CHAPTER 41

Is the whole universe a computer?

JACK COPELAND, MARK SPREVAK, and ORON SHAGRIR

The theory that the whole universe is a computer is a bold and striking one. It is a theory of *everything*: the entire universe is to be understood, fundamentally, in terms of the universal computing machine that Alan Turing introduced in 1936. We distinguish between two versions of this grand-scale theory and explain what the universe would have to be like for one or both versions to be true. Spoiler: the question is in fact wide open – at the present stage of science, nobody knows whether it's true or false that the whole universe is a computer. But the issues are as fascinating as they are important, so it's certainly worthwhile discussing them. We begin right at the beginning: what exactly *is* a computer?

What is a computer?

To start with the obvious, your laptop is a computer. But there are also computers very different from your laptop – tiny embedded computers inside watches, and giant networked supercomputers like China's *Tianhe-2*, for example. So what feature do all computers have in common? What is it that makes them all *computers*?

Colossus was a computer, even though (as Chapter 14 explained) it did not make use of stored programs and could do very few of the things that your laptop can do (not even long multiplication). Turing's ACE (Chapters 21 and 22) was a computer, even though its design was dissimilar from that of your laptop – for example, the ACE had no central processing unit (CPU), and moreover it stored its data and programs in the form of 'pings' of supersonic sound travelling along tubes of liquid. Turing's artificial neural nets were also computers (Chapter 29), and so are the modern brain-mimicking 'connectionist' networks that Turing anticipated. In connectionist networks – as in your brain, but unlike your laptop – there is no separation between memory and processing, and the very same 'hardware' that does the processing (the neurons and their connections) functions also as the memory. Even Babbage's Analytical Engine (Chapter 24) was a computer, despite being built of mechanical rather than electrical parts. As Turing said:[1]

'The fact that Babbage's Analytical Engine was to be entirely mechanical will help us to rid ourselves of a superstition. Importance is often attached to the fact that modern digital computers are electrical, and that the nervous system also is electrical. Since Babbage's

machine was not electrical … we see that this use of electricity cannot be of theoretical importance.'

In fact, there is astonishing variety among the computers that are currently being researched or prototyped by computer scientists. There are massively parallel and distributed computers, asynchronous computers (i.e. computers with no central 'clock' coordinating the processing), nano computers, quantum computers, chemical computers, DNA computers, evolutionary computers, slime-mould computers, computers that use billiard balls, and computers that use swarms of animals or insects to solve problems… The list goes on. There could in principle even be a universal computer consisting entirely of mirrors and beams of light.[2] What, then, do all these different forms of computer have in common? Let's examine what Turing said of relevance to this question.

Before the modern era, the word 'computer' referred to a human being. If someone spoke of a computer in the 19th century, or even in 1936, they would have been taken to be referring to a *human* computer – a clerk who performed the tedious job of routine numerical computation. There used to be many thousands of human computers employed in businesses, government departments, research establishments and elsewhere. In 1936, Turing introduced his 'logical computing machines' – Turing machines – so as to provide an idealized description of the human computer: in fact he began his account of the Turing machine: 'We may compare a man in the process of computing a ... number to a machine'.[3] Cambridge philosopher Ludwig Wittgenstein, well known for his pithy and penetrating statements, put the point like this:[4]

'Turing's "Machines". These machines are *humans* who calculate.'

Turing often emphasized the fundamental point that the Turing machine is a model (idealized in certain respects) of the human computer. For example:[5]

'A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine.'

Even in his discussions of the ACE (Automatic Computing Engine) Turing continued to use the word 'computer' to mean 'human computer':[6]

'Computers always spend just as long in writing numbers down and deciding what to do next as they do in actual multiplications, and it is just the same with ACE ... [T]he ACE will do the work of about 10,000 computers ... Computers will still be employed on small calculations.'

So there were on the one hand computers – human beings – and on the other hand *machines* that could take over aspects of the computers' work. The term 'computing machine' was used increasingly from the 1920s to refer to small calculating machines that mechanized elements of the human computer's work. When the phrase 'electronic computer' came along

in the 1940s, it also referred to a machine that mechanized the work of the human computer. Turing made this explicit:[7]

'The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer'.

Turing considered this characterization of the concept *digital computer* to be so important that he began his *Programmers' Handbook for Manchester Electronic Computer Mark II* with the following statement:[8]

'Electronic computers are intended to carry out any definite rule of thumb process which could have been done by a human operator working in a disciplined but unintelligent manner.'

Here, then, is the Turing-style answer to the question: 'What is a computer, in the modern sense?'.[9] Any physical mechanism that carries out the same work as the idealized human computer is itself a computer. (The human computer is idealized in that no limit is placed on the amount of time available to the human computer, nor on the quantity of paper and pencils available – idealized human computers live indefinitely long, and never get bored.) The computer carries out tasks that can, in principle, be done by a human rote-worker following an algorithm (and no other tasks) – tasks that, as Turing put it[10], can be done

'by human clerical labour, working to fixed rules, and without understanding.'

With this clarification in place we turn next to the important distinction between *conventional* and *unconventional* computers. Modern laptops, tablets, minis and mainframes are conventional computers, while slime-mould computers and swarm computers are not. Conventional computers derive ultimately from the design set out in the famous 1945 proposal 'First Draft of a Report on the EDVAC' (Chapter 20) and they consist fundamentally of two parts, the CPU and the memory. A conventional computer's basic cycle of activity is the 'fetch-operate-store' cycle: operands (numbers) are fetched from memory, operated on in the CPU (e.g. multiplied together), and the result of the operation (another number) is routed back to the memory and stored. Any computer that does not fit this description is *un*conventional.

Is the universe a conventional computer, a cosmic version of your laptop or of *Tianhe-2*? This seems to us logically possible but not terribly likely. Where is the cosmic computer's CPU? Where is the cosmic computer's memory? Where are the registers holding the operands, and the registers in which the results of the CPU's operations are stored? There's no evidence of the universe containing any of these core elements of the conventional computer.

However, the Californian philosopher John Searle argues that even your garden wall is a conventional computer; and other philosophers maintain that a simple rock standing

motionless on a beach is a (conventional) computer – and so, if Searle et al. are right, the entire universe is by the same token a gigantic conventional computer.[11] These claims about walls and rocks, even if ultimately absurd, deserve a detailed discussion, but since we are after more important quarry we shall not pause to give this discussion here: interested readers will find a critique of these claims in the references given in the endnote.[12] Turning away from the idea that the universe is a conventional computer, we are going to discuss the more promising hypothesis that the universe is a computer of a type first introduced by John von Neumann and mentioned by Stephen Wolfram in Chapter 5: a *cellular automaton*.[13]

Zuse's thesis

Konrad Zuse, who appears briefly in Chapters 6 and 31, built his first computers before the War – in the living room of his parents' Berlin apartment.[14] As an engineering student at the Technical University in Berlin-Charlottenburg, Zuse had become painfully aware that engineers must perform what he called 'big and awful calculations'.[15] 'That is really not right for a man', he said:[16]

'It's beneath a man. That should be accomplished with machines.'

After the War Zuse supplied Europe with cheap relay-based computers, and later transistorized computers, from his factory in Bad Hersfeld. Even though he had anticipated elements of the stored-program concept, in a 1936 patent application, it was not until the 1960s that he began to include stored programming in his computers.[17] (It is sometimes said in the historical literature that Zuse's 1941 Z3 computer was a stored-program machine but this is an error.) Whether Zuse and Turing ever met in person is uncertain. Interestingly Zuse stated that he had no knowledge of Turing's 1936 article 'On Computable Numbers' until 1948, the year that he was summoned from Germany to London to be interrogated by British computing experts.[18] Donald Davies, Turing's assistant at the NPL, was one of the interviewers: Zuse eventually 'got pretty cross', Davies recollected, and things 'degenerated into a glowering match'.[19] Zuse seemed 'quite convinced' (Davies continued) that he could make a smallish relay machine 'which would be the equal of any of the electronic calculators we were developing'.

Fig. 41.1 Konrad Zuse, 1910-1995.

Zuse's 1967 book *Rechnender Raum* ('Space Computes') sketched a new – even mind-bending – framework for fundamental physics. Zuse's thesis is that the universe is a giant digital computer, a cellular automaton (CA).[20] According to Zuse the universe is, at bottom, nothing more than a collection of 1s and 0s changing state according to

computational rules. Everything that is familiar in physics – force, energy, entropy, mass, particles – emerges from that cosmic computation.

Stephen Wolfram explains that cellular automata are lattice-like grids, all of whose properties are *discrete*: they are[21]

'systems in which space and time are discrete, and physical quantities take on a finite set of discrete values. … A cellular automaton evolves in discrete time steps.'

A CA is very different from a conventional computer. To visualize a CA, picture a two-dimensional grid made up from square cells. As the CA's time ticks forward in discrete steps, each cell in the grid is at any moment in one or other of two states, 'on' or 'off'. The CA's 'transition rules' describe how the cells' states at one time-step determine their states at the next time-step. At the start of the process, some of the grid's cells are 'on' and others are 'off'; and as time ticks forwards, cells turn on or off according to the transition rules. At some point the grid may reach what is called a 'halting' state: the computation is completed and the output can be read off from the remaining pattern of activity on the grid.

Just as your laptop can solve computational problems (such as calculating how many tiles of a certain size and shape you will need to tile your bathroom floor, or solving some humungous mathematical equation), CAs can also solve such problems. The problem is encoded in the grid's initial pattern of activity, and once the grid reaches its 'halting' state, the user reads off the solution from the residual pattern of activity. Different CAs can have different transition rules; and some may have different kinds of grid, or more than just two possible cell-states. Even though CAs are remarkably different from conventional computers, it nevertheless turns out that if a problem can be solved by a conventional computer then it can also be solved by a CA (and vice versa): different computational *architecture*, but same computational *power*.

In 1970 the British mathematician John Conway invented a CA engagingly called the 'Game of Life'. This CA has four very simple transition rules (see box). Conway noted an interesting fact about the Game of Life: through applying these simple rules to small-scale patterns on the grid, large-scale patterns of surprising complexity emerge.

Format as a box

The Game of Life has just four transition rules:

1. If a cell is **on**, and fewer than 2 of its neighbours are also on, it will turn **off** at the next time-step.
2. If a cell is **on**, and either 2 or 3 of its neighbours are also on, it will stay **on** at the next time-step.
3. If a cell is **on** and more than 3 of its neighbours are on, it will turn **off** at the next time-step.
4. If a cell is **off** and exactly 3 of its neighbours are on, it will turn **on** at the next time step.

If you were to zoom in and watch individual cells during the course of the Game of Life's computation, all you would see would be the cells switching on and off in accord with the four rules. Zoom out, though, and something else appears. Large structures, composed of many cells, are seen to grow and disintegrate over time. Some of these structures have recognisable characters: they maintain cohesion, move, reproduce, and interact with one another. Their behaviour can be dizzyingly complex. Patterns called 'oscillators' change shape, returning to the same shape that they began with after a certain number of time steps. A three-cell winking 'blinker' flips repeatedly from a vertical line to a horizontal line and back again, while the twelve-cell 'pentadecathlon' undergoes a beautiful fifteen step transformation that returns it to its original shape.

So-called 'spaceships' glide across the grid in the Game of Life: as time clicks forward, they morph into a new configuration that duplicates their original pattern but is displaced by one or more cells from their starting position, so creating movement. If you watch the game speeded up, spaceships appear to move smoothly. Spaceships are the main way in which information is transferred from one part of the grid to another. 'Gliders' are the smallest spaceship: they consist of five cells and will, over four time-steps, reproduce their original configuration but displaced one cell to the left and down. There are larger spaceships: in fact there is no known largest spaceship. The largest one discovered so far is an eleven-million cell monster, the 'Caterpillar'. Large-scale structures like the caterpillar are governed by their own rules, and to discover these 'higher-order' rules it is often better to experiment than to calculate. Observing the behaviour of the large structures under various conditions reveals the large-scale rules.

Some large-scale patterns, consisting of hundreds of thousands of cells, even behave as a universal Turing machine. Still larger patterns act like construction machines that assemble this universal Turing machine, and yet larger patterns – virtual creatures, perhaps? –

feed instructions to the universal machine. The virtual creatures inside the Game of Life can program their universal Turing machines to perform any computation – and that includes running their own simulation of the Game of Life. A simulation of the Game of Life on their machines – a game within the game – might contain other virtual creatures, and these may simulate the Game of Life on *their* Turing machines, which may in turn contain more virtual creatures, and so on. The nested levels of complexity that can emerge on a large grid are mind-blowing.[22] Nevertheless, everything that happens in the Game of Life is in a fundamental sense simple: the behaviour of every pattern, large and small, evolves as prescribed by the four simple transition rules. Nothing ever happens in the Game of Life that is not determined by these rules.

Zuse's thesis is that our universe is a CA governed by a small number of simple transition rules: he suggested that with the right rules a CA can generate patterns called 'digital particles' (*Digital-Teilchen*).[23] These digital particles correspond to the fundamental physical particles that conventional physicists regard as the basic building blocks of the universe. Zuse was writing before the Game of Life was invented, and so he wasn't suggesting that the specific transition rules in the Game of Life are the fundamental rules of our universe, but if he's right then *some* simple transition rules (and their associated grid structure) comprise the fundamental physics of the universe. More recently the Dutch Nobel Laureate and theoretical physicist Gerard 't Hooft (pronounced 'toft') has said:[24]

'I think Conway's Game of Life is the perfect example of a toy universe. I like to think that the universe we are in is something like this.'

If Zuse's thesis is right, then our universe is at its most fundamental level a computer: everything we observe in the universe – particles, matter, energy, fields – is a large-scale pattern that emerges from the activity of a CA. This CA's grid is not made up from traditional matter like electrons or protons: the CA operates at a more fundamental level, and electrons, protons, and all matter currently known to physics, emerge from the CA's activity – although what the CA's grid is in fact made of is far from clear, as we shall see below. This CA operates everywhere in the universe, at the smallest scale; and to describe it would be to produce a grand unifying theory of everything in the universe. All our other theories in physics – including general relativity and quantum mechanics – should fall out of this grand unifying theory. If Zuse is right then we humans are not so different from the virtual creatures that we can create in the Game of Life. In fact, 't Hooft suggests that our three-dimensional universe may be a sort hologram, arising from the transformation of digital information on a two-dimensional surface.[25]

Examining Zuse's thesis

Is Zuse's thesis right? The idea that the universe is a giant CA faces three big challenges. The first is the 'emergence problem': can it be demonstrated that the physics of our universe could in principle emerge out of a digital computation? The second challenge is the 'evidence problem': is there any experimental evidence to support Zuse's thesis? Third is the 'implementation problem': what 'hardware' is supposed to implement the universe's computation? Let's take the three challenges in turn.

The emergence problem is extremely hard. To solve it, the proponent of Zuse's thesis would need to find a way of showing how current physical theories could emerge from some simple underlying digital computation. Four large hurdles stand in the way. First, existing physics involves *continuous* quantities (position, energy, velocity, etc.); whereas CAs, and all digital computers, deal only in *discrete* units, not in continuous quantities. How could what is fundamentally continuous emerge from what is fundamentally discrete? To give a simple illustration: time is traditionally regarded as being continuous, whereas the movements of a digital watch are discrete: how could what is smooth and continuous arise from what is jerky and discontinuous? Second, physics seems to involve *non-deterministic* (i.e. random) processes, whereas CAs behave in a completely deterministic way. Third, current physics allows for *non-local* connections between particles: relationships without an intervening messenger (this is known as 'quantum entanglement'). Yet CAs don't allow such connections between distant cells of the grid. Fourth, and more worryingly still, our two best physical theories – general relativity and quantum mechanics – appear to be *incompatible*. How to unify general relativity and quantum mechanics is the hardest problem in current physics. But this is exactly what would need to be done by an underlying computational theory – no easy task! Perhaps each of these problems can be solved; if so, it is up to the advocates of Zuse's thesis to find the solutions.

Second, the evidence problem. To date, there is no experimental evidence at all for Zuse's thesis – so why believe it? It's also true that there's no experimental evidence against the thesis, and in fact evidence either way would be hard to find. This is because the CA that supposedly underlies the universe exists at extremely small spatial scales, of around one 'Planck length'. A Planck length, named after the famous quantum physicist Max Planck, is defined as 10^{-35} meters – that's just one zero short of a million million million million million millionth of a meter. Exploring events at this scale poses formidable obstacles. Let's use the size of the sub-atomic particle called the *proton* as our measuring stick. The Large Hadron Collider at CERN near Geneva can probe events that are one hundred thousand times smaller than a proton – the size difference between a mosquito and Mount Everest. However, a

Planck length is much, much smaller still: a Planck length is ten followed by nineteen zeroes times smaller than a proton – the same as the size difference between a mosquito and the Milky Way. Conventional particle colliders are never likely to be able to explore events at the Planck scale. In 2014, Craig Hogan's team at FermiLab in Chicago started a different kind of experiment to test whether space is a digital grid at the Planck scale. If it were, this would be one small step toward verifying Zuse's thesis. The experiment aims to detect jitter at the Planck scale, by measuring small movements in two laser interferometers.[26] So far, the experiment has not produced evidence for or against space being digital, and moreover there are serious doubts over whether the experiment will ever produce evidence one way or the other.[27] Collecting any evidence at the scale that is relevant to Zuse's thesis is *hard*.

Turning next to the implementation problem, the challenge here is to say what hardware could possibly implement the universe's computation. The computations that your laptop carries out are implemented by electrical activity in silicon chips and metal wires; the computations in your brain are implemented by electro-chemical activity in your neurons and synapses; and Conway's original version of the Game of Life was implemented by means of plastic counters and a Go board. Every computation requires some implementing medium, and the implementing hardware must exist in its own right. It cannot be something that itself *emerges* from the computation as a high-level pattern: Conway's plastic counters cannot emerge from the Game of Life – they are required in order to play the Game of Life in the first place.

According to Zuse's thesis, all matter, all energy, all fields, all particles emerge as patterns from the underlying cellular computation. What, though, could implement the cellular computation? Not something that we already know of in physics, since by hypothesis everything that we currently know of is an emergent pattern produced by the computation. Nor even something physical that we don't presently know of, since everything physical is supposed to emerge from the underlying computation. The implementing hardware must be something else: something beyond the realm of physics.

Some outlandish proposals have been made regarding this hardware. For example, the cosmologist Max Tegmark's 'Mathematical Universe Hypothesis' claims that the implementing hardware of the physical universe consists of *abstract mathematical objects* (existing in what mathematicians sometimes call 'Platonic heaven').[28] Tegmark's proposal inverts the usual way we think of computation: rather than physical objects – such as your electronic PC – implementing abstract mathematical objects – such as Turing machines or natural numbers – abstract objects implement all physical objects. On Tegmark's proposal, abstract mathematical objects are more fundamental to the universe than atoms and electrons!

Many objections can be raised to this proposal.[29] The most relevant here is that abstract mathematical entities don't seem to be the right kinds of things to implement computation. Time and change are essential to implementing a computation: computation is a process that unfolds through time, during which the hardware undergoes a series of changes (flip-flops flip, for example, neurons fire and go quiet, plastic counters appear and disappear on a Go board, and so on). Yet Tegmark's mathematical objects exist timelessly and unchangingly. What plays the role of time and change for this 'hardware'? How could these Platonic objects change over time in order to implement distinct computational steps? And how could one step give rise to the next if there is no time or change? Unchanging mathematical objects are just not the right kinds of things to implement a computation.

Currently, there are no plausible solutions to this chicken-and-egg implementation problem. Perhaps supporters of Zuse's thesis could say: we know that *something* must implement the universe's computation, but we should admit that we know nothing – and can know nothing — about this shadowy substratum. The proper aim of physics (Zuse's supporter continues) is simply to describe the universe's computation; physics must remain *silent* about the implementing medium. As Wittgenstein said in his usual pithy way:[30]

'Whereof one cannot speak, thereof one must be silent.'

If, however, you think that there is something unsatisfying about restricting the scope of physics in this way, then you are not alone. Red-blooded physicists want to know everything about the universe, and will not take well to this idea that the universe contains a fundamental substratum that must always remain beyond the reach of physics.

So far we have found no reason at all to think that the universe is a computer. In the Introduction, we mentioned a second version of the computer-universe theory and we now turn to this. More modest than the first version of the theory, this acknowledges that the universe may not *literally be* a computer, but maintains that nevertheless the physical universe is fundamentally computational, in a sense that we shall now explain.

Is the universe comput*able*?

A comput*able* system is a system whose behaviour could be computed by an idealized human computer. It's important to add the caveat 'idealized', since it might take a human clerk a million years to compute the behaviour of some large and complex system – and moreover, the calculations might require more paper and pencils than planet Earth is able to supply.

There are many systems that, although they are not computers themselves, are nevertheless computable. Consider, for example, an old-fashioned navigation lamp. The

function of the lamp is to flash out a signal marking, say, the eastern end of a particular sandbank in the Thames estuary (and to assist navigators the signal must be recognizably different from the signals emitted by all the other navigation lamps up and down that stretch of water). This lamp's signal is as follows: the lamp turns on for a second, then switches off for two seconds, then on for two seconds, then off for four seconds, and then repeats this cycle indefinitely. (The ons and offs are created by a sliding metal disk that is controlled mechanically: while the disc is positioned over the lamp's glass aperture the light is effectively turned off, and when the disc ceases to obscure the aperture, the light shines out – although the mechanical details do not matter for the example.) It is easy for a human computer to calculate the on-off behaviour of this lamp, and if you were asked whether the lamp would be on or off seventy-seven seconds (say) after its first flash, you would probably have little difficulty computing the answer. In summary: the lamp is not a computer but its flashing behaviour is comput*able*.

More complicated behaviours are also computable. For example, let's bring the irrational number π into the formula that determines whether the lamp is on or off. π, the ratio of a circle's circumference to its diameter, is 3.141592653589… There is no last digit: the digits of π continue on to infinity. Using π we can make the lamp's switching behaviour quite complex: if the 1st digit of π is odd then the lamp begins its sequence of operations by illuminating for a second, and if the 1st digit is even the lamp remains unilluminated during the first second; and if the 2nd digit of π is odd, the lamp illuminates for a second, and if the 2nd digit is even the lamp is unilluminated during this second second of its operating time; and so on. In this case, the lamp's behaviour during its first thirteen seconds of operating is: *flash*, *flash*, *no flash*, *flash*, *flash*, *flash*, *no flash*, *no flash*, *flash*, *flash*, *flash*, *no flash*, *flash*. As the sequence grows longer, an observer might think that the flashes and pauses are coming randomly. But this isn't so: there is nothing random about π.

Is the behaviour of the lamp still computable? Yes, it is. Turing showed that π is what he called a 'computable number': a Turing machine – and therefore a human computer – can calculate the digits of π, one by one. (Since there is no *last* digit of π, the Turing machine will work on forever, unless we stop it after it has produced some finite number of the digits.) So the Turing machine (or human computer) can calculate when the lamp is going to flash and when there will be no flash.

Randomness is one form of uncomputability: if the lamp were flashing randomly, its behaviour would *not* be computable, because if the human clerk could always predict the behaviour at the next second, then the behaviour would not be random.[31] Is there any conceivable way of arranging the behaviour of the lamp so that (a) its behaviour is *not*

random (i.e. is deterministic) and (b) its behaviour is nevertheless not computable? Answer: yes. This in fact follows from points explained in Chapters 7 and 37. As mentioned there, Turing proved that no Turing machine can solve the 'printing problem': that is to say, there is no way of programming a Turing machine so that it can decide, given any Turing-machine program, whether that program is ever going to print '1' or not (or '#', as in the example in Chapter 7: the choice of symbol makes no difference). Using this fact, we will explain how to arrange the flashes so that the sequence is not computable.

Let's assume, first, that all the infinitely many Turing machines are *ordered* in some way, so that we can speak of the 1st Turing machine, the 2nd Turing machine, and so on. The precise details of how the ordering is done need not concern us; one method would be to deem that the Turing machine with the *shortest* program is the 1st machine, and that the one with the next shortest program is the 2nd machine, and so on – although, of course, some 'tiebreaker' principles would be required for ordering machines whose programs are of the same length; and some further details would also be required to deal with the issue of how much data ('input') a machine has on its tape before it starts work. We are going to modify the above switching recipe (the one involving π) like this: if the 1st Turing machine is one of those that at some point prints a '1', then the lamp starts its sequence of operations by illuminating for a second, and if the 1st Turing machine never prints '1', then the lamp remains unilluminated during the first second; and if the 2nd Turing machine is one of those that at some point prints a '1', the lamp illuminates for a second, and if the 2nd Turing machine never prints '1', the lamp is unilluminated during the second second of its operating time; and so on. Now the resulting sequence of flashes and pauses is *not* computable.

This flashing light example helps to clarify what is being asked by the question 'Is the whole universe computable?': the question asks whether the behaviour of *everything* in the universe can be computed by an idealized human computer (equivalently, by a Turing machine), or whether the universe contain systems that, like the third lamp, are *un*computable. In the next three sections we examine a number of theses that are relevant to this question, starting with a famous – but sometimes misunderstood – thesis put forward by Turing.

Turing's thesis

In 1936 Turing stated (and argued for) what has come to be called simply 'Turing's thesis': *a Turing machine can do any task that the human computer can do*.[32] Essentially the thesis says that the Turing machine is a correct formal model of the human computer.

It is worth mentioning in passing that sometimes Turing's thesis is called the *Church-Turing thesis* (or even just *Church's thesis*). This is because Alonzo Church devised another formal model of the human computer, also in 1936, which Turing quickly proved to be equivalent to his own model.[33] Church's model was couched in terms of his highly technical concept of 'λ-definability' (the Greek letter 'λ' is pronounced 'lambda'). Something is said to be λ-definable if it can be produced by a certain process of repeated substitutions – the details need not concern us. Chapter 7 mentioned that Kurt Gödel much preferred Turing's model to Church's: Gödel said that he found Turing's model 'most satisfactory' but told Church that his approach was 'thoroughly unsatisfactory'.[34]

Nevertheless Church's own thesis, that his λ-definability model is a model of the human computer, is true – since Turing managed to prove that everything Turing machines can do is λ-definable (and vice versa). Nowadays there is a tendency to say that Turing's thesis and this thesis of Church's are 'the same' (in virtue of Turing's proof), but this is misleading, because the two theses have different *meanings*. One obvious and important difference between them is that Turing's thesis involves *computing machines* but Church's does not. In what follows we focus on Turing's thesis, not Church's.

Turing's thesis gets confused with some quite different claims about computability, and the implications of his thesis are not infrequently misunderstood. Searle, for example, gives this formulation of the thesis:[35]

'anything that can be given a precise enough characterization as a set of steps can be simulated on a digital computer.'

This statement of Searle's implies that *any* system that operates step by step is computable, but that is a much stronger claim than Turing's actual thesis, which says merely that *human computers* can be simulated by Turing machines. In fact Searle's thesis can readily be counter-exampled.[36]

Another example of confusion is Sam Guttenplan's statement (in the Blackwell *Companion to the Philosophy of Mind*) that for any systems whose 'relations between input and output are functionally well-behaved enough to be describable by ... mathematical relationships': [37]

'we know that some specific version of a Turing machine will be able to mimic them.'

Again this is very different from what Turing said: Turing's own thesis does *not* imply that a Turing machine can simulate (mimic) any input-output system that can be described by mathematics – only that it can simulate any human computer. In fact, since the universe is effectively an input-output system (one thing leads to another by physical causation), and

since the universe certainly appears to be mathematically describable, Guttenplan's thesis appears to imply that indeed the physical universe is computable. But no such thing is implied by Turing's thesis.

A third and final example of this tendency to misunderstand Turing and his work is provided by philosophers Paul and Patricia Churchland, who say that Turing's[38]

'results entail something remarkable, namely that a standard digital computer, given only the right program, a large enough memory and sufficient time, can compute *any* rule-governed input-output function. That is, it can display any systematic pattern of responses to the environment whatsoever.'

Turing's results certainly do not entail that every rule-governed input-output system is computable. That, as we have just seen, is tantamount to claiming that a *rule-governed* universe is a *computable* universe; and the uncomputable lamp example – where the rule that determines whether or not a flash comes at the n^{th} second involves whether or not the n^{th} Turing machine ever prints '1' – shows that this is wrong.

There is a thesis very different from Turing's lurking amid these confusions, and the fact that it isn't the *same* as Turing's thesis doesn't necessarily mean that it isn't true. So let's try to pin this thesis down and examine it. First attempt: *the behaviour of any imaginable law-governed deterministic physical system is computable*. Notice that if it's assumed that the universe contains *only* physical systems (and if it is also assumed that the universe is law-governed and deterministic) then this thesis certainly implies that the whole universe is computable. But because we don't want to get diverted by a discussion of the thorny question of whether *everything* in the universe is physical – or whether, on the other hand, it contains non-physical things such as souls and angels – we will henceforward concentrate on the claim that the whole *physical* universe is computable.

This thesis relating lawfulness and computability is, however, no more acceptable than the previously discussed thesis that all rule-governed systems are computable. The third lamp is a deterministic physical system that is governed by the laws of physics, yet is not a computable system. Another counter-example to the thesis is the hypothetical discovery, in some distant galaxy, of a naturally occurring and fully deterministic source of radio waves – perhaps an oscillating supernova remnant – that emits an unending sequence of radio-frequency pulses exhibiting the same pattern as the flashes emitted by the third lamp. The supernova remnant is a law-governed physical system whose behaviour is not computable. The underlying point is that there are imaginable physical laws that are deterministic but uncomputable, and so the thesis fails.[39]

The third lamp, though, was only very loosely specified: we did not actually explain how the flashing behaviour is brought about in such a way that it reflects the printing behaviour of the Turing machines. The same goes for the hypothetical supernova remnant. This leads on to a better statement of the thesis: *the behaviour of any rigorously specified deterministic physical system is computable*. We call this the *Specification Thesis*, or S-thesis for short.

The physical universe, we assume, can be rigorously specified mathematically – we don't know this specification yet, but this is what physics aims at. So the S-thesis seemingly entails that the physical universe, if deterministic, is computable. But is the S-thesis true? In fact it seems not to be. In the next section we will describe a rigorously specified deterministic machine – actually a kind of Turing machine – whose behaviour is not computable.

Accelerating Turing machines

As the name implies, accelerating Turing machines (ATMs) speed up as they compute; but apart from the fact that their speed of operation accelerates as the computation proceeds, ATMs are exactly like standard Turing machines.[40]

An ATM accelerates in accordance with a formula first introduced by the philosopher Bertrand Russell in a 1914 lecture. He described an unusual way of running around a racetrack:[41]

'If half the course takes half a minute, and the next quarter takes a quarter of a minute, and so on, the whole course will take a minute.'

Russell emphasized that although this is *'medically* impossible', it is not *logically* impossible.

An ATM follows the same pattern: it performs the second operation specified by its program in half the time taken to perform the first, and the third in half the time taken to perform the second, etc. If the time taken to perform the first operation is one second (say), then the next operation is done in half a second, the third operation is done in a quarter of a second, and so on. Adding up the times taken by the second and third operations and onwards, we get:

$1/2 + 1/4 + 1/8 + 1/16 + 1/32 + ...$

This total evidently cannot exceed one second (since what is added at each step is always less than the remaining amount that must be added in order to reach 1). So, allowing also for the second that it takes to do the first operation, the total time required for *all* the operations done

by the machine is no more than *two* seconds. Notice that this remains true even if the machine carries out an *infinite* number of operations – i.e., never stops computing. Which is to say: an ATM can carry out an infinite number of computations in no more than two seconds.

An ATM can exhibit behaviour that is not computable. For example, let's suppose that, for any selected integer n, we want to find out whether the n^{th} Turing machine ever prints '#'. If the n^{th} Turing machine is one of those that runs on forever, then you couldn't find this out simply by watching the machine at work, since no matter how long you had been watching without '#' appearing, you could never be sure that '#' would not be printed at some future time – and so you would never be in a position to say: 'The n^{th} Turing machine does not print "#"'. This is not a computable task, as Turing proved in 1936; but nevertheless an ATM can do it. Here's how. The ATM calculates the behaviour of the n^{th} Turing machine, step by step, and if it finds that the n^{th} machine does print '#', then it outputs the message: 'Yes, it prints "#"' (or, better, we can arrange for it to output an abbreviated form of this message). If, on the other hand, the n^{th} machine runs on forever without printing a single '#', then the ATM will itself run on through infinitely many calculations as it simulates the n^{th} machine, waiting in vain for it to print '#'. But since these infinitely many calculations will take the ATM no more than two seconds to complete, we know that if the message 'Yes, it prints "#"' has not arrived after two seconds, then the n^{th} machine does not print '#'.

It only remains to fill in the details. We prepare the ATM by writing the program of the n^{th} machine on its tape: the ATM will simulate the n^{th} machine by following this program. In order to make the message 'Yes, it prints "#"' as compact as possible, we adopt the convention that if the ATM writes '1' on the very first square of its tape (which we make sure we leave blank in the setting-up procedure), then this shall be taken to mean 'Yes, it prints "#"'. We accordingly program the ATM to go to this square and print '1' on it (and then halt) if it discovers that the n^{th} machine prints '#'; and the ATM is prohibited from printing anything on this square in any other circumstances.

Now we press the ATM's start button and wait two seconds before reading the output. If we see that the special square contains '1', then the n^{th} machine does print '#'; and if we see that the special square is still blank, then the n^{th} machine does not print '#'.

So the S-thesis appears to be false. The ATM has been specified carefully and yet its behaviour is not computable. Those interested in exploring ATMs further will find a reference in the endnote.[42]

However, the fact that the ATM is a logical possibility doesn't mean that it could actually exist in our physical universe. The ATM pushes us towards a sharper version of the

thesis that we are looking for; we call this sharper version the 'physical computability thesis' (PCT).

The physical computability thesis

The PCT states that *the behaviour of every genuinely possible deterministic physical system (that is, every deterministic physical system that is possible according to the physics of our universe) is computable*.

It is worth noting in passing that the PCT is often referred to as the 'physical version' of Turing's thesis, and is sometimes named the 'Physical Church-Turing Thesis'. However, the name 'Physical Church-Turing Thesis' is perhaps not ideal, because the PCT has little or nothing to do with the Church-Turing thesis, and neither Church nor Turing endorsed – nor even formulated – the PCT. Since using the name 'Physical Church-Turing Thesis' could open the door to confusions, we prefer to avoid it here.

The PCT is an interesting thesis, and entails an affirmative answer to our question 'Is the whole physical universe computable?' (assuming that the universe is deterministic). But is the PCT true? Some maintain not. In the remainder of this section we will describe a potential counterexample to the PCT involving a relativistic system (in the sense of Einstein's theory of relativity).

The idea of using relativity in order to formulate an uncomputable system was presented by Itamar Pitowsky in 1986, at an academic conference in Jerusalem: Pitowsky explained that under certain special conditions, a computer can perform infinitely many steps in what an observer who is outside the system (but communicating with it) experiences as a *finite* time.[43] What's more, this computer can be a perfectly ordinary laptop that functions just as usual – and as far as the laptop is concerned, there is no speed-up at all: it performs each step of the computation at the same rate. The speed-up that enables infinitely many steps to be performed in a finite time is seen only from the viewpoint of the distant observer.

This is not quite the same idea as an ATM, since the ATM described earlier would be seen as speeding up even by an observer who is part of the system (sitting on the scanner, say). Additionally, relativistic systems are governed by Einstein's theory of relativity, meaning that no signal can travel faster than light travels in a vacuum; whereas an ATM is not necessarily subject to this restriction. An ATM may accelerate to the point where the scanner is moving along the tape faster than the speed of light (although the scanner could be kept below light speed if the symbols on the tape were to get progressively smaller, with the result that the distances travelled by the scanner become shorter and shorter).

Pitowsky's original setup conformed to Einstein's special theory of relativity, but here we will describe a setup proposed by István Németi and his group (at the Hungarian Academy of Sciences in Budapest) that involves Einstein's general theory of relativity.[44] Németi's system is in effect a relativistic implementation of an ATM. He emphasizes that his system is a *physical* one, as opposed to some purely notional system that could exist only in fairy-land: the system is physical, Németi says, in the sense that it is 'not in conflict with presently accepted scientific principles', and, in particular, 'the principles of quantum mechanics are not violated'.[45] Németi suggests that humans might 'even build' his relativistic system 'sometime in the future'.[46]

Németi's system consists of two parts, one part being a standard Turing machine, *S*, located on Earth, and the other part being an observer, *O*, who journeys through space. Before beginning his or her journey, *O* sets up *S* to simulate the n^{th} Turing machine, the object of the exercise being to discover whether or not the n^{th} machine prints '#'. Associated with *S* is a piece of ancillary equipment that emits a signal if (and only if) the simulation done by *S* reveals that the n^{th} machine prints '#'. This arrangement is equivalent to the previous one of writing '1' on the first square of *S*'s tape if (and only if) the n^{th} machine prints '#'.

O then travels through space to a type of black hole known as a 'slow Kerr hole', after New Zealand mathematician Roy Kerr. Slow Kerr holes are huge, slowly rotating black holes. Cosmologists do not know for certain if any slow Kerr holes actually exist, but Németi points to 'mounting astronomical evidence for their existence'. He chooses a slow Kerr hole because these have special properties, one of which is that the observer *O* can, Németi says, pass through the hole 'and happily live on'. If *O* were to enter a more traditional type of black hole, he or she would be annihilated by the crushing gravitational forces generated by the black hole. In the case of a slow Kerr hole, however, these extreme gravitational forces are, Németi explains, counterbalanced by the Kerr hole's rotational forces – the gravitational forces are offset by the forces that result as the black hole spins, meaning that the observer is not crushed, and can in principle emerge safely to tell the tale.

Németi theorizes that as *O* starts to enter the Kerr hole, *S*'s rate of computation accelerates relative to *O*. This is due to 'gravitational time dilation', an effect predicted by Einstein's theory of relativity. The deeper into the hole *O* travels, the faster and faster *S* runs relative to *O*, in fact without any upper limit. The acceleration continues until, relative to a time *t* on *O*'s watch, the entire span of *S*'s computing is over; and, if any signal was emitted by *S*'s signal-generator, it will have been received by *O* before this time *t*. From *O*'s point of view, *S* has done its computation in a finite period of time. This is so even if *S* runs on through infinitely many calculations as it simulates the n^{th} machine, in the possible case

where the n^{th} machine computes forever without printing '#'. By time t, therefore, O knows whether or not a signal has been emitted, and so knows whether or not the n^{th} machine ever prints '#'.

If Németi is right then this is a counterexample to PCT: a deterministic physical system that is not computable but that nevertheless is possible according to the physical laws of our universe. His counterexample is certainly not universally accepted: for example, one can question whether the existence of a Turing machine that is able to compute forever without wearing out – as S must, if the n^{th} machine runs on forever without printing '#' – is *really* consistent with the actual laws of physics. But Németi's example certainly serves to show that it's far from *obvious* that the PCT is true. As we intimated in our introduction, the answer to 'Is the whole physical universe computable?' is currently unknown. Even if there is genuine randomness – and hence uncomputability – in the universe, the question would still remain whether there are (or could be) real physical systems not involving randomness that are uncomputable. Theorists disagree about the answer – and sometimes the debate gets heated.

In fact, to assume without warrant that the physical universe is computable might actually hinder scientific progress. If the universe is essentially uncomputable, and yet physicists are searching for a system of physical laws that would describe a computable universe, then bad physics is likely to ensue. Even in the case of brain science – let alone the study of the whole universe – simply assuming computability could be counterproductive. As philosopher and physicist Mario Bunge remarked, this assumption[47]

'involves a frightful impoverishment of psychology, by depriving it of nonrecursive [i.e. non-computable] functions.'

In the next section, we will examine Turing's views on this question of computability and the brain – a microcosm of the debate about the grandscale question of whether the whole universe is computable.

Turing's opinion

It used to be widely believed that Turing had said, or perhaps even proved, that every possible physical system is computable. Earlier we saw Paul and Patricia Churchland asserting that Turing's results *entail* that all rule-governed behaviour is computable. Another example comes from David Deutsch, one of the pioneers of quantum computing, who put forward this variant of the PCT, calling it 'the physical version of the Church–Turing principle':[48]

'Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.'

Deutsch went on to say that 'This formulation is both better defined and more physical than Turing's own way of expressing it.' Deutch's thesis is indeed more physical than Turing's thesis; but it is a completely *different* thesis from Turing's, not a 'better defined' version of what Turing said! Turing was talking about human computers, not physical systems in general.

In similar vein mathematician Roger Penrose (the co-discoverer of black holes) stated:[49]

'It seems likely that he [Turing] viewed physical action in general—which would include the action of a human brain—to be always reducible to some kind of Turing-machine action.'

Penrose even named this claim *Turing's thesis*. But, as we shall see, Turing never endorsed this thesis and was aware that the thesis might be false.

Andrew Hodges (the mathematician who wrote the biography that inspired the movie *The Imitation Game*) maintained in his book that Turing's work implied what is a close cousin of the PCT:[50]

'Alan had ... discovered something almost ... miraculous, the idea of a universal machine that could take over the work of any machine.'

Here Hodges, like Penrose, was suggesting that Turing's work entails that any physical mechanism is computable. He also stated that Turing claimed[51]

'that the action of the brain must be computable, and therefore can be simulated on a computer.'

However, that was back in the bad old days. Modern Turing scholarship, by Hodges and others, now paints a very different picture. In fact there is no evidence that Turing ever understood his work on computability to rule out the possibility of mechanisms whose action is not computable. In 1999, one of us – Jack Copeland, together with Diane Proudfoot – suggested in an article in *Scientific American* that Turing was an important forerunner of the modern debate concerning the possibility of uncomputability in physics and uncomputability in the action of the human brain.[52] In the same year Copeland also published a commentary on a lecture that Turing gave on BBC radio in 1951 titled *Can Digital Computers Think?*[53]: this commentary pointed out that, in the lecture, Turing noted the possibility that the physics of the brain might be uncomputable.[54] Turing was contemplating the possibility of a physical system that is not computable. Hodges was persuaded by these observations (and, in a public

lecture, generously thanked Copeland for this new view of Turing, even saying on his website 'I don't mind admitting that I wish I had thought of it').[55]

In the passage in question, Turing considered the claim that if 'some particular machine can be described as a brain we have only to programme our digital computer to imitate it and it will also be a brain'.[56] He immediately went on to argue that this 'can quite reasonably be challenged', pointing out that there is a difficulty if the behaviour of the brain is not 'predictable by calculation' – if a human computer cannot predict the behaviour of a system by calculation, then the system is not computable. Turing referred to the view of the physicist Sir Arthur Stanley Eddington who, he pointed out, held that 'no such prediction is even theoretically possible', on account of 'the indeterminacy principle in quantum mechanics'. If Eddington is right about the impossibility of prediction, Turing argued, then the brain is not computable.

Happily a consensus has emerged that Turing – far from claiming that every physical mechanism, including the brain, must be computable – was in fact open to the idea that the brain, at least, is not a computable system. As Hodges recently put it:[57]

'[Turing] was also one of the first to use a computer for simulating physical systems. In 1951, however, Turing gave a radio talk with a different take on this question, suggesting that the nature of quantum mechanics might make simulation of the physical brain impossible.'

Conclusion

Alan Turing never said that the physical universe is computable, and nor do any of his technical results entail that it is. Some computer scientists and physicists seem infuriated by the suggestion that the physical universe might be uncomputable; but it's an important issue and the truth is that we simply do not know.

Notes for CH 41 *Is the whole universe a computer?* (COPELAND, SPREVAK and SHAGRIR)

---

[1] Turing, 'Computing machinery and intelligence', p. 446 in *The Essential Turing*.

[2] Copeland, B. J., Sorensen, R. 'Multiple Realizability: the Copeland-Sorensen Optical Universal Computing Machine', in Copeland, B. J., Piccinini, G., Proudfoot, D., Shagrir, O. *The Philosophy of Computing* (forthcoming).

[3] Turing, 'On computable numbers', p. 59 in *The Essential Turing*.

[4] Wittgenstein, L. 1980. *Remarks on the Philosophy of Psychology*. Vol.1. Oxford: Blackwell, §1096.

5 Turing, 'Intelligent machinery', p. 416 in *The Essential Turing*.

[6] Turing, 'Lecture on the Automatic Computing Engine', pp. 387, 391 in *The Essential Turing*.

[7] Turing, 'Computing machinery and intelligence', p. 444 in *The Essential Turing*.

[8] Turing, A. M. *Programmers' Handbook for Manchester Electronic Computer Mark II*, p. 1; a digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/programmers_handbook.

[9] We canvas alternative answers in our forthcoming chapter 'Zuse's thesis, Gandy's thesis, and Penrose's thesis', in Cuffaro, M., Fletcher, S. (eds) *Physical Perspectives on Computation, Computational Perspectives on Physics*, Cambridge University Press.

[10] Turing, 'Proposed Electronic Calculator', p. 386 in Copeland, B. J. *Alan Turing's Automatic Computing Engine*.

[11] Searle, J. 'Is the Brain a Digital Computer?'. *Proceedings and Addresses of the American Philosophical Association*, 64 (1990), 21-37 (pp. 25-27); Searle, J. *The Rediscovery of the Mind.* Cambridge, Mass.: MIT Press (1992), 205-209; Putnam, H. *Representation and Reality*, Cambridge, Mass.: MIT Press (1988), 121-125.

[12] Copeland, B. J. 'What is Computation?', *Synthese*, vol. 108 (1996), pp. 335-359; Chalmers, D. J. 'Does a Rock Implement Every Finite-State Automaton?' *Synthese* 108 (1996), 309–333.

[13] von Neumann, J. 'The General and Logical Theory of Automata', in Vol. 5 of von Neumann's *Collected Works*, ed. A. H. Taub (Oxford: Pergamon Press, 1963).

[14] Copeland, B. J., Sommaruga, G. 'The Stored-Program Universal Computer: Did Zuse Anticipate Turing and von Neumann?', in Sommaruga, G., Strahm, T. (eds) *Turing's Revolution*, Basel: Springer, 2016.

[15] Zuse, K. 'Some Remarks on the History of Computing in Germany', in Metropolis, N., Howlett, J., Rota, G. C. (eds) *A History of Computing in the Twentieth Century* (New York: Academic Press, 1980), p. 611.

[16] Konrad Zuse interviewed by Uta Merzbach in 1968 (Computer Oral History Collection, Archives Centre, National Museum of American History, Washington D.C.).

[17] Copeland and Sommaruga, 'The Stored-Program Universal Computer: Did Zuse Anticipate Turing and von Neumann?'.

[18] Zuse in conversation with Brian Carpenter at CERN on 17 June 1992 (Copeland is grateful to Carpenter for sending some brief notes on the conversation that Carpenter made at the time); Zuse, K. *Der Computer – Mein Lebenswerk* [The computer—my life's work] (Berlin: Springer, 4th edn, 2007), p. 101.

[19] Donald Davies interviewed by Christopher Evans in 1975 ('The Pioneers of Computing: An Oral History of Computing', London: Science Museum; © Board of Trustees of the Science Museum).

[20] Robin Gandy and Wilfried Sieg view CAs as parallel computers that differ from Turing machines in allowing changes in *arbitrarily many* cells, whereas in a Turing machine only the content of one of the tape's cells can be altered at each step; Gandy, R. 'Church's Thesis and Principles of Mechanisms', in J. Barwise, H. J. Keisler and K. Kunen (eds), *The Kleene Symposium*, Amsterdam: North-Holland (1980),

123-148; Sieg, W. 'On Computability', in A. Irvine (ed.) *Handbook of the Philosophy of Mathematics*, Amsterdam: Elsevier (2009), 535– 630.

[21] Wolfram, S. *Theory and Applications of Cellular Automata*, World Scientific, 1986, 8.

[22] For more on this, see Poundstone, W. 1985. *The Recursive Universe*. New York, NY: William Morrow & Company.

[23] Zuse, K. 1969. *Rechnender Raum*. Braunschweig: Friedrich Vieweg & Sohn.

[24] 'Looking at life with Gerardus 't Hooft'. *Plus Magazine* (January 2002). http://plus.maths.org/issue18/features/thooft/.

[25] For a good summary, see Bekenstein, J. D. "Information in the Holographic Universe." *Scientific American* 17 (April 2007), 66–73.

[26] See Moyer, M. "Is Space Digital?" *Scientific American* 23 (August 2014), 104–11.

[27] Cho, A. "Controversial Test Finds No Sign of a Holographic Universe." *Science* 350 (2015), 1303.

[28] Tegmark, M. *Our Mathematical Universe*. New York, NY: Knopf, 2014.

[29] For a critical discussion of Tegmark's proposal, see the commentary by Scott Aaronson at http://www.scottaaronson.com/blog/?p=1753.

[30] Wittgenstein, L. *Tractatus Logico-Philosophicus*, Kegan Paul, Trench and Trubner, 1922, proposition 7.

[31] Church, A. On the concept of a random sequence, *Bulletin of American Mathematical Society* 46 (1940), 130-135.

[32] For a detailed examination of Turing's various formulations of his thesis, see Copeland, B. J. 'The Church-Turing Thesis', in Zalta, E. (ed.) *The Stanford Encyclopedia of Philosophy*, plato.stanford.edu/entries/church-turing.

[33] Church, A. 'An Unsolvable Problem of Elementary Number Theory', *American Journal of Mathematics* 58 (1936), 345-363; Turing, Appendix to 'On computable numbers', pp. 88-90 in *The Essential Turing*.

[34] See Kleene, S. C. 'Origins of Recursive Function Theory', *Annals of the History of Computing*, vol. 3 (1981), pp. 52-67 (pp. 59, 61); Gödel, K. 'Some Basic Theorems on the Foundations of Mathematics and their Implications' (1951), in his *Collected Works*, ed. Feferman, S. et al., Vol. 3 (Oxford: Oxford University Press, 1995), pp. 304-305.

[35] Searle mistakenly calls this 'Church's thesis': Searle, J. *The Rediscovery of the Mind*. Cambridge, Mass.: MIT Press (1992), 200-201; see also Searle, J. *The Mystery of Consciousness*. New York: New York Review (1997), 87.

[36] Copeland, B. J. 'The Broad Conception of Computation', *American Behavioral Scientist*, vol. 40 (1997), 690-716.

[37] Guttenplan, S. *A Companion to the Philosophy of Mind*. Oxford: Blackwell (1994), p. 595.

[38] Churchland, P. M., Churchland, P. S. 'Could a Machine Think?'. *Scientific American*, 262 (Jan. 1990), 26-31 (p. 26).

[39] Pour-El, M. B., Richards, I. 'The Wave Equation with Computable Initial Data such that its Unique Solution is not Computable'. *Advances in Mathematics*, 39 (1981), 215-239.

[40] Stewart, I. 'Deciding the Undecidable'. *Nature*, 352 (1991), 664-5; Copeland, B. J. 'Even Turing Machines Can Compute Uncomputable Functions', in Calude, C., Casti, J., Dinneen, M. (eds) *Unconventional Models of Computation*, London: Springer-Verlag (1998); Copeland, B. J. 'Super Turing-Machines', *Complexity*, 4 (1998), 30-32; Copeland, B. J. 'Accelerating Turing Machines', *Minds and Machines*, 12 (2002), 281-300. The term 'accelerating Turing machine' was introduced by Copeland in lectures in 1997. The variant term 'accelerated Turing machine' (see e.g. Calude, C. S., Staiger, L. 'A Note on Accelerated Turing Machines', *Centre for Discrete Mathematics and Theoretical Computer Science Research Reports*, 2009, http://hdl.handle.net/2292/3857; Fearnley, L. G. 'On Accelerated Turing Machines', Honours thesis in Computer Science, 2009, University of Auckland; Potgieter, P.H., Rosinger, E. 'Output Concepts for Accelerated Turing Machines', *Centre for Discrete Mathematics and Theoretical Computer Science Research Reports*, 2009, http://hdl.handle.net/2292/3858) originated in Copeland's 'Even Turing Machines Can Compute Uncomputable Functions'.

[41] Russell, B. A. W. *Our Knowledge of the External World as a Field for Scientific Method in Philosophy*, Chicago: Open Court (1915), pp. 172-3.

[42] Copeland, B. J., Shagrir, O. 'Do Accelerating Turing Machines Compute the Uncomputable?' *Minds and Machines*, vol. 21 (2011), pp. 221-239.

[43] The conference papers were published in 1990; Pitowsky, I. 'The Physical Church Thesis and Physical Computational Complexity'. *Iyyun* 39 (1990), pp. 81–99.

[44] Andréka, H., Németi, I., Németi, P. 'General Relativistic Hypercomputing and Foundation of Mathematics', *Natural Computing*, 8 (2009), 499-516; Németi, I., Dávid, G. 'Relativistic Computers and the Turing Barrier', *Applied Mathematics and Computation*, 178 (2006), 118-142; Etesi, G., Németi, I. 'Non-Turing Computations via Malament-Hogarth Space-Times'. *International Journal of Theoretical Physics*, 41 (2002), 341-370. David Malament (in private communications) and Mark Hogarth have described essentially similar setups; Hogarth, M. L. 'Does General Relativity Allow an Observer to View an Eternity in a Finite Time?' *Foundations of Physics Letters* 5 (1992), pp. 173–181, Hogarth, M. L. 'Non-Turing Computers and Non-Turing Computability' *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association* 1 (1994), pp. 126–138.

[45] Andréka, Németi and Németi, 'General Relativistic Hypercomputing and Foundation of Mathematics', 501.

[46] Andréka, Németi and Németi, 'General Relativistic Hypercomputing and Foundation of Mathematics', 511.

[47] Bunge, M., Ardila, R. *Philosophy of Psychology*, New York: Springer-Verlag (1987), p. 109.

[48] Deutsch, D. 'Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer' *Proceedings of the Royal Society*, series A, 400 (1985), 97-117 (p. 99).

[49] Penrose, R. *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford: Oxford University Press (1994), 21.

[50] Hodges, A., *Alan Turing: The Enigma*, London: Vintage (1992), p. 109.

[51] Hodges, A. 'What Would Alan Turing Have Done After 1954?', in C. Teuscher (ed.) *Alan Turing: Life and Legacy of a Great Thinker*, Berlin: Springer-Verlag (2003), p. 51.

[52] Copeland, B.J., Proudfoot, D. 'Alan Turing's Forgotten Ideas in Computer Science', *Scientific American* 280 (1999), pp. 99-103.

[53] Turing, 'Can Digital Computers Think?', first published in Copeland, B. J. 'A Lecture and Two Radio Broadcasts on Machine Intelligence by Alan Turing', in K. Furukawa, D. Michie, S. Muggleton (eds), *Machine Intelligence 15*, Oxford: Oxford University Press (1999), 445-476; and also in *The Essential Turing*.

[54] Copeland, 'A Lecture and Two Radio Broadcasts on Machine Intelligence by Alan Turing', 451-2. See also Copeland, B. J. 'Turing and the Physics of the Mind', in Cooper, S. B., van Leeuwen, J. (eds) *Alan Turing: His Work and Impact*, Amsterdam: Elsevier (2013), 651-666.

[55] Hodges, A. (2002), 'What Would Alan Turing Have Done After 1954?', Lecture at the Turing Day, Lausanne, Switzerland, June 2002.

[56] Turing, 'Can Digital Computers Think?', p. 483 in *The Essential Turing*.

[57] A. Hodges, 'Beyond Turing's machines', *Science*, 336 (13 April 2012), 163-4.