Universality, Invariance, and the Foundations of Computational Complexity in the light of the Quantum Computer[*,†]

Michael E. Cuffaro[a,b]

[a]Rotman Institute of Philosophy, University of Western Ontario

[b]Munich Center for Mathematical Philosophy, Ludwig-Maximilians-Universität München

1 Introduction

Computational complexity theory is a branch of computer science that is dedicated to classifying computational problems in terms of their difficulty. Unlike computability theory, whose object is to determine what we can compute in principle, the object of complexity theory[1] is to inform us with regards to our practical limits. It thus serves as a natural conceptual bridge between the study of

[*]**Note:** this is the submitted version of a chapter published in *Technology and Mathematics: Philosophical and Historical Investigations* (Springer-Verlag), Sven Ove Hansson (ed.). There may be changes made in the published version which are not reflected in this preprint. When citing this paper, **please refer to the published version**, which can be accessed via: doi.org/10.1007/978-3-319-93779-3_11.

[†]This project was supported financially by the Foundational Questions Institute (FQXi) and by the Rotman Institute of Philosophy. I am grateful to Scott Aaronson, Walter Dean, William Demopoulos, Armond Duwell, Laura Felline, Sona Ghosh, Amit Hagar, Gregory Lavers, and Markus Müller for their comments on a previous draft of this chapter. It has improved substantially as a result.

[1]There are a number of sciences (for example: complex systems theory, the study of Kolmogorov complexity, and so on) which are referred to as complexity theories. Unless otherwise noted, any occurrence of 'complexity theory' in what follows should be understood as referring in particular to *computational* complexity theory, and any conclusions made should be taken as pertaining only to it.

mathematics and the study of technology, in the sense that computational complexity theory informs us with respect to which computational procedures may reasonably be expected to be technologically feasible.

Quantum computer science is the study of algorithms and other aspects of computer systems whose construction involves an explicit appeal to various features of quantum physical theory. Strikingly, there are quantum algorithms that appear to significantly outperform algorithms which do not take advantage of quantum resources. What distinguishes, quantitatively, quantum from classical computation is not the number of problems that can be solved using one or the other model. Rather, what distinguishes the quantum from the classical model of computation is that the number of problems solvable *efficiently*—i.e. the number of problems whose solution is practically realisable—in the former model appears to be larger than the number of problems solvable efficiently in the latter. The study of quantum computer science therefore advances the goal of complexity theory in the sense that it adds to our knowledge of the class of practically realisable computational procedures.

More generally, as I will argue below, the study of quantum computation illuminates the very nature and subject matter of complexity theory. Yet it does not do so in a way that is often claimed. In particular it is not uncommon to come across statements in the philosophical and scientific literature to the effect that advances in quantum computing force a fundamental revision of the foundations of complexity theory (Hagar, 2007; Nielsen & Chuang, 2000; Bernstein & Vazirani, 1997). According to this view it is the traditional aim of complexity theory to understand the nature of concepts such as that of a 'tractable problem' *in themselves*; i.e., apart from the manner in which they are implemented under particular models of computation. Model-independence, in turn, is taken to rest upon an 'extended' or 'strong' version of the Church-Turing thesis, or alternately, upon an 'invariance' thesis. And because quantum computers seemingly violate these theses, it is concluded that complexity theory's foundations must be somehow rebuilt.

As I will argue, however, model-independence is not and never has been at the core of computational complexity theory. Its foundations are therefore not shaken by the advent of quantum computing. Complexity theory is fundamentally a practical science, whose aim is to guide us in making distinctions in practice among tractable and intractable problem sets. The model-independence of complexity-theoretic concepts is not a necessary condition for realising this aim. Quantum computation indeed illuminates the subject matter of complexity theory. But it does not do so by overturning its foundations. Rather, quantum computing illuminates complexity theory by reminding us of its practical nature.

This is both a virtue of the theory as well as a reason for increased philosophical attention to it.

Science does not always or only, or perhaps ever, progress through the absolute identification of fundamental entities, be they abstract or concrete. Complexity theory furnishes us with a particularly striking illustration that scientific progress—even in the mathematical sciences—is, in fact, often built upon pragmatically justified foundations and conceptual structures.[2] There is a general philosophical lesson in this, which in different contexts has been profitably analysed by some (for example, Carnap 1980 [1950]; 1962 [1950], ch. 1), though in my view too few, philosophers.

In the next section we will briefly review, from a historical perspective, the foundations of computability theory. §3 will then connect the foregoing discussion to the foundations of computational complexity theory, and will introduce the theory's basic concepts. In §4 we will discuss the 'universality of Turing efficiency' thesis, as well as the closely related 'invariance thesis'. §5 will introduce the basic concepts of quantum computing. In §6 we will discuss quantum computing's significance for the conceptual foundations of complexity theory. We will then conclude.

2 The *Entscheidungsproblem* and the origins of the Church-Turing thesis

With his second incompleteness theorem, Gödel demonstrated that any ω-consistent formalisation of number theory, whose formulas are primitively recursively definable, and which is rich enough to permit arithmetisation of syntax, cannot prove its own consistency.[3] For such a capability would be incompatible with Gödel's first incompleteness theorem, by which he demonstrated that within any such formalisation there are sentences neither provable nor refutable from the axioms. Finding a general and effective procedure for determining whether a given formula in such a system is one of these sentences, however, remained an open question.

This was the Decision Problem—in German: the *Entscheidungsproblem*—for validity, originally posed for first-order logic by (Hilbert & Ackermann, 1928, Pt. III); that is, to describe an 'effective procedure' by which one can decide whether an arbitrarily given expression of first-order logic is

[2]See also Dean (2016a), who reviews the arguably insurmountable problems that face any attempt to regard an algorithm as a mathematical object in the light of computer science practice.

[3]An ω-consistent theory is such that it is both consistent and satisfies a syntactic analogue of soundness (see Dawson Jr., 2007, p. 504). A primitively recursively definable formula is such that it can be built up from a finite number of successive basic operations. Arithmetisation of syntax refers to a procedure by which every sentence in a formal system is encoded uniquely into a natural number (called its 'Gödel number').

provable from the axioms.[4] Informally, an effective computational procedure consists of a finite number of precise finite-length instructions guaranteed to produce some desired result in a finite number of steps if followed exactly by a human being using nothing other than paper and pencil. An example of an effective procedure is the truth-table method as applied to sentential logic. Famously, Church and Turing were independently able to show that the *Entscheidungsproblem* for first-order logic could not be solved; i.e., no effective calculational procedure for determining the validity of an arbitrarily given expression in first-order logic exists.

Turing, to whom we will restrict our attention, showed this partly by means of a penetrating philosophical analysis of the notion of effective computation.[5] Turing (1936-7, pp. 249–51) argued that it is essential to the idea of carrying out a computation that the computer uses a notebook from which she reads, and onto which she writes, various symbols related to her work. These symbols, as they must be distinguishable from one another, are chosen from a finite alphabet. At any given moment during a computation, the computer will find herself in one of a finite number of relevant states of mind which summarise her memory of the actions she has performed up until that point along with her awareness of what she must now do (see pp. 253–4). The actions that are available to her are characterised by a finite set of elementary operations, such as 'read the next symbol' from the notebook, 'write symbol a' to the notebook, and so on. Turing then argued that one could design an automatic machine, which he called an α-machine, to instantiate each of these essential features of the practice of human computation (see Figure 1). In doing so he identified the concept 'effectively calculable' with the concept of 'computable by α-machine'. This identification is known as Turing's thesis, which he proved (p. 263ff) to be equivalent with Church's independently arrived at thesis that the class of effectively calculable functions is identical with the class of λ-definable functions (Church, 1936). For this reason it is also called the Church-Turing thesis.

[4]The reason for Hilbert & Ackermann's focus on the special case of first order logic is that it is the most restricted example of a general logic adequate for representing higher arithmetic, i.e. number theory. Its study was to constitute the first step in the development of a more encompassing logical framework for mathematics (Dawson Jr., 2007, p. 500). Note that Hilbert and Ackermann additionally posed a parallel Decision Problem for satisfiability. In the sequel, unless otherwise indicated, I will take the Decision Problem or *Entscheidungsproblem* to refer exclusively to the problem for validity.

[5]In what follows it must be kept in mind that computation, at the time of the publication of "On Computable Numbers," generally referred to an activity performed by human beings; a computer was a person employed to carry out computations.

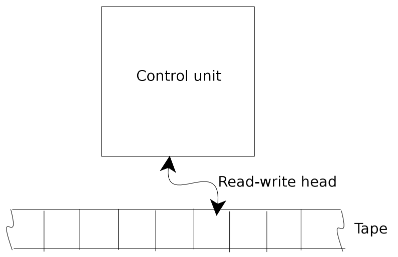


Figure 1: A version of what is now called a 'Turing machine'. The control unit houses the machine's 'state of mind', which in general changes after every operation of the read-write head. The read-write head reads, writes, and moves back and forth along portions of a one-dimensional tape (the machine's 'notebook'). Such a machine is an idealised representation of the components involved in human computation.

Turing then addressed the *Entscheidungsproblem* in an indirect way (Turing 1936-7, pp. 259–63, Turing 1938). He first showed that it is impossible to determine, for a given α-machine, whether it is 'circle-free'; i.e. whether it is not the case that it never outputs more than a finite number of symbols. He then showed that if the *Entscheidungsproblem* were solvable, one could determine, for any given α-machine, whether it is circle-free. Since this contradicts the first result, the *Entscheidungsproblem* is unsolvable.

3 Efficient computation

The period just discussed, during which the seminal papers by Church, Gödel, Turing, and others were published, is the period of the birth of computer science in the modern sense. It was to be nearly three more decades before the particular branch of modern computer science that furnishes the subject matter for this chapter, computational complexity theory, took shape with the work of Cobham (1965), Edmonds (1965), Hartmanis & Stearns (1965), and others. Yet one of its key questions was anticipated significantly earlier by none other than Gödel. Revisiting the *Entscheidungsproblem* in a letter he wrote to von Neumann in 1956, Gödel asked for von Neumann's opinion concerning the number, $\varphi(l)$, of steps needed, in the worst case, to decide whether some arbitrarily given formula of first-order logic has a proof of length l. In his letter Gödel asks:[6] "how fast does $\varphi(l)$ grow for an optimal [Turing] machine?" He notes that "One can show that $\varphi(l) \geq Kl$" (for some constant K), and then asserts:

[6]In the following quotations I have replaced the variable n with l.

> If there actually were a machine with $\varphi(l) \sim Kl$ (or even only with $\sim Kl^2$), this would have consequences of the greatest magnitude. That is to say, it would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely [Gödel's Footnote: Apart from the postulation of axioms] replaced by machines. One would indeed have to simply select an l so large that, if the machine yields no result, there would then also be no reason to think further about the problem (Gödel, 1956, p. 10).

To illustrate: take some proposition F of first-order logic and consider testing to see whether F has a proof, Ψ, of length l. Let l be a number of steps far too large for any unaided human being to survey in a lifetime, but small enough that a machine could survey them all relatively quickly. Gödel's point is that, from the machine's perspective, the Kl (or perhaps Kl^2) steps needed to discover whether Ψ exists is not very much greater than the l steps that would be needed to survey it. We would expect, therefore, that the machine will give us an answer to the question of whether F has a proof of length l in a reasonable amount of time. By assumption, however, surveying a proof of length $\geq l$ is beyond the practical capabilities of any human being. So if the machine yields a negative result, then we can conclusively say that, for the practical purposes of unaided human computation, F is unprovable. Indeed there would be no reason to bother with the practical computational purposes of unaided human mathematicians at all; if such a machine existed we could henceforth consider such questions exclusively with respect to it.

There is an additional, deeper, point that is implicit here as well. Gödel's question to von Neumann is stated in the context of the *Entscheidungsproblem*, where it is assumed that the procedure to be used by a human mathematician to answer the question of whether F can be proved is an effective one, in the sense described in the previous section. Recall that following an effective procedure requires no ingenuity on the part of the person doing the following; it is a purely mechanical procedure which, if followed exactly, is guaranteed to give one a result in a finite number of steps. It is precisely for this reason that we can model it with a machine. In general, however, theorem proving is an activity which we do take to require insight and ingenuity. We take there to be more to the process of discovering a proof of a particular theorem than blindly following a set of rules; we need insight into the 'essential nature' of the problem at hand in order to guide us to the most likely route to a solution, and we need ingenuity to proceed along this route in a skillful, efficient, way. Or so one could object. Be that as it may, if we could in fact build a machine to discover, in only Kl (or Kl^2) steps, whether any given proposition of first-order logic has a proof of length l, it would make, not just human beings themselves, but the ingenuity and insight associated with their activities in this context, dispensable.

Implicit in the above considerations is the idea that neither $\varphi(l) \sim Kl$ nor $\varphi(l) \sim Kl^2$ yields a significantly greater number than l from the point of view of a machine. This is consistent with the ideas of modern complexity theory, where in fact any decision problem (i.e., yes-or-no question) for which a solution exists whose worst-case running time is bounded by as much as a polynomial function of its input size, n, is considered to be a 'tractable' (a.k.a. 'feasible', 'efficiently solvable', 'easy', etc.[7]) problem. Indeed, these ideas are not just consistent; one way to motivate the modern complexity-theoretic identification is to begin with essentially Gödel's assertion that problems which require only Kn or Kn^2 steps to solve are tractable.[8] Combine this with the computer programmer's intuition that an efficient program, to which one adds a call to an efficient subroutine, should continue to be thought of as efficient (Arora & Barak, 2009, p. 27), and we naturally arrive at the conclusion that the set of efficiently solvable problems just is the set of problems solvable in a polynomial number of time steps. This 'polynomial principle' is generally considered to be at the heart of the theory of computational complexity. We will discuss it in more detail (and critically) in §6.

In the context of the Turing machine (TM) model, the set of decision problems solvable in polynomial time is referred to as the class P.[9] More formally, we can conceive of a decision problem as one whose goal is to yield a yes-or-no answer to the question of whether a given string x of length n is a member of the 'language' L. For example, the decision problem for determining whether a given number is prime can be represented as the problem to determine, for an arbitrarily given binary string, whether it is a member of the language {10, 11, 101, 111, 1011, 1101, 10001, 10011, ...} (the set of binary representations of prime numbers). Now call a given language L a member of the class DTIME($T(n)$) iff there is a Turing machine[10] for deciding membership in L whose running time, $t(n)$, is 'on the order of $T(n)$', or in symbols: $O(T(n))$. Here, $T(n)$ represents an upper bound for the growth rate of $t(n)$ in the sense that, by definition, $t(n)$ is

[7]I will be using these terms interchangeably below.

[8]Note that although Gödel's letter to von Neumann anticipates this and other ideas of modern complexity theory, I am not claiming that it actually influenced the theory's development. As far as I am aware, Gödel's letter was unknown prior to its translation and publication in Sipser (1992).

[9]It is also sometimes referred to as PTIME, in order to emphasise the distinction between it and PSPACE, the class of problems solvable using space resources bounded by a polynomial function of n.

[10]The 'D' in DTIME stands for 'deterministic'. It contrasts with 'nondeterministic time', which I will introduce later.

$O(T(n))$ if for every sufficiently large n, $t(n) \leq k \cdot T(n)$ for some constant k.[11] So for any language L in, for example, DTIME(n^2), there is a TM that will take no more than kn^2 steps to decide membership in L. We can now formally characterise P as (Arora & Barak, 2009, p. 25):

$$\mathsf{P} = \bigcup_{k \geq 1} \mathsf{DTIME}(n^k). \tag{1}$$

Note that the class DTIME($T(n)$) is defined, strictly speaking, to be a set of languages. Below I will sometimes use statements of the form: '(decision) problem R is in DTIME($T(n)$)', which is shorthand for the assertion that the language L_R, associated with R, is decidable in $O(T(n))$ steps.

We have just seen that L is in P iff one can construct a polynomial-time TM that will decide, for any given x, whether $x \in L$. Now suppose that one is presented with a proof that $x \in L$. If one can *verify* this proof using a polynomial-time TM, then we say that L is a member of the complexity class NP.[12] More formally (Arora & Barak, 2009, p. 39),

$$L \in NP \quad \text{whenever:} \quad x \in L \Leftrightarrow \exists u \quad \text{s.t.} \quad M(x,u) \overset{poly}{=} \text{'yes'}, \tag{2}$$

where u is string (usually called a 'certificate') whose length is given by a polynomial function of the length, n, of x, and $M(x,u) \overset{poly}{=}$ 'yes' asserts that the machine M accepts x, given u, in polynomial time.[13]

The restricted form of the *Entscheidungsproblem* described above by Gödel is certainly in NP; given a proposition x, and a proof u of x whose length is $\leq l$, one can obviously verify this in polynomial time. Indeed, the problem also happens to be 'NP-complete' (Hartmanis, 1993).[14] NP-complete problems are the hardest problems in NP, in the sense that if we have in hand a solution to an NP-complete problem, we can easily convert it into a solution to any other problem in NP. That is, a language $L \in$ NP is in the class NP-complete iff a procedure for deciding L can be converted, in polynomial time, into a procedure for deciding L', for any $L' \in$ NP. More concisely, $L \in$ NP is NP-complete iff $\forall L' \in NP$, L' is polynomial-time reducible, in the above

[11]The qualification 'for every sufficiently large n' can be rephrased as the assertion that there exists some finite $n_0 \geq 1$ such that $t(n) \leq k \cdot T(n)$ whenever $n \geq n_0$.

[12]NP stands for nondeterministic polynomial time. The reason for this name will become clear shortly.

[13]u must be of polynomial length in n to ensure that M can read u in polynomial time.

[14]Gödel himself gives no indication that he realises this in his letter.

sense,[15] to L (Arora & Barak, 2009, p. 42).

The proposition that there exists a general solution to the restricted *Entscheidungsproblem* which requires no more than Kl^2 steps to carry out—call this the 'Gödelian conjecture'[16]—does not amount merely to the proposition that this decision problem is in NP. Recall that the restricted *Entscheidungsproblem* is the problem to decide whether an arbitrarily given formula x has a proof of length l; it is not merely the problem of verifying this fact about x given a certificate u. The Gödelian conjecture, therefore, amounts to the claim that the restricted *Entscheidungsproblem* is in P. But since this problem is known to be NP-complete, the Gödelian conjecture, if correct, amounts to the claim that P = NP.[17]

Interestingly, there has been no proof or disproof to date of the statement that P = NP. Partly due to the intuitive implausibility of its consequences—that "the mental effort of the mathematician in the case of yes-or-no questions could be completely replaced by machines" (Gödel, 1956)—the statement is generally believed to be false. Besides this there are further, mathematical, reasons to believe that P $\neq$ NP (Aaronson, 2013a, p. 67). I will not mention these here as the P = NP question is not our focus. I will only say that the project to prove or disprove P = NP is a worthwhile one, not so much because the outcome is in doubt, but because a formal proof would likely enlighten us with regards to just what it is that insight and ingenuity contribute to the practice of mathematics.

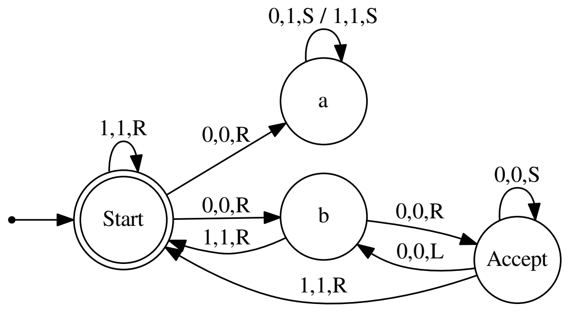
Our discussion of the Turing machine model of computation has thus far focused on the standard, i.e., deterministic, case. A standard TM is such that its behaviour at any given moment in time is wholly determined by the state that it finds itself in plus whatever input it receives. The machine can be fully characterised, that is, by a unique transition function over the domain of states and input symbols. One can, however, generalise the TM model by allowing the machine to instantiate more than one transition function simultaneously.[18] Upon being presented with a given

[15]What I have described above is actually called a Karp reduction. It is a weaker concept than the related one of Cook reduction. We will not discuss the distinction here. For more on this, see Aaronson (2013a, p. 58).

[16]Gödel does not himself actually conjecture this, although he comes close to doing so: "it seems to me ... to be totally within the realm of possibility that $\varphi(l)$ grows slowly." (Gödel, 1956, p. 10).

[17]Strictly speaking it only entails that NP $\subseteq$ P. But since obviously P $\subseteq$ NP, it would follow that P = NP.

[18]The idea of a machine with an ambiguous transition function can be found in Turing (1936-7). Turing calls this a 'choice machine' (p. 232), and notes its extensional equivalence with



$$\delta_1(Start, 0) = (a, 0, R) \qquad \delta_2(Start, 0) = (b, 0, R)$$
$$\delta_1(Start, 1) = (Start, 1, R) \qquad \delta_2(Start, 1) = (Start, 1, R)$$

etc. etc.

Figure 2: A nondeterministic Turing machine (NTM) is such that, for a given state and a given input, the state transitioned to is not predetermined; at any given step the machine is able select from more than one transition function (in this case, δ_1 and δ_2). The machine depicted accepts binary strings ending in '00', since there exists a series of transitions for which, given such a string, the machine will end in the 'Accept' state. But it is not guaranteed to do so. The machine additionally is guaranteed to reject any string not ending in '00'. In the diagram, an edge from s_1 to s_2 with the label α, β, P is read as: In state s_1, the machine reads α from its tape, writes β to the tape in the same position, moves its read/write head along the tape to the position P with respect to the current tape position (L = to the left, R = to the right, S = same), and finally transitions to state s_2.

input in a given state, a *nondeterministic Turing machine* (NTM) is allowed to 'choose' which of its transition functions to follow (see Figure 2). Exactly how this choice is made is left undefined, and for the purposes of the model can be thought of as arbitrary. We say that an NTM accepts a string x iff there exists a path through its state space that, given x, leads to an accepting state. It rejects x otherwise. We define the class NTIME($T(n)$), analogously to DTIME($T(n)$), as the set of languages for which there exists an NTM that will decide, in $O(T(n))$ steps, whether a given string x of length n is in the language L.

Recall that above I characterised NP as the set of languages for which one can construct a polynomial-time TM to verify, for any x, that $x \in L$, given a polynomial-length certificate u for x.

the automatic (i.e. deterministic) machine (p. 252, footnote ‡).

One can alternatively characterise NP as the set of languages for which there exists a polynomial-time NTM for determining membership in L:

$$\mathsf{NP} =_{df} \bigcup_{k \geq 1} \mathsf{NTIME}(n^k). \tag{3}$$

This definition is the source of the name NP, in fact, which stands for 'nondeterministic polynomial time'.

Defs. (2) and (3) are equivalent. Given a language L and a polynomial-time NTM that decides it, then for any $x \in L$, there is by definition a polynomial-length sequence of transitions of the NTM which will accept x. One can use this sequence as a certificate for x, and verify it in polynomial-time using a (deterministic) TM. Conversely, suppose there is a TM M_D that, given a polynomial-length certificate u for x, can verify in polynomial time that $x \in L$. Then one can construct a polynomial-time NTM M_N that will 'choose' certificates from among the set of possible polynomial-length strings (e.g., by randomly writing one down). Upon choosing a certificate u, M_N then calls M_D to verify x given u, and transitions to 'yes' only if M_D outputs 'yes' (Arora & Barak, 2009, p. 42).

For an NTM, no attempt is made to define how such a computer chooses, at any given moment, whether to follow one transition function rather than another. In particular, it is not assumed that any probabilities are attached to the machine's choices. Indeed, under Turing's original conception (1936-7, p. 232), these are thought of as the choices of an external operator. They are thus arbitrary from the machine's point of view. In a *probabilistic Turing machine* (PTM), on the other hand, we characterise the computer's choices by associating a particular probability with each of its transitions (see Figure 3).

Like TMs and NTMs, PTMs have associated with them a number of complexity classes. The most important of these is the class BPP (bounded-error probabilistic polynomial time). This is the class of languages such that there exists a polynomial-time PTM that, on any given run, will correctly determine whether or not a string x is in the language L with probability $\geq 2/3$. The particular threshold value of 2/3 is inessential to this definition. It is chosen in order to express the idea of a 'high probability'.[19] But any threshold probability $p_{min} \geq 1/2 + n^{-k}$, where k is a constant, will suffice for the definition of BPP. For given a polynomial-time PTM that correctly determines whether or not $x \in L$ with probability p_{min}, re-running it a number of additional times

[19]Note that the high probability requirement constitutes a key conceptual difference between BPP and NP. The latter demands only that it is *possible* for an NTM to arrive at a correct solution.

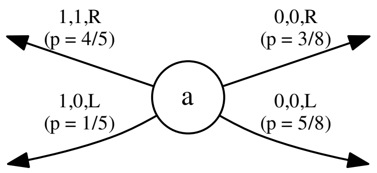


Figure 3: One node in a PTM. Given an input of 1 in the state **a**, the machine will write 1 to the tape and move right with probability 4/5, or write 0 and move left with probability 1/5. On an input of 0 it will write 0 and move right with probability 3/8, or write 0 and move left with probability 5/8. For a given state and a given input, edge probabilities must add up to 1. We can imagine that the machine's choices are made in accordance with these probabilities by repeatedly 'flipping a coin'.

that is no more than polynomial in n and taking the majority answer will yield a correct result with probability close to 1 (Arora & Barak, 2009, p. 132). Since, as I mentioned above, the time it takes to run a polynomial-time algorithm a polynomial number of times is still polynomial, varying p_{min} in this way will do nothing to alter the set of languages contained in BPP.

4 The universality and invariance theses

The Church-Turing (C-T) thesis claims nothing about the efficiency of any particular model of computation. Nor does it carry with it any implications concerning physically possible computing machines in general (see Turing, 1950, §§3, 5, 6.7). Both Church's and Turing's theses are, as we saw earlier, theses concerning the limits of effective procedures. Despite this, the C-T thesis is often misrepresented in this regard in the philosophical and even in the scientific literature (for further discussion of the reasons for the confusion, see Copeland, 2015; Timpson, 2013; Pitowsky, 1990). In more informed literature, however, these re-interpretations of the C-T thesis are explicitly distinguished from it. The thesis (I) that any reasonable model of computation can be simulated with at most a polynomial number of extra time steps by a PTM is often called the 'strong' C-T thesis (see, e.g., Nielsen & Chuang, 2000, p. 140).[20,21] The thesis (II) that a physical instantiation of a TM can simulate any physically possible machine that realises a finite instruction set and that works on finite data is often called the 'physical' C-T thesis (Andréka et al., Forthcoming; Piccinini, 2011). But confusingly, (II) is also sometimes called the 'strong' thesis (Goldin & Wegner, 2008), and (I) is sometimes called the 'physical' thesis (Hagar, 2007).

So as not to contribute to the confusion arising from this ambiguous labelling, and more importantly, to discourage any erroneous inferences to the intended scope of Church's and Turing's original theses themselves, I will, following Gandy (1980), refer to (II) as 'Thesis M'. I will refer to (I), the subject of this section, as the 'universality of Turing efficiency thesis'. For it follows from the truth of (I) that the set of problems efficiently solvable in general, i.e., on any reasonable digital machine model $\mathfrak{M}$, is identical with the set of problems efficiently solvable on a PTM. Formally this can be expressed as:

$$\bigcup \mathsf{Poly}_{\mathfrak{M}} = \mathsf{BPP}. \tag{4}$$

In other words, the thesis implies that the set of problems solvable in polynomial time does not grow beyond BPP if we allow ourselves to vary the underlying model.[22]

[20]Some textbooks state (I) as a thesis about the TM rather than the PTM model (see, e.g., Arora & Barak, 2009, p. 26). I will follow Nielsen & Chuang (2000), in order to leave open the possibility that P $\subsetneq$ BPP, and also because BPP constitutes a more natural contrast (see fn. 22 below) with its quantum analogue, BQP, which we will introduce in the next section. Until recently, P $\subsetneq$ BPP was thought to be very likely true, however evidence (e.g., Agrawal et al., 2004) has been mounting in favour of the conjecture that in fact P = BPP. Whether (I) is formulated with respect to TMs or PTMs makes little difference to what follows. A TM can be thought of as a special case of a PTM for which transition probabilities are always either 0 or 1.

[21]The qualification 'reasonable' will be explained shortly.

[22]There is a slight complication that I am glossing over here, namely that what it means for a machine to constitute a solution to a problem varies across computational models. In particular a TM solution to a problem is required to yield a correct answer with certainty, whereas (as I mentioned previously) a PTM solution in general need only yield a correct answer with probability close to 1. Implicit in (4), therefore, is an appeal to the more general criterion for solvability corresponding to that appropriate to a PTM rather than to a TM. This subtle distinction regarding what it means to solve a problem under various models of computation is one reason, that I alluded to in fn. 20 above, for expressing the universality thesis in terms of BPP rather than

A further closely related notion is what van Emde Boas (1990, p. 5) has called the ‘invariance thesis’. This states that any reasonable machine model can simulate any other reasonable machine model with no more than a polynomial slowdown (see also: Goldreich 2008, p. 33, who names it differently). The invariance thesis implies the universality thesis, but not vice versa. Note that in the context of both the universality and invariance theses, ‘reasonable’ is typically understood as *physically realisable*. Reasonable models include variants of the TM model, for example, but do not include models which employ unbounded parallelism.[23] This will be discussed further in §6.

There are reasons for believing in the truth of both the universality and invariance theses. Neither the standard variations on the Turing model, such as adding more tapes, increasing the number of squares readable or writable at a given moment, and so on (Arora & Barak, 2009), nor the alternative reasonable universal (classical) models of computation that have been developed since Turing’s work, are faster than PTMs by more than a polynomial factor, and all appear to be able to simulate one another efficiently in this sense (van Emde Boas, 1990).

Over the last three decades, however, evidence has nevertheless been mounting against universality and invariance, primarily as a result of the advent of quantum computing (Aaronson, 2013a, chs. 10, 15). We will discuss quantum computing in more detail in the next section.

5 Quantum computation

Consider the (non-quantum) machine depicted in Figure 4. This simple automaton has two possible states: $\{0, 1\}$. It has one possible input (omitted in the state-transition diagram), which essentially instructs the machine to ‘run’. This can be implemented, for example, by a button connected to the machine’s inner mechanism. At the end of any given run, the machine will either remain in the state it was previously in or else transition to the opposite state, with equal probability. One can imagine that the machine also includes a small door which, when opened, reveals a display indicating what state the machine is in. A typical session with the machine consists in: (a) opening the door to record the machine’s initial state; (b) pushing the ‘run’ button one or more times; (c) opening the door to reveal the machine’s final state.

Let us suppose that between the initial and final opening of the door, our experimenter pushes

P. For as we will see in the next section, a quantum computer, like a PTM, is a probabilistic machine and is subject to the same criterion for success. Expressing the universality thesis in terms of BPP thus allows for a more straightforward analysis of the quantum model’s significance for the thesis. A similar remark applies to the invariance thesis, which I now introduce.

[23]The parallel random access machine (PRAM) model, for example, is excluded.

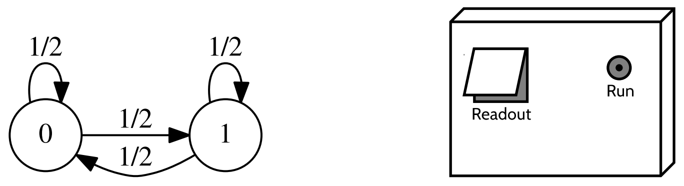


Figure 4: A simple automaton which, when run, randomly transitions to one of two possible states. In the state-transition diagram at the left, edge labels represent probabilities for the indicated transitions.

the button twice. Given that the initial reading was 0, what is the probability that the final reading is 0 as well? This is given by:

$$\begin{aligned}\text{Pr}(C^2 0 \to 0) &= \text{Pr}(C0 \to 0) \times \text{Pr}(C0 \to 0)\\ &+ \text{Pr}(C0 \to 1) \times \text{Pr}(C1 \to 0)\\ &= 1/2, \end{aligned} \tag{5}$$

where $C^n\psi \to \phi$ signifies that the computer is run n times after beginning in the state ψ, and ends in ϕ. Eq. (5) illustrates that there are two possible ways for the computer to begin and end in the state 0 after two runs. Either it remains in 0 after each individual run, or else it first flips to 1 and then flips back. From Figure 4, one can easily see that:

$$\text{Pr}(C^2 0 \to 0) = \text{Pr}(C^2 0 \to 1) = \text{Pr}(C^2 1 \to 0) = \text{Pr}(C^2 1 \to 1) = 1/2, \tag{6}$$

and indeed we have that $\text{Pr}(C^n\psi \to \phi) = 1/2$ for any n.

The internal state (what is revealed by opening the door) of the simple machine pictured in Figure 4 is describable by a single binary digit, or 'bit'. In general the internal state of any classical digital computer is describable by a sequence of n bits, and likewise for its inputs and outputs. A bit can be directly instantiated by any two-level classical physical system, for example by a circuit that can be either open or closed. In a *quantum* computer, on the other hand, the basic unit of representation is not the bit but the qubit. To directly instantiate it, we can use a two-level quantum system such an electron (specifically: its spin). The qubit generalises the bit. Like a bit, it can be 'on', i.e. in the state $|0\rangle$, or 'off', i.e. in the state $|1\rangle$. In general, however, the state of a qubit can be expressed as a normalised linear superposition:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{7}$$

where the 'amplitudes' α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. We refer to $|\psi\rangle$ as the 'state vector' for the qubit.[24]

An important difference between qubits and bits is that not all states of a qubit can be observed directly; in particular, one never observes a qubit in a linear superposition (aside from the trivial case in which one of α, β is 0).[25] According to the Born rule, a qubit in the state (7), when measured, will be found to be in the state $|0\rangle$ with probability $|\alpha|^2$, and in the state $|1\rangle$ with probability $|\beta|^2$. For example, consider a simple one-qubit quantum machine that implements the following transitions:

$$Q|0\rangle \to \frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \equiv |\chi\rangle, \tag{8}$$

$$Q|1\rangle \to \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle \equiv |\xi\rangle. \tag{9}$$

If the machine begins in the state $|0\rangle$, and the button is pushed once, it will transition to $|\chi\rangle$. Then with probability $|\frac{i}{\sqrt{2}}|^2$, opening the door will reveal $|0\rangle$, and with probability $|\frac{1}{\sqrt{2}}|^2$ it will reveal $|1\rangle$.

Since $|\frac{i}{\sqrt{2}}|^2 = |\frac{1}{\sqrt{2}}|^2 = 1/2$, a series of 'one-push' experiments with this quantum machine will produce identical statistics as will a series of one-push experiments with the classical machine depicted in Figure 4. Things become more interesting when we consider two-push experiments. If the machine is in the initial state $|0\rangle$, then after the first push the machine will effect the transition (8). If, before opening the door, we push the button again, the machine will make the

---

[24]The modulus squared (or 'absolute square'), $|c|^2$, of a complex number c is given by $c\bar{c}$, where $\bar{c}$ is the complex conjugate of c. $|\psi\rangle$ is normalised when $|\alpha|^2 + |\beta|^2 = 1$.

[25]To be more precise: one never observes a qubit in a linear superposition with respect to a particular measurement basis. Generally, in quantum computing, measurements are carried out in the computational, i.e. $\{|0\rangle, |1\rangle\}$, basis. In this basis the superposition $(|0\rangle + |1\rangle)/\sqrt{2}$, for example, can never be the result of a measurement. If one measures in the $\{|+\rangle, |-\rangle\}$ basis, however, then such a result is possible, since $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. On the other hand, a measurement in the $\{|+\rangle, |-\rangle\}$ basis will never yield the result $|0\rangle = (|+\rangle + |-\rangle)/\sqrt{2}$ even though a result of $|0\rangle$ is possible in the computational basis.

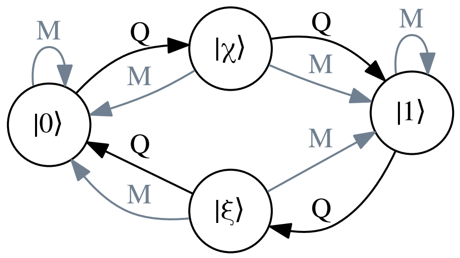


Figure 5: A simple quantum computer. With each button push, the machine deterministically oscillates, via the transition Q, between the states $|0\rangle, |\chi\rangle, |1\rangle, |\xi\rangle$ in the manner depicted. When the door is opened, the machine undergoes the 'measurement' transition M. This results, when the computer is in one of the states $|\chi\rangle$ and $|\xi\rangle$, in a reading of $|0\rangle$ or $|1\rangle$ with equal probability. Opening the door when the machine is in either $|0\rangle$ or $|1\rangle$ has no effect on the computer's state.

following transition:

$$\begin{aligned} Q\Big(\frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\Big) &= \frac{i}{\sqrt{2}}Q|0\rangle + \frac{1}{\sqrt{2}}Q|1\rangle \\ &= -\frac{1}{2}|0\rangle + \frac{i}{2}|1\rangle + \frac{1}{2}|0\rangle + \frac{i}{2}|1\rangle = i|1\rangle. \end{aligned} \tag{10}$$

Since $|i|^2 = 1$, opening the door will find the machine in the state $|1\rangle$ with certainty. Likewise, if the machine begins in $|1\rangle$, a two-push experiment will find it in the state $|0\rangle$ with certainty. A state transition diagram for the quantum machine is given in Figure 5.[26]

The probabilities for outcomes of two-push experiments with the quantum computer Q are significantly different from those associated with two-push experiments on C. This is despite the fact that if one performs two (or in general n) repetitions of a *one*-push experiment (i.e. in which one opens the door after every button push), the resulting statistics will be identical for both C and Q. One can think of a one-push experiment with C or Q as instantiating a 'maximally noisy'

[26]Note that overall phase factors have been abstracted away from in Figure 5. Two normalised state vectors which differ only in their overall phase factor yield all of the same probabilities for outcomes of experiments and are considered as equivalent according to quantum theory. For example, all of these express the same physical state: $|\psi\rangle$, $-|\psi\rangle$, $i|\psi\rangle$, $-i|\psi\rangle$. *Local* phase factors, however, are important; $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, for example, are different states.

(i.e. completely useless) NOT-gate. With a two-push experiment on Q, however, we have instantiated a perfect NOT-gate. We cannot do anything analogous with C.

The foregoing was a simple—almost trivial—illustration of some of the basic differences between classical and quantum computation. But by taking advantage of these and other differences, researchers have been able to develop quantum algorithms to achieve results that seem impossible for a classical computer. Quantum computers cannot solve non-Turing-computable problems (see Hagar & Korolev, 2007). However, as we will discuss shortly, quantum computers are able to efficiently solve problems that have no known efficient classical solution. This apparent ability of quantum computers to outperform classical computers is known as 'quantum speedup'.

A fascinating question, assuming that they indeed have this ability,[27] regards exactly which physical features of quantum systems are responsible for it. We will not be discussing this question further here.[28] Rather, let us return to Gödel's question regarding the resources required to solve the restricted form of the *Entscheidungsproblem*. Could a quantum computer be used to solve NP-complete problems such as this one efficiently? It turns out that a quantum computer can yield a performance improvement over a standard TM with respect to such problems. Recall (see def. 2) that a language L is in NP if there is a TM, C, such that $x \in L$ iff there is a certificate u whose length is polynomial with respect to the length of x, that if fed to C can be used by C to verify x's membership in L in polynomial time. If in addition, in polynomial time, for any given x, C can itself either find such a suitable certificate, or determine that one does not exist, then L is also in P.

Let the question be, 'Does the string x of length n have a "proof", i.e. a certificate, of length

[27]There is strong evidence (some of which we will discuss shortly), however there is still no proof, that quantum computers can efficiently solve more problems than classical computers can.

[28]In (10), the partial amplitudes contributing to the $|0\rangle$ component of the state vector cancel each other out. Many quantum algorithms include similar transitions, leading some to view quantum interference as the source of quantum speedup (Fortnow, 2003), although others have questioned whether interference is a truly quantum phenomenon (Spekkens, 2007). The fact that some quantum algorithms appear to spawn parallel computational processes has led to the idea of 'quantum parallelism' as the primary contributing mechanism (Duwell, 2007, forthcoming), and to the related but distinct idea that this processing occurs in parallel physical universes (Hewitt-Horsman 2009; for a criticism see Aaronson 2013b; Cuffaro 2012). Others view quantum entanglement (Cuffaro, 2017, forthcoming-; Steane 2003), or quantum contextuality (Howard et al., 2014), as providing the answer. Still others view the structure of quantum logic as the key (Bub, 2010).

$\leq n^k$?', for some constant k. The number of possible certificates u_i is $N = 2^{n^k}$. Assuming the space of certificates is unstructured, C will need $O(N)$ steps to decide whether x is in L; the computer will run through the possible certificates u_i one by one, testing each in turn to see if it is a valid certificate for x, moving on to the next certificate if it is not. Using a quantum computer and Grover's quantum search algorithm (Grover, 1996), however, only $O(\sqrt{N})$ steps are required. It turns out that this is the best we can do (Bennett et al., 1997). But while this quadratic speedup is impressive, the overall running time of the quantum computer remains exponential in the length, n, of x. Quantum computers, therefore, do not appear to allow us to affirm the Gödelian conjecture.[29]

However there is evidence that the class of languages efficiently decidable by a quantum computer is larger than that corresponding to either a deterministic or probabilistic classical computer. To be more precise, define the class BQP, analogously to the class BPP, as the class of languages such that there exists a polynomial-time quantum computer that will correctly determine, with probability $\geq 2/3$, whether or not a string x is in the language L. The question of whether a quantum computer can outperform a classical computer amounts to the question of whether BQP is larger than BPP. It is clear that BPP $\subseteq$ BQP; one invocation of the transition (8), for example, followed by a measurement, can serve to simulate a classical 'coin flip', and in polynomial time this procedure can be used to simulate any of a given PTM's transition probabilities.[30] As for the evidence for strict containment—i.e. for BPP $\subsetneq$ BQP—this comes mainly from the various quantum algorithms that have been developed.

Shor's quantum algorithm (Shor, 1997) for integer factorisation is a spectacular example. The best known classical factoring algorithm is the number field sieve (Lenstra et al., 1990), which requires $O(2^{(\log N)^{1/3}})$ steps to factor a given integer N. Popular encryption mechanisms such as RSA (Rivest et al., 1978) rely on the assumption that factoring is hard. Yet Shor's algorithm requires only a number of steps that is polynomial in $\log N$—an exponential speedup over the

[29]Note that above it was assumed that the space of certificates is unstructured. However it is possible that a given NP-complete language L possesses non-trivial structure that can be exploited to yield further performance improvements (Cerf et al., 2000). Therefore we cannot rule out that L is efficiently decidable by a classical computer, let alone by a quantum one.

[30]Rather than Q, one typically uses a 'Hadamard gate' (H) for this purpose, which acts as follows:

$$H|0\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \qquad H|1\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

number field sieve. There are also provable 'oracle' separations between the classical and quantum computational models. An oracle is a kind of imaginary magic black box, to which one puts a question chosen from a specified set, and from which one receives an answer in a single time step. For example, Simon's problem (Simon, 1994) is that of determining the period of a given function f that is periodic under bitwise modulo-2 addition. One can define an oracle $\mathcal{O}$ for evaluating arbitrary calls to f. Relative to $\mathcal{O}$, Simon's quantum algorithm requires $O(n)$ steps, while a classical algorithm requires $O(2^n)$. This is an exponential speedup.

None of these results are absolutely conclusive. On the one hand, not every complexity-theoretic proposition relativises to an oracle. The result that IP $=$ PSPACE, for example, does not hold under certain oracle relativisations (PSPACE is the class of problems solvable using polynomially many space resources; IP is the class of problems for which an affirmative answer can be verified using an interactive proof). Further, there are oracles relative to which P $=$ NP, as well as oracles relative to which P $\neq$ NP. Oracles are important tools that, among other things, help to simplify and clarify the conceptual content of complexity-theoretic propositions, however they cannot be used to resolve these and other questions (for a discussion, see Fortnow, 1994). Nor can they definitively show that BPP $\subsetneq$ BQP. Simon's problem, for instance, might contain some hitherto unknown structure, obscured by the relativisation of the problem to an oracle, that could be exploited by a classical algorithm. Regarding Shor's algorithm, on the other hand, unlike Simon's, it does not make essential use of an oracle. Yet this nevertheless does not conclusively demonstrate that BPP $\subsetneq$ BQP, for it is still an open question whether factoring is in BPP. While most complexity theorists believe this to be false, their conviction is not as strong as their conviction, for example, that P $\neq$ NP—for the factoring problem does have *some* internal structure, which is in fact exploited by the classical number field sieve algorithm (Aaronson, 2013a, 64-66).

While none of the individual pieces of evidence are conclusive, taken together they nevertheless do lend a great deal of plausibility to the thesis that quantum computers can solve more problems efficiently than can classical computers. That said, it is not the place here to evaluate this evidence. For the purposes of our discussion we will assume that this thesis is true. In the next section we will consider its consequences.

6 Quantum computing and the foundations of computational complexity theory

If BPP $\subsetneq$ BQP, then it follows that the universality of Turing efficiency thesis is false. Some authors view the consequences of this for complexity theory to be profound. Bernstein & Vazirani (1997), for example, take it that the theory "rests upon" this thesis (p. 1411), and that the advent of quantum computers forces us to "re-examine the foundations" (p. 1412) of the theory. The sense in which complexity theory rests upon universality is expressed by Nielsen & Chuang (2000), who write that the failure of the universality thesis implies that complexity theory cannot achieve an "elegant, model independent form" (p. 140). Of this, Hagar (2007) writes:

> To my mind, the strongest implication [of the violation of universality] is on the autonomous character of some of the theoretical entities used in computer science, ... given that quantum computers may be able to efficiently solve classically intractable problems, hence re-describe the abstract space of computational complexity, computational concepts and even computational kinds such as 'an efficient algorithm' or 'the class NP', will become machine dependent, and recourse to 'hardware' will become inevitable in any analysis of the notion of computational complexity. (pp. 244–5).

Given that the universality of Turing efficiency thesis states that any reasonable model of computation can be simulated with at most a polynomial number of extra time steps by a (probabilistic) *Turing machine*, however, the reader may be understandably confused by the claim that this thesis grounds the model-independence of complexity-theoretic concepts to begin with. At most, it seems that only a very weak sense of model-independence follows from universality. The truth of (4), that is, implies that any assertion, of the form 'language L is decidable efficiently by an instance of the reasonable machine model $\mathfrak{M}$', is replaceable by the assertion that 'language L is decidable efficiently by a PTM'. And since 'by a PTM' is thus made universally applicable, it can be omitted from all such statements in the knowledge that it is implicit (cf. Nakhnikian & Salmon, 1957). But while this yields a kind of model-independence in the sense that one need not explicitly mention the PTM model when speaking of the complexity-theoretic characteristics of L, it remains the case, nevertheless, that a reference to the PTM model is implicit in one's assertions about L.

To illustrate just how weak such a notion of model-independence is, note that, based on it, one could argue that, although quantum computing refutes the model-independence consequent upon the universality of *Turing* efficiency, it at the same time provides a replacement for it (cf. Deutsch 1985, Bernstein & Vazirani 1997, p. 1413). Nielsen & Chuang (2000) write that if the universality of Turing efficiency thesis were true, that it would be

> great news for the theory of computational complexity, for it implies that attention may be restricted to the probabilistic Turing machine model of computation. After all, if a problem has no polynomial resource solution on a probabilistic Turing machine, then the [universality of Turing efficiency] implies that it has no efficient solution on any computing device. Thus, the [universality of Turing efficiency] implies that the entire

theory of computational complexity will take on an elegant, model-independent form if the notion of efficiency is identified with polynomial resource algorithms (p. 140).

Putting aside for the moment the somewhat strange comment that an expansion of our knowledge of the extent of the space of efficiently decidable languages is 'bad news' for complexity theory, note that quite the same argument could be made if one replaces 'probabilistic Turing machine' with 'quantum computer' and 'universality of Turing efficiency' with 'universality of quantum efficiency'. For although BPP in (4) is now replaced with BQP, we have analogously given a characterisation, in terms of a single machine model, of $\bigcup \text{Poly}_{\mathfrak{M}}$. And yet if a computer based on the principles of *quantum physics* can be taken to ground an absolute model-independent characterisation of complexity-theoretic concepts, then the right conclusion to draw is that this is not a satisfactory notion of model-independence.[31]

One could, perhaps, counter that the model-independence consequent on the universality thesis actually stems from the nature of the Turing model itself. Assuming that one is convinced by Turing's philosophical analysis, the Turing model does, after all, represent the conceptual essence of effective computation (cf. Hartmanis & Stearns, 1965, p. 285). Be that as it may, there is no reason to think that such a model must also be the most efficient one.[32] The model-independence of complexity theory thus would turn out to be a contingent fact. This in itself is not a criticism. Nevertheless in that case it is not clear just what model-independence would contribute to the ground of complexity theory in the theoretical sense. A 'universality of quantum efficiency thesis' would be, perhaps, less metaphysically satisfying from the point of view of a computer scientist, but in itself would do just as much theoretical work as the universality of Turing efficiency thesis.

$\text{BPP} \subsetneq \text{BQP}$ also implies the failure of the invariance thesis. Because of its supposed relation to the Church-Turing thesis, it is universality and not invariance that has received the lion's share of attention from philosophers (an exception is Dean). But unlike the universality thesis, there is a sense of model-independence built right into the very statement of invariance. After all, it amounts to a direct claim that the particular machine model under consideration, since it can be efficiently simulated by any other reasonable model, is irrelevant for the purposes of providing a characterisation of the complexity of a problem. Note also that the statement of invariance itself

[31]One could, however, ground a *relative* notion of model-independence based on quantum principles. I will discuss this further below.

[32]For a discussion of some possible justifications, and the problems that go along with them, for the choice of the multi-tape TM as the benchmark for complexity-theoretic analyses, see Dean (2016c).

makes no reference to the Turing model, so it is not susceptible to the same sort of criticism I directed at the universality thesis above. It is true that the domain of 'reasonable' or physically realisable models does not, for example, include the 'unreasonable' parallel computational models, thus the invariance thesis cannot provide model-independence in a truly absolute sense. Still, the study of efficient algorithms in particular is mainly concerned with reasonable models. So invariance, if true, arguably provides absolute model-independence in the only sense that matters.[33]

Van Emde Boas takes the invariance thesis (he does not mention the universality thesis) to, as a consequence, constitute a foundational thesis for complexity theory:

> The fundamental complexity classes P and NP became part of a *fundamental hierarchy*: LOGSPACE, NLOGSPACE, P, NP, PSPACE, EXPTIME, ... And again theory faced the problem that each of these classes has a *machine-dependent* definition, and that efficient simulations are needed before one can claim that these classes are in fact *machine-independent* and represent fundamental concepts of computational complexity. It seems therefore that complexity theory, as we know it today, is based on the [assumption that the invariance thesis holds] (p. 5 van Emde Boas, 1990, ellipsis in original).

I will be criticising this statement. Before I do, however, it is important to note that it is clear that the simplifying assumption of invariance can be profoundly useful; its truth would imply that one can restrict one's attention to the (reasonable) model of one's choice when inquiring into the complexity-theoretic characteristics of particular problems. Further, and independently of this, studies such as van Emde Boas's, of the extent to which one model can simulate another, illuminate the structure of complexity theory by allowing one to characterise the space of machine models in terms of various 'equivalence classes'. Van Emde Boas, for instance, defines the models comprising the 'first machine class' as those for which the invariance thesis holds. The 'second machine class' is defined with respect to a different 'parallel computation thesis' (1990, p. 5), which I will not further describe here. Note that van Emde Boas is careful to point out the partly conventional and partly empirical (he does not use these words) nature of such theses:

> The escape in defending the Invariance Thesis ... is clearly present in the word *reasonable* ... For example, when in 1974 it was found that a RAM model with unit-time multiplication and addition instructions (together with bitwise Boolean

[33]See Dean (2016c) for a discussion of ways to circumscribe the space of operations that should be allowable in a reasonable model.

> operations) is as powerful as a parallel machine, this model (the MBRAM) was thrown out of the realm of reasonable (sequential) machines and was considered to be a "parallel machine" instead. The standard strategy seems to be to adjust the definition of "reasonable" when needed. The theses become a guiding rule for specifying the right classes of models rather than absolute truths and, once accepted, the theses will never be invalidated. This strategy is made explicit if we replace the word *reasonable* by some more neutral phrase. (van Emde Boas, 1990, pp. 5-6).

There is much to commend in this statement. But note first that it is not clear that the 'standard strategy' will work in the face of quantum computing. On the one hand, one would be hard-pressed to argue that quantum computers are not physically realisable machines. On the other hand, the 'quantum parallelism thesis' (Duwell, 2007, forthcoming; Pitowsky, 1990, 2002) is quite controversial (Cuffaro, 2012; Steane, 2003), so it is not obvious that quantum computers should be classed as parallel rather than sequential computers. This said, even if one takes the invariance thesis to be violated by quantum computing, the idea of it, not as an absolute truth but as a 'guiding rule'—a sort of intentional principle for characterising the extensions corresponding to different equivalence classes of models—remains, and it remains a highly illuminating methodological principle for studying the relations between computational models.

To illustrate what I mean by this,[34] note that 'quantum computer' is actually an umbrella term for a number of (universal) computational models: the quantum Turing model (Deutsch, 1985), the quantum circuit model (Deutsch, 1989), the cluster-state model (Briegel et al., 2009), the adiabatic model (Farhi et al., 2000), and many others. To date, all of these models have been found to be computationally equivalent in the sense that they all yield the same class of problems, BQP (see, for example, Aharonov et al., 2007; Raussendorf & Briegel, 2002; Nishimura & Ozawa, 2009). Thus it seems as though a third *quantum* machine class, in addition to van Emde Boas's first and second machine classes, exists. Fascinatingly, the differences between the first and this third machine class turn out to be related to the differences in the physics used to describe the machines which comprise them. The physics, in turn, informs our judgements regarding which of these equivalence classes should be deemed as 'reasonable'. On the basis of these judgements we are then enabled make conclusions with regards to what is feasible for us to accomplish in the real world (cf. Dean, 2016a, pp. 30, 56). If it were not for the existence of quantum computers, one would be warranted in the belief that only a single reasonable machine class exists. Quantum computing teaches us that there are at least two.

[34] I am indebted to Scott Aaronson and to Sona Ghosh for independently prompting the discussion in this and the next two paragraphs.

Invariance, thought of as a guiding rule or *methodological principle*, rather than as a thesis, is what is driving these investigations; through the search for equivalence classes we carve out and illuminate the structure of the space of computational models. This yields a notion of *relative* model-independence among the machine models comprising a particular class. To be clear, the existence of relative model-independence within the conceptual structure of complexity theory is itself not strictly speaking necessary for the theory. It is true that the theory would arguably be far less interesting if every abstract model of computation were different from every other; for one thing there would be no unified notion of 'classical computation' to compare quantum computation with—a quantum computer would be just another model among many. Yet one can still imagine what such a complexity theory would look like: a theory describing the computational power of various abstract models of computation and their interrelations. This is not so alien that it would be unrecognisable from our modern point of view.[35] In fact the early period of complexity theory, before the introduction of the polynomial principle (to be discussed below) looked much like this. Representative of this period are results such as that by Hartmanis & Stearns (1965), for example, who prove that the multi-tape TM model is capable of a quadratic speedup with respect to the single-tape TM model. Such analyses are indeed still present in the modern theory.

The fact that relative model-independence does exist, on the other hand, arguably tells us something deep, or anyway something, about how computer science connects up with the world. The invariance principle (rather than thesis) is a vitally important part of computational complexity theory partly for this reason. And as a methodological principle, it fulfils this role whether it is successful in its search for equivalence classes of computational models or not. For the lack of any relative model-independence within the theory would arguably also tell us something about computer science's relation to the physical world. A further, still methodological, role of invariance is as a simplifying principle. For from a practical perspective the theory would be exceedingly unwieldy, even if it would not strictly speaking be impossible to develop, if no equivalence classes of abstract computational models existed.

And yet none of this seems to imply or depend upon model-independence in the sense of the

[35]The very notion of an abstract model of computation presupposes *some* notion of complexity-theoretic invariance, of course, without which it would be impossible to unify various physical instantiations of a model under a single concept. I am perfectly ready to concede that if complexity-theoretic invariance failed to hold in this minimal sense then this would be disastrous for modern complexity theory. But then I cannot see how it would be possible to revise complexity theory in light of this; it would seem impossible to have a theory of complexity, or indeed any theory of any subject, if even basic abstraction were not possible.

first of my quotes of van Emde Boas above. Indeed it is not clear what one gains from model-independence in that sense. LOGSPACE, NLOGSPACE, P, NP, PSPACE, EXPTIME, and other complexity classes are each of them classes of *languages*, after all. To compare any two of them is to compare one class of languages with another; they are thus already machine-independent in that sense. For this reason it is a meaningful question to ask whether the class P is large enough to include all of the languages in NP, irrespective of how one defines either of them in terms of an underlying machine model. On the other hand, one *can* define P as the class of languages which are efficiently decidable by a TM. And one *can* define NP as the class of languages which are efficiently decidable by an NTM. And to be sure, deeper insight is gained by reflecting on how one translates the definition of NP given in terms of the NTM model (3), into the alternative definition of NP given in terms of the standard TM model (2). For then one sees clearly that the statement that P = NP amounts to the assertion that the restricted form of the *Entscheidungsproblem* is efficiently solvable. But in this case it is by reflecting on the characteristics of a *particular model* that one gains this insight, namely, the Turing machine model insofar as it represents the conceptual essence of *human* digital computation.[36]

I am not alone in my skepticism. The idea that it is the fundamental goal of complexity theory to get at some metaphysical notion of an independently existing *thing* called 'efficient computation' is certainly not shared by all complexity theorists. For example, Fortnow (2006) writes:

> By no means does computational complexity "rest upon" a [universality of Turing efficiency] thesis. The goals [sic.] of computational complexity is to consider different notions of efficient computation and compare the relative strengths of these models. Quantum computing does not break the computational complexity paradigm but rather fits nicely within it.

Fortnow's statement refers to the universality thesis; however it can clearly equally well be asserted as a counter to the claimed foundational status of the invariance thesis. A quick glance at the practice of complexity theorists seems to confirm that Fortnow is right, for since the advent of quantum computing in the mid-1990s, complexity theory appears to have continued on in much the same way as before. Classic textbooks such as Papadimitriou's (1994) excellent reference, written before Shor's (1994) breakthrough in quantum factoring, continue to be cited frequently in modern scholarly work; more modern textbooks such as Arora & Barak's (2009) book include a chapter on quantum computation but otherwise present the subject in much the same way as

[36]This is one reason why the question whether P = NP remains interesting even if $P \subsetneq BQP$.

similar textbooks always have done; BQP is just one of many classes in Aaronson's (2016) 'complexity zoo'. Despite the fact that the prospects for a practicable and scalable quantum computer are improving significantly every year (Veldhorst et al., 2015), and despite the fact that most computer scientists believe that BPP $\subsetneq$ BQP and thus that the universality and invariance theses are false, complexity theory—as a discipline—does not appear to be in crisis. Complexity theory as a whole has grown—many new and important questions have arisen regarding exactly how BQP relates to other complexity classes—but the basic conceptual framework within which we ask these questions remains much the same as before.

But if model-independence is not constitutive of complexity theory, what is? Built into the definition of both the universality and invariance theses is the more basic idea that an algorithm is efficient iff it requires no more than a polynomial number of steps to complete. As we have seen, the roots of this idea go back at least as far as Gödel's letter to von Neumann, although from the modern perspective, its main sources are the seminal articles of Cobham (1965) and Edmonds (1965). I will call it the 'polynomial efficiency principle' or 'polynomial principle' for short.[37] Unlike either the universality or invariance theses, there is no question that the polynomial principle is de facto foundational with respect to the modern framework of complexity theory, in the sense that the conceptual structure of the theory—the definitions of and relations between its most important complexity classes such as P, NP, BPP, BQP, and so on—depend crucially upon the principle.

And yet there is a different sense in which it can be said to be controversial. The goal of the polynomial principle is to capture our pre-theoretic notion of what it means for an algorithm to be efficient. Expressing this, Edmonds writes:

> ... my purpose is ... to show as attractively as I can that there is an efficient algorithm [for maximum matching]. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."
>
> I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph. (Edmonds, 1965, p. 420, emphasis in original).

[37]Forms of this principle are often referred to as the Cobham-Edmonds thesis (e.g., see Dean, 2016b). Unfortunately, this terminology is not always consistent. In Goldreich (2008, p. 33), for example, the Cobham-Edmonds thesis is the name given to what we have here called the invariance thesis.

One could argue, however, that the polynomial principle fails to achieve this goal. For example, a problem for which the best algorithm requires $O(n^{1000})$ steps to complete is considered to be tractable according to the principle, while a problem for which the best algorithm requires $O(2^{n/1000})$ steps is considered to be intractable. Yet despite these labels, the 'intractable' problem will take fewer steps to solve for all but very large values of n. Strictly speaking, of course, since it is defined asymptotically, the principle does not yield an incorrect answer even in such cases. However problems we are faced with in practice are invariably of bounded size, and an asymptotic measure—the preceding example illustrates this—seems to at least sometimes be ill-suited for their analysis. A further reason for doubting the polynomial principle is that it is a measure of worst-case complexity. Yet it does not seem implausible that an average-case measure might give us better insight into just how 'good' a given algorithm is.

All of this may be granted. And yet growth rates such as the above are extremely rare in practice. Generally speaking, polynomial-time algorithms have growth rates with small exponents, and the simplification made possible by the use of an asymptotic measure generally does more good than it does harm; "For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite." (Edmonds, 1965, p. 451). We also generally do not know in advance how the inputs to a particular problem will be distributed, and in such circumstances average case complexity analyses are impracticable (see Papadimitriou, 1994, pp. 6-7).

What the arguments for and against the polynomial principle illustrate is that its goal is not so much to provide an absolute or metaphysical distinction between good and bad algorithms. What these arguments show us is that the purpose of the principle is to guide us in making such distinctions in practice. In particular, what the arguments *for* the principle amount to is the—empirical—claim that the polynomial principle has been highly successful, in the sense that it has tended to provide us with extraordinarily good guidance for the problems with which we are generally faced. Aaronson sums this up as follows:

> Of the big problems solvable in polynomial time—matching, linear programming, primality testing, etc.—most of them really *do* have practical algorithms. And of the big problems that we think take exponential time—theorem-proving, circuit minimization, etc.—most of them really *don't* have practical algorithms. So, that's the empirical skeleton holding up our fat and muscle (2013a, p. 54).

The precise way in which the polynomial principle aids us in the search for good algorithms is by providing us with a mathematical explication of 'good' in the context of complexity theory. In doing so it provides us with a mathematical framework for our inquiries, within which we can express precise questions, and generate precise answers.

> … if only to motivate the search for good, practical algorithms, it is important to realise that it is mathematically sensible even to question their existence. For one thing the task can then be described in terms of concrete conjectures. (Edmonds, 1965, p. 451).

And yet, while the principle is generally a good guide, it is we who must ultimately decide, in every case upon which we bring it to bear, whether or not to follow its advice.

> It would be unfortunate for any rigid criterion to inhibit the practical development of algorithms which are either not known or known not to conform nicely to the criterion. Many of the best algorithmic ideas known today would suffer by such theoretical pedantry. … And, on the other hand, many important algorithmic ideas in electrical switching theory are obviously not "good" in our sense. (Edmonds, 1965, p. 451).

Just as the polynomial principle is a practical principle, so is complexity theory a practical science, in the sense that its fundamental aim is to inform us with regards to the practical difficulty of computing different classes of problems—i.e. with regards to the things we would like to do—on our various machines. Principles such as the polynomial and even the invariance principle (insofar as it serves as a methodological principle for carving out equivalence classes of machine models) illuminate the structure of this space of possible tasks. But they, and the structure with it, are ultimately guides which should be set aside whenever they cease to be useful. To some extent this is true in every science—the principles of Newtonian mechanics, say, must give way to the principles of modern spacetime theories. But principles such as the polynomial principle, and the theory of complexity that is built upon it, do not claim for themselves universal validity as Newtonian mechanics at one time did—nor do they even claim to have a well-defined sphere of application (how large must an exponent be before a polynomial-time algorithm is no longer considered to be 'really' efficient?). The polynomial principle, and complexity theory with it, are intrinsically practical in nature; they claim to be useful only 'most of the time' and for most of our practical purposes. This is the theory's core.

This said, if, with the development of the theory, the polynomial principle somehow ceased to be useful even in merely the majority of cases of practical interest, this would certainly require a substantial revision of much of the theory's framework, for so much of the conceptual structure of complexity theory is built upon the assumption of the polynomial principle. And yet even in this case the essential nature and subject matter—the metaphysical foundation, if you will—of the theory—a theory of what we are capable of doing in practice—would not change.

7 Conclusion

Cobham (1965) took complexity theory to be a science concerned with three general groups of questions: those related (i) to "specific computational systems", (ii) to "categories of computing machines", and (iii) to those questions which "are independent of any particular method of computation" (pp. 24–5). The third subdivision will always remain a part of complexity theory. In fact, machine-independent results can be had in the theory—though no one would argue that these provide a foundation for it—if one uses a very general and amorphous notion of a complexity measure (Seiferas, 1990). Indeed studies such as these suggest that further reflection may be needed on precisely what is meant by the notion of 'intrinsic complexity'. But model-independence is not all of the theory; nor is it a foundation for the other two groups of questions mentioned by Cobham. Computational complexity theory is, at its core, a practical science. As a mathematical theory, it employs idealised concepts and methods, and appeals to formal principles which are justified insofar as they are useful in providing us with practical advice regarding the problems we aim to solve. Our solutions to such problems are implemented on particular machine models. Computational complexity theory studies the various notions of efficiency associated with these different models, and studies how these notions relate to one another. "Quantum computing", to quote Fortnow once again, "does not break the computational complexity paradigm but rather fits nicely within it." Indeed, far from breaking the complexity-theoretic paradigm, quantum computing serves to *remind* us of the point of it all.

References

Aaronson, S. (2013a). *Quantum Computing Since Democritus*. New York: Cambridge University Press.

Aaronson, S. (2013b). Why philosophers should care about computational complexity. In B. J. Copeland, C. J. Posy, & O. Shagrir (Eds.) *Computability: Turing, Gödel, Church, and Beyond*, (pp. 261–327). Cambridge, MA: MIT Press.

Aaronson, S. (2016). Complexity zoo. complexityzoo.uwaterloo.ca/Complexity_Zoo.

Agrawal, M., Kayal, N., & Saxena, N. (2004). PRIMES is in P. *Annals of Mathematics*, *160*, 781–793.

Aharonov, D., van Dam, W., Kempe, J., Landau, Z., Lloyd, S., & Regev, O. (2007). Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal on Computing*, *37*, 166–194.

Andréka, H., Madarász, J. X., Németi, I., Németi, P., , & Székely, G. (Forthcoming). Relativistic computation. In M. E. Cuffaro, & S. C. Fletcher (Eds.) *Physical Perspectives on Computation, Computational Perspectives on Physics*. Cambridge: Cambridge University Press.

Arora, S., & Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge: Cambridge University Press.

Bennett, C. H., Bernstein, E., Brassard, G., & Vazirani, U. (1997). Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, *26*, 1510–1523.

Bernstein, E., & Vazirani, U. (1997). Quantum complexity theory. *SIAM Journal on Computing*, *26*, 1411–1473.

Briegel, H. J., Browne, D. E., Dür, W., Raussendorf, R., & den Nest, M. V. (2009). Measurement-based quantum computation. *Nature Physics*, *5*, 19–26.

Bub, J. (2010). Quantum computation: Where does the speed-up come from? In A. Bokulich, & G. Jaeger (Eds.) *Philosophy of Quantum Information and Entanglement*, (pp. 231–246). Cambridge: Cambridge University Press.

Carnap, R. (1962 [1950]). *Logical Foundations of Probability*. Chicago: The University of Chicago Press, second ed.

Carnap, R. (1980 [1950]). Empiricism, semantics, and ontology. In H. Morick (Ed.) *Challenges to Empiricism*, (pp. 28–45). Indianapolis: Hackett Publishing Company.

Cerf, N. J., Grover, L. K., & Williams, C. P. (2000). Nested quantum search and structured problems. *Physical Review A*, *61*, 032303.

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

Cobham, A. (1965). The intrinsic computational difficulty of functions. In Y. Bar-Hillel (Ed.) *Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress*, (pp. 24–30). Amsterdam: North-Holland.

Copeland, J. B. (2015). The Church-Turing thesis. In E. N. Zalta (Ed.) *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2015 ed.

Cuffaro, M. E. (2012). Many worlds, the cluster-state quantum computer, and the problem of the preferred basis. *Studies in History and Philosophy of Modern Physics*, *43*, 35–42.

Cuffaro, M. E. (2017). On the significance of the Gottesman-Knill theorem. *The British Journal for the Philosophy of Science*, *68*, 91–121.

Cuffaro, M. E. (forthcoming-). Reconsidering no-go-theorems from a practical perspective. *The British Journal for the Philosophy of Science*.

Dawson Jr., J. W. (2007). Classical logic's coming of age. In D. Jacquette (Ed.) *Philosophy of Logic*, (pp. 497–522). Amsterdam: Elsevier.

Dean, W. (2016a). Algorithms and the mathematical foundations of computer science. In L. Horsten, & P. Welch (Eds.) *Gödel's Disjunction: The Scope and Limits of Mathematical Knowledge*, (pp. 19–66). Oxford: Oxford University Press.

Dean, W. (2016b). Computational complexity theory. In E. N. Zalta (Ed.) *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 ed.

Dean, W. (2016c). Squeezing feasibility. In A. Beckmann, L. Bienvenu, & N. Jonoska (Eds.) *Pursuit of the Universal: Proceedings of the 12th Conference on Computability in Europe*, (pp. 78–88). Cham: Springer International Publishing.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, *400*, 97–117.

Deutsch, D. (1989). Quantum computational networks. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, *425*, 73–90.

Duwell, A. (2007). The many-worlds interpretation and quantum computation. *Philosophy of Science*, *74*, 1007–1018.

Duwell, A. (forthcoming). How to make orthogonal positions parallel: Revisiting the quantum parallelism thesis. In M. E. Cuffaro, & S. C. Fletcher (Eds.) *Physical Perspectives on Computation, Computational Perspectives on Physics*. Cambridge: Cambridge University Press.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, *17*, 449–467.

Farhi, E., Goldstone, J., Gutmann, S., & Sipser, M. (2000). Quantum computation by adiabatic evolution. Tech. Rep. MIT-CTP-2936, MIT. arXiv:quant-ph/0001106.

Fortnow, L. (1994). The role of relativization in complexity theory. *Bulletin of the European Association for Theoretical Computer Science*, *52*, 229–244.

Fortnow, L. (2003). One complexity theorist's view of quantum computing. *Theoretical Computer Science*, *292*, 597–610.

Fortnow, L. (2006). The efficient Church-Turing thesis. blog .computationalcomplexity.org/2006/12/efficient-church-turing-thesis.html. Posted: Thursday, December 07, 2006. Retrieved: Monday, April 26, 2010.

Gandy, R. (1980). Church's thesis and principles for mechanisms. In J. Barwise, H. J. Keisler, & K. Kunen (Eds.) *The Kleene Symposium*, Studies in Logic and the Foundations of Mathematics, (pp. 123–148). Amsterdam: Elsevier.

Gödel, K. (1956). Private letter to John von Neumann, 20 March 1956. Translated by Wensinger in: Sipser (1992).

Goldin, D., & Wegner, P. (2008). The interactive nature of computing: Refuting the strong Church-Turing thesis. *Minds & Machines*, *18*, 17–38.

Goldreich, O. (2008). *Computational Complexity: A Conceptual Perspective*. Cambridge: Cambridge University Press.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, (pp. 212–219). New York, NY, USA: Association for Computing Machinery.

Hagar, A. (2007). Quantum algorithms: Philosophical lessons. *Minds & Machines*, *17*, 233–247.

Hagar, A., & Korolev, A. (2007). Quantum hypercomputation - hype or computation? *Philosophy of Science*, *74*, 347–363.

Hartmanis, J. (1993). Gödel, von Neumann and the P =? NP problem. In G. Rozenberg, & A. Salomaa (Eds.) *Current Trends in Theoretical Computer Science*, (pp. 445–450). River Edge, NJ: World Scientific.

Hartmanis, J., & Stearns, R. E. (1965). On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, *117*, 285–306.

Hewitt-Horsman, C. (2009). An introduction to many worlds in quantum computation. *Foundations of Physics*, *39*, 869–902.

Hilbert, D., & Ackermann, W. (1928). *Principles of Mathematical Logic*. Berlin: Springer-Verlag.

Howard, M., Wallman, J., Veitch, V., & Emerson, J. (2014). Contextuality supplies the 'magic' for quantum computation. *Nature*, *510*, 351–355.

Lenstra, A. K., Lenstra Jr., H. W., Manasse, M. S., & Pollard, J. M. (1990). The number field sieve. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, (pp. 564–572). New York, NY: Association for Computing Machinery.

Nakhnikian, G., & Salmon, W. C. (1957). Exists as a predicate. *Philosophical Review*, *66*, 535–542.

Nielsen, M. A., & Chuang, I. L. (2000). *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press.

Nishimura, H., & Ozawa, M. (2009). Perfect computational equivalence between quantum turing machines and finitely generated uniform quantum circuit families. *Quantum Information Processing*, *8*, 13–24.

Papadimitriou, C. H. (1994). *Computational Complexity*. New York: Addison-Wesley.

Piccinini, G. (2011). The physical Church-Turing thesis: Modest or bold? *The British Journal for the Philosophy of Science*, *62*, 733–769.

Pitowsky, I. (1990). The physical Church thesis and physical computational complexity. *Iyyun: The Jerusalem Philosophical Quarterly*, *39*, 81–99.

Pitowsky, I. (2002). Quantum speed-up of computations. *Philosophy of Science*, *69*, S168–S177.

Raussendorf, R., & Briegel, H. J. (2002). Computational model underlying the one-way quantum computer. *Quantum Information and Computation*, *2*, 443–486.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*, 120–126.

Seiferas, J. I. (1990). Machine-independent complexity theory. In J. van Leeuwen (Ed.) *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, (pp. 165–186). Cambridge, MA: MIT Press/Elsevier.

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, (pp. 124–134).

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, *26*, 1484–1509.

Simon, D. R. (1994). On the power of quantum computation. In *1994 Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, (pp. 116–123). Los Alamitos, CA: IEEE Press.

Sipser, M. (1992). The history and status of the P versus NP question. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, (pp. 603–618). New York, NY: Association for Computing Machinery.

Spekkens, R. W. (2007). Evidence for the epistemic view of quantum states: A toy theory. *Physical Review A*, *75*, 032110.

Steane, A. M. (2003). A quantum computer only needs one universe. *Studies in History and Philosophy of Modern Physics*, *34*, 469–478.

Timpson, C. G. (2013). *Quantum Information Theory & the Foundations of Quantum Mechanics*. Oxford: Oxford University Press.

Turing, A. M. (1936-7). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society. Second Series*, *s2-42*, 230–265.

Turing, A. M. (1938). On computable numbers, with an application to the Entscheidungsproblem. A correction. *Proceedings of the London Mathematical Society. Second Series*, *s2-43*, 544–546.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460.

van Emde Boas, P. (1990). Machine models and simulations. In J. van Leeuwen (Ed.) *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, (pp. 1–66). Cambridge, MA: MIT Press/Elsevier.

Veldhorst, M., Yang, C. H., Hwang, J. C. C., Huang, W., Dehollain, J. P., Muhonen, J. T., Simmons, S., Laucht, A., Hudson, F. E., Itoh, K. M., Morello, A., & Dzurak, A. S. (2015). A two-qubit logic gate in silicon. *Nature*, *526*, 410–414.