all in fact effectively calculable, can I think easily be established simply on the basis of an intuitive notion of calculability. No axiomatization is required, just a direct argument. And the argument should be regarded as a rigorous intuitive proof of its result. [32, p. 78]

It is harder, no doubt—some would say impossibly harder, to prove that *all* effective means are included in a proposed formalism than that some particular formally defined function is effectively computable.

Primitive recursion, for instance, was considered at some point as a candidate formalism for characterizing effective computability, yet once general recursion was introduced, it was widely agreed to be effective and to exceed the computational power of primitive recursion.[3] Quoting Henk Barendregt [34]:

> One may wonder why doubting Church's Thesis is not a completely academic question. This becomes clear by realizing that [Skolem in 1923] had introduced the class of primitive recursive functions that for some time was thought to coincide with that of the intuitively computable ones. But then [Ackermann in 1928] showed that there is a function that is intuitively computable but not primitive recursive.

9.5 The Mathematical Thesis

What are the "Rules of the Game"?

The exact rules of the game are, of course, vague, as is the abstract notion of effectiveness. Still, some rules are clear enough. Computation is discrete, that is, done in steps, and the number of steps should be finite, though it may be unbounded. The computational process has a finite description, independent of the specific inputs. The space and time made available for the purpose of the computation are accordingly unbounded but finite. Computation is generally considered to be symbolic. "There is no calculation without representation" [35] (see [36]). That is, the data with which one works consist of finite symbolic configurations, where the symbols (or labels) are

[3] In [13, 33] it is shown that while strict containment of function sets is in general not enough to infer the presence of extra computational power, general recursion is indeed more powerful than primitive recursion, even under rigorous definitions of power comparison.

drawn from some finite set given in advance. It bears stressing that computation of a numerical or string function may involve additional domains, which need, likewise, to be represented symbolically.

This ideal, unbounded, nature of the computation—of both a Turing machine and an "effective computation"—is intuitively in tension with the effectivity of the computation. Indeed, in contrast to the common claim that Turing computablity is effective, there are counterclaims that even Turing machines are not effective, mentioned below. Yet, "effective" was never meant in the sense of reasonable or practical demands on time, or space, or other resources. This is the reason for the rich field of complexity theory, and for the distinction between computability and complexity. The latter relates to the extended Church-Turing thesis, while we concentrate on the former.

The Empirical Evidence

Since the seminal works of Church, Turing, Gödel, Herbrand, Kleene, and Post in the thirties, defining general computational models, and until current days, all of the general-purpose models were shown to have exactly the same computational power given finite and unbounded time and memory. Among these models are Turing machines, the recursive functions, lambda calculus, Post canonical systems, modern programming languages, and many others. This empirical fact, having so many different views of computation, all leading to the same set of functions, gave significant confidence in the validity of the Church-Turing thesis. As Hartley Rogers writes in his classic text in support of the thesis:

> The proposed characterizations of Turing and of Kleene, as well as those of Church, Post, Markov, and certain others, were all shown to be equivalent. [37, pp. 18–19]

9.5.1 Non-empirical Arguments in Favour of the Thesis

Next, we enumerate some of the efforts made to justify or prove the thesis on a non-empirical basis, starting with Church's and Turing's original ones.

Church's Arguments

Church, in his classic paper, provided two arguments in favour of identifying effectiveness with λ-definability or general recursiveness. The first concerns an effective algorithm and the second an effective logical proof:

> No more general definition of effective calculability than that proposed above can be obtained by either of two methods which naturally suggest themselves (1) by defining a function to be effectively calculable if there exists an algorithm for the calculation of its values (2) by defining a function F (of one positive integer) to be effectively calculable if, for every positive integer m, there exists a positive integer n such that $F(m) = n$ is a provable theorem. [4, p. 358]

In the same paper, it is shown that λ-definability and general recursiveness are equivalent in the sense that each can simulate computation in the other sense.

In the first argument for comprehensiveness of the definition of effectiveness, Church considers the following:

- An algorithm consists of a finite sequence of expressions: "For example, in the case of a function F of one positive integer, an algorithm consists in a method by which, given any positive integer n, a sequence of expressions (in some notation) $E_{n1}, E_{n2}, \cdots, E_{nr_n}$, can be obtained".
- Each step of the algorithm should be recursive: "We take the effective calculability of G and H to mean recursiveness" [4, p. 357], where G and H are functions over natural numbers that represent the algorithm's step computation when representing the algorithm's expressions with their Gödel numbers.

Under these assumptions, "the recursiveness (λ-definability) of F follows by a straightforward argument" [4, p. 357]. Church further claims:

> If this interpretation or some similar one is not allowed, it is difficult to see how the notion of an algorithm can be given any exact meaning at all. [4, p. 357]

Overall, the logic community was not at all convinced by Church's arguments, as demonstrated by Gödel's response (reported by Kleene):

> According to a November 29, 1935, letter from Church to me, "Gödel regarded as thoroughly unsatisfactory" Church's proposal to use λ-definability as a definition of effective calculability. [38]

In the second argument, Church considers a proof to be in some "symbolic logic, which contains a symbol ... for equality", starting with "*formal axioms*" and proceeding by "the *rules of procedure*". Analogously to the first argument, he assumes that "each rule of procedure must be a recursive operation", and that "the complete set of formal axioms must be recursively enumerable" [4, p. 357]. Under these assumptions, the recursiveness of the entire proof directly follows.

The main claim against this argument is that it's circular, demanding that the basic steps of the computation be themselves recursive (e.g. [27, 39, §6.5]). All the same, similar arguments were given by Turing (in addition to other arguments), and they have been further developed by Kripke [32]. See below.

Turing's Arguments

In his classic paper, Turing gave three different arguments in favour of identifying effective computability with computability via Turing machines. The first builds on the limitations of a human being working in a rigorous, bureaucratic way. His second argument concerns the equivalence of Turing computability and "Hilbert functional calculus", and resembles Church's arguments, namely, "The steps of any effective procedure (governing proofs in a system of symbolic logic) must be recursive" [39]. This has been further developed by Kripke to reduce the mathematical thesis to a thesis about first-order logic (see below). The third provides "empirical evidence" that large classes of real numbers are Turing computable.

We elaborate on Turing's first argument. It builds on the following limitations of a [human] computer [5, pp. 249–250]:

- "Computing is normally done by writing certain symbols on paper."
- "The two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares."
- "The number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent."
- "The behaviour of the computer at any moment is determined by the symbols which he is observing, and his 'state of mind' at that moment."
- "There is a bound B to the number of symbols or squares which the computer can observe at one moment."
- "The number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be 'arbitrarily close' and will be confused."
- "In a simple operation not more than one symbol is altered.... The squares whose symbols are changed are always 'observed' squares."
- "The new observed squares must be immediately recognisable by the computer.... They can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount."

Turing's argument is considered by many to be a most convincing one, some even going so far as to view it as an actual proof. For example, Robin Gandy called it "Turing's proof", and said, "The proof is quite as rigorous as many accepted mathematical proofs ..." [29, p. 82], and likewise Mendelson [28] and Shapiro [23].

Church, reviewing Turing's classic paper, was convinced by the arguments: "Computability by a Turing machine ... has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately—i.e. without the necessity of proving preliminary theorems" [40, p. 43].

Gödel adopted Turing's argument, saying [41, p. 203], "The resulting definition of the concept of mechanical by the sharp concept of 'performable by a Turing machine' is both correct and unique." In later years, he did, however, raise doubts about Turing's arguments:

> What Turing disregards completely is the fact that mind, in its use, is not static, but constantly developing.... Although at each stage the number and precision of the abstract terms at our disposal may be finite, both (and, therefore, also Turing's number of distinguishable states of mind) may converge toward infinity in the course of the application of the procedure. [42, p. 306]

The common response to this objection of Gödel is that an algorithm should be completely given in advance. Kleene wrote:

> Thus algorithms have been procedures that mathematicians can describe completely to one another in advance of their application for various choices of the arguments. How could someone describe completely to me in a finite interview a process for finding the values of a number-theoretic function, the execution of which process for various arguments would be keyed to more than the finite subset of our mental states that would have developed by the end of the interview, though the total number of our mental states might converge to infinity if we were immortal? [21, p. 50]

Wang notes that

> Gödel realizes that his incompleteness theorem does not by itself imply that the human mind surpasses all machines. [43]

> There is no reason why the number of states of the mind should not converge to infinity in the course of its development. [44, pp. 325–326]

In other words, human mathematical reasoning—as opposed to algorithmic reasoning, by human or machine—need not be held to the requirement that states of mind be bounded [45]. The assertion that humans or humankind *cannot* outperform Turing machines even in such non-algorithmic ways has been referred to as the "Non-mathematical Church-Turing Thesis" [46], to avoid confusion with the algorithmic, mathematical one we are discussing.

Another criticism of Turing's arguments, or more accurately a sort of request for further development, is that they are not formal enough. The quest is for a set of postulates that define the limitations of effective computability along with a proof that under these limitations no computation exceeds Turing computability. There are several works along this line, and we elaborate on some of them next.

Kripke's Arguments

Saul Kripke reduces the Church-Turing Thesis to "Hilbert's Thesis" (as Martin Davis called it [47, p. 41]; [48]), namely, "The steps of any mathematical argument can be given in a language based on first-order logic (with identity)" [32, p. 81]. He asserts, as did others before him,[4] that all "computation is a special form of a mathematical argument", and that "computation is a deductive argument from a finite number of instructions… albeit one of a very specialized form" [32, p. 80].

Thus, by Hilbert's Thesis, the premises of the computation as well as its steps can be expressed in first-order logic. This assumption in hand, Kripke obtains, as a corollary of Gödel's completeness theorem for first-order logic (with identity), the *First-Order Algorithm Theorem*, as he calls it:

> Any algorithm whose steps are meaningfully expressed in a first-order language, the conclusion of the computation does indeed follow from the premises. [50, at 1:18]

Since every first-order proof is Turing computable, as already shown by Turing, a proof of the thesis directly follows. "What [Turing] gives as argument 2 in his paper is essentially that any first-order algorithm can be computed by a Turing machine, as expressed in my own terminology" [50, at 1:19]. The first-order inference system works with terms and formulas to derive the output of the algorithm.

[4] Sieg [49] in explaining Church [4]: "Calculability is explicated by that of derivability in a logic."

It should be remarked that Kripke further claims that the proof also holds with respect to machine computations, as their computation steps must follow some deductive rules:

> Any machine—think of it as an abstract object, it is a program. Do the steps in the program follow from the initial instructions in the program as given or don't they? And can they be expressed in a final conventional mathematical language formalized in first order logic or can't they? If the answer is no, I don't regard it as a computation at all. If the answer is yes, the theorem that I just stated is enough to give an affirmative answer that the function that comes out of it must be Turing computable. [50, at 1:27]

An objection to Kripke's argument, for example by Michael Rabin [50, at 1:54], is that it is circular: A computation can be viewed as a logical proof, and the proof should be susceptible to mechanical verification. As Kripke puts it: "Granted that the proof relation of such a [formal logical] system is recursive (computable)" [32, p. 81]. Yet, disbelieving the thesis, there might be a stronger logic that can be mechanically checked (by an effective machine stronger than a Turing machine). Rabin would be equally surprised if a machine stronger than Turing's is found or if a notion of computation not expressible in first-order logic is found.

Sieg's Arguments

Gandy [51], Wilfried Sieg [52], and Robert Soare [9] divide Turing's first argument into a philosophical part, in which there are assumptions on what an idealized human (a "computor" with an *o*) can rigorously do with paper and pencil, and a mathematical part, in which it is proved that under the above philosophical assumptions, every computation is Turing computable.

The assumptions on the computor are reduced by Sieg to principles of boundedness and locality:

> The constraints Turing imposes on symbolic processes derive from his central goal of isolating the most basic steps of computations, that is, steps that need not be further subdivided. This objective leads to the normative demand that the configurations, which are directly operated on, must be *immediately recognizable* by the computor. This demand and the evident limitation of the computor's sensory apparatus motivate most convincingly two central restrictive conditions.
>
> (Boundedness) A computor can immediately recognize only a bounded number of configurations.
>
> (Locality) A computor can change only immediately recognizable configurations. [52]

Sieg's formal axioms aim at capturing these informal assumptions.

> The axioms are formulated for discrete dynamical systems and capture the above general ideas precisely; they should be viewed as determining classes of "algebraic structures" of which particular models of computations are instantiations. [52]

Sieg considers a computation to be a state transition system, in which states (as in [51]) are finite objects that can be represented by nonempty hereditarily finite sets over an infinite set of atoms. The classes of states that are considered are closed under isomorphism. His axioms formalize the locality and boundedness principles in this setting.

If x is a given state, regions of the next state are determined *locally* from particular *parts for* x on which the computor can operate. *Boundedness* requires that there are only finitely many different kinds of such parts. [52]

These "kinds" are formalized by isomorphism types, termed *stereotypes*.

A Turing computor is defined to comprise a class S of states closed under isomorphism, a finite set T of stereotypes, and an operation G on the stereotypes of T, satisfying the requirement that for every state x of S there exists a state z of S (the "next state") such that the following three axioms hold:

◇ There exists in x a unique maximal stereotype from T. This unique stereotype is the *causal neighbourhood* of x.
◇ Applying the operation G on the causal neighbourhood of x yields, up to isomorphism, a unique region of the state z.
◇ The state z consists of its above region and the whole of x, excluding the causal neighbourhood of x.

The first two axioms formalize the principal of *local causality* and the third the principle of *global assembly*.

A computation of a Turing computor is defined to be a finite sequence of transitions between states using operation G, *halting* when the next state is the same as the preceding one.

As for computations over domains that do not consist of hereditarily finite sets, Sieg speaks of "suitable encoding and decoding" without detailing what is to be considered suitable:

A function **F** is (Turing) *computable* if and only if there is a Turing computor **M** whose computation results determine, under suitable encoding and decoding, the values of **F** for any of its arguments. [52]

Sieg's work served to clarify and sharpen Turing's arguments. Some researchers view the Turing-Sieg arguments as actually constituting a proof of the thesis. Sieg, himself, views his work as proving a representation theorem for the class of systems that can be formulated in a way that meets the stated criteria. Unsurprisingly, those who are not convinced by Turing's arguments tend not to be convinced by Sieg's more formal arguments either. Another criticism of Sieg's approach (and of Gandy before him) is that expressing computability axioms in terms of hereditarily finite sets representing states—however sensible under the standard set-theoretical, foundational view of the mathematical universe[5]—is nonetheless insufficiently general for faithfully modelling effective computations [32, 54].

Gurevich, Boker, and Dershowitz's Arguments

As Church argued in his classic paper [4], a possible take on the mathematical thesis is that a Turing machine can compute every function that can be computed algorithmically, where the term "algorithm" stands for the intensional means of a

[5] Gandy's idea of basing a formalization of computability on hereditarily finite sets, which are precisely the finite objects of set theory, was presaged by Moto-o Takahashi [53].

computation. A central point of note is that the equivalence claim is extensional and not intensional—the algorithm itself is not supposed to be captured by the computation of a Turing machine and may have various abstraction levels. It is only the result of the algorithm, namely the function it computes, that is supposed to be captured by the result of the Turing machine.

When aiming at proving the thesis, one approach is to first define what an algorithm is [2, 54], and what atomic operations it can effectively employ.

At the turn of the millennium, Yuri Gurevich proposed three basic postulates for sequential (non-parallel and non-interactive) algorithms over arbitrary mathematical structures [55]. By this view, every sequential algorithm should satisfy the following properties:

- ◇ It can be viewed as a set of states and a transition function from state to state. The importance of this generic view has been emphasized by Don Knuth [56].
- ◇ Its states are (first-order) structures sharing the same finite vocabulary. States are closed under isomorphism, and the transition function preserves isomorphism. The relevance of logical structures appears already in Emil Post's notes [57]; the importance of isomorphism is stressed by Gandy [51].
- ◇ There is a finite bound on the number of vocabulary-terms that affect the transition function. This formalizes ideas expressed by Kolmogorov [58].

Based on these three principles, Gurevich proved that every such sequential algorithm is captured by an *abstract state machine*, a formalism he defined that expresses generic algorithms as sets of parallel conditional assignments employing the operations of the structure [55].

Gurevich's postulates and proof are a natural starting point for axiomatizing effectiveness over arbitrary domains, not just strings. Yet, there are a couple of "loose edges" that allow an algorithm that satisfies Gurevich's postulates to yield an ineffective computation:

- The atomic operations, which can be viewed as the initial state of the algorithm, are not limited and might thus introduce ineffectiveness.
- The postulates consider a single algorithm and not an entire computational model. Interpreted over an arbitrary domain, the computed function might be ineffective with respect to a different domain or even with respect to the same domain, under a different representation.

In 2005, the present authors addressed the above issues by considering the whole of a *computational model*, comprising the computations of a *set* of functions, rather than a single function, requiring all computations within the same model to share the same domain view, and restricting the initial states to only have finite data. The additional restrictions were set out in a fourth postulate [59]:

- ◇ There is an initial state to the computation and it comprises only finite data in addition to the domain representation. The latter is isomorphic to a Herbrand universe.

They proved that every computational model that satisfies the above four postulates cannot compute more than Turing machines [14].

In 2008, Dershowitz and Gurevich addressed the two issues above in a slightly different fashion [60]. They required an injective mapping between the arbitrary domain and the natural numbers, and limited the initial data to be tracked, under that representation, by a recursive function.

Shortly afterwards, the present authors also showed that the three different approaches taken to define effectiveness on top of Gurevich's postulates for a sequential algorithm—their's from 2005, Dershowitz and Gurevich's from 2008, and Wolfgang Reisig's from 2008 [61], yield the exact same notion of effectiveness [62]. Whereas the latter two involve a measure of circularity, using recursiveness or Turing-computabilty to guarantee that initial states are effective (see the objection in [63, Postscriptum]), the former, equivalent approach builds only on first principles.

In favour of their view is their notion of state, namely, any mathematical first-order structure, which might be considered more general and natural than strings or hereditarily finite sets. Furthermore, their axioms are built on top of Gurevich's postulates for sequential algorithms, which aim at capturing the essence of classical algorithms. As a result, their formalization of effective computability captures a most general setting of computational models. (Supportive reactions include [64, p. 351n] and [32].)

The natural criticism of this attempt is that the result just "moves the goalpost", shifting the question to whether the provided postulates capture all of what is effective (e.g. [65, p. 431], [63, Postscriptum]). Gurevich himself believes that his postulates properly capture the notion of a sequential algorithm, but avers that no definition can capture what a most "general" algorithm is [54]. Another criticism is that the postulates are not sufficiently formal [63, Postscriptum], but that can be rectified.

9.5.2 Disproving the Thesis

One way to disprove the mathematical thesis would be to find a function that is not Turing-computable but which one can convincingly claim can be computed effectively by an idealized human (immortal, untiring, unerring, etc.) equipped with pencil and paper and following predetermined instructions. This would be analogous to the rejection of primitive recursion as capturing all effective computation on the basis of the effectivity of Ackermann's function.

As popular textbooks explain:

> If one ever found a procedure that fitted the intuitive notions, but could not be described by means of a Turing machine, it would indeed be of an unusual nature since it could not possibly be programmed for any existing computer. [66, p. 80]

> It is theoretically possible, however, that Church's thesis could be overthrown at some future date, if someone were to propose an alternative model of computation that was publicly acceptable as fulfilling the requirement of finite labor at each step and yet was provably capable of carrying out computations that cannot be carried out by any Turing machine. No one considers this likely. [67, pp. 168–169]

There have been several attempts in this direction, as we will see in this section. One of the earliest was by Kalmár [25] (explicated in [68]). In particular, he questioned the commonplace requirement that an algorithmic method be uniform for all inputs:

> We regard as effectively calculable any arithmetical function, the value of which can be effectively calculated for any given arguments in a finite number of steps, irrespective how these steps are and how they depend on the arguments for which the function value is to be calculated. In particular, I do not suppose the calculation method to be "uniform". [25, p. 73]

We will encounter such misgivings again.

Considering the computation of specific instances of the halting problem, Turing commented already in his 1936 paper as follows:

> It is an immediate consequence of the theorem of §8 that δ is not computable. It is (so far as we know at present) possible that any assigned number of figures of §8 can be calculated, but not by a uniform process. When sufficiently many figures of δ have been calculated, an essentially new method is necessary in order to obtain more figures. [5, p. 253]

In other words, the thesis places no limits on non-uniform human problem solving abilities, let alone those of endless generations of humankind. Alfred Tarski is quoted by Benjamin Wells [69] as referring to problems for which there is a series of non-uniform solutions as (intuitively) "decidable".

A related debate surrounds the so called "Artificial Intelligence Thesis" and the question whether a computational device can act with general human-level intelligence. This AI thesis demands more of the mechanical device in that it must also be creative, imaginative, and intuitive like human beings. We do not address the AI question but rather concentrate on mathematical proficiency and ignore other aspects of human intelligence.

Early on, several objections were voiced that the thesis was wrong for the opposite reason, that Turing's formalism was overly rich, and that not every Turing-computable function is effective. See [70, 71]. With the growth of programming and programming languages, this view is no longer tenable and were dispensed with already in [72, 73].

Nowadays, one occasionally hears objections from the point of view that unbounded time and space are wholly unrealistic idealizations, so these imaginary machines are not "really" effective in any practical sense. From the point of view of this survey, these are tangential issues.

Another issue, first raised in [74], is how a human interprets the strings that a Turing machine uses for input and output. Michael Rescorla, for instance, has pointed out correctly that if one imposes "a deviant interpretation upon numerals then Turing machines can 'compute' intuitively non-computable numerical functions" [75].

See [75–78] for a lively discussion. Our preferred solution is to require that all recursive functions remain computable under whatever representation is chosen, so that any model that ostensibly computes something that a Turing machine cannot must also be capable of computing whatever a Turing machine can with the identical encoding scheme [13, 79].

Penrose's Arguments

In *The Emperor's New Mind* [80] and especially in *Shadows of the Mind* [81], the eminent physicist, Roger Penrose, contends that human reasoning cannot be captured by a mechanical device because humans detect nontermination of programs in cases where digital machines cannot [82]. Penrose thus adapts the similar argumentation of Lucas [83], which was based on Gödel's incompleteness results.

How this impinges on the Church-Turing Thesis depends on whether the means by which Penrose contends that humans can solve the specific halting problem he has in mind should be deemed "effective" or just "intuitive". This is somewhat unclear, because Penrose does not claim to know himself how it is that people achieve what he asserts they do. Were it by somehow engaging a quantum mechanism embedded in the human brain (as Penrose speculates in [81]), then this would be an argument against the physical thesis, but not the mathematical one. But if the basis of the super-algorithm used for this purpose is ordinary human mathematical knowledge, such as that taught in an undergraduate computability course (as actually employed in the diagonalization argument in [81]), then this would be a frontal attack on the mathematical thesis.

In a nutshell, Penrose's argument runs as follows:

1. Consider all current sound human knowledge about nontermination.
2. Suppose now that one could reduce said knowledge to a computer program.
3. Then one could also create a self-referential version of said program.
4. From the assumed existence of such a "diagonal" program, a contradiction to its correct performance can be derived.

Penrose's resolution of this contradiction is to deny the validity of the second step: No finite program can incorporate all the algorithmic and mathematical knowledge that finitely many humans currently have at their disposal.

Since Penrose outlines the procedure by which a human can leverage all the knowledge of the first step to determine that the program in question is nonterminating, we may conclude that his argument serves to indicate that humans are somehow more capable than Turing machines, and that the Church-Turing thesis is false.

Penrose's conclusions, namely that human reasoning cannot be faithfully simulated by Turing machines, have been heavily critiqued (e.g. [84–86]). The possible rejoinders are classified in [87]. Assuming that the *idealized* humans in question will not die before they decide what to respond and that they would neither knowingly lie nor impetuously respond without thinking (something that real humans are apt to do under the circumstances), resolutions of the conundrum raised by Penrose fall mainly in two categories: (a) humans can err in judgement while believing they are being truthful; and (b) a human may realize or suspect that there are matters that may lie beyond one's ability to fully comprehend or to express, such as the soundness or consistency of one's own vastly complicated internal workings.

An example of objection (a) is this:

> No human mathematician can claim infallibility. We all make mistakes! So there is nothing in Gödel's theorem to preclude the mathematical powers of a human mind being equivalent to an algorithmic process that produces false as well as true statements. [88]

An example of the second objection is

> The possibility exists that each of the rules that a human mathematician explicitly relies on, or can be rationally persuaded to rely on, can be known to be sound and that the program generates all and only these rules but that the program itself cannot be rendered sufficiently "perspicuous" for us to know that that is what it does.... A program which simulated the brain of an idealized mathematician might well consist of hundreds of thousands (or millions or billions) of lines of code. Imagine it given in the form of a volume the size of the New York telephone book. Then we might not be able to appreciate it in a perfectly conscious way, in the sense of understanding it or of being able to say whether it is plausible or implausible that it should output correct mathematical proofs and only correct mathematical proofs. [89]

In other words, one may in fact only know correct rules, but at the same time be unable to establish that fact. Or, one may even realize that declaring soundness out loud might ipso facto make it false. In such cases, it would seem to us that a cautious human may simply prefer, or feel compelled, to maintain judicious silence [87].

9.6 The Physical Thesis

What are the "Rules of the Game"?

In the mathematical thesis, the number of computation steps is finite but unbounded. This no longer stands as a clear rule in the physical thesis—a step is not always a physical notion. We might want to consider analogue phenomena and how discrete entities are conceived in the physical world. The input and output of such a computation are to be understood by the human user as natural numbers, whereas the computation and entities involved are continuous, analogue processes.

We do not address the issue of what constitutes an effective real-valued function, normally expressed in terms of approximations of increasing accuracy (e.g. [90]).

Analysing the physical thesis, one needs to look into current physical theories of the universe and how humans can accordingly take advantage of physical phenomena for performing computations. Whereas the resources consumed in a computation must be finite, though unlimited upfront, there might indeed be limitations on them stemming from the constraints of physical theories, such as the speed of light, the uncertainty principle, etc. As put by Oron Shagrir [91, p. 231]:

> In this context, a physical system is any system, real or potential, such that (a) its states occupy a finite space, (b) its terminating dynamics are completed in finite real time, and (c) its dynamics are consistent with the laws of physics.

9.6.1 Proving the Thesis

Several arguments have been aired in support of the physical thesis. These arguments provide axioms that are claimed to express limitations on physical computations in consonance with current physical theories. Then, properties of all systems obeying those axioms are derived. In considering an attempt to disprove the physical thesis, one may ask which of the axioms is violated and argue whether a particular axiom should be adopted.

Gandy's Arguments

The most notable effort to prove the physical thesis was initiated by Robin Gandy (who was Turing's student) in 1980 [51]. He defined four principles that, according to his understanding of physics, every "discrete mechanical device" should satisfy. The principles are derived from two basic physical assumptions: that there is a lower bound on the size of atoms and that there is an upper bound on the speed of signal propagation. The main difference between Gandy's principles and Turing's assumptions on effectiveness is that Gandy allows for unbounded parallelism: There can be unboundedly many different units computing in parallel. Gandy then proved that whatever is computed by such a device is also Turing computable.

Gandy's four principles are the following, stated roughly:

◇ The machine can be described by a transition system, where the states are hereditarily finite sets.
◇ The set-theoretic rank of the states is bounded.
◇ There is a finite set of basic parts from which all states can be assembled.
◇ For each region of the machine, its next state only depends on its neighbourhood of a bounded size.

Gandy's work was much appreciated for its novelty and depth, though perceived as somewhat overly complicated [23, 54]. Sieg and John Byrnes simplified Gandy's principles and arguments [92], and they were later further modified by Sieg [52]. (See below.) They are critiqued in [91].

One natural claim against Gandy's arguments are that his four principles are too restrictive, not covering all of what is physically effective. In particular,

- The restriction to discrete devices is improper as physical devices may be analogue; we should also consider what the limits of analogue computations are [32, 93, 94].
- The first principle of finite information for each state and the fourth principle of local causality are not obeyed by quantum physics [95, 96].
- The parallelism is only synchronous, namely the different units proceed in synchronous steps, whereas actual independent computations are not synchronous [54].
- The requirement to have finitely many steps in a terminating computation is disobeyed by devices that allow for unbounded acceleration, which might be possible by laws of special and general relativity [94].

Sieg's Arguments

Following in Gandy's footsteps, Sieg extends the suppositions about what an idealized human "computor" can rigorously do with paper and pencil (Sect. 9.5) to assertions regarding what a parallel computing device (Gandy machine) can do [52]. These assertions still stem from the same principles of boundedness and locality, justified in this case by the physical lower bound on the size of atoms (or fundamental particles) and the physical upper bound on the speed of signal propagation, yet generalized to allow for parallelism. After formalizing these assertions, he proves that every computation of such a Gandy machine is in fact Turing computable.

Recall that Sieg defines a computor to comprise a class S of states closed under isomorphism, a finite set T of stereotypes, and an operation G on the stereotypes of T, satisfying some requirements. Now, he defines a Gandy machine to have in addition another set T' of stereotypes and another operation G' on stereotypes T', which allow it to coördinate parallel computations. He then generalizes the axioms that formalize *local causality* and *global assembly* for such a parallel machine.

Compared to Gandy's original four principles, as roughly expressed above, Sieg retains the first and fourth, while relaxing the second and third. Since Sieg's formulation puts limits only on causal neighbourhoods, entire states need not have a bounded set-theoretic rank and need not be assembled from a finite set of basic parts. Accordingly, objections to Gandy's arguments that concern his second and third principles do not hold for Sieg's arguments, while claims against Gandy's first and fourth principle, as described above, still carry over to Sieg's.

Arrighi and Dowek's Arguments

Pablo Arrighi and Gilles Dowek generalized Gandy's principle and proof to quantum computations [95].

Gandy's first principle, according to which states can be defined by finite sets, is based on his assumption that information has a finite density, which no longer holds in quantum theory. (See [97].) Arrighi and Dowek provide a different assumption that also holds in quantum theory and allows for the first principle—that each projective measurement of the system, at any given point in time, may only yield a finite number of possible outcomes.

Gandy's fourth principle of local causality is based on bounded velocity, which fails to stand in the quantum setting. Arrighi and Dowek provide instead a quantum version of causality, based on the assumption that the global evaluation of the system is "localizable", namely that it is implementable by local mechanisms [98, 99].

9.6.2 Disproving the Thesis

There are two types of claims against the thesis; the first is that there are currently computing devices that exceed Turing computability, and the second is that our physical theories allow for theoretical devices that exceed Turing computability and might possibly be built in the future. The first claim, concerning existing machines, is mostly based on randomness and learnability. We have already encountered several claims that the human mind may somehow enjoy such "super-Turing" capabilities, particularly that of Penrose. One more is given below.

The second sees no reason for nature to submit to the constrictions imposed on it by computability theory. Kripke [32, p. 91] writes:

> Consider such dimensionless physical constants as the electron-proton mass ratio, or the fine structure constant. As far as I have heard, at the present time physical theory has nothing much to say about the mathematical properties of these numbers, whether they are algebraic or transcendental, or even rational or irrational. In particular, it might well be the case that the decimal expansions of these numbers are not Turing computable. Assume this is so, and also assume that time is infinite in extent and that there are no limitations in energy that prevent computations of these decimals to any number of places. As far as I can see, nothing in principle rules this out. Then an empirically possible machine would exist that calculates a decimal not computable in Turing's sense.

Many arguments for and against hypercomputation are collected in [100].

9.6.2.1 Existing Devices

Biological Devices

Allowing one to compute with the aid of any physical device and taking biology as an aspect of physics, we end up with having humans on both sides of the computation—both as those who perform the computation using a physical-biological device and as the device itself. With such an outlook, the border between the physical Church-Turing thesis and the AI Thesis gets even blurrier.

Bringsjord's Arguments

Selmer Bringsjord and his colleagues claim to have mustered strong evidence that human beings can compute more than Turing machines. They claim that people can solve the "busy beaver" problem, which is known to be beyond the capabilities of Turing machines. This is intended as an attack on computationalism, along the lines of Gödel's reflections.[6]

Bringsjord embarked on an NSF-funded project to compute explicit values of the Busy Beaver function, called here Σ^B. Given its success in evaluating several values, he then proceeds to make the following claim:

> Persons must in some real way be capable of infinitary information processing....
>
> The idea is really quite simple: If humans are smart enough to determine $\Sigma^B(n)$, they will eventually (perhaps after 100 years, perhaps after 1000, perhaps after 1,000,000,000, ...) be smart enough to determine $\Sigma^B(n+1)$ .... While [this premise] may not be unassailable, a variant would seem to be, viz., that if humans are smart enough to determine $\Sigma^B(n)$, it's then mathematically possible that they determine $\Sigma^B(n+1)$. [102]

[6] Another argument of Bringsjord, explicitly touted as "a case against Church's thesis", takes its cue from the perceived ability of humans to judge the literary quality of writings that they cannot for the life of them produce themselves [101].

Owen Kellett, Bringsjord's student and one of the authors of [102], elaborates:

> Human visual reasoning system possesses [natural, physical] computational powers beyond the Turing limit, and these powers can be employed in efforts to solve $\Sigma(n)$.... The Busy Beaver function ... is still computable by the fantastic capabilities of the human mind.
>
> While Turing machines hold their place in conventional computability theory, the laws of the physical world are beyond the computational expressive power of Turing machines. Therefore, humans, a part of this physical universe, can harness this power to perform computations beyond the Turing limit. [103]

Thus, this argument is akin to Penrose's, appealing to unknown natural hypercomputational mechanisms that are somehow harnessed by biological brains. As such it is an attack on the physical thesis (or the non-mathematical one). In his book with Michael Zenzen, *Superminds*, Bringsjord summarizes his point of view:

> Persons are superminds: they are simple souls capable of cognition capturable by standard computationalism, cognition capturable by *hyper*computationalism, *and* cognition that will forever resist capturing in any third-person scheme. [104, §7.3]

Bringsjord conveniently lists the main objections to this claim, presumably culled from reviews and other reactions:

- This assertion resembles the claim that the human record for the 100 m run will converge to 0 s, since we achieve a new record with each passing year.... Solving the busy beaver problem for input up to 6 ... doesn't tell us much about the limits of the human mind.[7]
- The theorem ... only implies that humans will get stuck billions of orders of magnitude past where current research on computing the function stands.
- Humans are far from immune to errors.
- There is some confusion of a specific person with all of humanity, past, present, and future.
- There is ... a hidden assumption that people are unaffected by nature.... Historically, many scientific breakthroughs happen serendipitously.

Randomness

True randomness is believed by most physicists to exist in nature [106], whereas an ordinary Turing machine is incapable of generating truly random values. Hence, we simply get that random sequences, which can be computed by actual devices, falsify the physical thesis [107, p. 74], [108, p. 10].

A counterargument is that a *random number* (qua indexed sequence of digits) is not a calculation in the intended sense of the thesis. Proper functions always provide the identical output to the same input, which is not the case if a device randomly generates output values or uses such values to guide computation.

[7] Herbert Robbins, the statistician: "Nobody is going to run 100-meters in five seconds, no matter how much is invested in training and machines. The same can be said about using the brain. The human mind is no different now from what it was five thousand years ago. And when it comes to mathematics, you must realize that this is the human mind at an extreme limit of its capacity" [105].

> The above remarks converge on the conclusion that genuine random processes are not computations. Unlike computations properly so-called, random processes cannot be used to generate the desired values of a function or solve the desired instances of a general problem. [16, p. 751]

Quantum physicist, David Deutsch, writes:

> Quantum computing machines, and indeed classical stochastic computing machines, do not "compute functions" in the above sense: the output state of a stochastic machine is random with only the probability distribution function for the possible outputs depending on the input state. [109]

So we might ask instead that the distribution of outputs satisfy some requirements.

In cases where a random process provides the same one output with high probability, it can be considered an *estimation* of a proper function, and we may ask whether the function that it estimates is computable by a Turing machine. The answer to this question is "yes", as one may derandomize such a process, trying all paths until some required percentage give the same answer [110] (though this does depend on the precise definitions [111]).

A different way to handle random numbers in a "proper function" setting might be to fix a computation by augmenting it with a memory cache. Consider a device M that maintains an unbounded array R of binary values, initially blank. Whenever M is presented with an input $n \in \mathbb{N}$, it does the following: If $R[n]$ is blank, it randomly chooses between 0 and 1, stores this bit in $R[n]$, and returns it as the value of the function on n; on the other hand, if $R[n]$ already has value v, it returns that v. Such a device computes a proper function $R : \mathbb{N} \to \{0, 1\}$, but has the characteristics of an *interactive* process, which in itself need not make it ineffective or non-Turing-computable, as discussed in Sect. 9.2. Yet, because it *learns* and *evolves* with time, one may argue whether it is better viewed as an infinite series of distinct computations, rather than as a device for repeatedly computing a single function. An additional reason to contend that this is not a valid attack on the thesis is the uniqueness of the device—any attempt to clone it will result in a different device, with different behaviour, computing a different function.

Evolution and Learnability

An interactive process evolves over time—it maintains a memory that may grow and change from one invocation of the process to another. A central question is then how does it evolve: Is its evolution predefined or not?

There are claims against the physical thesis, arguing that internet applications, such as Google Translate, transcend Turing computability (e.g. [112]). So as to consider such an evolving process as a proper function that always returns the same output for the same input, one may augment every "genuine" call to the application with a unique number and cache all previous queries as explained above with respect to a random process.

The main argument against such claims is that the evolvement of the application is not predefined. Google Translate, for example, changes over time not as a result of a predefined learning algorithm, but as a result of humans modifying it. Hence, claims that such computations exceed Turing computability are analogous to claims that the human mind, and possibly all of future humanity, exceed Turing computability in the limit. Such processes are usually not considered to lie under the rubric of effective means [21, p. 50].

9.6.2.2 Futuristic Devices

We list below some of the claims about theoretical devices that may agree with modern physical theories and compute more than Turing machines.

Zeno Machines

A *Zeno machine* or *accelerating machine* can be thought of as a standard Turing machine in which each step is performed in, say, half the time of the previous step [113–116]. Having such an accelerating machine, one can solve the halting problem: If the first step takes, for example, a second, any number of steps takes less than two seconds. Hence, after two seconds we (the machine users) can inspect the machine's tape and check if it has a halting mark. In case that it does not, we know that it does not halt. The logical possibility of such a device has long been debated [113, 117]. On the other hand, there are claims that such an accelerating device, as well as communication with it, is theoretically possible in the vicinity of certain kinds of black holes by certain interpretations of special and general relativity [118–122].

One criticism of this view is that its physical assumptions do not match current physical theories and that, even under these assumptions, communication is only possible from Earth to the vicinity of the black hole, and not the other way round, providing a "one-way ticket" to knowing whether computation has ended [123, 124], as well as logical considerations [125].

Accessible Reals

Equipped somehow with an infinite precision real number, a device can easily compute beyond Turing machines. A simple example is Fred Abramson's Extended Turing Machine, which can store a real number on a single square of its tape [126].

A more classic example of calculating with reals is ruler (straightedge) and compass (*RC*) computations. A common algorithmic definition runs something like this:

> Given a finite set of points $\mathcal{B} = \{B_0, \ldots, B_m\}$ in the Euclidean plane, a point P is *RC-constructible from the set* $\mathcal{B}$ if there is a finite set of points $\{P_0, \ldots, P_n\}$ such that $P = P_n$, $P_0 \in \mathcal{B}$ and every point P_i $(1 \leq P_i \leq n)$ is either a point of $\mathcal{B}$ or is at the intersection either of two lines, or of a line and a circle, or of two circles, themselves obtained as follows:
>
> – any considered line passes through two distinct points from the set $\{P_0, \ldots, P_{i-1}\}$;
>
> – any considered circle has its centre in the set $\{P_0, \ldots, P_{i-1}\}$ and its radius is equal to the distance $P_j P_k$ for some $j < i$ and $k < i$. [127, trans. [128]]

The widespread view is that RC-computations do not exceed Turing computability, merely allowing one to compute a certain family of algebraic numbers (Wantzel's Theorem; see [129]). Yet, if initial points are arbitrary real points in the plane—given either as inputs or as elements of the initial state of a device, then with the addition of a "unit" segment that represents the natural number 1 and basic geometric operations, one can compute non-Turing-computable functions over the natural numbers [130].

A more involved example is Hava Siegelmann's version of neural networks [131].

The main criticism of these types of computation, from an effectiveness point of view, is the inaccessibility of the infinite precision needed to set up the computation framework [132], and that "the claimed noncomputability was nothing more than that of the real numbers provided at the beginning" [7]. But were there a primordial uncomputable fundamental constant in the universe—and a priori there is no physical reason to preclude that possibility, then accessing its digits by means of some idealized experiment might serve to violate the physical thesis, as already noted [32, p. 91].

Quantum Computers

The dominant view of quantum computation is that, if it is feasible and scalable, it will allow one to compute faster than Turing machines but will not enable one to compute more [95, 109]. If this view holds true, it would disprove the "extended" thesis, but not the original one. Still, there are also more audacious claims that quantum mechanics might allow for computing devices that exceed Turing computability [96, 133, 134]. Tien Kieu, for example, suggests a way to use the Quantum Adiabatic Theorem to provide a computational solution to Hilbert's Tenth Problem, which is known to be Turing uncomputable [96]. Criticisms of Kieu's proposal include issues of representation and concerns that adiabatic evolution fails to actually reach the required solution [7, 135–137].

Deutsch's Arguments

Another reaction to the stance that natural random processes falsify the physical thesis is to assert that computability at its core is indeed Turing-machine computability. Having said that, to extend the Turing model's scope from classical functions to stochastic functions, one merely needs to augment it with a (physical or oracular) source of randomness. This, as we have seen, accords with Turing's own perspective when describing the "choice machine" variant of his eponymous machines: "When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator" [5]. In this context, the "external operator" would be a physical apparatus, rather than a human guide.

Deutsch reformulated the physical Church-Turing thesis as an explicit postulate about nature herself, what is now known as "Deutsch's Principle":

> Every finitely realizible physical system can be perfectly simulated by a universal model computing machine operating by finite means. [109]

Specifically, he posits that a *quantum* generalization of the Turing machine can simulate every *physical* process, in the sense that for every possible input state, represented in some fixed way, the distribution of outcomes of output experiments is the same (up to representation) as for the physical system:

A class of model computing machines that is the quantum generalization of the class of Turing machines is described, and it is shown that quantum theory and the "universal quantum computer" are compatible with the principle. [109]

Empowered this way, Deutsch's proposed quantum machine computes discrete functions faster than Turing's machine can, can compute stochastic outputs that no classical system can reproduce, and can compute continuous functions to arbitrarily high (albeit imperfect) accuracy [109].

It is, of course, an open (unanswerable) question whether there is at all any complete, immutable, finite description of the (observable) behaviour of the physical universe [138].

9.7 Conclusion

In summary, most computer scientists take the mathematical thesis for granted, based on the strong arguments put forth by Turing and those who followed him.

Moreover, there is a long list of highly regarded logicians who are of the opinion that there is room for formalization and perhaps proof of the mathematical thesis, including: Gödel, Gandy, Mendelson, Shoenfield, Shapiro, George Kreisel [139], Sieg, Kripke, Soare, Robert Black [140], Friedman, and Gurevich. Some have even proposed formal proofs or sketches of proofs. Other scholars, starting with Church, Turing, and Kleene and continuing to the present, maintain that there is an impassable dissonance between intuitive and formal notions of effectiveness.

The physical thesis, on the other hand, enjoys far less support, as also purported claims of hypercomputability.

Acknowledgements We are enormously grateful for comments and suggestions from Arnon Avron, Erwin Engeler, Oron Shagrir, Wilfried Sieg, and an anonymous reader.

References

1. Emil L. Post. Finite combinatory processes—formulation 1. *Journal of Symbolic Logic*, 1 (3): 103–105, September 1936.
2. Yiannis N. Moschovakis. What is an algorithm? In Björn Engquist and Wilfried Schmid, editors, *Mathematics Unlimited — 2001 and Beyond*, pages 919–936. Springer, Berlin, 2001.
3. Erwin Engeler. *The Combinatory Programme*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1995.
4. Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58: 345–363, 1936.
5. Alan. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42: 230–265, 1936–37. URL http://www.abelard.org/turpap2/tp2-ie.asp. Corrections in vol. 43 (1937), pp. 544–546. Reprinted in M. Davis (ed.), *The Undecidable*, Raven Press, Hewlett, NY, 1965.

6. Stephen C. Kleene. Lambda-definability and recursiveness. *Duke Mathematical Journal*, 2: 340–353, 1936.
7. Martin Davis. The Church-Turing thesis: Consensus and opposition. In *Logical Approaches to Computational Barriers*, pages 125–132, Berlin, 2006. Springer.
8. Andrew Hodges. Did Church and Turing have a thesis about machines? In Adam Olszewski, Jan Wolenski, and Robert Janusz, editors, *Church's Thesis After 70 Years*, pages 214–224. Ontos Verlag, 2006.
9. Robert I. Soare. The history and concept of computability. In *Handbook of Computability Theory*, pages 3–36. Elsevier, 1999.
10. Ian Parberry. Parallel speedup of sequential machines: A defense of parallel computation thesis. *SIGACT News*, 18 (1): 54–67, March 1986.
11. Peter van Emde Boas. Machine models and simulations. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, pages 1–66. North-Holland, Amsterdam, 1990. URL http://www.illc.uva.nl/Research/Reports/CT-1988-05.text.pdf.
12. Scott Aaronson. The toaster-enhanced Turing machine, 2012. URL https://www.scottaaronson.com/blog/?p=1121. The Blog of Scott Aaronson.
13. Udi Boker and Nachum Dershowitz. Comparing computational power. *Logic Journal of the IGPL*, 14 (5): 633–648, 2006. URL http://nachum.org/papers/ComparingComputationalPower.pdf.
14. Udi Boker and Nachum Dershowitz. The Church-Turing thesis over arbitrary domains. In Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, editors, *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 199–229. Springer, 2008. URL http://nachum.org/papers/ArbitraryDomains.pdf.
15. Martin Davis. The myth of hypercomputation. In Christof Teuscher, editor, *Alan Turing: Life and Legacy of a Great Thinker*, pages 195–212. Springer, 2003.
16. Gualtiero Piccinini. The physical Church-Turing thesis: Modest or bold? *The British Journal for the Philosophy of Science*, 62: 733–769, 2011.
17. Dina Goldin and Peter Wegner. The Church-Turing Thesis: Breaking the myth. In S. Barry Cooper, Benedikt Löwe, and Leen Torenvliet, editors, *New Computational Paradigms: First Conference on Computability in Europe (CiE 2005, Amsterdam)*, volume 3526 of *Lecture Notes in Computer Science*, pages 152–168, Berlin, June 2005.
18. Dina Goldin and Peter Wegner. The interactive nature of computing: Refuting the strong church-turing thesis. *Minds and Machines*, 18: 17–38, March 2008.
19. Manfred Broy. Computability and realizability for interactive computations. *Information and Computation*, 241: 277–301, 2015.
20. Dina Q. Goldin, Scott A. Smolka, and Peter Wegner, editors. *Interactive Computation: The New Paradigm*. Springer, Berlin, 2006.
21. Stephen C. Kleene. Turing's analysis of computability, and major applications of it. In *A Half-century Survey on The Universal Turing Machine*, pages 17–54. Oxford University Press, Inc., 1988.
22. Janet Folina. Church's thesis: Prelude to a proof. *Philosophia Mathematica*, 6 (3): 302–323, 1998.
23. Stewart Shapiro. Understanding Church's thesis, again. *Acta Analytica*, 11: 59–77, 1993.
24. Stephen C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.
25. László Kalmár. An argument against the plausibility of Church's thesis. In A. Heyting, editor, *Constructivity in Mathematics, Proceedings of the Colloquium Held at Amsterdam, 1957*, pages 72–80, Amsterdam, 1959. North-Holland.
26. Martin Davis. Why Gödel didn't have Church's Thesis. *Information and Control*, 54 (1/2): 3–24, 1982.
27. Joseph R. Shoenfield. *Recursion Theory*, volume 1 of *Lecture Notes In Logic*. Springer-Verlag, Heidelberg, New York, 1991.

28. Elliott Mendelson. Second thoughts about Church's thesis and mathematical proofs. *Journal of Philosophy*, 87 (5): 225–233, 1990.
29. Robin Gandy. The confluence of ideas in 1936. In *A Half-Century Survey on the Universal Turing Machine*, pages 55–111, New York, NY, 1988. Oxford University Press, Inc. URL http://dl.acm.org/citation.cfm?id=57249.57252.
30. Stewart Shapiro. Proving things about the informal. In G. Sommaruga and T. Strahm, editors, *Turing's Revolution: The Impact of his Ideas About Computability*, pages 283–296. Springer, Cham, January 2015.
31. Harvey M. Friedman. Mathematical logic in the 20th and 21st centuries, 2000. URL http://cs.nyu.edu/pipermail/fom/2000-April/003913.html. FOM mailing list. April 27, 2000.
32. Saul A. Kripke. The Church-Turing "thesis" as a special corollary of Gödel's completeness theorem. In J. Copeland, C. Posy, and O. Shagrir, editors, *Computability: Turing, Gödel, Church, and Beyond*, pages 77–104. MIT Press, 2013.
33. Udi Boker and Nachum Dershowitz. The influence of domain interpretations on computational models. *Journal of Applied Mathematics and Computation*, 215 (4): 1323–1339, 2009.
34. Henk Barendregt. The impact of the lambda calculus in logic and computer science. *Bulletin of Symbolic Logic*, 3 (2): 181–215, 1997. URL http://www.math.ucla.edu/~asl/bsl/0302/0302-003.ps.
35. Solomon Feferman. Theses for computation and recursion on concrete and abstract structures. In *Turing's Revolution: The Impact of His Ideas about Computability*, pages 105–126. Springer International Publishing, 2015.
36. David Chalmers. A computational foundation for the study of cognition. *Journal of Cognitive Science*, 12 (4): 325–359, 2011. Written in 1993.
37. Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1966.
38. Stephen C. Kleene. Origins of recursive function theory. *Annals of the History of Computing*, 3 (1): 52–67, 1981.
39. Wilfried Sieg. Step by recursive step: Church's analysis of effective calculability. *Bulletin of Symbolic Logic*, 3: 154–180, June 1997.
40. Alonzo Church. Review of "On computable numbers, with an application to the Entscheidungsproblem". *The Journal of Symbolic Logic*, 2 (1): 42–43, 1937.
41. Hao Wang. *A Logical Journey. From Gödel to Philosophy*. MIT Press, 1996.
42. Kurt Gödel. Some remarks on the undecidability results. In Solomon Feferman, John Dawson, and Stephen Kleene, editors, *Kurt Gödel: Collected Works, Vol. II*, pages 305–306. Oxford University Press, 1972.
43. Hao Wang. *Reflections on Kurt Gödel*. Bradford Books. MIT Press, 1990.
44. Hao Wang. *From Mathematics to Philosophy*. Kegan Paul, London,UK, 1974.
45. Oron Shagrir. Gödel on Turing on computability. In Adam Olszewski, Jan Wolenski, and Robert Janusz, editors, *Church's Thesis after 70 Years*, pages 393–419. Ontos-Verlag, 2006.
46. Arnon Avron. The problematic nature of Gödel's disjunctions and Lucas-Penrose's theses. *Semiotic Studies*, 34(1): 83–108, 2020.
47. Jon Barwise. An introduction to first-order logic. In Jon Barwise, editor, *Handbook of Mathematical Logic*, chapter A.1, pages 5–46. North-Holland, 1977.
48. Reinhard Kahle. Is there a "Hilbert Thesis?" *Studia Logica*, 107: 145–165, 2019.
49. Wilfried Sieg. Mechanical procedures and mathematical experience. In Alexander George, editor, *Mathematics and Mind*, pages 71–117. Oxford University Press, 1994.
50. Saul A. Kripke. The origins and nature of computation, 2006. URL https://www.youtube.com/watch?v=D9SP5wj882w. Presented at the 21st International Workshop on the History and Philosophy of Science. Jerusalem, Israel.
51. Robin Gandy. Church's thesis and principles for mechanisms. In J. Barwise, D. Kaplan, H. J. Keisler, P. Suppes, and A. S. Troelstra, editors, *The Kleene Symposium*, volume 101 of *Studies in Logic and The Foundations of Mathematics*, pages 123–148. North-Holland, 1980.
52. Wilfried Sieg. Church without dogma: Axioms for computability. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms: Changing Conceptions of What is Computable*, pages 139–152, New York, 2007. Springer.

53. Moto-o Takahashi. A foundation of finite mathematics. *Publications of the Research Institute for Mathematical Sciences*, 12: 577–708, 1977.
54. Yuri Gurevich. What is an algorithm? In *SOFSEM 2012: Theory and Practice of Computer Science*, pages 31–42, 2012.
55. Yuri Gurevich. Sequential abstract state machines capture sequential algorithms. *ACM Transactions on Computational Logic*, 1: 77–111, 2000.
56. Donald E. Knuth. Algorithm and program; information and data. *Communications of the ACM*, 9: 654, 1968.
57. Emil L. Post. Absolutely unsolvable problems and relatively undecidable propositions: Account of an anticipation. In M. Davis, editor, *Solvability, Provability, Definability: The Collected Works of Emil L. Post*, pages 375–441. Birkhaüser, Boston, MA, 1994. Unpublished notes, 1941.
58. Andreĭ N. Kolmogorov. O ponyatii algoritma [On the concept of algorithm] (in Russian). *Uspekhi Matematicheskikh Nauk [Russian Mathematical Surveys]*, 8 (4): 1175–1176, 1953. English version in: Vladimir A. Uspensky and Alexei L. Semenov, *Algorithms: Main Ideas and Applications*, Kluwer, Norwell, MA, 1993, pp. 18–19.
59. Udi Boker and Nachum Dershowitz. Abstract effective models. In M. Fernández and I. Mackie, editors, *New Developments in Computational Models: Proceedings of the First International Workshop on Developments in Computational Models (DCM 2005), Lisbon, Portugal (July 2005)*, volume 135 of *Electronic Notes in Theoretical Computer Science*, pages 15–23, 2006.
60. Nachum Dershowitz and Yuri Gurevich. A natural axiomatization of computability and proof of Church's Thesis. *Bulletin of Symbolic Logic*, 14 (3): 299–350, 2008. https://doi.org/10.2178/bsl/1231081370.
61. Wolfgang Reisig. The computable kernel of Abstract State Machines. *Theoretical Computer Science*, 409: 126–136, 2008.
62. Udi Boker and Nachum Dershowitz. Three paths to effectiveness. In Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig, editors, *Fields of Logic and Computation: Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *Lecture Notes in Computer Science*, pages 36–47, Berlin, 2010. Springer. URL http://nachum.org/papers/ThreePathsToEffectiveness.pdf.
63. Wilfried Sieg. Axioms for computability: Do they allow a proof of Church's thesis? In Hector Zenil, editor, *A Computable Universe. Understanding and Exploring Nature as Computation*, pages 99–123. World Scientific/Imperial College Press, Singapore, 2013.
64. Elliott Mendelson. *Introduction to Mathematical Logic*. Discrete Mathematics and Its Applications. CRC Press, 5th edition, 2009.
65. William J. Rapaport. *Philosophy of Computer Science*. Online draft, 2020. URL https://cse.buffalo.edu/~rapaport/Papers/phics.pdf.
66. John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley, Reading, MA, 1968.
67. Harry R. Lewis and Cristos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, NJ, 1981.
68. Máté Szabó. Kalmár's argument against the plausibility of Church's thesis. *History and Philosophy of Logic*, 39 (2): 140–157, 2018.
69. Benjamin Wells. Is there a nonrecursive decidable equational theory? *Minds and Machines*, 12 (2): 301–324, 2002.
70. Rózsa Péter. Rekursivität und konstruktivität. In A. Heyting, editor, *Constructivity in Mathematics, Proceedings of the Colloquium Held at Amsterdam, 1957*, pages 226–233, Amsterdam, 1959. North-Holland.
71. Jean Porte. Quelques pseudo-paradoxes de la 'calculabilite effective'. In *Actes du 2me Congrès International de Cybernétique*, pages 332–334, Namur, Belgium, 1960.
72. Elliott Mendelson. On some recent criticism of Church's thesis. *Notre Dame Journal of Formal Logic*, IV (3): 201–205, July 1963.
73. Yiannis N. Moschovakis. Review of four recent papers on Church's thesis. *Journal of Symbolic Logic*, 33 (3): 471–472, 1968.

74. Stewart Shapiro. Acceptable notation. *Notre Dame Journal of Formal Logic*, 23 (1): 14–20, 1982.
75. Michael Rescorla. Copeland and Proudfoot on computability. *Studies in History and Philosophy of Science Part A*, 43 (1): 199–202, 2012. Reconsidering the Dynamics of Reason: A Symposium in Honour of Michael Friedman.
76. B. Jack Copeland and Diane Proudfoot. Deviant encodings and Turing's analysis of computability. *Studies in History and Philosophy of Science*, 41: 247–252, September 2010.
77. Michael Rescorla. Church's thesis and the conceptual analysis of computability. *Notre Dame Journal of Formal Logic*, 48 (2): 253–280, 2007.
78. Michał Wroclawski. Representations of natural numbers and computability of various functions. In Florin Manea, Barnaby Martin, Daniël Paulusma, and Giuseppe Primiero, editors, *Proceedings of the 15th Conference on Computability in Europe - Computing with Foresight and Industry (CiE 2019, Durham, UK)*, volume 11558 of *Lecture Notes in Computer Science*, pages 298–309. Springer, July 2019.
79. Udi Boker and Nachum Dershowitz. A hypercomputational alien. *Applied Mathematics and Computation*, 178 (1): 44–57, 2006.
80. Roger Penrose. *The Emperor's New Mind: Concerning Computers, Minds, and The Laws of Physics*. Oxford University Press, New York, 1989.
81. Roger Penrose. *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford University Press, Oxford, 1994.
82. Christopher Strachey. An impossible program. *Computer Journal*, 7 (4): 313, 1965.
83. John R. Lucas. Minds, machines and Gödel. *Philosophy*, XXXVI: 112–127, 1961. Reprinted in *The Modeling of Mind*, K. M. Sayre and F. J. Crosson, eds., Notre Dame Press, 1963, pp. 269–270. https://doi.org/10.1017/S0031819100057983
84. Arnon Avron. *Mishpete Gedel u-ve'ayat ha-yesodot shel ha-matematikah (= Gödel's Theorems and the Problem of the Foundations of Mathematics)*. Broadcast University, Ministry of Defence, Jerusalem, Israel, 1998. In Hebrew.
85. David Chalmers, editor. *Symposium on Roger Penrose's Shadows of the Mind*, volume 2, 1995. Association for the Scientific Study of Consciousness. URL http://journalpsyche.org/files/0xaa25.pdf.
86. Geoffrey LaForte, Patrick J. Hayes, and Kenneth M. Ford. Why Gödel's theorem cannot refute computationalism. *Artificial Intelligence*, 104 (1–2): 265–286, 1998.
87. Nachum Dershowitz. The four sons of Penrose. In *Proceedings 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR) (Montego Bay, Jamaica)*, volume 3835 of *Lecture Notes in Computer Science*, pages 125–138. Springer, December 2005.
88. Martin Davis. *The Universal Computer: The Road from Leibniz to Turing*. CRC Press, 3rd edition, 2018.
89. Hilary Putnam. Book review: *Shadows of the Mind* by Roger Penrose. *Bulletin of the American Mathematical Society*, 32 (3): 370–373, July 1995.
90. John. C. Shepherdson. On the definition of computable function of a real variable. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik (Mathematical Logic Quarterly)*, 22 (1): 391–402, 1976.
91. Oron Shagrir. Effective computation by humans and machines. *Minds and Machines*, 12: 221–240, 2002.
92. Wilfried Sieg and John Byrnes. An abstract model for parallel computations: Gandy's thesis. *The Monist*, 82 (1): 150–164, 1999.
93. Olivier Bournez, Nachum Dershowitz, and Pierre Néron. An axiomatization of analog algorithms. In *Computability in Europe 2016: Pursuit of the Universal (CiE, Paris, France)*, volume 9709 of *Lecture Notes in Computer Science*, pages 215–224, Switzerland, June 2016. Springer. URL http://nachum.org/papers/AxiomatizationAnalog.pdf. Full version at https://arxiv.org/pdf/1604.04295v2.pdf.
94. Jack B. Copeland and Oron Shagrir. Physical computation: How general are Gandy's principles for mechanisms? *Minds and Machines*, 17 (2): 217–231, 2007.

95. Pablo Arrighi and Gilles Dowek. The physical Church-Turing thesis and the principles of quantum theory. *International Journal of Foundations of Computer Science*, 23 (5): 1131–1145, 2012.
96. Tien D. Kieu. Quantum algorithm for Hilbert's Tenth Problem. *International Journal of Theoretical Physics*, 42: 1461–1478, 2003.
97. Pablo Arrighi and Gilles Dowek. The principle of a finite density of information. In H. Zenil, editor, *Irreducibility and Computational Equivalence*, volume 2 of *Emergence, Complexity and Computation*, pages 127–134. Springer, Berlin, 2013.
98. David Beckman, Daniel Gottesman, Michael A. Nielsen, and John Preskill. Causal and localizable quantum operations. *Phys. Rev. A*, 64, 2001.
99. Benjamin Schumacher and Michael D. Westmoreland. Locality and information transfer in quantum operations. *Quantum Information Processing*, 4 (1): 13–34, 2005.
100. Apostolos Syropoulos. *Hypercomputation: Computing Beyond the Church-Turing Barrier*. Springer Science & Business Media, 2008.
101. Selmer Bringsjord and David A. Ferrucci. The narrative-based refutation of Church's thesis. In *Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a Storytelling Machine*, chapter 5, pages 105–148. Lawrence Erlbaum, Mahwah, NJ, 2000.
102. Selmer Bringsjord, Owen Kellett, Andrew Shilliday, Joshua Taylor, Bram van Heuveln, Yingrui Yang, Jeffrey Baumes, and Kyle Ross. A new Gödelian argument for hypercomputing minds based on the Busy Beaver problem. *Applied Mathematics and Computation*, 176 (2): 516–530, 2006.
103. Owen Kellett. A multi-faceted attack on the Busy Beaver problem. Master's thesis, Rensselaer Polytechnic Institute, Troy, New York, July 2005.
104. Selmer Bringsjord and Michale Zenzen. *Superminds: People Harness Hypercomputation, and More*, volume 29 of *Studies in Cognitive Systems*. Kluwer Academic, Dordrecht, The Netherlands, 2003.
105. Warren Page. An interview with Herbert Robbins. *The Two-Year College Mathematics Journal*, 15 (1): 2–24, 1984.
106. Mario Stipčević and Çetin Kaya Koç. True random number generators. In Çetin Kaya Koç, editor, *Open Problems in Mathematics and Computational Science*, pages 275–315. Springer International Publishing, 2014.
107. G. Lee Bowie. An argument against Church's thesis. *The Journal of Philosophy*, 70 (3): 66–76, 1973.
108. Cristian S. Calude. Algorithmic randomness, quantum physics, and incompleteness. In *Proceedings of the Conference on Machines, Computations and Universality (MCU 2004)*, volume 3354, pages 1–17, september 2004.
109. David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 400: 97–117, 1985.
110. John Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6 (4): 675–695, 1977.
111. Karl de Leeuw, Edward F. Moore, Claude E. Shannon, and Norman Shapiro. Computability by probabilistic machines. In Claude E. Shannon and John McCarthy, editors, *Automata Studies*, volume 34 of *Annals of Mathematics Studies*, pages 183–212. Princeton University Press, 1956.
112. Yuri Gurevich. Unconstrained Church-Turing thesis cannot possibly be true. *Bull. EATCS*, 127, 2019. URL http://bulletin.eatcs.org/index.php/beatcs/article/view/566/565.
113. B. Jack Copeland. Accelerating Turing machines. *Minds and Machines*, 12 (2): 281–300, 2002.
114. E. Mark Gold. Limiting recursion. *J. Symbolic Logic*, 30 (1): 28–48, 1965.
115. Hilary Putnam. Trial and error predicates and the solution to a problem of Mostowski. *J. Symbolic Logic*, 30 (1): 49–57, 1965.
116. Hermann Weyl. *Philosophy of Mathematics and Natural Science*. Princeton University Press, 1949.

117. Adolf Grünbaum. Messrs. Black and Taylor on temporal paradoxes. *Analysis*, 12 (6): 144–148, 1952. URL http://www.jstor.org/stable/3326977.
118. Gábor Etesi and István Németi. Non-Turing computations via Malament-Hogarth space-times. *International Journal of Theoretical Physics*, 41 (2): 341–370, 2002.
119. Mark L. Hogarth. Non-Turing computers and non-Turing computability. In *Proceedings of the Philosophy of Science Association*, volume 1994, pages 126–138, 1994.
120. Mark L. Hogarth. Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, 5 (2): 173–181, 1992.
121. Itamar Pitowsky. The physical Church thesis and physical computational complexity. *Iyyun: The Jerusalem Philosophical Quarterly*, 39: 81–99, 1990.
122. Oron Shagrir and Itamar Pitowsky. Physical hypercomputation and the Church-Turing thesis. *Minds and Machines*, 13 (1): 87–101, 2003.
123. John Earman and John D. Norton. Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science*, 60 (1): 22–42, 1993.
124. Antony Galton. The Church-Turing thesis: Still valid after all these years? *Applied Mathematics and Computation*, 178: 93–102, 2006.
125. Paolo Cotogno. Hypercomputation and the physical Church-Turing thesis. *The British Journal for the Philosophy of Science*, 54 (2): 181–223, 2003.
126. Fred G. Abramson. Effective computation over the real numbers. In *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*, pages 33–37, 1971.
127. Jean-Claude. Carréga. *Théorie des corps - La règle et le compas*. Hermann, Paris, 1981.
128. Pascal Schreck. On the mechanization of straightedge and compass constructions. *Journal of Systems Science and Complexity*, 32: 124–149, February 2019.
129. Jesper Lützen. Why was Wantzel overlooked for a century? The changing importance of an impossibility result. *Historia Mathematica*, 36 (4): 374–394, 2009.
130. Arnon Avron. Personal communication, 2020.
131. Hava T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, Boston, 1998.
132. Paul Cockshotta, Lewis Mackenzie, and Greg Michaelson. Physical constraints on hypercomputation. *Theoretical Computer Science*, 394: 159–174, 2008.
133. Cristian S. Calude and Boris Pavlov. Coins, quantum measurements, and Turing's barrier. *Quantum Information Processing*, 1 (1): 107–127, 2002.
134. Michael A. Nielsen. Computable functions, quantum measurements, and quantum dynamics. *Physical Review Letters*, 79 (15): 2915–2918, 1997.
135. Andrew Hodges. Can quantum computing solve classically unsolvable problems? arXiv, 2005. URL http://arXiv.org/abs/quant-ph/0512248.
136. Warren D. Smith. Three counterexamples refuting Kieu's plan for "quantum adiabatic hypercomputation"; and some uncomputable quantum mechanical tasks. *Applied Mathematics and Computation*, 178 (1): 184–193, 2006. Special Issue on Hypercomputation.
137. Boris Tsirelson. The quantum algorithm of Kieu does not solve the Hilbert's tenth problem. arXiv, November 2001. URL http://arXiv.org/abs/quant-ph/0111009.
138. John W. Carroll. Laws of nature. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2016 edition, 2016. URL https://plato.stanford.edu/archives/fall2016/entries/laws-of-nature.
139. George Kreisel. Mathematical logic: What has it done for the philosophy of mathematics? In Ralph Schoenman, editor, *Bertrand Russell: Philosopher of the Century*. George Allen and Unwin, London, 1967.
140. Robert Black. Proving Church's Thesis. *Philosophia Mathematica*, 8 (3): 244–258, October 2000.

Udi Boker is Professor of Computer Science at the Reichman University Herzliya, Israel. He specializes in logic, formal verification, computational models, game theory, and automata theory. He received his Ph.D. in computer science from the Tel Aviv University, and did postdoctoral research in the Hebrew University and in the Institute of Science and Technology (IST) Austria. Prior to joining the academy, he served as an R&D Director in the high-tech company Mercury Interactive, initiating and leading the area of load-testing over the Internet.

Nachum Dershowitz is Professor Emeritus of Computer Science and incumbent of the Chair in Computational Logic at Tel Aviv University. His graduate degrees in applied mathematics are from the Weizmann Institute in Israel. He is an international authority on program verification and equational reasoning and has made major contributions to the computer analysis of historical manuscripts. He has authored or coauthored 200 research papers and several books, held visiting positions at prominent institutions around the globe, was elected to Academia Europaea, and has won numerous awards for research and teaching, including the Herbrand Award for Distinguished Contributions to Automated Reasoning, several "test of time" awards for past research, and the Choice Outstanding Academic Title Award for his book, *Calendrical Calculations*, with Ed Reingold, now in its fourth edition (Cambridge University Press, 2018).