Why Philosophers Should Care About Computational Complexity

Scott Aaronson*

Abstract

One might think that, once we know something is computable, how *efficiently* it can be computed is a practical question with little further philosophical importance. In this essay, I offer a detailed case that one would be wrong. In particular, I argue that *computational complexity theory*—the field that studies the resources (such as time, space, and randomness) needed to solve computational problems—leads to new perspectives on the nature of mathematical knowledge, the strong AI debate, computationalism, the problem of logical omniscience, Hume's problem of induction, Goodman's grue riddle, the foundations of quantum mechanics, economic rationality, closed timelike curves, and several other topics of philosophical interest. I end by discussing aspects of complexity theory itself that could benefit from philosophical analysis.

Contents

*MIT. Email: aaronson@csail.mit.edu. This material is based upon work supported by the National Science Foundation under Grant No. 0844626. Also supported by a DARPA YFA grant, the Sloan Foundation, and a TIBCO Chair.

1 Introduction

> *The view that machines cannot give rise to surprises is due, I believe, to a fallacy to which philosophers and mathematicians are particularly subject. This is the assumption that as soon as a fact is presented to a mind all consequences of that fact spring into the mind simultaneously with it. It is a very useful assumption under many circumstances, but one too easily forgets that it is false. —Alan M. Turing [126]*

The theory of computing, created by Alan Turing, Alonzo Church, Kurt Gödel, and others in the 1930s, didn't only change civilization; it also had a lasting impact on philosophy. Indeed, clarifying philosophical issues was the original *point* of their work; the technological payoffs only came later! Today, it would be hard to imagine a serious discussion about (say) the philosophy of mind, the foundations of mathematics, or the prospects of machine intelligence that was uninformed by this revolution in human knowledge three-quarters of a century ago.

However, as computers became widely available starting in the 1960s, computer scientists increasingly came to see computability theory as not asking quite the right questions. For almost *all* the problems we actually want to solve turn out to be computable in Turing's sense; the real question is which problems are *efficiently* or *feasibly* computable. The latter question gave rise to a new field, called computational complexity theory (not to be confused with the "other" complexity theory, which studies complex systems such as cellular automata). Since the 1970s, computational complexity theory has witnessed some spectacular discoveries, which include NP-completeness, public-key cryptography, new types of mathematical proof (such as probabilistic, interactive, and zero-knowledge proofs), and the theoretical foundations of machine learning and quantum computation. To people who work on these topics, the work of Gödel and Turing may look in retrospect like just a warmup to the "big" questions about computation.

Because of this, I find it surprising that complexity theory has *not* influenced philosophy to anything like the extent computability theory has. The question arises: *why hasn't it?* Several possible answers spring to mind: maybe computability theory just had richer philosophical implications. (Though as we'll see, one can make a strong case for exactly the opposite.) Maybe complexity has essentially the *same* philosophical implications as computability, and computability got there first. Maybe outsiders are scared away from learning complexity theory by the "math barrier." Maybe the explanation is social: the world where Gödel, Turing, Wittgenstein, and Russell participated in the same intellectual conversation vanished with World War II; after that, theoretical computer science came to be driven by technology and lost touch with its philosophical origins. Maybe recent advances in complexity theory simply haven't had enough time to enter philosophical consciousness.

However, I suspect that part of the answer is just *complexity theorists' failure to communicate* what they can add to philosophy's conceptual arsenal. Hence this essay, whose modest goal is to help correct that failure, by surveying some aspects of complexity theory that might interest philosophers, as well as some philosophical problems that I think a complexity perspective can clarify.

To forestall misunderstandings, let me add a note of humility before going further. This essay will touch on many problems that philosophers have debated for generations, such as strong AI, the problem of induction, the relation between syntax and semantics, and the interpretation of quantum mechanics. *In none of these cases* will I claim that computational complexity theory "dissolves" the philosophical problem—only that it contributes useful perspectives and insights. I'll often explicitly mention philosophical puzzles that I think a complexity analysis either leaves untouched or else introduces itself. But even where I don't do so, one shouldn't presume that I think there are no such puzzles! Indeed, one of my hopes for this essay is that computer scientists, mathematicians, and other technical people who read it will come away with a better appreciation for the subtlety of some of the problems considered in modern analytic philosophy.[1]

1.1 What This Essay *Won't* Cover

I won't try to discuss every *possible* connection between computational complexity and philosophy, or even every connection that's already been made. A small number of philosophers have long invoked computational complexity ideas in their work; indeed, the "philpapers archive" lists 32 papers under the heading Computational Complexity.[2] The majority of those papers prove theorems about the computational complexities of various logical systems. Of the remaining papers, some use "computational complexity" in a different sense than I do—for example, to encompass

[1] When I use the word "philosophy" in this essay, I'll *mean* philosophy within the analytic tradition. I don't understand Continental or Eastern philosophy well enough to say whether they have any interesting connections with computational complexity theory.

[2] See philpapers.org/browse/computational-complexity

computability theory—and some invoke the *concept* of computational complexity, but no particular results from the *field* devoted to it. Perhaps the closest in spirit to this essay are the interesting articles by Cherniak [40] and Morton [97]. In addition, many writers have made some version of the observations in Section 4, about computational complexity and the Turing Test: see for example Block [30], Parberry [101], Levesque [87], and Shieber [116].

In deciding which connections to include in this essay, I adopted the following ground rules:

(1) The connection must involve a "properly philosophical" problem—for example, the justification for induction or the nature of mathematical knowledge—and not just a technical problem in logic or model theory.

(2) The connection must draw on *specific insights* from the field of computational complexity theory: not just the *idea* of complexity, or the *fact* that there exist hard problems.

There are many philosophically-interesting ideas in modern complexity theory that this essay mentions only briefly or not at all. One example is *pseudorandom generators* (see Goldreich [63]): functions that convert a short random "seed" into a long string of bits that, while not truly random, is so "random-looking" that no efficient algorithm can detect any regularities in it. While pseudorandom generators in this sense are not yet proved to exist,[3] there are many plausible candidates, and the belief that at least some of the candidates work is central to modern cryptography. (Section 7.1 will invoke the related concept of pseudorandom *functions*.) A second example is *fully homomorphic encryption*: an extremely exciting new class of methods, the first of which was announced by Gentry [60] in 2009, for performing arbitrary computations on encrypted data *without ever decrypting the data*. The output of such a computation will look like meaningless gibberish to the person who computed it, but it can nevertheless be understood (and even recognized as the correct output) by someone who knows the decryption key. What are the implications of pseudorandom generators for the foundations of probability, or of fully homomorphic encryption for debates about the semantic meaning of computations? I very much hope that this essay will inspire others to tackle these and similar questions.

Outside of computational complexity, there are at least three other major intersection points between philosophy and modern theoretical computer science. The first one is the *semantics of programming languages*, which has large and obvious connections to the philosophy of language.[4] The second is *distributed systems theory*, which provides both an application area and a rich source of examples for philosophical work on reasoning about knowledge (see Fagin et al. [53] and Stalnaker [123]). The third is *Kolmogorov complexity* (see Li and Vitányi [89]) which studies the *length* of the shortest computer program that achieves some functionality, disregarding time, memory, and other resources used by the program.[5]

In this essay, I won't discuss *any* of these connections, except in passing (for example, Section 5 touches on logics of knowledge in the context of the "logical omniscience problem," and Section 7 touches on Kolmogorov complexity in the context of PAC-learning). In defense of these omissions, let me offer four excuses. First, these other connections fall outside my stated topic. Second, they

[3] The conjecture that pseudorandom generators exist implies the $\mathsf{P} \neq \mathsf{NP}$ conjecture (about which more later), but might be even stronger: the converse implication is unknown.

[4] The *Stanford Encyclopedia of Philosophy* entry on "The Philosophy of Computer Science," plato.stanford.edu/entries/computer-science, devotes most of its space to this connection.

[5] A variant, "resource-bounded Kolmogorov complexity," *does* take time and memory into account, and is part of computational complexity theory proper.

would make this essay even longer than it already is. Third, I lack requisite background. And fourth, my impression is that philosophers—at least *some* philosophers—are already more aware of these other connections than they are of the computational complexity connections that I want to explain.

2 Complexity 101

Computational complexity theory is a huge, sprawling field; naturally this essay will only touch on small parts of it. Readers who want to delve deeper into the subject are urged to consult one of the many outstanding textbooks, such as those of Sipser [122], Papadimitriou [100], Moore and Mertens [95], Goldreich [62], or Arora and Barak [15]; or survey articles by Wigderson [133, 134], Fortnow and Homer [58], or Stockmeyer [124].

One might think that, once we know something is *computable*, whether it takes 10 seconds or 20 seconds to compute is obviously the concern of engineers rather than philosophers. But that conclusion would *not* be so obvious, if the question were one of 10 seconds versus $10^{10^{10}}$ seconds! And indeed, in complexity theory, the quantitative gaps we care about are usually so vast that one has to consider them qualitative gaps as well. Think, for example, of the difference between reading a 400-page book and reading *every possible* such book, or between writing down a thousand-digit number and counting to that number.

More precisely, complexity theory asks the question: how do the resources needed to solve a problem scale with some measure n of the problem size: "reasonably" (like n or n^2, say), or "unreasonably" (like 2^n or $n!$)? As an example, two n-digit integers can be multiplied using $\sim n^2$ computational steps (by the grade-school method), or even $\sim n \log n \log \log n$ steps (by more advanced methods [112]). Either method is considered efficient. By contrast, the fastest known method for the reverse operation—*factoring* an n-digit integer into primes—uses $\sim 2^{n^{1/3}}$ steps, which is considered inefficient.[6] Famously, this conjectured gap between the inherent difficulties of multiplying and factoring is the basis for most of the cryptography currently used on the Internet.

Theoretical computer scientists generally call an algorithm "efficient" if its running time can be upper-bounded by any polynomial function of n, and "inefficient" if its running time can be lower-bounded by any exponential function of n.[7] These criteria have the great advantage of theoretical convenience. While the exact complexity of a problem might depend on "low-level encoding details," such as whether our Turing machine has one or two memory tapes, or how the inputs are encoded as binary strings, where a problem falls on the polynomial/exponential dichotomy can be shown to be independent of almost all such choices.[8] Equally important are the *closure properties* of polynomial and exponential time: a polynomial-time algorithm that calls a polynomial-time subroutine still yields an overall polynomial-time algorithm, while a polynomial-

[6] This method is called the *number field sieve*, and the quoted running time depends on plausible but unproved conjectures in number theory. The best *proven* running time is $\sim 2^{\sqrt{n}}$. Both of these represent nontrivial improvements over the naïve method of trying all possible divisors, which takes $\sim 2^n$ steps. See Pomerance [105] for a good survey of factoring algorithms.

[7] In some contexts, "exponential" means c^n for some constant $c > 1$, but in most complexity-theoretic contexts it can also mean c^{n^d} for constants $c > 1$ and $d > 0$.

[8] This is not to say that *no* details of the computational model matter: for example, some problems are known to be solvable in polynomial time on a quantum computer, but *not* known to be solvable in polynomial time on a classical computer! But in my view, the fact that the polynomial/exponential distinction can "notice" a modelling choice of this magnitude is a feature of the distinction, not a bug.

time algorithm that calls an exponential-time subroutine (or vice versa) yields an exponential-time algorithm. There are also more sophisticated reasons why theoretical computer scientists focus on polynomial time (rather than, say, n^2 time or $n^{\log n}$ time); we'll explore some of those reasons in Section 5.1.

The polynomial/exponential distinction is open to obvious objections: an algorithm that took 1.00000001^n steps would be much faster in practice than an algorithm that took n^{10000} steps! Furthermore, there are many growth rates that fall between polynomial and exponential, such as $n^{\log n}$ and $2^{2^{\sqrt{\log n}}}$. But empirically, polynomial-time *turned out* to correspond to "efficient in practice," and exponential-time to "inefficient in practice," so often that complexity theorists became comfortable making the identification. *Why* the identification works is an interesting question in its own right, one to which we will return in Section 12.

A priori, insisting that programs terminate after reasonable amounts of time, that they use reasonable amounts of memory, etc. might sound like relatively-minor amendments to Turing's notion of computation. In practice, though, these requirements lead to a theory with a completely different character than computability theory. Firstly, complexity has much closer connections with the *sciences*: it lets us pose questions about (for example) evolution, quantum mechanics, statistical physics, economics, or human language acquisition that would be meaningless from a computability standpoint (since *all* the relevant problems are computable). Complexity also differs from computability in the diversity of mathematical *techniques* used: while initially complexity (like computability) drew mostly on mathematical logic, today it draws on probability, number theory, combinatorics, representation theory, Fourier analysis, and nearly every other subject about which yellow books are written. Of course, this contributes not only to complexity theory's depth but also to its perceived inaccessibility.

In this essay, I'll argue that complexity theory has direct relevance to major issues in philosophy, including syntax and semantics, the problem of induction, and the interpretation of quantum mechanics. Or that, at least, whether complexity theory *does or does not* have such relevance is an important question for philosophy! My personal view is that complexity will ultimately prove *more* relevant to philosophy than computability was, precisely because of the rich connections with the sciences mentioned earlier.

3 The Relevance of Polynomial Time

Anyone who doubts the importance of the polynomial/exponential distinction needs only ponder how many basic intuitions in math, science, and philosophy already implicitly rely on that distinction. In this section I'll give three examples.

3.1 The Entscheidungsproblem Revisited

The *Entscheidungsproblem* was the dream, enunciated by David Hilbert in the 1920s, of designing a mechanical procedure to determine the truth or falsehood of any well-formed mathematical statement. According to the usual story, Hilbert's dream was irrevocably destroyed by the work of Gödel, Church, and Turing in the 1930s. First, the Incompleteness Theorem showed that no recursively-axiomatizable formal system can encode *all and only* the true mathematical statements. Second, Church's and Turing's results showed that, even if we settle for an incomplete system F, there is *still* no mechanical procedure to sort mathematical statements into the three categories "provable in F," "disprovable in F," and "undecidable in F."

However, there is a catch in the above story, which was first pointed out by Gödel himself, in a 1956 letter to John von Neumann that has become famous in theoretical computer science since its rediscovery in the 1980s (see Sipser [121] for an English translation). Given a formal system F (such as Zermelo-Fraenkel set theory), Gödel wrote, consider the problem of deciding whether a mathematical statement S has a proof in F *with n symbols or fewer.* Unlike Hilbert's original problem, this "truncated Entscheidungsproblem" is clearly decidable. For, if nothing else, we could always just program a computer to search through all 2^n possible bit-strings with n symbols, and check whether any of them encodes a valid F-proof of S. The issue is "merely" that this approach takes an astronomical amount of time: if $n = 1000$ (say), then the universe will have degenerated into black holes and radiation long before a computer can check 2^{1000} proofs!

But as Gödel also pointed out, it's far from obvious how to *prove* that there isn't a much better approach: an approach that would avoid brute-force search, and find proofs of size n in time polynomial in n. Furthermore:

> If there actually were a machine with [running time] $\sim Kn$ (or even only with $\sim Kn^2$) [for some constant K independent of n], 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 [added in a footnote: apart from the postulation of axioms] replaced by machines. One would indeed have to simply select an n so large that, if the machine yields no result, there would then also be no reason to think further about the problem.

If we replace the "$\sim Kn$ or $\sim Kn^2$" in Gödel's challenge by $\sim Kn^c$ for an *arbitrary* constant c, then we get precisely what computer science now knows as the P versus NP problem. Here P (Polynomial-Time) is, roughly speaking, the class of all computational problems that are solvable by a polynomial-time algorithm. Meanwhile, NP (Nondeterministic Polynomial-Time) is the class of computational problems for which a solution can be *recognized* in polynomial time, even though a solution might be very hard to find.[9] (Think, for example, of factoring a large number, or solving a jigsaw or Sudoku puzzle.) Clearly $\mathsf{P} \subseteq \mathsf{NP}$, so the question is whether the inclusion is strict. If $\mathsf{P} = \mathsf{NP}$, then the ability to *check* the solutions to puzzles efficiently would imply the ability to *find* solutions efficiently. An analogy would be if anyone able to *appreciate* a great symphony could also compose one themselves!

Given the intuitive implausibility of such a scenario, essentially all complexity theorists proceed (reasonably, in my opinion) on the assumption that $\mathsf{P} \neq \mathsf{NP}$, even if they publicly claim open-mindedness about the question. Proving or disproving $\mathsf{P} \neq \mathsf{NP}$ is one of the seven million-dollar Clay Millennium Prize Problems[10] (alongside the Riemann Hypothesis, the Poincaré Conjecture

[9] Contrary to a common misconception, NP does *not* stand for "Non-Polynomial"! There *are* computational problems that are *known* to require more than polynomial time (see Section 10), but the NP problems are not among those. Indeed, the classes NP and "Non-Polynomial" have a nonempty intersection exactly if $\mathsf{P} \neq \mathsf{NP}$.

For detailed definitions of P, NP, and several hundred other complexity classes, see my Complexity Zoo website: www.complexityzoo.com.

[10] For more information see www.claymath.org/millennium/P_vs_NP/

My own view is that P versus NP is manifestly the *most important* of the seven problems! For if $\mathsf{P} = \mathsf{NP}$, then by Gödel's argument, there is an excellent chance that we could program our computers to solve the other six problems as well.

proved in 2002 by Perelman, etc.), which should give some indication of the problem's difficulty.[11]

Now return to the problem of whether a mathematical statement S has a proof with n symbols or fewer, in some formal system F. A suitable formalization of this problem is easily seen to be in NP. For *finding* a proof might be intractable, but if we're *given* a purported proof, we can certainly check in time polynomial in n whether each line of the proof follows by a simple logical manipulation of previous lines. Indeed, this problem turns out to be NP-*complete*, which means that it belongs to an enormous class of NP problems, first identified in the 1970s, that "capture the entire difficulty of NP." A few other examples of NP-complete problems are Sudoku and jigsaw puzzles, the Traveling Salesperson Problem, and the satisfiability problem for propositional formulas.[12] Asking whether $\mathsf{P} = \mathsf{NP}$ is equivalent to asking whether *any* NP-complete problem can be solved in polynomial time, and is also equivalent to asking whether *all* of them can be.

In modern terms, then, Gödel is saying that if $\mathsf{P} = \mathsf{NP}$, then whenever a theorem had a proof of reasonable length, we could *find* that proof in a reasonable amount of time. In such a situation, we might say that "for all practical purposes," Hilbert's dream of mechanizing mathematics had prevailed, despite the undecidability results of Gödel, Church, and Turing. If you accept this, then it seems fair to say that until P versus NP is solved, the story of Hilbert's Entscheidungsproblem—its rise, its fall, and the consequences for philosophy—is not yet over.

3.2 Evolvability

Creationists often claim that Darwinian evolution is as vacuous an explanation for complex adaptations as "a tornado assembling a 747 airplane as it passes through a junkyard." Why is this claim false? There are several related ways of answering the question, but to me, one of the most illuminating is the following. In principle, one *could* see a 747 assemble itself in a tornado-prone junkyard—but before that happened, one would need to wait for an expected number of tornadoes that grew *exponentially* with the number of pieces of self-assembling junk. (This is similar to how, in thermodynamics, n gas particles in a box *will* eventually congregate themselves in one corner of the box, but only after $\sim c^n$ time for some constant c.) By contrast, evolutionary processes can often be observed in simulations—and in some cases, even proved theoretically—to find interesting solutions to optimization problems after a number of steps that grows only *polynomially* with the number of variables.

Interestingly, in a 1972 letter to Hao Wang (see [130, p. 192]), Kurt Gödel expressed his own doubts about evolution as follows:

> I believe that mechanism in biology is a prejudice of our time which will be disproved. In this case, one disproof, in my opinion, will consist in a mathematical theorem to the effect that the formation within geological time of a human body by the laws of

[11] One might ask: can we *explain* what makes the $\mathsf{P} \neq \mathsf{NP}$ problem so hard, rather than just pointing out that many smart people have tried to solve it and failed? After four decades of research, we *do* have partial explanations for the problem's difficulty, in the form of formal "barriers" that rule out large classes of proof techniques. Three barriers identified so far are *relativization* [21] (which rules out diagonalization and other techniques with a "computability" flavor), *algebrization* [8] (which rules out diagonalization even when combined with the main non-relativizing techniques known today), and *natural proofs* [108] (which shows that many "combinatorial" techniques, if they worked, could be turned around to get faster algorithms to distinguish random from pseudorandom functions).

[12] By contrast, and contrary to a common misconception, there is strong evidence that factoring integers is *not* NP-complete. It is known that if $\mathsf{P} \neq \mathsf{NP}$, then there are NP problems that are neither in P nor NP-complete [85], and factoring is one candidate for such a problem. This point will become relevant when we discuss quantum computing.

physics (or any other laws of similar nature), starting from a random distribution of the elementary particles and the field, is as unlikely as the separation by chance of the atmosphere into its components.

Personally, I see no reason to accept Gödel's intuition on this subject over the consensus of modern biology! But pay attention to Gödel's characteristically-careful phrasing. He does not ask whether evolution can *eventually* form a human body (for he knows that it can, given exponential time); instead, he asks whether it can do so on a "merely" geological timescale. Just as Gödel's letter to von Neumann anticipated the P versus NP problem, so Gödel's letter to Wang might be said to anticipate a recent effort, by the celebrated computer scientist Leslie Valiant, to construct a quantitative "theory of evolvability" [128]. Building on Valiant's earlier work in computational learning theory (discussed in Section 7), evolvability tries to formalize and answer questions about the *speed* of evolution. For example: "what sorts of adaptive behaviors can evolve, with high probability, after only a polynomial number of generations? what sorts of behaviors can be learned in polynomial time, but *not* via evolution?" While there are some interesting early results, it should surprise no one that evolvability is nowhere close to being able to calculate, from first principles, whether four billion years is a "reasonable" or "unreasonable" length of time for the human brain to evolve out of the primordial soup.

As I see it, this difficulty reflects a general point about Gödel's "evolvability" question. Namely, even *supposing* Gödel was right, that the mechanistic worldview of modern biology was "as unlikely as the separation by chance of the atmosphere into its components," computational complexity theory seems hopelessly far from being able to *prove* anything of the kind! In 1972, one could have argued that this merely reflected the subject's newness: no one had thought terribly deeply yet about how to prove *lower bounds* on computation time. But by now, people *have* thought deeply about it, and have identified huge obstacles to proving even such "obvious" and well-defined conjectures as $\mathsf{P} \neq \mathsf{NP}$.[13] (Section 4 will make a related point, about the difficulty of proving nontrivial lower bounds on the time or memory needed by a computer program to pass the Turing Test.)

3.3 Known Integers

My last example of the philosophical relevance of the polynomial/exponential distinction concerns the concept of "knowledge" in mathematics.[14] As of 2011, the "largest known prime number," as reported by GIMPS (the Great Internet Mersenne Prime Search),[15] is $p := 2^{43112609} - 1$. But on reflection, what do we mean by saying that p is "known"? Do we mean that, if we desired, we could literally print out its decimal digits (using about $30,000$ pages)? That seems like too restrictive a criterion. For, given a positive integer k together with a proof that $q = 2^k - 1$ was prime, I doubt most mathematicians would hesitate to call q a "known" prime,[16] even if k were so large that printing out its decimal digits (or storing them in a computer memory) were beyond the

[13] Admittedly, one might be able to prove that *Darwinian natural selection* would require exponential time to produce some functionality, without thereby proving that *any* algorithm would require exponential time.

[14] This section was inspired by a question of A. Rupinski on the website *MathOverflow*. See mathoverflow.net/questions/62925/philosophical-question-related-to-largest-known-primes/

[15] www.mersenne.org

[16] Here we implicitly mean: "known" by the entire mathematical community, rather than known by some particular individual when asked about it. Section 5 will consider the latter type of knowledge.

Earth's capacity. Should we call $2^{2^{1000}}$ an "unknown power of 2," just because it has too many decimal digits to list before the Sun goes cold?

All that should *really* matter, one feels, is that

(a) the expression '$2^{43112609}-1$' picks out a unique positive integer, and

(b) that integer has been proven (in this case, via computer, of course) to be prime.

But wait! If those are the criteria, then why can't we immediately beat the largest-known-prime record, like so?

$$p' = \text{The first prime larger than } 2^{43112609}-1.$$

Clearly p' exists, it is unambiguously defined, and it is prime. If we want, we can even write a program that is guaranteed to find p' and output its decimal digits, using a number of steps that can be upper-bounded *a priori*.[17] Yet our intuition stubbornly insists that $2^{43112609}-1$ is a "known" prime in a sense that p' is not. Is there any principled basis for such a distinction?

The clearest basis that I can suggest is the following. We know an algorithm that takes as input a positive integer k, and that outputs the decimal digits of $p = 2^k - 1$ *using a number of steps that is polynomial—indeed, linear—in the number of digits of* p. But we do not know any similarly-efficient algorithm that provably outputs the first prime larger than $2^k - 1$.[18]

3.4 Summary

The point of these examples was to illustrate that, beyond its utility for theoretical computer science, the polynomial/exponential gap is also a fertile territory for philosophy. I think of the polynomial/exponential gap as occupying a "middle ground" between two other sorts of gaps: on the one hand, small quantitative gaps (such as the gap between n steps and $2n$ steps); and on the other hand, the gap between a finite number of steps and an infinite number. The trouble with small quantitative gaps is that they are too sensitive to "mundane" modeling choices and the details of technology. But the gap between finite and infinite has the opposite problem: it is serenely *in*sensitive to distinctions that we actually care about, such as that between finding a solution and verifying it, or between classical and quantum physics.[19] The polynomial/exponential gap avoids both problems.

4 Computational Complexity and the Turing Test

Can a computer think? For almost a century, discussions about this question have often conflated two issues. The first is the "metaphysical" issue:

[17] For example, one could use *Chebyshev's Theorem* (also called *Bertrand's Postulate*), which says that for all $N > 1$ there exists a prime between N and $2N$.

[18] *Cramér's Conjecture* states that the spacing between two consecutive n-digit primes never exceeds $\sim n^2$. This conjecture appears staggeringly difficult: even assuming the Riemann Hypothesis, it is only known how to deduce the much weaker upper bound $\sim n2^{n/2}$. But interestingly, if Cramér's Conjecture is proved, expressions like "the first prime larger than $2^k - 1$" will *then* define "known primes" according to my criterion.

[19] In particular, it is easy to check that the set of *computable* functions does not depend on whether we define computability with respect to a classical or a quantum Turing machine, or a deterministic or nondeterministic one. At most, these choices can change a Turing machine's running time by an exponential factor, which is irrelevant for computability theory.

> Supposing a computer program passed the Turing Test (or as strong a variant of the Turing Test as one wishes to define),[20] would we be right to ascribe to it "consciousness," "qualia," "aboutness," "intentionality," "subjectivity," "personhood," or whatever other charmed status we wish to ascribe to other humans and to ourselves?

The second is the "practical" issue:

> Could a computer program that passed (a strong version of) the Turing Test actually be written? Is there some fundamental reason why it couldn't be?

Of course, it was precisely in an attempt to separate these issues that Turing proposed the Turing Test in the first place! But despite his efforts, a familiar feature of anti-AI arguments to this day is that they first assert AI's metaphysical impossibility, and then try to bolster that position with claims about AI's practical difficulties. "Sure," they say, "a computer program might mimic a few minutes of witty banter, but unlike a human being, it would never show fear or anger or jealousy, or compose symphonies, or grow old, or fall in love..."

The obvious followup question—and what if a program *did* do all those things?—is often left unasked, or else answered by listing more things that a computer program could self-evidently never do. Because of this, I suspect that many people who *say* they consider AI a metaphysical impossibility, really consider it only a practical impossibility: they simply have not carried the requisite thought experiment far enough to see the difference between the two.[21] Incidentally, this is as clear-cut a case as I know of where people would benefit from studying more philosophy!

Thus, the anti-AI arguments that interest me most have always been the ones that target the practical issue from the outset, by proposing empirical "sword-in-the-stone tests" (in Daniel Dennett's phrase [46]) that it is claimed humans can pass but computers cannot. The most famous such test is probably the one based on Gödel's Incompleteness Theorem, as proposed by John Lucas [91] and elaborated by Roger Penrose in his books *The Emperor's New Mind* [102] and *Shadows of the Mind* [103].

Briefly, Lucas and Penrose argued that, according to the Incompleteness Theorem, one thing that a computer making deductions via fixed formal rules can never do is to "see" the consistency of its own rules. Yet this, they assert, is something that human mathematicians *can* do, via some sort of intuitive perception of Platonic reality. Therefore humans (or at least, human mathematicians!) can never be simulated by machines.

Critics pointed out numerous holes in this argument,[22] to which Penrose responded at length in *Shadows of the Mind*, in my opinion unconvincingly. However, even *before* we analyze some

[20] The Turing Test, proposed by Alan Turing [126] in 1950, is a test where a human judge interacts with either another human or a computer conversation program, by typing messages back and forth. The program "passes" the Test if the judge can't reliably distinguish the program from the human interlocutor.

By a "strong variant" of the Turing Test, I mean that besides the usual teletype conversation, one could add additional tests requiring vision, hearing, touch, smell, speaking, handwriting, facial expressions, dancing, playing sports and musical instruments, etc.—even though many perfectly-intelligent *humans* would then be unable to pass the tests!

[21] One famous exception is John Searle [113], who has made it clear that, if (say) his best friend turned out to be controlled by a microchip rather than a brain, then he would regard his friend as never having been a person at all.

[22] See Dennett [46] and Chalmers [37] for example. To summarize:

(1) Why should we assume a computer operates within a knowably-sound formal system? If we grant a computer the same freedom to make occasional mistakes that we grant humans, then the Incompleteness Theorem is no longer relevant.

proposed sword-in-the-stone test, it seems to me that there is a much more basic question. Namely, what does one even *mean* in saying one has a task that "humans can perform but computers cannot"?

4.1 The Lookup-Table Argument

There is a fundamental difficulty here, which was noticed by others in a slightly different context [30, 101, 87, 116]. Let me first explain the difficulty, and then discuss the difference between my argument and the previous ones.

In practice, people judge each other to be conscious after interacting for a very short time, perhaps as little as a few seconds. This suggests that we can put a finite upper bound—to be generous, let us say 10^{20}—on the number of bits of information that two people A and B would ever realistically exchange, before A had amassed enough evidence to conclude B was conscious.[23] Now imagine a lookup table that stores every possible history H of A and B's conversation, and next to H, the action $f_B(H)$ that B *would* take next given that history. Of course, like Borges' Library of Babel, the lookup table would consist almost entirely of meaningless nonsense, and it would also be much too large to fit inside the observed universe. But all that matters for us is that the lookup table would be *finite*, by the assumption that there is a finite upper bound on the conversation length. This implies that the function f_B is computable (indeed, it can be recognized by a finite automaton!). From these simple considerations, we conclude that if there *is* a fundamental obstacle to computers passing the Turing Test, then it is not to be found in computability theory.[24]

In *Shadows of the Mind* [103, p. 83], Penrose recognizes this problem, but gives a puzzling and unsatisfying response:

> One could equally well envisage computers that contain nothing but lists of totally false mathematical 'theorems,' or lists containing random jumbles of truths and falsehoods. How are we to tell which computer to trust? The arguments that I am trying to make here do not say that an effective simulation of the output of conscious human activity

(2) Why should we assume that human mathematicians have "direct perception of Platonic reality"? Human mathematicians (such as Frege) have been wrong before about the consistency of formal systems.

(3) A computer could, of course, be programmed to output "I believe that formal system F is consistent"—and even to output answers to various followup questions about *why* it believes this. So in arguing that such affirmations "wouldn't really count" (because they wouldn't reflect "true understanding"), AI critics such as Lucas and Penrose are forced to retreat from their vision of an empirical "sword-in-the-stone test," and fall back on other, unspecified criteria related to the AI's internal structure. But then *why put the sword in the stone in the first place?*

[23] People interacting over the Internet, via email or instant messages, regularly judge each other to be humans rather than spam-bots after exchanging a much smaller number of bits! In any case, cosmological considerations suggest an upper bound of roughly 10^{122} bits in any observable process [34].

[24] Some readers might notice a tension here: I explained in Section 2 that complexity theorists care about the *asymptotic* behavior as the problem size n goes to infinity. So why am I now saying that, for the purposes of the Turing Test, we should restrict attention to finite values of n such as 10^{20}? There are two answers to this question. The first is that, in contrast to mathematical problems like the factoring problem or the halting problem, it is unclear whether it even makes *sense* to generalize the Turing Test to arbitrary conversation lengths: for the Turing Test is defined in terms of human beings, and human conversational capacity is finite. The second answer is that, to whatever extent it *does* make sense to generalize the Turing Test to arbitrary conversation lengths n, I *am* interested in whether the asymptotic complexity of passing the test grows polynomially or exponentially with n (as the remainder of the section explains).

> (here mathematics) is impossible, since purely by chance the computer might 'happen' to get it right—even without any understanding whatsoever. But the odds against this are absurdly enormous, and the issues that are being addressed here, namely how one decides *which* mathematical statements are true and which are false, are not even being touched...

The trouble with this response is that it amounts to a retreat from the sword-in-the-stone test, back to murkier internal criteria. If, in the end, we are going to have to look inside the computer anyway to determine whether it truly "understands" its answers, *then why not dispense with computability theory from the beginning?* For computability theory only addresses whether or not Turing machines *exist* to solve various problems, and we have already seen that that is not the relevant issue.

To my mind, there is *one* direction that Penrose could take from this point to avoid incoherence—though disappointingly, it is not the direction he chooses. Namely, he could point out that, while the lookup table "works," it requires computational resources that grow exponentially with the length of the conversation! This would lead to the following speculation:

> (*) *Any* computer program that passed the Turing Test would need to be exponentially-inefficient in the length of the test—as measured in some resource such as time, memory usage, or the number of bits needed to write the program down. In other words, the astronomical lookup table is essentially the best one can do.[25]

If true, speculation (*) would do what Penrose wants: it would imply that the human brain can't even be *simulated* by computer, within the resource constraints of the observable universe. Furthermore, unlike the earlier computability claim, (*) has the advantage of not being trivially false!

On the other hand, to put it mildly, (*) is not trivially *true* either. For AI proponents, the lack of compelling evidence for (*) is hardly surprising. After all, if you believe that the brain *itself* is basically an efficient,[26] classical Turing machine, then you have a simple explanation for why no one has proved that the brain can't be simulated by such a machine! However, complexity theory also makes it clear that, *even if we supposed (*) held*, there would be little hope of *proving* it in our current state of mathematical knowledge. After all, we can't even prove plausible, well-defined conjectures such as $\mathsf{P} \neq \mathsf{NP}$.

4.2 Relation to Previous Work

As mentioned before, I'm far from the first person to ask about the *computational resources* used in passing the Turing Test, and whether they scale polynomially or exponentially with the conversation length. While many writers ignore this crucial distinction, Block [30], Parberry [101], Levesque

[25] As Gil Kalai pointed out to me, one could speculate instead that an efficient computer program *exists* to pass the Turing Test, but that *finding* such a program would require exponential computational resources. In that situation, the human brain could indeed be simulated efficiently by a computer program, but maybe not by a program that humans could ever *write*!

[26] Here, by a Turing machine M being "efficient," we mean that M's running time, memory usage, and program size are modest enough that there is no real problem of principle understanding how M could be simulated by a classical physical system consisting of $\sim 10^{11}$ neurons and $\sim 10^{14}$ synapses. For example, a Turing machine containing a lookup table of size $10^{10^{20}}$ would not be efficient in this sense.

[87], Shieber [116], and several others all discussed it explicitly. The main difference is that the previous discussions took place in the context of Searle's Chinese Room argument [113].

Briefly, Searle proposed a thought experiment—the details don't concern us here—purporting to show that a computer program could pass the Turing Test, even though the program manifestly lacked anything that a reasonable person would call "intelligence" or "understanding." (Indeed, Searle argues that *no* physical system can understand anything "purely by virtue of" the computations that it implements.) In response, many critics said that Searle's argument was deeply misleading, because it implicitly encouraged us to imagine a computer program that was *simplistic* in its internal operations—something like the giant lookup table described in Section 4.1. And while it was true, the critics went on, that a giant lookup table wouldn't "truly understand" its responses, that point is also *irrelevant*. For the giant lookup table is a philosophical fiction anyway: something that can't even fit in the observable universe! If we instead imagine a *compact, efficient* computer program passing the Turing Test, then the situation changes drastically. For now, in order to *explain* how the program can be so compact and efficient, we'll need to posit that the program includes representations of abstract concepts, capacities for learning and reasoning, and all sorts of other internal furniture that we would expect to find in a mind.

Personally, I find this response to Searle extremely interesting—since if correct, it suggests that the distinction between polynomial and exponential complexity has *metaphysical* significance. According to this response, an exponential-sized lookup table that passed the Turing Test would not be sentient (or conscious, intelligent, self-aware, etc.), but a polynomially-bounded program with exactly the same input/output behavior *would* be sentient. Furthermore, the latter program would be sentient *because* it was polynomially-bounded.

Yet, as much as that criterion for sentience flatters my complexity-theoretic pride, I find myself reluctant to take a position on such a weighty matter. My point, in Section 4.1, was a simpler and (hopefully) less controversial one: namely, that if you want to claim that passing the Turing Test is *flat-out impossible*, then like it or not, you *must* talk about complexity rather than just computability. In other words, the previous writers [30, 101, 87, 116] and I are all interested in the computational resources needed to pass a Turing Test of length n, but for different reasons. Where others invoked complexity considerations to argue with Searle about the metaphysical question, I'm invoking them to argue with Penrose about the practical question.

4.3 Can Humans Solve NP-Complete Problems Efficiently?

In that case, what can we actually *say* about the practical question? Are there any reasons to accept the claim I called (*)—the claim that humans are *not* efficiently simulable by Turing machines? In considering this question, we're immediately led to some speculative possibilities. So for example, *if* it turned out that humans could solve arbitrary instances of NP-complete problems in polynomial time, then that would certainly constitute excellent empirical evidence for (*).[27] However, despite occasional claims to the contrary, I personally see no reason to believe that humans *can* solve NP-complete problems in polynomial time, and excellent reasons to believe the opposite.[28] Recall, for

[27] And amusingly, if we could solve NP-complete problems, then we'd presumably find it much easier to prove that computers *couldn't* solve them!

[28] Indeed, it is not even clear to me that we should think of humans as being able to solve all P problems efficiently, let alone NP-complete problems! Recall that P is the class of problems that *are* solvable in polynomial time by a deterministic Turing machine. Many problems are known to belong to P for quite sophisticated reasons: two examples are testing whether a number is prime (though not factoring it!) [9] and testing whether a graph has a

example, that the integer factoring problem is in NP. Thus, if humans could solve NP-complete problems, then presumably we ought to be able to factor enormous numbers as well! But factoring does not exactly seem like the most promising candidate for a sword-in-the-stone test: that is, a task that's easy for humans but hard for computers. As far as anyone knows today, factoring is hard for humans and (classical) computers *alike*, although with a definite advantage on the computers' side!

The basic point can hardly be stressed enough: when complexity theorists talk about "intractable" problems, they generally mean mathematical problems that all our experience leads us to believe are at least as hard for humans as for computers. This suggests that, *even if* humans were not efficiently simulable by Turing machines, the "direction" in which they were hard to simulate would almost certainly be different from the directions usually considered in complexity theory. I see two (hypothetical) ways this could happen.

First, the tasks that humans were uniquely good at—like painting or writing poetry—could be *incomparable* with mathematical tasks like solving NP-complete problems, in the sense that neither was efficiently reducible to the other. This would mean, in particular, that there could be no polynomial-time algorithm even to *recognize* great art or poetry (since if such an algorithm existed, then the task of *composing* great art or poetry would be in NP). Within complexity theory, it's known that there exist pairs of problems that are incomparable in this sense. As one plausible example, no one currently knows how to reduce the simulation of quantum computers to the solution of NP-complete problems *or* vice versa.

Second, humans could have the ability to solve interesting *special cases* of NP-complete problems faster than any Turing machine. So for example, even if computers were better than humans at factoring large numbers or at solving randomly-generated Sudoku puzzles, humans might still be better at search problems with "higher-level structure" or "semantics," such as proving Fermat's Last Theorem or (ironically) designing faster computer algorithms. Indeed, even in limited domains such as puzzle-solving, while computers can examine solutions millions of times faster, humans (for now) are vastly better at noticing *global patterns* or *symmetries* in the puzzle that make a solution either trivial or impossible. As an amusing example, consider the *Pigeonhole Principle*, which says that $n+1$ pigeons can't be placed into n holes, with at most one pigeon per hole. It's not hard to construct a propositional Boolean formula φ that encodes the Pigeonhole Principle for some fixed value of n (say, 1000). However, if you then feed φ to current Boolean satisfiability algorithms, they'll assiduously set to work trying out possibilities: "let's see, if I put *this* pigeon here, and *that* one there ... darn, it *still* doesn't work!" And they'll continue trying out possibilities for an exponential number of steps, oblivious to the "global" reason why the goal can never be achieved. Indeed, beginning in the 1980s, the field of *proof complexity*—a close cousin of computational complexity—has been able to show that large classes of algorithms *require* exponential time to prove the Pigeonhole Principle and similar propositional tautologies (see Beame and Pitassi [24] for a survey).

Still, if we want to build our sword-in-the-stone test on the ability to detect "higher-level patterns" in combinatorial search problems, then the burden is on us to explain what we *mean* by higher-level patterns, and why we think that *no* polynomial-time Turing machine—even much

perfect matching. In principle, of course, a human could laboriously run the polynomial-time algorithms for such problems using pencil and paper. But is the use of pencil and paper legitimate, where use of a computer would *not* be? What is the computational power of the "unaided" human intellect? Recent work of Drucker [51], which shows how to use a stock photography collection to increase the "effective memory" available for mental calculations, provides a fascinating empirical perspective on these questions.

more sophisticated ones than we can imagine today—could ever detect those patterns as well. For an initial attempt to understand NP-complete problems from a cognitive science perspective, see Baum [22].

4.4 Summary

My conclusion is that, if you oppose the possibility of AI in principle, then either

(i) you can take the "metaphysical route" (as Searle [113] does with the Chinese Room), conceding the possibility of a computer program passing every conceivable empirical test for intelligence, but arguing that that isn't enough, or

(ii) you can conjecture an *astronomical lower bound on the resources* needed either to run such a program or to write it in the first place—but here there is little question of proof for the foreseeable future.

Crucially, because of the lookup-table argument, one option you do *not* have is to assert the flat-out impossibility of a computer program passing the Turing Test, with no mention of quantitative complexity bounds.

5 The Problem of Logical Omniscience

Giving a formal account of *knowledge* is one of the central concerns in modern analytic philosophy; the literature is too vast even to survey here (though see Fagin et al. [53] for a computer-science-friendly overview). Typically, formal accounts of knowledge involve conventional "logical" axioms, such as

- If you know P and you know Q, then you also know $P \wedge Q$

supplemented by "modal" axioms having to do with knowledge itself, such as

- If you know P, then you also know that you know P
- If you don't know P, then you know that you don't know P[29]

While the details differ, what most formal accounts of knowledge have in common is that they treat an agent's knowledge as *closed* under the application of various deduction rules like the ones above. In other words, agents are considered *logically omniscient*: if they know certain facts, then they also know all possible logical consequences of those facts.

Sadly and obviously, no mortal being has ever attained or even approximated this sort of omniscience (recall the Turing quote from the beginning of Section 1). So for example, I can know the rules of arithmetic without knowing Fermat's Last Theorem, and I can know the rules of chess without knowing whether White has a forced win. Furthermore, the difficulty is *not* (as sometimes claimed) limited to a few domains, such as mathematics and games. As pointed out by Stalnaker [123], if we assumed logical omniscience, then we couldn't account for *any* contemplation of facts

[29] Not surprisingly, these particular axioms have engendered controversy: they leave no possibility for Rumsfeldian "unknown unknowns," or (a category that he left out) "unknown knowns."

already known to us—and thus, for the main activity and one of the main subjects of philosophy itself!

We can now loosely state what Hintikka [72] called the *problem of logical omniscience*:

> Can we give some formal account of "knowledge" able to accommodate people learning new things without leaving their armchairs?

Of course, one vacuous "solution" would be to declare that your knowledge is simply a list of all the true sentences[30] that you "know"—and that, if the list happens not to be closed under logical deductions, so be it! But this "solution" is no help at all at explaining *how* or *why* you know things. Can't we do better?

Intuitively, we want to say that your "knowledge" consists of various facts ("grass is green"), together with *some* simple logical consequences of those facts ("grass is not pink"), but not necessarily *all* the consequences, and certainly not all consequences that involve difficult mathematical reasoning. Unfortunately, as soon as we try to formalize this idea, we run into problems.

The most obvious problem is the lack of a sharp boundary between the facts you know right away, and those you "could" know, but only after significant thought. (Recall the discussion of "known primes" from Section 3.3.) A related problem is the lack of a sharp boundary between the facts you know "only if asked about them," and those you know even if you're *not* asked. Interestingly, these two boundaries seem to cut across each other. For example, while you've probably already encountered the fact that 91 is composite, it might take you some time to remember it; while you've probably *never* encountered the fact that 83190 is composite, once asked you can probably assent to it immediately.

But as discussed by Stalnaker [123], there's a third problem that seems much more serious than either of the two above. Namely, you might "know" a particular fact if asked about it one way, but not if asked in a different way! To illustrate this, Stalnaker uses an example that we can recognize immediately from the discussion of the P versus NP problem in Section 3.1. If I asked you whether $43 \times 37 = 1591$, you could probably answer easily (e.g., by using $(40+3)(40-3) = 40^2 - 3^2$). On the other hand, if I instead asked you what the prime factors[31] of 1591 were, you probably *couldn't* answer so easily.

> But the answers to the two questions have the same content, even on a very fine-grained notion of content. Suppose that we fix the threshold of accessibility so that the information that 43 and 37 are the prime factors of 1591 is accessible in response to the second question, but not accessible in response to the first. Do you know what the prime factors of 1591 are or not? ... Our problem is that we are not just trying to say what an agent would know upon being asked certain questions; rather, we are trying to use the facts about an agent's question answering capacities in order to get at what the agent knows, even if the questions are not asked. [123, p. 253]

To add another example: does a typical four-year-old child "know" that addition of reals is commutative? Certainly not if we asked her in those words—and if we tried to *explain* the words, she probably wouldn't understand us. Yet if we showed her a stack of books, and asked her

[30] If we don't require the sentences to be *true*, then presumably we're talking about *belief* rather than *knowledge*.

[31] If you lack the *concept* of a prime factor, then I could simply ask instead for two 2-digit numbers that yield 1591 when multiplied.

whether she could make the stack higher by shuffling the books, she probably wouldn't make a mistake that involved imagining addition was non-commutative. In that sense, we might say she already "implicitly" knows what her math classes will later make explicit.

In my view, these and other examples strongly suggest that only a small part of what we mean by "knowledge" is knowledge about the truth or falsehood of individual propositions. And crucially, this remains so even if we restrict our attention to "purely verbalizable" knowledge—indeed, *knowledge used for answering factual questions*—and not (say) knowledge of how to ride a bike or swing a golf club, or knowledge of a person or a place.[32] Many everyday uses of the word "know" support this idea:

> Do you know calculus?
> Do you know Spanish?
> Do you know the rules of bridge?

Each of the above questions could be interpreted as asking: *do you possess an internal algorithm, by which you can answer a large (and possibly-unbounded) set of questions of some form?* While this is rarely made explicit, the examples of this section and of Section 3.3 suggest adding the proviso: *...answer in a reasonable amount of time?*

But suppose we accept that "knowing how" (or "knowing a good algorithm for") is a more fundamental concept than "knowing that." How does that help us *at all* in solving the logical omniscience problem? You might worry that we're right back where we started. After all, if we try to give a formal account of "knowing how," then just like in the case of "knowing that," it will be tempting to write down axioms like the following:

> If you know how to compute $f(x)$ and $g(x)$ efficiently, then you also know how to compute $f(x)+g(x)$ efficiently.

Naturally, we'll then want to take the logical closure of those axioms. But then, before we know it, won't we have conjured into our imaginations a computationally-omniscient superbeing, who could efficiently compute anything at all?

5.1 The Cobham Axioms

Happily, the above worry turns out to be unfounded. We *can* write down reasonable axioms for "knowing how to compute efficiently," and then *go ahead and take the closure of those axioms*, without getting the unwanted consequence of computational omniscience. Explaining this point will involve a digression into an old and fascinating corner of complexity theory—one that probably holds independent interest for philosophers.

As is well-known, in the 1930s Church and Kleene proposed definitions of the "computable functions" that turned out to be precisely equivalent to Turing's definition, but that differed from Turing's in making no explicit mention of machines. Rather than analyzing the *process* of computation, the Church-Kleene approach was simply to list *axioms* that the computable functions of natural numbers $f:\mathbb{N}\rightarrow\mathbb{N}$ ought to satisfy—for example, "if $f(x)$ and $g(x)$ are both computable,

[32] For "knowing" a person suggests having actually met the person, while "knowing" a place suggests having visited the place. Interestingly, in Hebrew, one uses a completely different verb for "know" in the sense of "being familiar with" (*makir*) than for "know" in the intellectual sense (*yodeya*).

then so is $f(g(x))$"—and then to define "the" computable functions as the smallest set satisfying those axioms.

In 1965, Alan Cobham [42] asked whether the same could be done for the *efficiently* or *feasibly* computable functions. As an answer, he offered axioms that precisely characterize what today we call FP, or Function Polynomial-Time (though Cobham called it $\mathcal{L}$). The class FP consists of all functions of natural numbers $f : \mathbb{N} \to \mathbb{N}$ that are computable in polynomial time by a deterministic Turing machine. Note that FP is "morally" the same as the class P (Polynomial-Time) defined in Section 3.1: they differ only in that P is a class of *decision* problems (or equivalently, functions $f : \mathbb{N} \to \{0, 1\}$), whereas FP is a class of functions with integer range.

What was noteworthy about Cobham's characterization of polynomial time was that it didn't involve *any* explicit mention of either computing devices or bounds on their running time. Let me now list a version of Cobham's axioms, adapted from Arora, Impagliazzo, and Vazirani [16]. Each of the axioms talks about which functions of natural numbers $f : \mathbb{N} \to \mathbb{N}$ are "efficiently computable."

(1) Every constant function f is efficiently computable, as is every function which is nonzero only finitely often.

(2) **Pairing:** If $f(x)$ and $g(x)$ are efficiently computable, then so is $\langle f(x), g(x)\rangle$, where $\langle , \rangle$ is some standard pairing function for the natural numbers.

(3) **Composition:** If $f(x)$ and $g(x)$ are efficiently computable, then so is $f(g(x))$.

(4) **Grab Bag:** The following functions are all efficiently computable:

- the arithmetic functions $x + y$ and $x \times y$
- $|x| = \lfloor \log_2 x \rfloor + 1$ (the number of bits in x's binary representation)
- the projection functions $\Pi_1(\langle x, y\rangle) = x$ and $\Pi_2(\langle x, y\rangle) = y$
- $\operatorname{bit}(\langle x, i\rangle)$ (the i^{th} bit of x's binary representation, or 0 if $i > |x|$)
- $\operatorname{diff}(\langle x, i\rangle)$ (the number obtained from x by flipping its i^{th} bit)
- $2^{|x|^2}$ (called the "smash function")

(5) **Bounded Recursion:** Suppose $f(x)$ is efficiently computable, and $|f(x)| \leq |x|$ for all $x \in \mathbb{N}$. Then the function $g(\langle x, k\rangle)$, defined by

$$g(\langle x, k\rangle) = \begin{cases} f(g(\langle x, \lfloor k/2 \rfloor\rangle)) & \text{if } k > 1 \\ x & \text{if } k = 1 \end{cases},$$

is also efficiently computable.

A few comments about the Cobham axioms might be helpful. First, the axiom that "does most of the work" is (5). Intuitively, given any natural number $k \in \mathbb{N}$ that we can generate starting from the original input $x \in \mathbb{N}$, the Bounded Recursion axiom lets us set up a "computational process" that runs for $\log_2 k$ steps. Second, the role of the "smash function," $2^{|x|^2}$, is to let us map n-bit integers to n^2-bit integers to n^4-bit integers and so on, and thereby (in combination with the Bounded Recursion axiom) set up computational processes that run for arbitrary *polynomial* numbers of steps. Third, although addition and multiplication are included as "efficiently computable functions," it is crucial that exponentiation is *not* included. Indeed, if x and y are n-bit integers, then x^y might require exponentially many bits just to write down.

The basic result is then the following:

Theorem 1 ([42, 110]) *The class* FP*, of functions* $f : \mathbb{N} \to \mathbb{N}$ *computable in polynomial time by a deterministic Turing machine, satisfies axioms (1)-(5), and is the smallest class that does so.*

To prove Theorem 1, one needs to do two things, neither of them difficult: first, show that any function f that can be defined using the Cobham axioms can also be computed in polynomial time; and second, show that the Cobham axioms are enough to simulate any polynomial-time Turing machine.

One drawback of the Cobham axioms is that they seem to "sneak in the concept of polynomial-time through the back door"—both through the "smash function," and through the arbitrary-looking condition $|f(x)| \leq |x|$ in axiom (5). In the 1990s, however, Leivant [86] and Bellantoni and Cook [25] both gave more "elegant" logical characterizations of FP that avoid this problem. So for example, Leivant showed that a function f belongs to FP, if and only if f is computed by a program that can be proved correct in second-order logic with comprehension restricted to positive quantifier-free formulas. Results like these provide further evidence—if any was needed—that polynomial-time computability is an extremely natural notion: a "wide target in conceptual space" that one hits even while aiming in purely logical directions.

Over the past few decades, the idea of defining complexity classes such as P and NP in "logical, machine-free" ways has given rise to an entire field called *descriptive complexity theory*, which has deep connections with finite model theory. While further discussion of descriptive complexity theory would take us too far afield, see the book of Immerman [77] for the definitive introduction, or Fagin [52] for a survey.

5.2 Omniscience Versus Infinity

Returning to our original topic, how exactly do axiomatic theories such as Cobham's (or Church's and Kleene's, for that matter) escape the problem of omniscience? One straightforward answer is that, unlike the set of true sentences in some formal language, which is only *countably* infinite, the set of functions $f : \mathbb{N} \to \mathbb{N}$ is *uncountably* infinite. And therefore, even if we define the "efficiently-computable" functions $f : \mathbb{N} \to \mathbb{N}$ by taking a countably-infinite logical closure, we are sure to miss *some* functions f (in fact, almost all of them!).

The observation above suggests a general strategy to tame the logical omniscience problem. Namely, we could refuse to define an agent's "knowledge" in terms of which individual questions she can quickly answer, and insist on speaking instead about which infinite *families* of questions she can quickly answer. In slogan form, we want to "fight omniscience with infinity."

Let's see how, by taking this route, we can give semi-plausible answers to the puzzles about knowledge discussed earlier in this section. First, the reason why you can "know" that $1591 = 43 \times 37$, but at the same time *not* "know" the prime factors of 1591, is that, when we speak about knowing the answers to these questions, we really mean knowing *how* to answer them. And as we saw, there need not be any contradiction in knowing a fast multiplication algorithm but *not* a fast factoring algorithm, even if we model your knowledge about algorithms as deductively closed. To put it another way, by embedding the two questions

Q1 = "Is $1591 = 43 \times 37$?"
Q2 = "What are the prime factors of 1591?"

into *infinite families of related questions*, we can break the symmetry between the knowledge entailed in answering them.

Similarly, we could think of a child as possessing an internal algorithm which, given any statement of the form $x + y = y + x$ (for specific x and y values), immediately outputs *true*, without even examining x and y. However, the child does not yet have the ability to process *quantified* statements, such as "$\forall x, y \in \mathbb{R}\ \ x + y = y + x$." In that sense, she still lacks the explicit knowledge that addition is commutative.

Although the "cure" for logical omniscience sketched above solves some puzzles, not surprisingly it raises many puzzles of its own. So let me end this section by discussing three major objections to the "infinity cure."

The first objection is that we've simply pushed the problem of logical omniscience somewhere else. For suppose an agent "knows" how to compute every function in some restricted class such as FP. Then how can we ever make sense of the agent *learning a new algorithm?* One natural response is that, even if you have the "latent ability" to compute a function $f \in \mathsf{FP}$, you might not *know* that you have the ability—either because you don't know a suitable algorithm, or because you *do* know an algorithm, but don't know that it's an algorithm for f. Of course, if we wanted to pursue things to the bottom, we'd next need to tell a story about *knowledge of algorithms*, and how logical omniscience is avoided there. However, I claim that this represents progress! For notice that, even without such a story, we can already explain *some* failures of logical omniscience. For example, the reason why you don't know the factors of a large number might *not* be your ignorance of a fast factoring method, but rather that no such method exists.

The second objection is that, when I advocated focusing on infinite families of questions rather than single questions in isolation, I never specified *which* infinite families. The difficulty is that the same question could be generalized in wildly different ways. As an example, consider the question

Q = "Is 432150 composite?"

Q is an instance of a computational problem that humans find very hard: "given a large integer N, is N composite?" However, Q is *also* an instance of a computational problem that humans find very easy: "given a large integer N *ending in* 0, is N composite?" And indeed, we'd expect a person to know the answer to Q *if* she noticed that 432150 ends in 0, but not otherwise. To me, what this example demonstrates is that, *if* we want to discuss an agent's knowledge in terms of individual questions such as Q, then the relevant issue will be whether there *exists* a generalization G of Q, such that the agent knows a fast algorithm for answering questions of type G, and *also* recognizes that Q is of type G.

The third objection is just the standard one about the relationship between asymptotic complexity and finite statements. For example, if we model an agent's knowledge using the Cobham axioms, then we can indeed explain why the agent doesn't know how to play perfect chess on an $n \times n$ board, for *arbitrary* values of n.[33] But on a standard 8×8 board, playing perfect chess would "merely" require (say) $\sim 10^{60}$ computational steps, which is a constant, and therefore certainly polynomial! So strictly on the basis of the Cobham axioms, what explanation could we

[33] For chess on an $n \times n$ board is known to be EXP-complete, and it is also known that $\mathsf{P} \neq \mathsf{EXP}$. See Section 10, and particularly footnote 62, for more details.

possibly offer for why a rational agent, who knew the rules of 8×8 chess, didn't also know how to play it optimally? While this objection might sound devastating, it's important to understand that it's no different from the usual objection leveled against complexity-theoretic arguments, and can be given the usual response. Namely: asymptotic statements are *always* vulnerable to being rendered irrelevant, if the constant factors turned out to be ridiculous. However, experience has shown that, for whatever reasons, that happens rarely enough that one can usually take asymptotic behavior as "having explanatory force until proven otherwise." (Section 12 will say more about the explanatory force of asymptotic claims, as a problem requiring philosophical analysis.)

5.3 Summary

Because of the difficulties pointed out in Section 5.2, my own view is that computational complexity theory has not yet come close to "solving" the logical omniscience problem, in the sense of giving a satisfying formal account of knowledge that also avoids making absurd predictions. I have no idea whether such an account is even possible.[34] However, what I've tried to show in this section is that complexity theory provides a well-defined "limiting case" where the logical omniscience problem *is* solvable, about as well as one could hope it to be. The limiting case is where the size of the questions grows without bound, and the solution there is given by the Cobham axioms: "axioms of knowing how" whose logical closure one *can* take without thereby inviting omniscience.

In other words, when we contemplate the omniscience problem, I claim that we're in a situation similar to one often faced in physics—where we might be at a loss to understand some phenomenon (say, gravitational entropy), *except* in limiting cases such as black holes. In epistemology just like in physics, the limiting cases that we *do* more-or-less understand offer an obvious starting point for those wishing to tackle the general case.

6 Computationalism and Waterfalls

Over the past two decades, a certain argument about computation—which I'll call the *waterfall argument*—has been widely discussed by philosophers of mind.[35] Like Searle's famous Chinese Room argument [113], the waterfall argument seeks to show that computations are "inherently syntactic," and can never be "about" anything—and that for this reason, the doctrine of "computationalism" is false.[36] But unlike the Chinese Room, the waterfall argument supplements the bare appeal to intuition by a further claim: namely, that the "meaning" of a computation, to whatever extent it has one, is always *relative to some external observer*.

More concretely, consider a waterfall (though any other physical system with a large enough state space would do as well). Here I do not mean a waterfall that was specially engineered

[34] Compare the pessimism expressed by Paul Graham [68] about knowledge representation more generally:

> In practice formal logic is not much use, because despite some progress in the last 150 years we're still only able to formalize a small percentage of statements. We may never do that much better, for the same reason 1980s-style "knowledge representation" could never have worked; many statements may have no representation more concise than a huge, analog brain state.

[35] See Putnam [106, appendix] and Searle [114] for two instantiations of the argument (though the formal details of either will not concern us here).

[36] "Computationalism" refers to the view that the mind is literally a computer, and that thought is literally a type of computation.

to perform computations, but *really* a naturally-occurring waterfall: say, Niagara Falls. Being governed by laws of physics, the waterfall implements some mapping f from a set of possible initial states to a set of possible final states. If we accept that the laws of physics are *reversible*, then f must also be injective. Now suppose we restrict attention to some finite subset S of possible initial states, with $|S| = n$. Then f is just a one-to-one mapping from S to some output set $T = f(S)$ with $|T| = n$. The "crucial observation" is now this: given *any* permutation σ from the set of integers $\{1, \ldots, n\}$ to itself, there is some way to label the elements of S and T by integers in $\{1, \ldots, n\}$, such that we can interpret f as implementing σ. For example, if we let $S = \{s_1, \ldots, s_n\}$ and $f(s_i) = t_i$, then it suffices to label the initial state s_i by i and the final state t_i by $\sigma(i)$. But the permutation σ could have any "semantics" we like: it might represent a program for playing chess, or factoring integers, or simulating a different waterfall. Therefore "mere computation" cannot give rise to semantic meaning. Here is how Searle [114, p. 57] expresses the conclusion:

> If we are consistent in adopting the Turing test or some other "objective" criterion for intelligent behavior, then the answer to such questions as "Can unintelligent bits of matter produce intelligent behavior?" and even, "How exactly do they do it" are ludicrously obvious. Any thermostat, pocket calculator, or waterfall produces "intelligent behavior," and we know in each case how it works. Certain artifacts are designed to behave as if they were intelligent, and since everything follows laws of nature, then everything will have some description under which it behaves as if it were intelligent. But this sense of "intelligent behavior" is of no psychological relevance at all.

The waterfall argument has been criticized on numerous grounds: see Haugeland [71], Block [30], and especially Chalmers [37] (who parodied the argument by proving that a cake recipe, being merely syntactic, can never give rise to the semantic attribute of crumbliness). To my mind, though, perhaps the easiest way to demolish the waterfall argument is through computational complexity considerations.

Indeed, suppose we actually wanted to use a waterfall to help us calculate chess moves. How would we do that? In complexity terms, what we want is a *reduction* from the chess problem to the waterfall-simulation problem. That is, we want an efficient algorithm that somehow *encodes* a chess position P into an initial state $s_P \in S$ of the waterfall, in such a way that a good move from P can be read out efficiently from the waterfall's corresponding final state, $f(s_P) \in T$.[37] But *what would such an algorithm look like?* We cannot say for sure—certainly not without detailed knowledge about f (i.e., the physics of waterfalls), as well as the means by which the S and T elements are encoded as binary strings. But for *any* reasonable choice, it seems overwhelmingly likely that any reduction algorithm would just *solve the chess problem itself*, without using the waterfall in an essential way at all! A bit more precisely, I conjecture that, given any chess-playing algorithm A that accesses a "waterfall oracle" W, there is an equally-good chess-playing algorithm A', with similar time and space requirements, that does *not* access W. If this conjecture holds, then it gives us a perfectly observer-independent way to formalize our intuition that the "semantics" of waterfalls have nothing to do with chess.[38]

[37] Technically, this describes a restricted class of reductions, called *nonadaptive* reductions. An *adaptive* reduction from chess to waterfalls might solve a chess problem by some procedure that involves initializing a waterfall and observing its final state, then using the results of that aquatic computation to initialize a *second* waterfall and observe *its* final state, and so on for some polynomial number of repetitions.

[38] The perceptive reader might suspect that we smuggled our conclusion into the assumption that the waterfall

6.1 "Reductions" That Do All The Work

Interestingly, the issue of "trivial" or "degenerate" reductions also arises *within* complexity theory, so it might be instructive to see how it is handled there. Recall from Section 3.1 that a problem is NP-*complete* if, loosely speaking, it is "maximally hard among all NP problems" (NP being the class of problems for which solutions can be checked in polynomial time). More formally, we say that L is NP-complete if

(i) $L \in \mathsf{NP}$, and

(ii) given any *other* NP problem L', there exists a polynomial-time algorithm to solve L' using access to an oracle that solves L. (Or more succinctly, $L' \in \mathsf{P}^L$, where P^L denotes the complexity class P augmented by an L-oracle.)

The concept of NP-completeness had incredible explanatory power: it showed that *thousands* of seemingly-unrelated problems from physics, biology, industrial optimization, mathematical logic, and other fields were all *identical* from the standpoint of polynomial-time computation, and that not one of these problems had an efficient solution unless $\mathsf{P} = \mathsf{NP}$. Thus, it was natural for theoretical computer scientists to want to define an analogous concept of P-*completeness*. In other words: among all the problems that *are* solvable in polynomial time, which ones are "maximally hard"?

But how should P-completeness even be defined? To see the difficulty, suppose that, by analogy with NP-completeness, we say that L is P-complete if

(i) $L \in \mathsf{P}$ and

(ii) $L' \in \mathsf{P}^L$ for every $L' \in \mathsf{P}$.

Then it is easy to see that the second condition is vacuous: *every* P problem is P-complete! For in "reducing" L' to L, a polynomial-time algorithm can always just ignore the L-oracle and solve L' by itself, much like our hypothetical chess program that ignored its waterfall oracle. Because of this, condition (ii) must be replaced by a stronger condition; one popular choice is

(ii') $L' \in \mathsf{LOGSPACE}^L$ for every $L' \in \mathsf{P}$.

Here $\mathsf{LOGSPACE}$ means, informally, the class of problems solvable by a deterministic Turing machine with a read/write memory consisting of only $\log n$ bits, given an input of size n.[39] It's not hard to show that $\mathsf{LOGSPACE} \subseteq \mathsf{P}$, and this containment is strongly believed to be strict (though just like with $\mathsf{P} \neq \mathsf{NP}$, there is no proof yet). The key point is that, if we want a *non-vacuous* notion of completeness, then the reducing complexity class needs to be *weaker* (either provably

states $s_P \in S$ and $f(s_P) \in T$ were encoded as binary strings in a "reasonable" way (and not, for example, in a way that encodes the solution to the chess problem). But a crucial lesson of complexity theory is that, when we discuss "computational problems," we *always* make an implicit commitment about the input and output encodings anyway! So for example, if positive integers were given as input via their prime factorizations, then the factoring problem would be trivial (just apply the identity function). But who cares? If, in mathematically defining the waterfall-simulation problem, we required input and output encodings that entailed solving chess problems, then it would no longer be reasonable to call our problem (solely) a "waterfall-simulation problem" at all.

[39] Note that a $\mathsf{LOGSPACE}$ machine does not even have enough memory to store its input string! For this reason, we think of the input string as being provided on a special *read-only* tape.

or conjecturally) than the class being reduced to. In fact complexity classes even smaller than LOGSPACE almost always suffice in practice.

In my view, there is an important lesson here for debates about computationalism. Suppose we want to claim, for example, that a computation that plays chess is "equivalent" to some other computation that simulates a waterfall. Then our claim is only non-vacuous if it's possible to *exhibit* the equivalence (i.e., give the reductions) within a model of computation that isn't *itself* powerful enough to solve the chess or waterfall problems.

7 PAC-Learning and the Problem of Induction

Centuries ago, David Hume [76] famously pointed out that learning from the past (and, by extension, science) seems logically impossible. For example, if we sample 500 ravens and every one of them is black, why does that give us *any* grounds—even probabilistic grounds—for expecting the 501^{st} raven to be black also? Any modern answer to this question would probably refer to *Occam's razor*, the principle that simpler hypotheses consistent with the data are more likely to be correct. So for example, the hypothesis that all ravens are black is "simpler" than the hypothesis that most ravens are green or purple, and that only the 500 we happened to see were black. Intuitively, it seems Occam's razor *must* be part of the solution to Hume's problem; the difficulty is that such a response leads to questions of its own:

(1) What do we mean by "simpler"?

(2) *Why* are simple explanations likely to be correct? Or, less ambitiously: what properties must reality have for Occam's Razor to "work"?

(3) How much data must we collect before we can find a "simple hypothesis" that will probably predict future data? How do we go about finding such a hypothesis?

In my view, the theory of *PAC (Probabilistically Approximately Correct) Learning*, initiated by Leslie Valiant [127] in 1984, has made large enough advances on all of these questions that it deserves to be studied by anyone interested in induction.[40] In this theory, we consider an idealized "learner," who is presented with points $x_1, \ldots, x_m$ drawn randomly from some large set $\mathcal{S}$, together with the "classifications" $f(x_1), \ldots, f(x_m)$ of those points. The learner's goal is to infer the function f, well enough to be able to predict $f(x)$ for *most* future points $x \in \mathcal{S}$. As an example, the learner might be a bank, $\mathcal{S}$ might be a set of people (represented by their credit histories), and $f(x)$ might represent whether or not person x will default on a loan.

For simplicity, we often assume that $\mathcal{S}$ is a set of binary strings, and that the function f maps each $x \in \mathcal{S}$ to a single bit, $f(x) \in \{0,1\}$. Both assumptions can be removed without significantly changing the theory. The important assumptions are the following:

(1) Each of the sample points $x_1, \ldots, x_m$ is drawn *independently* from some (possibly-unknown) "sample distribution" $\mathcal{D}$ over $\mathcal{S}$. Furthermore, the future points x on which the learner will need to predict $f(x)$ are drawn from the same distribution.

[40] See Kearns and Vazirani [82] for an excellent introduction to PAC-learning, and de Wolf [136] for previous work applying PAC-learning to philosophy and linguistics: specifically, to fleshing out Chomsky's "poverty of the stimulus" argument. De Wolf also discusses several formalizations of Occam's Razor other than the one based on PAC-learning.

(2) The function f belongs to a known "hypothesis class" $\mathcal{H}$. This $\mathcal{H}$ represents "the set of possibilities the learner is willing to entertain" (and is typically much smaller than the set of all $2^{|\mathcal{S}|}$ possible functions from $\mathcal{S}$ to $\{0,1\}$).

Under these assumptions, we have the following central result.

Theorem 2 (Valiant [127]) *Consider a finite hypothesis class $\mathcal{H}$, a Boolean function $f : \mathcal{S} \rightarrow \{0,1\}$ in $\mathcal{H}$, and a sample distribution $\mathcal{D}$ over $\mathcal{S}$, as well as an error rate $\varepsilon > 0$ and failure probability $\delta > 0$ that the learner is willing to tolerate. Call a hypothesis $h : \mathcal{S} \rightarrow \{0,1\}$ "good" if*

$$\Pr_{x \sim \mathcal{D}} [h(x) = f(x)] \geq 1 - \varepsilon.$$

Also, call sample points $x_1, \ldots, x_m$ "reliable" if any hypothesis $h \in \mathcal{H}$ that satisfies $h(x_i) = f(x_i)$ for all $i \in \{1, \ldots, m\}$ is good. Then

$$m = \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta}$$

sample points $x_1, \ldots, x_m$ drawn independently from $\mathcal{D}$ will be reliable with probability at least $1 - \delta$.

Intuitively, Theorem 2 says that the behavior of f on a small number of randomly-chosen points *probably* determines its behavior on *most* of the remaining points. In other words, if, by some unspecified means, the learner manages to find any hypothesis $h \in \mathcal{H}$ that makes correct predictions on all its past data points $x_1, \ldots, x_m$, then provided m is large enough (and as it happens, m doesn't need to be very large), the learner can be statistically confident that h will also make the correct predictions on most future points.

The part of Theorem 2 that bears the unmistakable imprint of complexity theory is the bound on sample size, $m = \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta}$. This bound has three notable implications. First, even if the class $\mathcal{H}$ contains exponentially many hypotheses (say, 2^n), one can still learn an arbitrary function $f \in \mathcal{H}$ using a *linear* amount of sample data, since m grows only logarithmically with $|\mathcal{H}|$: in other words, like the number of bits needed to *write down* an individual hypothesis. Second, one can make the probability that the hypothesis h will fail to generalize *exponentially small* (say, $\delta = 2^{-n}$), at the cost of increasing the sample size m by only a linear factor. Third, assuming the hypothesis *does* generalize, its error rate ε decreases inversely with m. It is not hard to show that each of these dependencies is tight, so that for example, if we demand either $\varepsilon = 0$ or $\delta = 0$ then no finite m suffices. This is the origin of the name "PAC-learning": the most one can hope for is to output a hypothesis that is "probably, approximately" correct.

The proof of Theorem 2 is easy: consider any hypothesis $h \in \mathcal{H}$ that is *bad*, meaning that

$$\Pr_{x \sim \mathcal{D}} [h(x) = f(x)] < 1 - \varepsilon.$$

Then by the independence assumption,

$$\Pr_{x_1, \ldots, x_m \sim \mathcal{D}} [h(x_1) = f(x_1) \wedge \cdots \wedge h(x_m) = f(x_m)] < (1 - \varepsilon)^m.$$

Now, the number of bad hypotheses is no more than the total number of hypotheses, $|\mathcal{H}|$. So by the union bound, the probability that there *exists* a bad hypothesis that agrees with f on all of $x_1, \ldots, x_m$ can be at most $|\mathcal{H}| \cdot (1 - \varepsilon)^m$. Therefore $\delta \leq |\mathcal{H}| \cdot (1 - \varepsilon)^m$, and all that remains is to solve for m.

The relevance of Theorem 2 to Hume's problem of induction is that the theorem describes a nontrivial class of situations where induction is *guaranteed to work* with high probability. Theorem 2 also illuminates the role of Occam's Razor in induction. In order to learn using a "reasonable" number of sample points m, the hypothesis class $\mathcal{H}$ must have a sufficiently small cardinality. But that is equivalent to saying that every hypothesis $h \in \mathcal{H}$ must have a *succinct description*—since the number of bits needed to specify an arbitrary hypothesis $h \in \mathcal{H}$ is simply $\lceil \log_2 |\mathcal{H}| \rceil$. If the number of bits needed to specify a hypothesis is too large, then $\mathcal{H}$ will always be vulnerable to the problem of *overfitting*: some hypotheses $h \in \mathcal{H}$ surviving contact with the sample data just by chance.

As pointed out to me by Agustín Rayo, there are several possible interpretations of Occam's Razor that have nothing to do with descriptive complexity: for example, we might want our hypotheses to be "simple" in terms of their ontological or ideological commitments. However, to whatever extent we interpret Occam's Razor as saying that *shorter* or *lower-complexity* hypotheses are preferable, Theorem 2 comes closer than one might have thought possible to a mathematical justification for why the Razor works.

Many philosophers might be familiar with alternative formal approaches to Occam's Razor. For example, within a Bayesian framework, one can choose a prior over all possible hypotheses that gives greater weight to "simpler" hypotheses (where simplicity is measured, for example, by the length of the shortest program that computes the predictions). However, while the PAC-learning and Bayesian approaches are related, the PAC approach has the advantage of requiring only a *qualitative* decision about which hypotheses one wants to consider, rather than a quantitative prior over hypotheses. Given the hypothesis class $\mathcal{H}$, one can then seek learning methods that work for *any* $f \in \mathcal{H}$. (On the other hand, the PAC approach requires an assumption about the probability distribution over *observations*, while the Bayesian approach does not.)

7.1 Drawbacks of the Basic PAC Model

I'd now like to discuss three drawbacks of Theorem 2, since I think the drawbacks illuminate philosophical aspects of induction as well as the advantages do.

The first drawback is that Theorem 2 works only for *finite* hypothesis classes. In science, however, hypotheses often involve continuous parameters, of which there is an uncountable infinity. Of course, one could solve this problem by simply discretizing the parameters, but then the number of hypotheses (and therefore the relevance of Theorem 2) would depend on how fine the discretization was. Fortunately, we can avoid such difficulties by realizing that *the learner only cares about the "differences" between two hypotheses insofar as they lead to different predictions.* This leads to the fundamental notion of *VC-dimension* (after its originators, Vapnik and Chervonenkis [129]).

Definition 3 (VC-dimension) *A hypothesis class $\mathcal{H}$ shatters the sample points $\{x_1, \ldots, x_k\} \subseteq \mathcal{S}$ if for all 2^k possible settings of $h(x_1), \ldots, h(x_k)$, there exists a hypothesis $h \in \mathcal{H}$ compatible with those settings. Then* $\mathrm{VCdim}(\mathcal{H})$*, the VC-dimension of $\mathcal{H}$, is the largest k for which there exists a subset $\{x_1, \ldots, x_k\} \subseteq \mathcal{S}$ that $\mathcal{H}$ shatters (or if no finite maximum exists, then* $\mathrm{VCdim}(\mathcal{H}) = \infty$*).*

Clearly any finite hypothesis class has finite VC-dimension: indeed, $\mathrm{VCdim}(\mathcal{H}) \leq \log_2 |\mathcal{H}|$. However, even an infinite hypothesis class can have finite VC-dimension if it is "sufficiently simple." For example, let $\mathcal{H}$ be the class of all functions $h_{a,b} : \mathbb{R} \to \{0, 1\}$ of the form

$$h_{a,b}(x) = \begin{cases} 1 & \text{if } a \leq x \leq b \\ 0 & \text{otherwise.} \end{cases}$$

Then it is easy to check that $\text{VCdim}(\mathcal{H}) = 2$.

With the notion of VC-dimension in hand, we can state a powerful (and harder-to-prove!) generalization of Theorem 2, due to Blumer et al. [31].

Theorem 4 (Blumer et al. [31]) *For some universal constant $K > 0$, the bound on m in Theorem 2 can be replaced by*

$$m = \frac{K\,\text{VCdim}(\mathcal{H})}{\varepsilon} \ln \frac{1}{\delta\varepsilon},$$

with the theorem now holding for any hypothesis class $\mathcal{H}$, finite or infinite.

If $\mathcal{H}$ has infinite VC-dimension, then it is easy to construct a probability distribution $\mathcal{D}$ over sample points such that *no finite number m of samples from D suffices to PAC-learn a function $f \in \mathcal{H}$*: one really is in the unfortunate situation described by Hume, of having no grounds at all for predicting that the next raven will be black. In some sense, then, Theorem 4 is telling us that finite VC-dimension is a necessary and sufficient condition for scientific induction to be possible. Once again, Theorem 4 also has an interpretation in terms of Occam's Razor, with the smallness of the VC-dimension now playing the role of simplicity.

The second drawback of Theorem 2 is that it gives us no clues about how to *find* a hypothesis $h \in \mathcal{H}$ consistent with the sample data. All it says is that, *if* we find such an h, then h will probably be close to the truth. This illustrates that, even in the simple setup envisioned by PAC-learning, induction *cannot* be merely a matter of seeing enough data and then "generalizing" from it, because immense computations might be needed to *find* a suitable generalization! Indeed, following the work of Kearns and Valiant [81], we now know that many natural learning problems—as an example, inferring the rules of a regular or context-free language from random examples of grammatical and ungrammatical sentences—are computationally intractable in an extremely strong sense:

> *Any polynomial-time algorithm for finding a hypothesis consistent with the data would imply a polynomial-time algorithm for breaking widely-used cryptosystems such as RSA!*[41]

The appearance of *cryptography* in the above statement is far from accidental. In a sense that can be made precise, learning and cryptography are "dual" problems: a learner wants to find patterns in data, while a cryptographer wants to generate data whose patterns are *hard* to find. More concretely, one of the basic primitives in cryptography is called a *pseudorandom function family*. This is a family of efficiently-computable Boolean functions $f_s : \{0,1\}^n \to \{0,1\}$, parameterized by a short random "seed" s, that are *virtually indistinguishable from random functions* by a polynomial-time algorithm. Here, we imagine that the would-be distinguishing algorithm can query the function f_s on various points x, and also that it *knows* the mapping from s to f_s, and so is ignorant only of the seed s itself. There is strong evidence in cryptography that pseudorandom function families exist: indeed, Goldreich, Goldwasser, and Micali [64] showed how to construct one starting from any pseudorandom *generator* (the latter was mentioned in Section 1.1).

[41] In the setting of "proper learning"—where the learner needs to output a hypothesis in some specified format—it is even known that many natural PAC-learning problems are NP-complete (see Pitt and Valiant [104] for example). But in the "improper" setting—where the learner can describe its hypothesis using any polynomial-time algorithm—it is only known how to show that PAC-learning problems are hard under cryptographic assumptions, and there seem to be inherent reasons for this (see Applebaum, Barak, and Xiao [14]).

Now, given a pseudorandom function family $\{f_s\}$, imagine a PAC-learner whose hypothesis class $\mathcal{H}$ consists of f_s for all possible seeds s. The learner is provided some randomly-chosen sample points $x_1, \ldots, x_m \in \{0,1\}^n$, together with the values of f_s on those points: $f_s(x_1), \ldots, f_s(x_m)$. Given this "training data," the learner's goal is to figure out how to compute f_s for itself—and thereby predict the values of $f_s(x)$ on new points x, points *not* in the training sample. Unfortunately, it's easy to see that *if* the learner could do that, then it would thereby distinguish f_s from a truly random function—and thereby contradict our starting assumption that $\{f_s\}$ was pseudorandom! Our conclusion is that, *if* the basic assumptions of modern cryptography hold (and in particular, if there exist pseudorandom generators), then there must be situations where learning is impossible purely because of computational complexity (and not because of insufficient data).

The third drawback of Theorem 2 is the assumption that the distribution $\mathcal{D}$ from which the learner is tested is the same as the distribution from which the sample points were drawn. To me, this is the most serious drawback, since it tells us that PAC-learning models the "learning" performed by an undergraduate cramming for an exam by solving last year's problems, or an employer using a regression model to identify the characteristics of successful hires, or a cryptanalyst breaking a code from a collection of plaintexts and ciphertexts. It does not, however, model the "learning" of an Einstein or a Szilard, making predictions about phenomena that are different in kind from anything yet observed. As David Deutsch stresses in his recent book *The Beginning of Infinity* [49], the goal of science is not merely to summarize observations, and thereby let us make predictions about similar observations. Rather, the goal is to discover explanations with "reach," meaning the ability to predict what would happen even in novel or hypothetical situations, like the Sun suddenly disappearing or a quantum computer being built. In my view, developing a compelling mathematical model of *explanatory* learning—a model that "is to explanation as the PAC model is to prediction"—is an outstanding open problem.[42]

7.2 Computational Complexity, Bleen, and Grue

In 1955, Nelson Goodman [67] proposed what he called the "new riddle of induction," which survives the Occam's Razor answer to Hume's original induction problem. In Goodman's riddle, we are asked to consider the hypothesis "All emeralds are green." The question is, why do we favor *that* hypothesis over the following alternative, which is equally compatible with all our evidence of green emeralds?

> "All emeralds are green before January 1, 2030, and then blue afterwards."

The obvious answer is that the second hypothesis adds superfluous complications, and is therefore disfavored by Occam's Razor. To that, Goodman replies that the definitions of "simple" and "complicated" depend on our language. In particular, suppose we had no words for green or blue, but we did have a word *grue*, meaning "green before January 1, 2030, and blue afterwards," and a word *bleen*, meaning "blue before January 1, 2030, and green afterwards." In that case, we could only express the hypothesis "All emeralds are green" by saying

[42] This problem is not as hopeless as it might sound! Important progress includes the work of Angluin [11] on learning finite automata from queries and counterexamples, and that of Angluin et al. [12] on learning a circuit by injecting values. Both papers study natural learning models that generalize the PAC model by allowing "controlled scientific experiments," whose results confirm or refute a hypothesis and thereby provide guidance about which experiments to do next.

"All emeralds are grue before January 1, 2030, and then bleen afterwards."

—a manifestly more complicated hypothesis than the simple "All emeralds are grue"!

I confess that, when I contemplate the grue riddle, I can't help but recall the joke about the Anti-Inductivists, who, when asked why they continue to believe that the future *won't* resemble the past, when that false belief has brought their civilization nothing but poverty and misery, reply, "because anti-induction has never worked before!" Yes, if we artificially define our primitive concepts "against the grain of the world," then we shouldn't be surprised if the world's actual behavior becomes more cumbersome to describe, or if we make wrong predictions. It would be as if we were using a programming language that had no built-in function for multiplication, but only for $F(x,y) := 17x - y - x^2 + 2xy$. In that case, a normal person's first instinct would be either to switch programming languages, or else to *define* multiplication in terms of F, and forget about F from that point onward![43] Now, there *is* a genuine philosophical problem here: why *do* grue, bleen, and $F(x,y)$ go "against the grain of the world," whereas green, blue, and multiplication go with the grain? But to me, that problem (like Wigner's puzzlement over "the unreasonable effectiveness of mathematics in natural sciences" [135]) is more about the world itself than about human concepts, so we shouldn't expect any purely linguistic analysis to resolve it.

What about computational complexity, then? In my view, while computational complexity doesn't solve the grue riddle, it does contribute a useful insight. Namely, that when we talk about the simplicity or complexity of hypotheses, we should distinguish two issues:

(a) The *asymptotic scaling* of the hypothesis size, as the "size" n of our learning problem goes to infinity.

(b) The constant-factor overheads.

In terms of the basic PAC model in Section 7, we can imagine a "hidden parameter" n, which measures the number of bits needed to specify an individual point in the set $\mathcal{S} = \mathcal{S}_n$. (Other ways to measure the "size" of a learning problem would also work, but this way is particularly convenient.) For convenience, we can identify $\mathcal{S}_n$ with the set $\{0,1\}^n$ of n-bit strings, so that $n = \log_2 |\mathcal{S}_n|$. We then need to consider, not just a *single* hypothesis class, but an infinite *family* of hypothesis classes $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \ldots\}$, one for each positive integer n. Here $\mathcal{H}_n$ consists of hypothesis functions h that map $\mathcal{S}_n = \{0,1\}^n$ to $\{0,1\}$.

Now let L be a *language* for specifying hypotheses in $\mathcal{H}$: in other words, a mapping from (some subset of) binary strings $y \in \{0,1\}^*$ to $\mathcal{H}$. Also, given a hypothesis $h \in \mathcal{H}$, let

$$\kappa_L(h) := \min\{|y| : L(y) = h\}$$

be the length of the *shortest* description of h in the language L. (Here $|y|$ just means the number of bits in y.) Finally, let

$$\kappa_L(n) := \max\{\kappa_L(h) : h \in \mathcal{H}_n\}$$

[43] Suppose that our programming language provides only multiplication by constants, addition, and the function $F(x,y) := ax^2 + bxy + cy^2 + dx + ey + f$. We can assume without loss of generality that $d = e = f = 0$. Then provided $ax^2 + bxy + cy^2$ factors into two independent linear terms, $px + qy$ and $rx + sy$, we can express the product xy as

$$\frac{F(sx - qy, -rx + py)}{(ps - qr)^2}.$$

be the number of bits needed to specify an *arbitrary* hypothesis in $\mathcal{H}_n$ using the language L. Clearly $\kappa_L(n) \geq \lceil \log_2 |\mathcal{H}_n| \rceil$, with equality if and only if L is "optimal" (that is, if it represents each hypothesis $h \in \mathcal{H}_n$ using as few bits as possible). The question that concerns us is how quickly $\kappa_L(n)$ grows as a function of n, for various choices of language L.

What does any of this have to do with the grue riddle? Well, we can think of the details of L (its syntax, vocabulary, etc.) as affecting the "lower-order" behavior of the function $\kappa_L(n)$. So for example, suppose we are unlucky enough that L contains the words *grue* and *bleen*, but not *blue* and *green*. That might increase $\kappa_L(n)$ by a factor of ten or so—since now, every time we want to mention "green" when specifying our hypothesis h, we instead need a wordy circumlocution like "grue before January 1, 2030, and then bleen afterwards," and similarly for blue.[44] However, a crucial lesson of complexity theory is that the "higher-order" behavior of $\kappa_L(n)$—for example, whether it grows polynomially or exponentially with n—is almost completely unaffected by the details of L! The reason is that, if two languages L_1 and L_2 differ only in their "low-level details," then *translating* a hypothesis from L_1 to L_2 or vice versa will increase the description length by no more than a polynomial factor. Indeed, as in our grue example, there is usually a "universal translation constant" c such that $\kappa_{L_1}(h) \leq c\kappa_{L_2}(h)$ or even $\kappa_{L_1}(h) \leq \kappa_{L_2}(h) + c$ for *every* hypothesis $h \in \mathcal{H}$.

The one exception to the above rule is if the languages L_1 and L_2 have different *expressive powers*. For example, maybe L_1 only allows nesting expressions to depth two, while L_2 allows nesting to arbitrary depths; or L_1 only allows propositional connectives, while L_2 also allows first-order quantifiers. In those cases, $\kappa_{L_1}(h)$ could indeed be much greater than $\kappa_{L_2}(h)$ for some hypotheses h, possibly even exponentially greater ($\kappa_{L_1}(h) \approx 2^{\kappa_{L_2}(h)}$). A rough analogy would be this: suppose you hadn't learned what differential equations were, and had no idea how to solve them even approximately or numerically. In that case, Newtonian mechanics might seem just as complicated to you as the Ptolemaic theory with epicycles, if not *more* complicated! For the only way you could make predictions with Newtonian mechanics would be using a huge table of "precomputed" differential equation solutions—and *to you*, that table would seem just as unwieldy and inelegant as a table of epicycles. But notice that in this case, your perception would be the result, not of some arbitrary choice of vocabulary, but of an *objective* gap in your mathematical expressive powers.

To summarize, our choice of vocabulary—for example, whether we take green/blue or bleen/grue as primitive concepts—could indeed matter if we want to use Occam's Razor to predict the future color of emeralds. But I think that complexity theory justifies us in treating grue as a "small-n effect": something that becomes less and less important in the asymptotic limit of more and more complicated learning problems.

[44] Though note that, if the language L is expressive enough to allow this, we can simply define green and blue in terms of bleen and grue *once*, then refer back to those definitions whenever needed! In that case, taking bleen and grue (rather than green and blue) to be the primitive concepts would increase $\kappa_L(n)$ by only an *additive* constant, rather than a multiplicative constant.

The above fact is related to a fundamental result from the theory of Kolmogorov complexity (see Li and Vitányi [89] for example). Namely, if P and Q are any two Turing-universal programming languages, and if $K_P(x)$ and $K_Q(x)$ are the lengths of the shortest programs in P and Q respectively that output a given string $x \in \{0,1\}^*$, then there exists a universal "translation constant" c_{PQ}, such that $|K_P(x) - K_Q(x)| \leq c_{PQ}$ for *every* x. This c_{PQ} is just the number of bits needed to write a P-interpreter for Q-programs or vice versa.

8 Quantum Computing

Quantum computing is a proposal for using quantum mechanics to solve certain computational problems much faster than we know how to solve them today.[45] To do so, one would need to build a new type of computer, capable of exploiting the quantum effects of superposition and interference. Building such a computer—one large enough to solve interesting problems—remains an enormous challenge for physics and engineering, due to the fragility of quantum states and the need to isolate them from their external environment.

In the meantime, though, theoretical computer scientists have extensively studied what we could and couldn't do with a quantum computer if we had one. For certain problems, remarkable quantum algorithms are known to solve them in polynomial time, even though the best-known classical algorithms require exponential time. Most famously, in 1994 Peter Shor [117] gave a polynomial-time quantum algorithm for factoring integers, and as a byproduct, breaking most of the cryptographic codes used on the Internet today. Besides the practical implications, Shor's algorithm also provided a key piece of evidence that switching from classical to quantum computers would enlarge the class of problems solvable in polynomial-time. For theoretical computer scientists, this had a profound lesson: if we want to know the limits of efficient computation, we may need to "leave our armchairs" and incorporate actual facts about physics (at a minimum, the truth or falsehood of quantum mechanics!).[46]

Whether or not scalable quantum computers are built anytime soon, my own (biased) view is that quantum computing represents one of the great scientific advances of our time. But here I want to ask a different question: does quantum computing have any implications for *philosophy*—and specifically, for the interpretation of quantum mechanics?

From one perspective, the answer seems like an obvious "no." Whatever else it is, quantum computing is "merely" an application of quantum mechanics, as that theory has existed in physics textbooks for 80 years. Indeed, *if* you accept that quantum mechanics (as currently understood) is true, then presumably you should also accept the possibility of quantum computers, and make the same predictions about their operation as everyone else. Whether you describe the "reality" behind quantum processes via the Many-Worlds Interpretation, Bohmian mechanics, or some other view (or, following Bohr's Copenhagen Interpretation, refuse to discuss the "reality" at all), seems irrelevant.

From a different perspective, though, a scalable quantum computer would *test* quantum mechanics in an extremely novel regime—and for that reason, it could indeed raise new philosophical issues. The "regime" quantum computers would test is characterized not by an energy scale or a temperature, but by computational complexity. One of the most striking facts about quantum mechanics is that, to represent the state of n entangled particles, one needs a vector of size *exponential* in n. For example, to specify the state of a thousand spin-1/2 particles, one needs 2^{1000} complex numbers called "amplitudes," one for every possible outcome of measuring the spins in the $\{\text{up}, \text{down}\}$ basis. The quantum state, denoted $|\psi\rangle$, is then a linear combination or "superposition"

[45] The authoritative reference for quantum computing is the book of Nielsen and Chuang [99]. For gentler introductions, try Mermin [92, 93] or the survey articles of Aharonov [10], Fortnow [57], or Watrous [132]. For a general discussion of polynomial-time computation and the laws of physics (including speculative models beyond quantum computation), see my survey article "NP-complete Problems and Physical Reality" [4].

[46] By contrast, if we only want to know what is *computable* in the physical universe, with no efficiency requirement, then it remains entirely consistent with current knowledge that Church and Turing gave the correct answer in the 1930s—and that they did so without incorporating any physics beyond what is "accessible to intuition."

of the possible outcomes, with each outcome $|x\rangle$ weighted by its amplitude α_x:

$$|\psi\rangle = \sum_{x \in \{\text{up},\text{down}\}^{1000}} \alpha_x |x\rangle .$$

Given $|\psi\rangle$, one can calculate the probability p_x that any particular outcome $|x\rangle$ will be observed, via the rule $p_x = |\alpha_x|^2$.[47]

Now, there are only about 10^{80} atoms in the visible universe, which is a much smaller number than 2^{1000}. So assuming quantum mechanics is true, it seems Nature has to invest *staggering* amounts of "computational effort" to keep track of small collections of particles—certainly more than anything classical physics requires![48,49] In the early 1980s, Richard Feynman [55] and others called attention to this point, noting that it underlay something that had long been apparent in practice: the extraordinary difficulty of simulating quantum mechanics using conventional computers. But Feynman also raised the possibility of turning that difficulty around, by building our computers out of quantum components. Such computers could conceivably solve certain problems faster than conventional computers: if nothing else, then at least the problem of simulating quantum mechanics!

Thus, quantum computing is interesting not just because of its applications, but (even more, in my opinion) because *it is the first technology that would directly "probe" the exponentiality inherent in the quantum description of Nature.* One can make an analogy here to the experiments in the 1980s that first convincingly violated the Bell Inequality. Like quantum algorithms today, Bell's refutation of local realism was "merely" a mathematical consequence of quantum mechanics. But that refutation (and the experiments that it inspired) made conceptually-important *aspects* of quantum mechanics no longer possible to ignore—and for that reason, it changed the philosophical landscape. It seems overwhelmingly likely to me that quantum computing will do the same.

Indeed, we can extend the analogy further: just as there were "local realist diehards" who denied that Bell Inequality violation would be possible (and tried to explain it away after it was achieved),

[47] This means, in particular, that the amplitudes satisfy the normalization condition $\sum_x |\alpha_x|^2 = 1$.

[48] One might object that even in the classical world, if we simply don't *know* the value of (say) an n-bit string, then we *also* describe our ignorance using exponentially-many numbers: namely, the *probability* p_x of each possible string $x \in \{0,1\}^n$! And indeed, there is an extremely close connection between quantum mechanics and classical probability theory; I often describe quantum mechanics as just "probability theory with complex numbers instead of nonnegative reals." However, a crucial difference is that we can always describe a classical string x as "really" having a definite value; the vector of 2^n probabilities p_x is then just a mental representation of our own ignorance. With a quantum state, we do not have the same luxury, because of the phenomenon of *interference* between positive and negative amplitudes.

[49] One might also object that, even in classical physics, it takes *infinitely* many bits to record the state of even a single particle, if its position and momentum can be arbitrary real numbers. And indeed, Copeland [43], Hogarth [73], Siegelmann [118], and other writers have speculated that the continuity of physical quantities might actually allow "hypercomputations"—including solving the halting problem in a finite amount of time! From a modern perspective, though, quantum mechanics and quantum gravity strongly suggest that *the "continuity" of measurable quantities such as positions and momenta is a theoretical artifact.* In other words, it ought to suffice for simulation purposes to approximate these quantities to some finite precision, probably related to the Planck scale of 10^{-33} centimeters or 10^{-43} seconds.

But the exponentiality of quantum states is different, for at least two reasons. Firstly, it doesn't lead to computational speedups that are nearly as "unreasonable" as the hypercomputing speedups. Secondly, no one has any idea where the theory in question (quantum mechanics) *could* break down, in a manner consistent with current experiments. In other words, there is no known "killer obstacle" for quantum computing, analogous to the Planck scale for hypercomputing. See Aaronson [2] for further discussion of this point, as well as a proposed complexity-theoretic framework (called "Sure/Shor separators") with which to study such obstacles.

so today a vocal minority of computer scientists and physicists (including Leonid Levin [88], Oded Goldreich [61], and Gerard 't Hooft [75]) denies the possibility of scalable quantum computers, even in principle. While they admit that quantum mechanics has passed every experimental test for a century, these skeptics are *confident* that quantum mechanics will fail in the regime tested by quantum computing—and that whatever new theory replaces it, that theory will allow only classical computing.

As most quantum computing researchers are quick to point out in response, they would be *thrilled* if the attempt to build scalable quantum computers led instead to a revision of quantum mechanics! Such an outcome would probably constitute the largest revolution in physics since the 1920s, and ultimately be much *more* interesting than building a quantum computer. Of course, it is also possible that scalable quantum computing will be given up as too difficult for "mundane" technological reasons, rather than fundamental physics reasons. But that "mundane" possibility is not what skeptics such as Levin, Goldreich, and 't Hooft are talking about.

8.1 Quantum Computing and the Many-Worlds Interpretation

But let's return to the original question: suppose the skeptics are wrong, and it *is* possible to build scalable quantum computers. Would that have any relevance to the interpretation of quantum mechanics? The best-known argument that the answer is "yes" was made by David Deutsch, a quantum computing pioneer and staunch defender of the Many-Worlds Interpretation. To be precise, Deutsch thinks that quantum mechanics *straightforwardly* implies the existence of parallel universes, and that it does so independently of quantum computing: on his view, even the double-slit experiment can only be explained in terms of two parallel universes interfering. However, Deutsch also thinks that quantum computing adds emotional punch to the argument. Here is how he put it in his 1997 book *The Fabric of Reality* [48, p. 217]:

> Logically, the possibility of complex quantum computations adds nothing to a case [for the Many-Worlds Interpretation] that is already unanswerable. But it does add psychological impact. With Shor's algorithm, the argument has been writ very large. To those who still cling to a single-universe world-view, I issue this challenge: *explain how Shor's algorithm works.* I do not merely mean predict that it will work, which is merely a matter of solving a few uncontroversial equations. I mean provide an explanation. When Shor's algorithm has factorized a number, using 10^{500} or so times the computational resources that can be seen to be present, where was the number factorized? There are only about 10^{80} atoms in the entire visible universe, an utterly minuscule number compared with 10^{500}. So if the visible universe were the extent of physical reality, physical reality would not even remotely contain the resources required to factorize such a large number. Who did factorize it, then? How, and where, was the computation performed?

There is plenty in the above paragraph for an enterprising philosopher to mine. In particular, how *should* a nonbeliever in Many-Worlds answer Deutsch's challenge? In the rest of this section, I'll focus on two possible responses.

The first response is to deny that, if Shor's algorithm works as predicted, that can only be explained by postulating "vast computational resources." At the most obvious level, complexity theorists have not yet ruled out the possibility of a fast *classical* factoring algorithm.[50] More generally, that quantum computers can solve certain problems superpolynomially faster than classical computers is not a theorem, but a (profound, plausible) *conjecture*.[51,52] If the conjecture failed, then the door would seem open to what we might call "polynomial-time hidden-variable theories": theories that reproduce the predictions of quantum mechanics without invoking any computations outside P.[53] These would be analogous to the *local* hidden variable theories that Einstein and others had hoped for, before Bell ruled such theories out.

A second response to Deutsch's challenge is that, even if we agree that Shor's algorithm demonstrates the reality of vast *computational resources* in Nature, it is not obvious that we should think of those resources as "parallel universes." Why not simply say that there is *one* universe, and that it is quantum-mechanical? Doesn't the parallel-universes language reflect an ironic *parochialism*: a desire to impose a familiar science-fiction image on a mathematical theory that is *stranger* than fiction, that doesn't match *any* of our pre-quantum intuitions (including computational intuitions) particularly well?

One can sharpen the point as follows: *if* one took the parallel-universes explanation of how a quantum computer works too seriously (as many popular writers do!), then it would be natural to make further inferences about quantum computing that are flat-out wrong. For example:

> *"Using only a thousand quantum bits (or qubits), a quantum computer could store* 2^{1000} *classical bits."*

This is true only for a bizarre definition of the word "store"! The fundamental problem is that, when you measure a quantum computer's state, you see only *one* of the possible outcomes; the rest disappear. Indeed, a celebrated result called *Holevo's Theorem* [74] says that, using n qubits, there is no way to store more than n classical bits so that the bits can be reliably retrieved later. In other words: for at least one natural definition of "information-carrying capacity," qubits have exactly the same capacity as bits.

To take another example:

[50] Indeed, one *cannot* rule that possibility out, without first proving $\mathsf{P} \neq \mathsf{NP}$! But even if $\mathsf{P} \neq \mathsf{NP}$, a fast classical factoring algorithm might *still* exist, again because factoring is not thought to be NP-complete.

[51] A formal version of this conjecture is $\mathsf{BPP} \neq \mathsf{BQP}$, where BPP (Bounded-Error Probabilistic Polynomial-Time) and BQP (Bounded-Error Quantum Polynomial-Time) are the classes of problems efficiently solvable by classical randomized algorithms and quantum algorithms respectively. Bernstein and Vazirani [29] showed that $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PSPACE}$, where PSPACE is the class of problems solvable by a deterministic Turing machine using a polynomial amount of *memory* (but possibly exponential time). For this reason, any proof of the $\mathsf{BPP} \neq \mathsf{BQP}$ conjecture would immediately imply $\mathsf{P} \neq \mathsf{PSPACE}$ as well. The latter would be considered almost as great a breakthrough as $\mathsf{P} \neq \mathsf{NP}$.

[52] Complicating matters, there *are* quantum algorithms that provably achieve exponential speedups over any classical algorithm: one example is Simon's algorithm [119], an important predecessor of Shor's algorithm. However, all such algorithms are formulated in the "black-box model" (see Beals et al. [23]), where the resource to be minimized is the number of queries that an algorithm makes to a hypothetical black box. Because it is relatively easy to analyze, the black-box model is a crucial source of insights about what *might* be true in the conventional Turing machine model. However, it is also known that the black-box model sometimes misleads us about the "real" situation. As a famous example, the complexity classes IP and PSPACE are equal [115], despite the existence of a black box that separates them (see Fortnow [56] for discussion).

Besides the black-box model, *unconditional* exponential separations between quantum and classical complexities are known in several other restricted models, including communication complexity [107].

[53] Technically, if the hidden-variable theory involved classical randomness, then it would correspond more closely to the complexity class BPP (Bounded-Error Probabilistic Polynomial-Time). However, today there is strong evidence that $\mathsf{P} = \mathsf{BPP}$ (see Impagliazzo and Wigderson [79]).

"Unlike a classical computer, which can only factor numbers by trying the divisors one by one, a quantum computer could try all possible divisors in parallel."

If quantum computers can harness vast numbers of parallel worlds, then the above seems like a reasonable guess as to how Shor's algorithm works. But *it's not how it works at all.* Notice that, if Shor's algorithm *did* work that way, then it could be used not only for factoring integers, but also for the much larger task of solving NP-complete problems in polynomial time. (As mentioned in footnote 12, the factoring problem is strongly believed *not* to be NP-complete.) But contrary to a common misconception, quantum computers are neither known nor believed to be able to solve NP-complete problems efficiently.[54] As usual, the fundamental problem is that measuring reveals just a single random outcome $|x\rangle$. To get around that problem, and ensure that the *right* outcome is observed with high probability, a quantum algorithm needs to generate an *interference pattern*, in which the computational paths leading to a given wrong outcome cancel each other out, while the paths leading to a given right outcome reinforce each other. This is a delicate requirement, and as far as anyone knows, it can only be achieved for a few problems, most of which (like the factoring problem) have special structure arising from algebra or number theory.[55]

A Many-Worlder might retort: "sure, I agree that quantum computing involves harnessing the parallel universes in subtle and non-obvious ways, but it's still *harnessing parallel universes!*" But even here, there's a fascinating irony. Suppose we choose to think of a quantum algorithm in terms of parallel universes. Then to put it crudely, not only must many universes interfere to give a large final amplitude to the right answer; they must also, by interfering, *lose their identities as parallel universes!* In other words, to whatever extent a collection of universes is useful for quantum computation, to that extent it is arguable whether we ought to call them "parallel universes" at all (as opposed to parts of one exponentially-large, self-interfering, quantum-mechanical blob). Conversely, to whatever extent the universes have unambiguously separate identities, to that extent they're now "decohered" and out of causal contact with each other. Thus we can explain the outputs of any future computations by invoking only one of the universes, and treating the others as unrealized hypotheticals.

To clarify, I don't regard either of the above objections to Deutsch's argument as decisive, and am unsure what I think about the matter. My purpose, in setting out the objections, was simply to illustrate the potential of quantum computing theory to inform debates about the Many-Worlds Interpretation.

9 New Computational Notions of Proof

Since the time of Euclid, there have been two main notions of mathematical proof:

(1) A "proof" is a verbal explanation that induces a sense of certainty (and ideally, understanding) about the statement to be proved, in any human mathematician willing and able to follow it.

[54] There is a remarkable quantum algorithm called *Grover's algorithm* [69], which can search any space of 2^N possible solutions in only $\sim 2^{N/2}$ steps. However, Grover's algorithm represents a *quadratic* (square-root) improvement over classical brute-force search, rather than an exponential improvement. And without any further assumptions about the structure of the search space, Grover's algorithm is optimal, as shown by Bennett et al. [27].

[55] Those interested in further details of how Shor's algorithm works, but still not ready for a mathematical exposition, might want to try my popular essay "Shor, I'll Do It" [1].

(2) A “proof” is a finite sequence of symbols encoding syntactic deductions in some formal system, which start with axioms and end with the statement to be proved.

The tension between these two notions is a recurring theme in the philosophy of mathematics. But theoretical computer science deals regularly with a third notion of proof—one that seems to have received much less philosophical analysis than either of the two above. This notion is the following:

(3) A “proof” is any computational process or protocol (real or imagined) that can terminate in a certain way if and only if the statement to be proved is true.

9.1 Zero-Knowledge Proofs

As an example of this third notion, consider *zero-knowledge proofs*, introduced by Goldwasser, Micali, and Rackoff [66]. Given two graphs G and H, each with $n \approx 10000$ vertices, suppose that an all-powerful but untrustworthy wizard Merlin wishes to convince a skeptical king Arthur that G and H are *not* isomorphic. Of course, one way Merlin could do this would be to list all $n!$ graphs obtained by permuting the vertices of G, then note that none of these equal H. However, such a proof would clearly exhaust Arthur's patience (indeed, it could not even be written down within the observable universe). Alternatively, Merlin could point Arthur to some *property* of G and H that differentiates them: for example, maybe their adjacency matrices have different eigenvalue spectra. Unfortunately, it is not yet proven that, if G and H are non-isomorphic, there is always a differentiating property that Arthur can verify in time polynomial in n.

But as noticed by Goldreich, Micali, and Wigderson [65], there is something Merlin can do instead: he can let Arthur *challenge* him. Merlin can say:

> Arthur, send me a new graph K, which you obtained *either* by randomly permuting the vertices of G, *or* randomly permuting the vertices of H. Then I guarantee that I will tell you, without fail, whether $K \cong G$ or $K \cong H$.

It is clear that, if G and H are really non-isomorphic, then Merlin can always answer such challenges correctly, by the assumption that he (Merlin) has unlimited computational power. But it is equally clear that, if G and H are isomorphic, then Merlin must answer some challenges incorrectly, regardless of his computational power—since a random permutation of G is statistically indistinguishable from a random permutation of H.

This protocol has at least four features that merit reflection by anyone interested in the nature of mathematical proof.

First, the protocol is *probabilistic*. Merlin cannot convince Arthur with certainty that G and H are non-isomorphic, since even if they were isomorphic, there's a 1/2 probability that Merlin would get lucky and answer a given challenge correctly (and hence, a $1/2^k$ probability that he would answer k challenges correctly). All Merlin can do is offer to repeat the protocol (say) 100 or 1000 times, and thereby make it less likely that his proof is unsound than that an asteroid will strike Camelot, killing both him and Arthur.

Second, the protocol is *interactive*. Unlike with proof notions (1) and (2), Arthur is no longer a passive recipient of knowledge, but an active player who challenges the prover. We know from experience that the ability to *interrogate* a seminar speaker—to ask questions that the speaker could not have anticipated, evaluate the responses, and then possibly ask followup questions—often speeds up the process of figuring out whether the speaker knows what he or she is talking about. Complexity theory affirms our intuition here, through its discovery of interactive proofs for statements (such as "G and H are not isomorphic") whose shortest known conventional proofs are exponentially longer.

The third interesting feature of the graph non-isomorphism protocol—a feature seldom mentioned—is that its soundness implicitly relies on a *physical* assumption. Namely, if Merlin had the power (whether through magic or through ordinary espionage) to "peer into Arthur's study" and *directly observe* whether Arthur started with G or H, then clearly he could answer every challenge correctly even if $G \cong H$. It follows that the persuasiveness of Merlin's "proof" can only be as strong as Arthur's extramathematical belief that Merlin does *not* have such powers. By now, there are many other examples in complexity theory of "proofs" whose validity rests on assumed limitations of the provers.

As Shieber [116] points out, all three of the above properties of interactive protocols *also* hold for the Turing Test discussed in Section 4! The Turing Test is interactive by definition, it is probabilistic because even a program that printed random gibberish would have *some* nonzero probability of passing the test by chance, and it depends on the physical assumption that the AI program doesn't "cheat" by (for example) secretly consulting a human. For these reasons, Shieber argues that we can see the Turing Test *itself* as an early interactive protocol—one that convinces the verifier not of a mathematical theorem, but of the prover's capacity for intelligent verbal behavior.[56]

However, perhaps the most striking feature of the graph non-isomorphism protocol is that it is *zero-knowledge*: a technical term formalizing our intuition that "Arthur learns nothing from the protocol, beyond the truth of the statement being proved."[57] For all Merlin ever tells Arthur is which graph he (Arthur) started with, G or H. But Arthur *already knew* which graph he started with! This means that, not only does Arthur gain no "understanding" of what makes G and H non-isomorphic, he does not even gain the ability to prove to a third party what Merlin proved to him. This is another aspect of computational proofs that has no analogue with proof notions (1) or (2).

One might complain that, as interesting as the zero-knowledge property is, so far we've only shown it's achievable for an extremely specialized problem. And indeed, just like with factoring integers, today there is strong evidence that the graph isomorphism problem is *not* NP-complete [33].[58,59] However, in the same paper that gave the graph non-isomorphism protocol, Goldreich,

[56]Incidentally, this provides a good example of how notions from computational complexity theory can influence philosophy even just at the level of metaphor, forgetting about the actual results. In this essay, I didn't try to collect such "metaphorical" applications of complexity theory, simply because there were too many of them!

[57]Technically, the protocol is "*honest-verifier* zero-knowledge," meaning that Arthur learns nothing from his conversation his Merlin besides the truth of the statement being proved, *assuming* Arthur follows the protocol correctly. If Arthur cheats—for example, by sending a graph K for which he *doesn't* already know an isomorphism either to G or to H—then Merlin's response could indeed tell Arthur something new. However, Goldreich, Micali, and Wigderson [65] also gave a more sophisticated proof protocol for graph non-isomorphism, which remains zero-knowledge even in the case where Arthur cheats.

[58]Indeed, there is not even a consensus belief that graph isomorphism is outside P! The main reason is that, in contrast to factoring integers, graph isomorphism turns out to be extremely easy *in practice*. Indeed, finding non-isomorphic graphs that *can't* be distinguished by simple invariants is itself a hard problem! And in the past, several problems (such as linear programming and primality testing) that were long known to be "efficiently solvable for practical purposes" were eventually shown to be in P in the strict mathematical sense as well.

[59]There is also strong evidence that there are short *conventional* proofs for graph non-isomorphism—in other words,

Micali, and Wigderson [65] also gave a celebrated zero-knowledge protocol (now called the *GMW protocol*) for the NP-complete problems. By the definition of NP-complete (see Section 3.1), the GMW protocol meant that *every mathematical statement that has a conventional proof (say, in Zermelo-Fraenkel set theory) also has a zero-knowledge proof of comparable size!* As an example application, suppose you've just proved the Riemann Hypothesis. You want to convince the experts of your triumph, but are paranoid about them stealing credit for it. In that case, "all" you need to do is

(1) rewrite your proof in a formal language,

(2) encode the result as the solution to an NP-complete problem, and then

(3) like a 16^{th}-century court mathematician challenging his competitors to a duel, invite the experts to run the GMW protocol with you over the Internet!

Provided you answer all their challenges correctly, the experts can become *statistically certain* that you possess a proof of the Riemann Hypothesis, without learning anything *about* that proof besides an upper bound on its length.

Better yet, unlike the graph non-isomorphism protocol, the GMW protocol does not assume a super-powerful wizard—only an ordinary polynomial-time being who happens to know a proof of the relevant theorem. As a result, today the GMW protocol is much more than a theoretical curiosity: it and its variants have found major applications in Internet cryptography, where clients and servers often need to prove to each other that they are following a protocol correctly without revealing secret information as they do so.

However, there is one important caveat: unlike the graph-nonisomorphism protocol, the GMW protocol relies essentially on a *cryptographic hypothesis*. For here is how the GMW protocol works: you (the prover) first publish thousands of encrypted messages, each one "committing" you to a randomly-garbled piece of your claimed proof. You then offer to decrypt a tiny fraction of those messages, as a way for skeptical observers to "spot-check" your proof, while learning nothing about its structure besides the useless fact that, say, the 1729^{th} step is valid (but how could it *not* be valid?). If the skeptics want to increase their confidence that your proof is sound, then you simply run the protocol over and over with them, using a fresh batch of encrypted messages each time. If the skeptics could decrypt all the messages in a single batch, *then* they could piece together your proof—but to do that, they would need to break the underlying cryptographic code.

9.2 Other New Notions

Let me mention four other notions of "proof" that complexity theorists have explored in depth over the last twenty years, and that might merit philosophical attention.

- *Multi-prover interactive proofs* [26, 20], in which Arthur exchanges messages with *two* (or more) computationally-powerful but untrustworthy wizards. Here, Arthur might become convinced of some mathematical statement, but only under the assumption that the wizards could not communicate with *each other* during the protocol. (The usual analogy is to a police detective who puts two suspects in separate cells, to prevent them from coordinating their answers.) Interestingly, in some multi-prover protocols, even non-communicating wizards could

that not just graph isomorphism but also graph non-isomorphism will ultimately turn out to be in NP [84].

successfully coordinate their responses to Arthur's challenges (and thereby convince Arthur of a falsehood) through the use of *quantum entanglement* [41]. However, other protocols are conjectured to remain sound even against entangled wizards [83].

- *Probabilistically checkable proofs* [54, 18], which are mathematical proofs encoded in a special error-correcting format, so that one can become confident of their validity by checking only 10 or 20 bits chosen randomly in a correlated way. The *PCP (Probabilistically Checkable Proofs) Theorem* [17, 50], one of the crowning achievements of complexity theory, says that *any* mathematical theorem, in any standard formal system such as Zermelo-Fraenkel set theory, can be converted in polynomial time into a probabilistically-checkable format.

- *Quantum proofs* [131, 6], which are proofs that depend for their validity on the output of a quantum computation—possibly, even a quantum computation that requires a special entangled "proof state" fed to it as input. Because n quantum bits might require $\sim 2^n$ classical bits to simulate, quantum proofs have the property that it might never be possible to list all the "steps" that went into the proof, within the constraints of the visible universe. For this reason, one's belief in the mathematical statement being proved might depend on one's belief in the correctness of quantum mechanics as a physical theory.

- *Computationally-sound proofs and arguments* [35, 94], which rely for their validity on the assumption that the prover was limited to polynomial-time computations—as well as the mathematical conjecture that crafting a convincing argument for a falsehood would have taken the prover more than polynomial time.

What implications do these new types of proof have for the foundations of mathematics? Do they merely make more dramatic what "should have been obvious all along": that, as David Deutsch argues in *The Beginning of Infinity* [49], proofs are physical processes taking place in brains or computers, which therefore have no validity independent of our beliefs about physics? Are the issues raised essentially the same as those raised by "conventional" proofs that require extensive computations, like Appel and Haken's proof of the Four-Color Theorem [13]? Or does appealing, in the course of a "mathematical proof," to (say) the validity of quantum mechanics, the randomness of apparently-random numbers, or the lack of certain superpowers on the part of the prover represent something qualitatively new? Philosophical analysis is sought.

10 Complexity, Space, and Time

What can computational complexity tell us about the nature of space and time? A first answer might be "not much": after all, the definitions of standard complexity classes such as P can be shown to be insensitive to such details as the number of spatial dimensions, and even whether the speed of light is finite or infinite.[60] On the other hand, I think complexity theory does offer insight about the *differences* between space and time.

[60] More precisely, Turing machines with one-dimensional tapes are polynomially equivalent to Turing machines with k-dimensional tapes for any k, and are also polynomially equivalent to *random-access machines* (which can "jump" to any memory location in unit time, with no locality constraint).

On the other hand, if we care about polynomial differences in speed, and *especially* if we want to study parallel computing models, details about the spatial layout of the computing and memory elements (as well as the speed of communication among the elements) can become vitally important.

The class of problems solvable using a polynomial amount of memory (but possibly an exponential amount of time[61]) is called PSPACE, for Polynomial Space. Examples of PSPACE problems include simulating dynamical systems, deciding whether a regular grammar generates all possible strings, and executing an optimal strategy in two-player games such as Reversi, Connect Four, and Hex.[62] It is not hard to show that PSPACE is at least as powerful as NP:

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}.$$

Here EXP represents the class of problems solvable using an exponential amount of time, and also possibly an exponential amount of memory.[63] Every one of the above containments is believed to be strict, although the only one currently *proved* to be strict is $\mathsf{P} \neq \mathsf{EXP}$, by an important 1965 result of Hartmanis and Stearns [70] called the Time Hierarchy Theorem[64,65].

Notice, in particular, that $\mathsf{P} \neq \mathsf{NP}$ implies $\mathsf{P} \neq \mathsf{PSPACE}$. So while $\mathsf{P} \neq \mathsf{PSPACE}$ is not yet proved, it is an extremely secure conjecture by the standards of complexity theory. In slogan form,

[61]Why "only" an exponential amount? Because a Turing machine with B bits of memory can run for no more than 2^B time steps. After that, the machine must either halt or else return to a configuration previously visited (thereby entering an infinite loop).

[62]Note that, in order to speak about the computational complexity of such games, we first need to generalize them to an $n \times n$ board! But if we do so, then for many natural games, the problem of determining which player has the win from a given position is not only in PSPACE, but PSPACE-*complete* (i.e., it captures the entire difficulty of the class PSPACE). For example, Reisch [109] showed that this is true for Hex.

What about a suitable generalization of *chess* to an $n \times n$ board? That's also in PSPACE—but as far as anyone knows, only if we impose a polynomial upper bound on the number of moves in a chess game. Without such a restriction, Fraenkel and Lichtenstein [59] showed that chess is EXP-complete; with such a restriction, Storer [125] showed that chess is PSPACE-complete.

[63]In this context, we call a function $f(n)$ "exponential" if it can be upper-bounded by $2^{p(n)}$, for some polynomial p. Also, note that *more* than exponential memory would be useless here, since a Turing machine that runs for T time steps can visit at most T memory cells.

[64]More generally, the Time Hierarchy Theorem shows that, if f and g are any two "sufficiently well-behaved" functions that satisfy $f(n) \ll g(n)$ (for example: $f(n) = n^2$ and $g(n) = n^3$), then *there are computational problems solvable in $g(n)$ time but not in $f(n)$ time.* The proof of this theorem uses diagonalization, and can be thought of as a scaled-down version of Turing's proof of the unsolvability of the halting problem. That is, we argue that, *if* it were always possible to simulate a $g(n)$-time Turing machine by an $f(n)$-time Turing machine, then we could construct a $g(n)$-time machine that "predicted its own output in advance" and then output something else—thereby causing a contradiction.

Using similar arguments, we can show (for example) that there exist computational problems solvable using n^3 bits of memory but not using n^2 bits, and so on in most cases where we want to compare *more versus less of the same computational resource.* In complexity theory, the hard part is comparing two *different* resources: for example, determinism versus nondeterminism (the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem), time versus space ($\mathsf{P} \stackrel{?}{=} \mathsf{PSPACE}$), or classical versus quantum computation ($\mathsf{BPP} \stackrel{?}{=} \mathsf{BQP}$). For in those cases, diagonalization by itself no longer works.

[65]The fact that $\mathsf{P} \neq \mathsf{EXP}$ has an amusing implication, often attributed to Hartmanis: namely, *at least one* of the three inequalities

(i) $\mathsf{P} \neq \mathsf{NP}$

(ii) $\mathsf{NP} \neq \mathsf{PSPACE}$

(iii) $\mathsf{PSPACE} \neq \mathsf{EXP}$

must be true, even though proving any one of them to be true *individually* would represent a titanic advance in mathematics!

The above observation is sometimes offered as circumstantial evidence for $\mathsf{P} \neq \mathsf{NP}$. Of all our hundreds of unproved beliefs about inequalities between pairs of complexity classes, a large fraction of them *must* be correct, simply to avoid contradicting the hierarchy theorems. So then why not $\mathsf{P} \neq \mathsf{NP}$ in particular (given that our intuition there is stronger than our intuitions for most of the other inequalities)?

complexity theorists believe that *space is more powerful than time.*

Now, some people have asked how such a claim could possibly be consistent with modern physics. For didn't Einstein teach us that space and time are merely two aspects of the same structure? One immediate answer is that, even *within* relativity theory, space and time are not interchangeable: space has a positive signature whereas time has a negative signature. In complexity theory, the difference between space and time manifests itself in the straightforward fact that you can *reuse* the same memory cells over and over, but you can't reuse the same moments of time.[66]

Yet, as trivial as that observation sounds, it leads to an interesting thought. Suppose that the laws of physics let us travel *backwards* in time. In such a case, it's natural to imagine that time would become a "reusable resource" just like space is—and that, as a result, arbitrary PSPACE computations would fall within our grasp. But is that just an idle speculation, or can we rigorously justify it?

10.1 Closed Timelike Curves

Philosophers, like science-fiction fans, have long been interested in the possibility of closed timelike curves (CTCs), which arise in certain solutions to Einstein's field equations of general relativity.[67] On a traditional understanding, the central philosophical problem raised by CTCs is the *grandfather paradox*. This is the situation where you go back in time to kill your own grandfather, therefore you are never born, therefore your grandfather is *not* killed, therefore you *are* born, and so on. Does this contradiction immediately imply that CTCs are impossible?

No, it doesn't: we can only conclude that, *if* CTCs exist, then the laws of physics must somehow prevent grandfather paradoxes from arising. How could they do so? One classic illustration is that "when you go back in time to try and kill your grandfather, the gun jams"—or some other "unlikely" event inevitably occurs to keep the state of the universe consistent. But why should we imagine that such a convenient "out" will always be available, in every physical experiment involving CTCs? Normally, we like to imagine that we have the freedom to design an experiment however we wish, without Nature imposing conditions on the experiment (for example: "every gun must jam sometimes") whose reasons can only be understood in terms of distant or hypothetical events.

In his 1991 paper "Quantum mechanics near closed timelike lines," Deutsch [47] gave an elegant proposal for eliminating grandfather paradoxes. In particular he showed that, as long as we assume the laws of physics are quantum-mechanical (or even just classically probabilistic), every experiment involving a CTC admits at least one *fixed point*: that is, a way to satisfy the conditions of the experiment that ensures consistent evolution. Formally, if S is the mapping from quantum states to themselves induced by "going around the CTC once," then a fixed point is any quantum mixed state[68] ρ such that $S(\rho) = \rho$. The existence of such a ρ follows from simple linear-algebraic arguments. As one illustration, the "resolution of the grandfather paradox" is now that you are born with probability $1/2$, and *if* you are born, you go back in time to kill your grandfather—from

[66] See my blog post www.scottaaronson.com/blog/?p=368 for more on this theme.

[67] Though it is not known whether those solutions are "physical": for example, whether or not they can survive in a quantum theory of gravity (see [96] for example).

[68] In quantum mechanics, a *mixed state* can be thought of as a classical probability distribution over quantum states. However, an important twist is that the same mixed state can be represented by *different* probability distributions: for example, an equal mixture of the states $|0\rangle$ and $|1\rangle$ is physically indistinguishable from an equal mixture of $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. This is why mixed states are represented mathematically using Heisenberg's density matrix formalism.

which it follows that you are born with probability 1/2, and so on. Merely by treating states as probabilistic (as, in some sense, they *have* to be in quantum mechanics[69]), we have made the evolution of the universe consistent.

But Deutsch's account of CTCs faces at least three serious difficulties. The first difficulty is that the fixed points might not be *unique*: there could be many mixed states ρ such that $S(\rho) = \rho$, and then the question arises of how Nature chooses one of them. To illustrate, consider the *grandfather anti-paradox*: a bit $b \in \{0,1\}$ that travels around a CTC without changing. We can consistently assume $b = 0$, or $b = 1$, or any probabilistic mixture of the two—and unlike the usual situation in physics, here there is no possible boundary condition that could resolve the ambiguity.

The second difficulty, pointed out Bennett et al. [28], is that Deutsch's proposal violates the statistical interpretation of quantum mixed states. So for example, if half of an entangled pair

$$\frac{|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B}{\sqrt{2}}$$

is placed inside the CTC, while the other half remains outside the CTC, then the process of finding a fixed point will "break" the entanglement between the two halves. As a "remedy" for this problem, Bennett et al. suggest requiring the CTC fixed point ρ to be independent of the entire rest of the universe. To my mind, this remedy is so drastic that it basically amounts to defining CTCs out of existence!

Motivated by these difficulties, Lloyd et al. [90] recently proposed a completely different account of CTCs, based on *postselected teleportation*. Lloyd et al.'s account avoids both of the problems above—though perhaps not surprisingly, introduces other problems of its own.[70] My own view, for whatever it is worth, is that Lloyd et al. are talking less about "true" CTCs as I would understand the concept, as about postselected quantum-mechanical experiments that *simulate* CTCs in certain interesting respects. If there are any controversies in physics that call out for expert philosophical attention, surely this is one of them.

10.2 The Evolutionary Principle

Yet so far, we have not even mentioned what I see as the *main* difficulty with Deutsch's account of CTCs. This is that *finding* a fixed point might require Nature to solve an astronomically-hard computational problem! To illustrate, consider a science-fiction scenario wherein you go back in time and dictate Shakespeare's plays to him. Shakespeare thanks you for saving him the effort, publishes verbatim the plays that you dictated, and centuries later the plays come down to you, whereupon you go back in time and dictate them to Shakespeare, etc.

Notice that, in contrast to the grandfather paradox, here there is no logical contradiction: the story as we told it is entirely consistent. But most people find the story "paradoxical" anyway. After all, somehow *Hamlet* gets written, without anyone ever doing the work of writing it! As Deutsch [47] perceptively observed, if there is a "paradox" here, then it is not one of logic but of

[69]In more detail, Deutsch's proposal works if the state space consists of classical probability distributions $\mathcal{D}$ or quantum mixed states ρ, but *not* if it consists of pure states $|\psi\rangle$. Thus, *if* one believed that only pure states were fundamental in physics, and that probability distributions and mixed states always reflected subjective ignorance, one might reject Deutsch's proposal on that ground.

[70]In particular, in Lloyd et al.'s proposal, the only way to deal with the grandfather paradox is by some variant of "the gun jams": there *are* evolutions with no consistent solution, and it needs to be postulated that the laws of physics are such that they never occur.

computational complexity. Specifically, the story violates a commonsense principle that we can loosely articulate as follows:

Knowledge requires a causal process to bring it into existence.

Like many other important principles, this one might not be recognized as a "principle" at all before we contemplate situations that violate it! Deutsch [47] calls this principle the *Evolutionary Principle* (EP). Note that some version of the EP was invoked both by William Paley's blind-watchmaker argument, and (ironically) by the arguments of Richard Dawkins [45] and other atheists against the existence of an intelligent designer.

In my survey article "NP-Complete Problems and Physical Reality" [4], I proposed and defended a complexity-theoretic analogue of the EP, which I called the NP *Hardness Assumption*:

There is no physical means to solve NP-complete problems in polynomial time.

The above statement implies $\mathsf{P} \neq \mathsf{NP}$, but is stronger in that it encompasses probabilistic computing, quantum computing, and *any other computational model* compatible with the laws of physics. See [4] for a survey of recent results bearing on the NP Hardness Assumption, analyses of claimed counterexamples to the assumption, and possible implications of the assumption for physics.

10.3 Closed Timelike Curve Computation

But can we show more rigorously that closed timelike curves would *violate* the NP Hardness Assumption? Indeed, let us now show that, in a universe where arbitrary computations could be performed inside a CTC, and where Nature had to find a fixed point for the CTC, we could solve NP-complete problems using only polynomial resources.

We can model any NP-complete problem instance by a function $f : \{0, \ldots, 2^n - 1\} \to \{0, 1\}$, which maps each possible solution x to the bit 1 if x is valid, or to 0 if x is invalid. (Here, for convenience, we identify each n-bit solution string x with the nonnegative integer that x encodes in binary.) Our task, then, is to find an $x \in \{0, \ldots, 2^n - 1\}$ such that $f(x) = 1$. We can solve this problem with just a *single* evaluation to f, provided we can run the following computer program C inside a closed timelike curve [36, 4, 7]:

Given input $x \in \{0, \ldots, 2^n - 1\}$:
If $f(x) = 1$, then output x
Otherwise, output $(x + 1) \bmod 2^n$

Assuming there exists at least one x such that $f(x) = 1$, the only *fixed points* of C—that is, the only ways for C's output to equal its input—are for C to input, and output, such a valid solution x, which therefore appears in C's output register "as if by magic." (If there are no valid solutions, then C's fixed points will simply be uniform superpositions or probability distributions over *all* $x \in \{0, \ldots, 2^n - 1\}$.)

Extending the above idea, John Watrous and I [7] (following a suggestion by Fortnow) recently showed that a CTC computer in Deutsch's model could solve all problems in PSPACE. (Recall that PSPACE is believed to be even larger than NP.) More surprisingly, we also showed that PSPACE constitutes the *limit* on what can be done with a CTC computer; and that this is true whether the CTC computer is classical or quantum. One consequence of our results is that the "naïve intuition" about CTC computers—that their effect would be to "make space and time equivalent as computational resources"—is ultimately correct, although not for the naïve reasons.[71] A second, amusing consequence is that, once closed timelike curves are available, switching from classical to quantum computers provides no *additional* benefit!

It is important to realize that our algorithms for solving hard problems with CTCs do *not* just boil down to "using huge amounts of time to find the answer, then sending the answer back in time to before the computer started." For even in the exotic scenario of a time travel computer, we still require that all resources used *inside* the CTC (time, memory, etc.) be polynomially-bounded. Thus, the ability to solve hard problems comes solely from *causal consistency*: the requirement that Nature must find some evolution for the CTC computer that avoids grandfather paradoxes.

In Lloyd et al.'s alternative account of CTCs based on postselection [90], hard problems can *also* be solved, though for different reasons. In particular, building on an earlier result of mine [5], Lloyd et al. show that the power of their model corresponds to a complexity class called PP (Probabilistic Polynomial-Time), which is believed to be strictly smaller than PSPACE but strictly larger than NP. Thus, one might say that Lloyd et al.'s model "improves" the computational situation, but not by much!

So one might wonder: is there any way that the laws of physics could allow CTCs, *without* opening the door to implausible computational powers? There remains at least one interesting possibility, which was communicated to me by the philosopher Tim Maudlin.[72] Maybe the laws of physics have the property that, no matter what computations are performed inside a CTC, Nature always has an "out" that avoids the grandfather paradox, but *also* avoids solving hard computational problems—analogous to "the gun jamming" in the original grandfather paradox. Such an out might involve (for example) an asteroid hitting the CTC computer, or the computer failing for other mysterious reasons. Of course, *any* computer in the physical world has some nonzero probability of failure, but ordinarily we imagine that the failure probability can be made negligibly small. However, in situations where Nature is being "forced" to find a fixed point, maybe "mysterious computer failures" would become the norm rather than the exception.

To summarize, I think that computational complexity theory *changes* the philosophical issues raised by time travel into the past. While discussion traditionally focused on the grandfather paradox, we have seen that there is no shortage of ways for Nature to avoid logical inconsistencies, even in a universe with CTCs. The "real" problem, then, is how to escape the *other* paradoxes that arise in the course of taming the grandfather paradox! Probably foremost among those is the "computational complexity paradox," of NP-complete and even harder problems getting solved as if by magic.

11 Economics

In classical economics, agents are modeled as rational, Bayesian agents who take whatever actions will maximize their expected utility $\mathrm{E}_{\omega\in\Omega}\left[U\left(\omega\right)\right]$, given their subjective probabilities $\{p_\omega\}_{\omega\in\Omega}$ over

[71] Specifically, it is *not* true that in a CTC universe, a Turing machine tape head could just travel back and forth in time the same way it travels back and forth in space. If one thinks this way, then one really has in mind a second, "meta-time," while the "original" time has become merely one more dimension of space. To put the point differently: even though a CTC would make time *cyclic*, time would still retain its *directionality*. This is the reason why, if we want to show that CTC computers have the power of PSPACE, we need a nontrivial argument involving causal consistency.

[72] This possibility is also discussed at length in Deutsch's paper [47].

all possible states ω of the world.[73] This, of course, is a caricature that seems almost designed to be attacked, and it *has* been attacked from almost every angle. For example, humans are not even close to rational Bayesian agents, but suffer from well-known cognitive biases, as explored by Kahneman and Tversky [80] among others. Furthermore, the classical view seems to leave no room for critiquing people's beliefs (i.e., their prior probabilities) or their utility functions as irrational—yet it is easy to cook up prior probabilities or utility functions that would lead to behavior that almost anyone would consider insane. A third problem is that, in games with several cooperating or competing agents who act simultaneously, classical economics guarantees the existence of at least one *Nash equilibrium* among the agents' strategies. But the usual situation is that there are multiple equilibria, and then there is no general principle to predict which equilibrium will prevail, even though the choice might mean the difference between war and peace.

Computational complexity theory can contribute to debates about the foundations of economics by showing that, even in the idealized situation of rational agents who all have perfect information about the state of the world, it will often be *computationally intractable* for those agents to act in accordance with classical economics. Of course, some version of this observation has been recognized in economics for a long time. There is a large literature on *bounded rationality* (going back to the work of Herbert Simon [120]), which studies the behavior of economic agents whose decision-making abilities are limited in one way or another.

11.1 Bounded Rationality and the Iterated Prisoners' Dilemma

As one example of an insight to emerge from this literature, consider the Finite Iterated Prisoner's Dilemma. This is a game where two players meet for some fixed number of rounds N, which is finite and common knowledge between the players. In each round, both players can either "Defect" or "Cooperate" (not knowing the other player's choice), after which they receive the following payoffs:

	Defect_2	Cooperate_2
Defect_1	$1,1$	$4,0$
Cooperate_1	$0,4$	$3,3$

Both players remember the entire previous history of the interaction.

It is clear that the players will be jointly best off (i.e., will maximize the sum of their payoffs) if they both cooperate. Unfortunately, it is equally clear that, *if* the players meet for only one round ($N = 1$), then cooperation is not an *equilibrium*. Regardless of what Player 1 does, Player 2 will do better by defecting than by cooperating, and vice versa.

On the other hand, *if the number of rounds N were unknown or infinite*, then the players could rationally decide to cooperate, similarly to how humans decide to cooperate in real life. That is, Player 1 reasons that if he defects, then Player 2 might retaliate by defecting in future rounds, and Player 2 reasons likewise. Both players conclude that, over the long run, they'll do best *even just for themselves* by cooperating.

The "paradox" is now the following: as soon as the number of rounds N becomes *known*, the above reasoning completely collapses! For clearly, neither player can have anything to lose by defecting *in the very last round*, when the other player no longer has any chance to retaliate. So if both players are rational, then that is exactly what they will do. But if both players *know* that

[73] Here we assume for simplicity that the set Ω of possible states is countable; otherwise we could of course use a continuous probability measure.

both of them will defect in round N, then neither one has anything to lose by defecting in round $N-1$ *either*. Of course, they can continue inductively like this all the way back to the first round. We therefore get the "prediction" that both players will defect in every round, even though that is neither in the players' own interests, nor what actual humans do in experiments. And all because the number of rounds N was *known*: something that seems intuitively like it shouldn't make any difference!

In 1985, Neyman [98] proposed an ingenious resolution of this paradox. Specifically, he showed that if the two players have *sufficiently small memories*—technically, if they are finite automata with k states, for $2 \leq k < N$—then cooperation becomes an equilibrium once again! The basic intuition is that, if both players lack enough memory to count up to N, and both of them know that, and both know that they both know that, and so on, then the inductive argument in the last paragraph fails, since it assumes intermediate strategies that neither player can implement.

While complexity considerations vanquish *some* of the counterintuitive conclusions of classical economics, equally interesting to me is that they do not vanquish others. As one example, I showed in [3] that Robert Aumann's celebrated *agreement theorem* [19]—perfect Bayesian agents with common priors can never "agree to disagree"—persists even in the presence of limited communication between the agents.

There are many other interesting results in the bounded rationality literature, too many to do them justice here (but see Rubinstein [111] for a survey). On the other hand, "bounded rationality" is something of a catch-all phrase, encompassing almost every imaginable deviation from rationality—including human cognitive biases, limits on information-gathering and communication, and the restriction of strategies to a specific form (for example, linear threshold functions). Many of these deviations have little to do with computational complexity *per se*. So the question remains of whether computational complexity *specifically* can provide new insights about economic behavior.

11.2 The Complexity of Equilibria

There are some very recent advances suggesting that the answer is yes. Consider the problem of finding an equilibrium of a two-player game, given the $n \times n$ payoff matrix as input. In the special case of *zero-sum games* (which von Neumann studied in 1928), it has long been known how to solve this problem in an amount of time polynomial in n, for example by reduction to linear programming. But in 2006, Daskalakis, Goldberg, and Papadimitriou [44] (with improvements by Chen and Deng [39]) proved the spectacular result that, for a *general* (not necessarily zero-sum) two-player game, finding a Nash equilibrium is "PPAD-complete." Here PPAD ("Polynomial Parity Argument, Directed") is, roughly speaking, the class of *all* search problems for which a solution is guaranteed to exist for the same combinatorial reason that every game has at least one Nash equilibrium. Note that finding a Nash equilibrium *cannot* be NP-complete, for the technical reason that NP is a class of *decision* problems, and the answer to the decision problem "does this game have a Nash equilibrium?" is always yes. But Daskalakis et al.'s result says (informally) that the search problem of *finding* a Nash problem is "as close to NP-complete as it could possibly be," subject to its decision version being trivial. Similar PPAD-completeness results are now known for other fundamental economic problems, such as finding market-clearing prices in Arrow-Debreu markets [38].

Of course, one can debate the economic relevance of these results: for example, how often does the computational hardness that we now know[74] to be inherent in economic equilibrium theorems actually rear its head in practice? But one can similarly debate the economic relevance of the equilibrium theorems themselves! In my opinion, if the theorem that Nash equilibria *exist* is considered relevant to debates about (say) free markets versus government intervention, then the theorem that *finding* those equilibria is PPAD-complete should be considered relevant also.

12 Conclusions

The purpose of this essay was to illustrate how philosophy could be enriched by taking computational complexity theory into account, much as it was enriched almost a century ago by taking computability theory into account. In particular, I argued that computational complexity provides new insights into the explanatory content of Darwinism, the nature of mathematical knowledge and proof, computationalism, syntax versus semantics, the problem of logical omniscience, debates surrounding the Turing Test and Chinese Room, the problem of induction, the foundations of quantum mechanics, closed timelike curves, and economic rationality.

Indeed, one might say that the "real" question is which philosophical problems *don't* have important computational complexity aspects! My own opinion is that there probably *are* such problems (even within analytic philosophy), and that one good candidate is the problem of what we should take as "bedrock mathematical reality": that is, the set of mathematical statements that are objectively true or false, regardless of whether they can be proved or disproved in a given formal system. To me, if we are not willing to say that a given Turing machine M either accepts, rejects, or runs forever (when started on a blank tape)—and that which one it does is an objective fact, independent of our formal axiomatic theories, the laws of physics, the biology of the human brain, cultural conventions, etc.—then *we have no basis to talk about any of those other things* (axiomatic theories, the laws of physics, and so on). Furthermore, M's resource requirements are irrelevant here: even if M only halts after $2^{2^{10000}}$ steps, its output is as mathematically definite as if it had halted after 10 steps.[75]

Can we say anything *general* about when a computational complexity perspective is helpful in philosophy, and when it isn't? Extrapolating from the examples in this essay, I would say that computational complexity tends to be helpful when we want to know whether a particular fact *does any explanatory work*: Sections 3.2, 3.3, 4, 6, and 7 all provided examples of this. Other "philosophical applications" of complexity theory come from the Evolutionary Principle and the NP Hardness Assumption discussed in Section 10.2. *If* we believe that certain problems are computationally intractable, then we may be able to draw interesting conclusions from that belief about economic rationality, quantum mechanics, the possibility of closed timelike curves, and other issues. By contrast, computational complexity tends to be *un*helpful when we only want to know whether a particular fact "determines" another fact, and don't care about the length of the inferential chain.

[74]Subject, as usual, to widely-believed complexity assumptions.

[75]The situation is very different for mathematical statements like the Continuum Hypothesis, which *can't* obviously be phrased as predictions about idealized computational processes (since they're not expressible by first-order or even second-order quantification over the integers). For those statements, it really *is* unclear to me what one means by their truth or falsehood apart from their provability in some formal system.

12.1 Criticisms of Complexity Theory

Despite its explanatory reach, complexity theory has been criticized on various grounds. Here are four of the most common criticisms:

(1) Complexity theory only makes *asymptotic* statements (statements about how the resources needed to solve problem instances of size n scale as n goes to infinity). But as a matter of logic, asymptotic statements need not have *any implications whatsoever* for the finite values of n (say, $10,000$) that humans care actually about, nor can any finite amount of experimental data confirm or refute an asymptotic claim.

(2) Many of (what we would like to be) complexity theory's basic principles, such as $\mathsf{P} \neq \mathsf{NP}$, are currently unproved mathematical conjectures, and will probably remain that way for a long time.

(3) Complexity theory focuses on only a limited type of computer—the serial, deterministic Turing machine—and fails to incorporate the "messier" computational phenomena found in nature.

(4) Complexity theory studies only the *worst-case* behavior of algorithms, and does not address whether that behavior is representative, or whether it merely reflects a few "pathological" inputs. So for example, even if $\mathsf{P} \neq \mathsf{NP}$, there might still be excellent heuristics to solve *most* instances of NP-complete problems that actually arise in practice; complexity theory tells us nothing about such possibilities one way or the other.

For whatever it's worth, criticisms (3) and (4) have become much less accurate since the 1980s. As discussed in this essay, complexity theory has by now branched out far beyond deterministic Turing machines, to incorporate (for example) quantum mechanics, parallel and distributed computing, and stochastic processes such as Darwinian evolution. Meanwhile, although worst-case complexity remains the best-understood kind, today there is a large body of work—much of it driven by cryptography—that studies the *average-case* hardness of computational problems, for various probability distributions over inputs. And just as almost all complexity theorists believe that $\mathsf{P} \neq \mathsf{NP}$, so almost all subscribe to the stronger belief that there exist *hard-on-average* NP problems—indeed, that belief is one of the underpinnings of modern cryptography. A few problems, such as calculating discrete logarithms, are even known to be *just as hard on random inputs as they are on the hardest possible input* (though whether such "worst-case/average-case equivalence" holds for any NP-complete problem remains a major open question). For these reasons, although speaking about average-case rather than worst-case complexity would complicate some of the arguments in this essay, I don't think it would change the conclusions much.[76] See Bogdanov and Trevisan [32] for an excellent recent survey of average-case complexity, and Impagliazzo [78] for an evocative discussion of complexity theory's "possible worlds" (for example, the "world" where NP-complete problems turn out to be hard in the worst case but easy on average).

The broader point is that, even if we admit that criticisms (1)-(4) have merit, that does not give us a license to dismiss complexity-theoretic arguments whenever we dislike them! In science,

[76] On the other hand, it *would* presuppose that we knew how to define reasonable probability distributions over inputs. But as discussed in Section 4.3, it seems hard to explain what we mean by "structured instances," or "the types of instances that normally arise in practice."

we only ever deal with imperfect, approximate theories—and if we reject the conclusions of the *best* approximate theory in some area, then the burden is on us to explain why.

To illustrate, suppose you believe that quantum computers will never give a speedup over classical computers for any practical problem. Then as an explanation for your stance, you might assert any of the following:

(a) Quantum mechanics is false or incomplete, and an attempt to build a scalable quantum computer would instead lead to falsifying or extending quantum mechanics itself.

(b) There exist polynomial-time *classical* algorithms for factoring integers, and for all the other problems that admit polynomial-time quantum algorithms. (In complexity terms, the classes BPP and BQP are equal.)

(c) The "constant-factor overheads" involved in building a quantum computer are so large as to negate their asymptotic advantages, for any problem of conceivable human interest.

(d) While we don't yet know which of (a)-(c) holds, we can know on some *a priori* ground that at least one of them has to hold.

The point is that, even if we can't answer every possible shortcoming of a complexity-theoretic analysis, we can still use it to *clarify the choices*: to force people to lay some cards on the table, committing themselves either to a prediction that might be falsified or to a mathematical conjecture that might be disproved. Of course, this is a common feature of *all* scientific theories, not something specific to complexity theory. If complexity theory is unusual here, it is only in the number of "predictions" it juggles that could be confirmed or refuted by mathematical proof (and indeed, *only* by mathematical proof).[77]

12.2 Future Directions

Even if the various criticisms of complexity theory don't negate its relevance, it would be great to address those criticisms head-on—and more generally, to get a clearer understanding of the relationship between complexity theory and the real-world phenomena that it tries to explain. Toward that end, I think the following questions would all benefit from careful philosophical analysis:

- What is the empirical status of asymptotic claims? What sense can we give to an asymptotic statement "making predictions," or being supported or ruled out by a finite number of observations?

- How can we explain the empirical facts on which complexity theory relies: for example, that we rarely see n^{10000} or 1.0000001^n algorithms, or that the computational problems humans care about tend to organize themselves into a relatively-small number of equivalence classes?

- Short of proof, how do people form intuitions about the truth or falsehood of mathematical conjectures? What *are* those intuitions, in cases such as $\mathsf{P} \neq \mathsf{NP}$?

[77] One other example that springs to mind, of a scientific theory many of whose "predictions" take the form of mathematical conjectures, is string theory.

- If we think of complexity classes (P, NP, etc.) as collections of problems, then it becomes harder to understand results like those of Baker, Gill, and Solovay [21], which says that there exists an "oracle" A such that $\mathsf{P}^A \neq \mathsf{NP}^A$ (and furthermore, this remains true even if $\mathsf{P} = \mathsf{NP}$ in the "real" world). In philosophical terms, should we think of a complexity class as corresponding to its *definition* rather than its *extension*?

- Do the conceptual conclusions that people sometimes want to draw from conjectures such as $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{BPP} \neq \mathsf{BQP}$—for example, about the nature of mathematical creativity or the interpretation of quantum mechanics—actually depend on those conjectures being true? Are there easier-to-prove statements that would arguably support the same conclusions?

- If $\mathsf{P} \neq \mathsf{NP}$, then how have humans managed to make such enormous mathematical progress, even in the face of the general intractability of theorem-proving? Is there a "selection effect," by which mathematicians favor problems with special structure that makes them easier to solve than arbitrary problems? If so, then what does this structure consist of?

In short, I see plenty of scope for the converse essay to this one: "Why Computational Complexity Theorists Should Care About Philosophy."

13 Acknowledgments

I am grateful to Oron Shagrir for pushing me to finish this essay, for helpful comments, and for suggesting Section 7.2; to Alex Byrne for suggesting Section 6; to Agustín Rayo for suggesting Section 5; and to David Aaronson, Seamus Bradley, Terrence Cole, Michael Collins, Andy Drucker, Michael Forbes, Oded Goldreich, Bob Harper, Gil Kalai, Dana Moshkovitz, Jan Arne Telle, Dylan Thurston, Ronald de Wolf, Avi Wigderson, and Joshua Zelinsky for their feedback.

References

[1] S. Aaronson. Shor, I'll do it (weblog entry). www.scottaaronson.com/blog/?p=208.

[2] S. Aaronson. Multilinear formulas and skepticism of quantum computing. In *Proc. ACM STOC*, pages 118–127, 2004. quant-ph/0311039.

[3] S. Aaronson. The complexity of agreement. In *Proc. ACM STOC*, pages 634–643, 2005. ECCC TR04-061.

[4] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, March 2005. quant-ph/0502072.

[5] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proc. Roy. Soc. London*, A461(2063):3473–3482, 2005. quant-ph/0412187.

[6] S. Aaronson and G. Kuperberg. Quantum versus classical proofs and advice. *Theory of Computing*, 3(7):129–157, 2007. Previous version in Proceedings of CCC 2007. quant-ph/0604056.

[7] S. Aaronson and J. Watrous. Closed timelike curves make quantum and classical computing equivalent. *Proc. Roy. Soc. London*, (A465):631–647, 2009. arXiv:0808.2669.

[8] S. Aaronson and A. Wigderson. Algebrization: a new barrier in complexity theory. *ACM Trans. on Computation Theory*, 1(1), 2009. Conference version in Proc. ACM STOC 2008.

[9] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. www.cse.iitk.ac.in/users/manindra/primality.ps, 2002.

[10] D. Aharonov. Quantum computation - a review. In Dietrich Stauffer, editor, *Annual Review of Computational Physics*, volume VI. 1998. quant-ph/9812037.

[11] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[12] D. Angluin, J. Aspnes, J. Chen, and Y. Wu. Learning a circuit by injecting values. *J. Comput. Sys. Sci.*, 75(1):60–77, 2009. Earlier version in STOC'2006.

[13] K. Appel and W. Haken. *Every planar map is four-colorable.* American Mathematical Society, 1989.

[14] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proc. IEEE FOCS*, pages 211–220, 2008.

[15] S. Arora and B. Barak. *Complexity Theory: A Modern Approach.* Cambridge University Press, 2009. Online draft at www.cs.princeton.edu/theory/complexity/.

[16] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. Manuscript, 1992.

[17] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[18] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[19] R. J. Aumann. Agreeing to disagree. *Annals of Statistics*, 4(6):1236–1239, 1976.

[20] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[21] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM J. Comput.*, 4:431–442, 1975.

[22] E. B. Baum. *What Is Thought?* Bradford Books, 2004.

[23] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Earlier version in IEEE FOCS 1998, pp. 352-361. quant-ph/9802049.

[24] P. Beame and T. Pitassi. Propositional proof complexity: past, present, and future. *Current Trends in Theoretical Computer Science*, pages 42–70, 2001.

[25] S. Bellantoni and S. A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992. Earlier version in STOC 1992, p. 283-293.

[26] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove the intractability assumptions. In *Proc. ACM STOC*, pages 113–131, 1988.

[27] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. quant-ph/9701001.

[28] C. H. Bennett, D. Leung, G. Smith, and J. A. Smolin. Can closed timelike curves or nonlinear quantum mechanics improve quantum state discrimination or help solve hard problems? *Phys. Rev. Lett.*, 103(170502), 2009. arXiv:0908.3023.

[29] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. First appeared in ACM STOC 1993.

[30] N. Block. Searle's arguments against cognitive science. In J. Preston and M. Bishop, editors, *Views into the Chinese Room: New Essays on Searle and Artificial Intelligence*, pages 70–79. Oxford, 2002.

[31] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[32] A. Bogdanov and L. Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. ECCC TR06-073.

[33] R. B. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Inform. Proc. Lett.*, 25:127–132, 1987.

[34] R. Bousso. Positive vacuum energy and the N-bound. *J. High Energy Phys.*, 0011(038), 2000. hep-th/0010252.

[35] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Sys. Sci.*, 37(2):156–189, 1988.

[36] T. Brun. Computers with closed timelike curves can solve hard problems. *Foundations of Physics Letters*, 16:245–253, 2003. gr-qc/0209061.

[37] D. J. Chalmers. *The Conscious Mind: In Search of a Fundamental Theory*. Oxford, 1996.

[38] X. Chen, D. Dai, Y. Du, and S.-H. Teng. Settling the complexity of Arrow-Debreu equilibria in markets with additively separable utilities. In *Proc. IEEE FOCS*, pages 273–282, 2009.

[39] X. Chen and X. Deng. Settling the complexity of two-player Nash equilibrium. In *Proc. IEEE FOCS*, pages 261–271, 2006.

[40] C. Cherniak. Computational complexity and the universal acceptance of logic. *The Journal of Philosophy*, 81(12):739–758, 1984.

[41] R. Cleve, P. Høyer, B. Toner, and J. Watrous. Consequences and limits of nonlocal strategies. In *Proc. IEEE Conference on Computational Complexity*, pages 236–249, 2004. quant-ph/0404076.

[42] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of Logic, Methodology, and Philosophy of Science II*. North Holland, 1965.

[43] J. Copeland. Hypercomputation. *Minds and Machines*, 12:461–502, 2002.

[44] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *Commun. ACM*, 52(2):89–97, 2009. Earlier version in Proceedings of STOC'2006.

[45] R. Dawkins. *The God Delusion*. Houghton Mifflin Harcourt, 2006.

[46] D. C. Dennett. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon & Schuster, 1995.

[47] D. Deutsch. Quantum mechanics near closed timelike lines. *Phys. Rev. D*, 44:3197–3217, 1991.

[48] D. Deutsch. *The Fabric of Reality*. Penguin, 1998.

[49] D. Deutsch. *The Beginning of Infinity: Explanations that Transform the World*. Allen Lane, 2011.

[50] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[51] A. Drucker. Multiplying 10-digit numbers using Flickr: the power of recognition memory. people.csail.mit.edu/andyd/rec_method.pdf, 2011.

[52] R. Fagin. Finite model theory - a personal perspective. *Theoretical Comput. Sci.*, 116:3–31, 1993.

[53] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[54] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[55] R. P. Feynman. Simulating physics with computers. *Int. J. Theoretical Physics*, 21(6-7):467–488, 1982.

[56] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–244, February 1994.

[57] L. Fortnow. One complexity theorist's view of quantum computing. *Theoretical Comput. Sci.*, 292(3):597–610, 2003.

[58] L. Fortnow and S. Homer. A short history of computational complexity. *Bulletin of the EATCS*, (80):95–133, 2003.

[59] A. Fraenkel and D. Lichtenstein. Computing a perfect strategy for nxn chess requires time exponential in n. *Journal of Combinatorial Theory A*, 31:199–214, 1981.

[60] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. ACM STOC*, pages 169–178, 2009.

[61] O. Goldreich. On quantum computing. www.wisdom.weizmann.ac.il/~oded/on-qc.html, 2004.

[62] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. Earlier version at www.wisdom.weizmann.ac.il/~oded/cc-drafts.html.

[63] O. Goldreich. *A Primer on Pseudorandom Generators*. American Mathematical Society, 2010. www.wisdom.weizmann.ac.il/~oded/PDF/prg10.pdf.

[64] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1984.

[65] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(1):691–729, 1991.

[66] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186208, 1989.

[67] N. Goodman. *Fact, Fiction, and Forecast*. Harvard University Press, 1955.

[68] P. Graham. How to do philosophy. www.paulgraham.com/philosophy.html, 2007.

[69] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

[70] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[71] J. Haugeland. Syntax, semantics, physics. In J. Preston and M. Bishop, editors, *Views into the Chinese Room: New Essays on Searle and Artificial Intelligence*, pages 379–392. Oxford, 2002.

[72] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[73] M. Hogarth. Non-Turing computers and non-Turing computability. *Biennial Meeting of the Philosophy of Science Association*, 1:126–138, 1994.

[74] A. S. Holevo. Some estimates of the information transmitted by quantum communication channels. *Problems of Information Transmission*, 9:177–183, 1973. English translation.

[75] G. 't Hooft. Quantum gravity as a dissipative deterministic system. *Classical and Quantum Gravity*, 16:3263–3279, 1999. gr-qc/9903084.

[76] D. Hume. *An Enquiry concerning Human Understanding*. 1748. 18th.eserver.org/hume-enquiry.html.

[77] N. Immerman. *Descriptive Complexity*. Springer, 1998.

[78] R. Impagliazzo. A personal view of average-case complexity. In *Proc. IEEE Conference on Computational Complexity*, pages 134–147, 1995.

[79] R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the XOR Lemma. In *Proc. ACM STOC*, pages 220–229, 1997.

[80] D. Kahneman, P. Slovic, and A. Tversky. *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, 1982.

[81] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[82] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[83] J. Kempe, H. Kobayashi, K. Matsumoto, B. Toner, and T. Vidick. Entangled games are hard to approximate. *SIAM J. Comput.*, 40(3):848–877, 2011. Earlier version in FOCS'2008. arXiv:0704.2903.

[84] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31:1501–1526, 2002. Earlier version in ACM STOC 1999.

[85] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22:155–171, 1975.

[86] D. Leivant. A foundational delineation of poly-time. *Information and Computation*, 110(2):391–420, 1994. Earlier version in LICS (Logic In Computer Science) 1991, p. 2-11.

[87] H. J. Levesque. Is it enough to get the behavior right? In *Proceedings of IJCAI*, pages 1439–1444, 2009.

[88] L. A. Levin. Polynomial time and extravagant models, in The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003. cs.CR/0012023.

[89] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (3rd ed.)*. Springer, 2008.

[90] S. Lloyd, L. Maccone, R. Garcia-Patron, V. Giovannetti, and Y. Shikano. The quantum mechanics of time travel through post-selected teleportation. *Phys. Rev. D*, 84(025007), 2011. arXiv:1007.2615.

[91] J. R. Lucas. Minds, machines, and Gödel. *Philosophy*, 36:112–127, 1961.

[92] N. D. Mermin. From cbits to qbits: teaching computer scientists quantum mechanics. *American J. Phys.*, 71(1):23–30, 2003. quant-ph/0207118.

[93] N. D. Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007.

[94] S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[95] C. Moore and S. Mertens. *The Nature of Computation*. Oxford University Press, 2011.

[96] M. S. Morris, K. S. Thorne, and U. Yurtsever. Wormholes, time machines, and the weak energy condition. *Phys. Rev. Lett.*, 61:1446–1449, 1988.

[97] A. Morton. Epistemic virtues, metavirtues, and computational complexity. *Noûs*, 38(3):481–502, 2004.

[98] A. Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters*, 19(3):227–229, 1985.

[99] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[100] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[101] I. Parberry. Knowledge, understanding, and computational complexity. In D. S. Levine and W. R. Elsberry, editors, *Optimality in Biological and Artificial Networks?*, pages 125–144. Lawrence Erlbaum Associates, 1997.

[102] R. Penrose. *The Emperor's New Mind*. Oxford, 1989.

[103] R. Penrose. *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford, 1996.

[104] L. Pitt and L. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

[105] C. Pomerance. A tale of two sieves. *Notices of the American Mathematical Society*, 43(12):1473–1485, 1996.

[106] H. Putnam. *Representation and Reality*. Bradford Books, 1991.

[107] R. Raz. Exponential separation of quantum and classical communication complexity. In *Proc. ACM STOC*, pages 358–367, 1999.

[108] A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. Sys. Sci.*, 55(1):24–35, 1997.

[109] S. Reisch. Hex is PSPACE-complete. *Acta Informatica*, 15:167–191, 1981.

[110] H. E. Rose. *Subrecursion: Functions and Hierarchies*. Clarendon Press, 1984.

[111] A. Rubinstein. *Modeling Bounded Rationality*. MIT Press, 1998.

[112] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, (7):281–292, 1971.

[113] J. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(417-457), 1980.

[114] J. Searle. *The Rediscovery of the Mind*. MIT Press, 1992.

[115] A. Shamir. IP=PSPACE. *J. ACM*, 39(4):869–877, 1992.

[116] S. M. Shieber. The Turing test as interactive proof. *Noûs*, 41(4):686–713, 2007.

[117] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. quant-ph/9508027.

[118] H. T. Siegelmann. Neural and super-Turing computing. *Minds and Machines*, 13(1):103–114, 2003.

[119] D. Simon. On the power of quantum computation. In *Proc. IEEE FOCS*, pages 116–123, 1994.

[120] H. A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.

[121] M. Sipser. The history and status of the P versus NP question. In *Proc. ACM STOC*, pages 603–618, 1992.

[122] M. Sipser. *Introduction to the Theory of Computation (Second Edition)*. Course Technology, 2005.

[123] R. Stalnaker. The problem of logical omniscience, I and II. In *Context and Content: Essays on Intentionality in Speech and Thought*, Oxford Cognitive Science Series, pages 241–273. Oxford University Press, 1999.

[124] L. J. Stockmeyer. Classifying the computational complexity of problems. *J. Symbolic Logic*, 52(1):1–43, 1987.

[125] J. A. Storer. On the complexity of chess. *J. Comput. Sys. Sci.*, 27(1):77–100, 1983.

[126] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[127] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[128] L. G. Valiant. Evolvability. *J. ACM*, 56(1), 2009. Conference version in MFCS 2007. ECCC TR06-120.

[129] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[130] H. Wang. *A Logical Journey: From Gödel to Philosophy*. MIT Press, 1997.

[131] J. Watrous. Succinct quantum proofs for properties of finite groups. In *Proc. IEEE FOCS*, pages 537–546, 2000. cs.CC/0009002.

[132] J. Watrous. Quantum computational complexity. In *Encyclopedia of Complexity and Systems Science*. Springer, 2008. arXiv:0804.3401.

[133] A. Wigderson. P, NP and mathematics - a computational complexity perspective. In *Proceedings of the International Congress of Mathematicians 2006 (Madrid)*, pages 665–712. EMS Publishing House, 2007. www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/W06/w06.pdf.

[134] A. Wigderson. Knowledge, creativity and P versus NP, 2009. www.math.ias.edu/˜avi/PUBLICATIONS/MYPAPERS/AW09/AW09.pdf.

[135] E. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications in Pure and Applied Mathematics*, 13(1), 1960.

[136] R. de Wolf. Philosophical applications of computational learning theory: Chomskyan innateness and occam's razor. Master's thesis, Erasmus University, 1997. homepages.cwi.nl/˜rdewolf/publ/philosophy/phthesis.pdf.