Membrane Systems and Hypercomputation

Mike Stannett[*]

Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello, Sheffield S1 4DP, United Kingdom

Abstract. We present a brief analysis of hypercomputation and its relationship to membrane systems theory, including a re-evaluation of Turing's analysis of computation and the importance of timing structure, and suggest a 'cosmological' variant of tissue P systems that is capable of super-Turing behaviour. No prior technical background in hypercomputation theory is assumed.

1 Re-evaluating Turing's Analysis

In his seminal paper [Tur36], Turing gave a careful and powerfully intuitive analysis of what it means for a human being to compute something, and described how the processes involved could be captured mechanistically via the machine model that now bears his name. By analysing the behaviour of his model, Turing was then able to show that certain problems could not be solved computationally. Against this, hypercomputation theorists, myself included, claim that certain physical forms of computation may in fact be more powerful than Turing envisaged. It is incumbent on us, therefore, to explain why and where Turing's analysis is incomplete, and why computation might indeed be capable of solving problems that appear on first analysis to be formally undecidable.

1.1 The Halting Problem revisited

Let us begin by recalling the reasons underpinning the insolubility of the Halting Problem (essentially a recasting of Richard's Paradox [Ric05], see also [vH77, pp. 142–144]). Our goal in doing so is to not to re-establish the standard underlying paradox, but to investigate its possible sources. For ease of argument, we will express things in terms of modern computers and programming languages. Our focus is the set of programs that accept a single natural number as input.

Preliminaries
A standard (Western) computer keyboard allows users to express roughly 105 distinct characters. Think of the characters on the keyboard as distinct digits in

[*] The author is partially supported under the Royal Society International Exchanges Scheme (ref. IE110369). This work was partially undertaken whilst the author was a visiting fellow at the Isaac Newton Institute for the Mathematical Sciences in the programme *Semantics & Syntax: A Legacy of Alan Turing.*

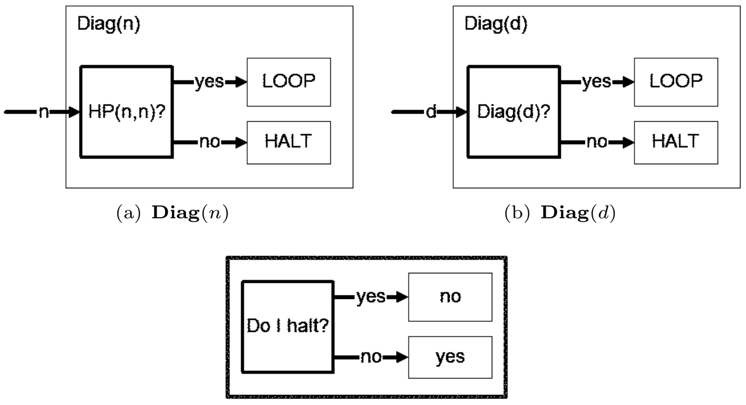
this base. Since computer programs can be expressed as finite strings typed on such a keyboard, each program can be regarded as a natural number (written in base 105). We can therefore arrange the set of all programs, acting on a single natural number input, in some definite order: $P_0, P_1, P_2, \ldots$.

Step 1. Suppose **HP** can be built

We would like someone to build us a labour saving tool (**HP**). Given two natural numbers m and n, **HP** should output *yes* if $P_m(n)$ eventually halts, and *no* if it doesn't. Let us assume that **HP** can indeed be built.

Step 2. Use **HP** to build **Diag**

The clerk can use **HP** to build a program, **Diag**, that halts if $P_n(n)$ runs forever, and loops forever if $P_n(n)$ will eventually halt (Fig. 1(a)).



(a) **Diag**(n) (b) **Diag**(d)

(c) **Paradox**

Fig. 1. Behaviour of **Diag**(n), **Diag**(d) and **Paradox**.

Step 3. Apply **Diag** to its own code

Because **Diag** takes a single natural number input, it must occur somewhere in the list $P_0, P_1, P_2, \ldots$. Suppose then that **Diag** is P_d, and consider what happens when we compute **Diag**(d). Because P_d is just **Diag**, the subroutine **HP**(d, d) resolves the question "Does **Diag**(d) halt?", indicated "**Diag**(d)?" in Fig. 1(b).

Step 4. Establish a paradox

The value d is simply a fixed natural number. We can therefore consider it to be 'hard-wired' as **Diag**(d)'s input, and re-interpret **Diag**(d) as a program, **Paradox**, taking no inputs. In informal terms, **Paradox** asks itself the question '*Do I halt?*', and behaves paradoxically in the sense that it apparently halts if and only if it runs forever (Fig. 1(c)).

Step 5. Resolve the contradiction

Since a paradox appears to have been generated, one of our assumptions must be wrong. However, the only assumption we seem to have made is that **HP** can be built, since the other steps presumably follow automatically. Therefore, **HP** cannot in fact be built.

2 Towards hypercomputation

Given the paradoxical behaviour established in Step 4, Turing's argument relies on the judgment (Step 5) that the only questionable assumption is that **HP** can be built (Step 1). However, there are at least three other key assumptions built into this argument, any of which can also be used to provide an alternative resolution of the paradox:

- Can **HP** be implemented via a physical system (an *oracle* [Tur39, BCLT08]) that is not itself computational?

- Is the ability to use tools as components in the construction of larger systems reflected only in those systems' structure, or does it also affect *what* can be computed? If so, it need not be **HP**'s existence that should be called into question, but the way in which it has been used as a component in the larger system, **Diag**.

- The argument tacitly assumes that computational systems can be built that behave deterministically, since we are using our definite knowledge of what **Diag**'s output *ought to be* in order to derive the paradox. While the uncertainties of quantum theory obviously throw this assumption into question, we will see below that mechanistic determinism cannot be achieved even in the setting of classical Newtonian dynamics.

The first of these caveats needs careful analysis. The existence of a physical **HP** solver would not prevent the construction of **Diag**, and one can still envisage a situation where **Diag** must be one of the programs $P_0, P_1, P_2, \ldots$, because there is nothing to stop us using a language whose commands include statements of the form "*Feed the values into the black box on the table, and use its subsequent output in what follows*". It is important to remember, however, that the problem **HP** depends on the computational system under analysis. If we add the ability to use **HP** as a basic instruction type, Turing's proof then shows that the halting problem for this *extended* system cannot be solved by a (Turing+**HP**)-machine. In other words, we may well be able to find a physical oracle that solves **HP** in the context of standard digital computing, but there will always be other problems that remain undecidable no matter how powerful the components we allow. The existence of a physical **HP** solver cannot be answered definitively given our present knowledge of physics, but the evidence (reviewed below) is encouraging; it appears, in particular, that relativistic phenomena could be exploited to solve problems like **HP** that are insoluble via Turing machine.

Addressing the third issue leads to somewhat more surprising results. Underpinning all our discussions of computation so far is the notion that the physics of computation is essentially deterministic. We are used to the idea that quantum theory introduces inherent uncertainties, but as we now explain, even such classical systems as Newtonian dynamics *must* be non-deterministic. This is quite surprising, since the Newtonian model has traditionally been seen as an archetype of deterministic physics: as Laplace puts it [Lap51, p. 4]

> An intellect [*Laplace's demon*] which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.

The failure of this claim follows from a remarkable result of Zhihong Xia, published some 20 years ago [Xia92], demonstrating that the Newtonian n-body problem possesses 'non-collision singularities.' That is, we can have a system of objects interacting gravitationally, one or more of which are propelled to infinity in finite time. The key point here is that the laws of Newtonian physics are unaffected if we mentally reverse the direction of time. If we do so in the context of Xia's result, this tells us that objects can appear *from* infinity in finite time, and indeed there is no limit to how quickly they can do so. Consequently, even if Laplace's demon were equipped with complete knowledge of the current state of the universe, it would *not* be able to determine the subsequent state even one second later, because the spontaneous arrival of new material during the intervening period would inevitably introduce gravitational forces, or even collisions, that could not have been forecast in advance.

2.1 The significance of interaction

In Step 2 the clerk uses **HP** as a component in the construction of **Diag**. For the sake of argument we will assume that **HP** is implemented as a separate agent (i.e., on a separate machine) with which the clerk and **Diag** can interact; furthermore, to avoid circular reasoning, we will assume throughout that the agent is essentially just a digital computer, with no hypercomputational capabilities of its own. Using agents in this way is permitted under Turing's analysis, as he explains in another of his landmark papers [Tur50, emphasis added]:

> The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail. We may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. He has also an unlimited supply of paper on which he does his calculations. *He may also do his multiplications and additions on a "desk machine," but this is not important.*

As we shall see, the problem with this analysis lies not in the description of the human computer's (i.e., the clerk's) behaviour, but in the 'throw-away' claim that allowing the clerk to interact with another agent – in this case a desk machine – is "not important." We stress again that we are not assuming that the agent itself has 'super-Turing' capabilities of any kind; indeed, the agent in question might simply be another clerk. The important feature of agent-assisted computation is, rather, the physical separation between clerk and agent, since this implies that they can be in motion relative to one another, subject to different forces and accelerations – and as Einstein has taught us, this means that their perceptions of space and time need not agree with one another [Ein20].

2.2 Accelerating machines

Suppose the agent is based high above the Earth's surface, while the clerk operates at sea level. The difference in gravitational potential between the two locations will ensure that time for the agent appears to run faster than for the clerk.[1] While there are limits to the speed-up that can be achieved in this way, the scenario naturally raises the question whether *accelerating machines* can be implemented. In its simplest form, an accelerating machine is one in which each instruction takes half as long to execute as its predecessor, so that if the first instruction takes 1 sec, even a non-terminating computation will have run to completion after 2 sec – for example, if we placed the agent on board a rocket so that it moves ever further away from us and with ever increasing acceleration, this could result in each instruction taking less time to run (from our point of view) than its predecessor. Such a simple scheme is fraught with physical and logical difficulties [Tho54], but it is nonetheless instructive to consider how it might be used to solve **HP**, and what the difficulties would be in doing so. We will then be in a position to relate our findings to the (arguably more realistic) hypercomputational potential of, e.g., tissue P systems [BG10].

Accelerating machines have been discussed in the context of P systems by Calude and Păun [CP04], based on the observation that reactions tend to run faster in smaller volumes (assuming that concentration increases accordingly). By recursively constructing ever smaller subregions and having them compute subroutines ever faster, one can achieve exactly the speed-up required to solve **HP** and its kindred problems. More recently, Gheorghe and Stannett [GS12] have extended this principle to solve problems at all levels of the *arithmetical hierarchy* (and beyond) [AK00]. Taking $\mathcal{P}_0$ to be the class of 'standard' P systems, we can define a hierarchy of systems $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \ldots$, where a $\mathcal{P}_{n+1}$ system is a Calude-Păun accelerating P system in which the systems replicated at each stage are $\mathcal{P}_n$ systems. For example, the original Calude-Păun accelerating P system model generates a $\mathcal{P}_1$ system under this scheme, since the replicated components are all standard $\mathcal{P}_0$ systems. As shown in [GS12], each $\mathcal{P}_n$ is strictly more powerful than its predecessor, and together they exhaust the entire hierarchy.

[1] See, e.g., [CHRW10] for experimental confirmation of this long-standing claim.

How might an accelerating machine be implemented physically? Notice first that neither the agent nor the clerk can solve **HP** on their own, because their separate behaviours are still susceptible to the limitations identified by Turing's analysis; solving **HP** requires the agent and clerk to *cooperate* with one another. Provided they agree to do so, deciding whether or not some computation $P_m(n)$ eventually terminates is simple.

1. The clerk transmits the values m and n to the accelerating agent.
2. The accelerating agent executes $P_m(n)$, and has agreed that in the event of the computation halting, a message will be sent back to the clerk saying so.
3. The clerk waits two seconds to allow the agent sufficient time to run the program, adds a further delay corresponding to the maximum transit time of the potential signals involved, and then checks to see whether a message has been received from the agent. If so, the computation halted, and the clerk reports *yes*. If not, the clerk reports *no*.

Let us analyse the timing structure of this system in more detail, since it is the same for *any* system in which (a) the clerk uses an agent to execute $P_m(n)$; and (b) the clerk has to determine in finite time whether or not the agent's execution of $P_m(n)$ terminated.

– The clerk (A) and the agent (B) communicate at the start of the procedure;
– The agent may need to run the program forever, but even in this case the clerk has to perceive the computation as requiring only finite time relative to her own clock.
– There must come a point later in the clerk's life where the termination or otherwise of the agent's program execution can be identified.

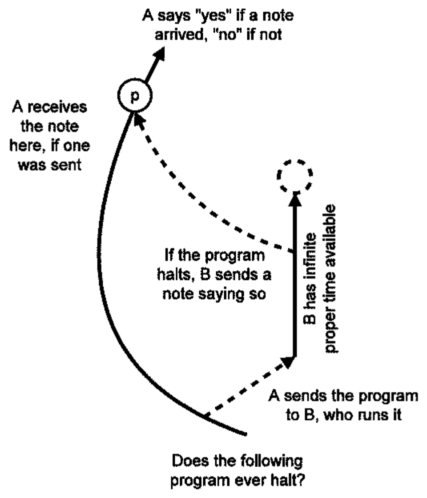

A says "yes" if a note arrived, "no" if not
p
A receives the note here, if one was sent
If the program halts, B sends a note saying so
B has infinite proper time available
A sends the program to B, who runs it
Does the following program ever halt?

Fig. 2. The underlying timing structure involved in the cooperative solution of **HP** is that of Malament-Hogarth spacetime.

In relativistic language, we are saying (Fig. 2) that

1. the agent's worldline should allow the agent *infinite proper time*;
2. there is a point, p, on the clerk's worldline such that: at any point x on the agent's worldline it is possible for the agent (eventually) to send an agreed signal s from x to the clerk, so that s is received by the clerk earlier than p.

Cosmological spacetimes that include timing structures of this nature are called *Malament-Hogarth* (MH) spacetimes [EN93], and schemes have been proposed showing that the existence of stable MH-spacetimes is sufficient to allow *cosmological hypercomputation* to be implemented [EN02]. The analysis above suggests that the use of MH timing structures is also *necessary* if problems like **HP** are to be solved cooperatively by standard computational systems.

While MH structures seem exotic at first sight, they are associated with large slowly-rotating (*slow Kerr*) blackholes of the kind thought to exist at the centre of many galaxies (including our own [GET$^+$09]), and this makes them usable by the clerk for computational purposes. It might be argued, of course, that using the Galactic centre in this way is of only technical interest but has no practical relevance due to the vast distances involved. However, this neglects another important aspect of Turing's analysis. In proposing his model of human computation, Turing placed no limitations on how long a task might take to complete; and indeed complexity theory has shown that even fairly simple tasks may take longer than the current age of the Universe to run to completion on a standard PC. In contrast, a rocket travelling at 11 km s^{-1} (escape velocity at the Earth's surface) towards the Galactic centre (roughly 28,000 light years away [GET$^+$09, Maj10]) would require only around 763 million years to arrive there. While this is certainly a long time, it nonetheless compares well with the expected runtime of certain computations; it is therefore hard to see why the use of the Galactic centre should be considered any less reasonable than the use of arbitrarily long-lived Turing machines when determining what can and can't be computed.

The use of slowly rotating massive blackholes for hypercomputational purposes[2] is discussed in detail by Etesi and Németi [EN02]. As one falls into the blackhole one is inexorably drawn through a region linking an outer to an inner event horizon, but thereafter things return to 'normal' in the sense that one can move freely, and in particular one can avoid hitting the singularity. In their scenario the clerk chooses to fall into the blackhole, leaving the agent orbiting outside. Due to time dilation effects the agent's time appears to run ever faster the nearer the clerk gets to the horizon, thereby implementing the required MH timing structure. After crossing the horizon, the clerk knows whether or not a signal has been received from the agent, and then continues into the inner 'safe zone' where she makes use of the information. This scheme is not without its problems, however, since there are clear indications that the blackholes in question may exhibit inherent instabilities [Pen68, Hod12]. An important open question, therefore, is whether other cosmological examples of MH timing structures can be identified for which these instabilities are provably absent.

2.3 Cosmological P systems

One can easily adapt the MH-spacetime model of (hyper)computation to produce a new hypercomputational tissue-based model, which we will refer to as a *cos-*

[2] Other cosmological approaches also exist e.g., the exploitation of closed timelike curves (CTCs) and wormholes [ANS12, Sta13].

mological P system. Looking again at Fig. 2, we begin by envisaging a contiguous population of membrane systems ("cells") which begins as a small population based where A and B first diverge. This population replicates, spreading at the same, constant, average speed in all directions. It is not the cells which generate the hypercomputational speed-up, but the geometry of the spacetime in which they are replicating, for by the time a new cell has been created at p, it will 'perceive' infinitely many cells to have been generated along B. From the viewpoint of any cell on B, however, there is nothing unusual happening locally – the regeneration time remains unchanged from its own point of view.

To see how the computation proceeds, we observe that the original model involves three distinct entities: the clerk, the agent, and the spacetime through which signals are propagated. Accordingly, we need the cells that form along A and B to differentiate themselves both from each other and from those which fill the rest of space at any given moment. We therefore assume that the cell strain is initially *spacetime* – this cell type simply propagates signals in straight lines (in other words, it includes rewrite rules of the form "*if* signal *is present in the cell, place a copy of* signal *in all neighbouring cells*", thereby ensuring recursive re-transmission of incoming signals). In contrast, we assume that as the clerk and agent move along their respective trajectories, they emit promoters into the cells' environment which trigger the conversion of *spacetime* cells into A-type or B-type cells, respectively. This ensures the generation of two filaments within the general population, one composed of A-type cells, the other of B-types (Fig. 3).

The behaviour of A- and B-type cells is essentially straightforward.

– A-type cells respond to the presence of a *signal* compound by converting it into a *yes*, which is then replicated in all descendants. At all times, an A-type cell responds to the clerk's chemical promoter by extending the filament along A's trajectory, while replicating standard *spacetime* cells in all other directions. If at any point a *signal* or a *yes* is present in the cell, it includes *yes* in the 'genome' of its immediate descendants.
– B-type cells perform the actual computation of $P_m(n)$. We encode the program counter and registers as chemical species in the cell, along with the program itself. Each cell simulates the execution of one instruction, and then generates the next cell along the B-trajectory so that it

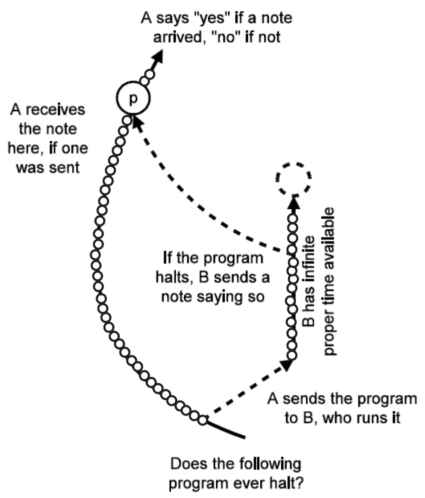

A says "yes" if a note arrived, "no" if not
p
A receives the note here, if one was sent
If the program halts, B sends a note saying so
B has infinite proper time available
A sends the program to B, who runs it
Does the following program ever halt?

Fig. 3. Building a tissue P system with hypercomputational power.

contains a full copy of the program, together with the coded versions of the updated program counter and registers. If at any point the cell determines that the program counter and registers remain unchanged (i.e., the program halts), the cell emits a *signal* in all directions.

In this way, $P_m(n)$ is executed by the growing B-filament, and a *signal* is received at p if and only if the program eventually terminates – thereby solving **HP**. This is, of course, unsurprising, since neither the clerk nor the agent need be entities with continuous existence. All that matters is the information they carry with them as they travel along their respective trajectories. By replicating these information flows, we automatically replicate the associated computational power.

3 Summary and Further Research

In this paper we have revisited Turing's analysis of computation, and considered how it can be subverted by taking into account the physical separation between cooperative agents. This in turn leads to analysis of cosmological models of hypercomputation based on Malament-Hogarth spacetimes, and their simulation via tissue P systems. This suggests a number of avenues for further research, for example:

– The behaviour of a 'cosmological' P system can clearly be replicated instead using a cellular automaton. The advantage of the P system approach lies in the system's self-generation – instead of presupposing a pre-existing infinite population of communicating automata, the cells of the tissue simply replicate as time goes by, filling spacetime as they do so. However, the model relies on interactions between the three component cell strains and two 'external' entities (the clerk and the agent) which move through the underlying spacetime scattering promoter molecules. Can these be modelled directly within the P system paradigm, or is their autonomous nature necessary for the model to work?

– We have only discussed one approach to hypercomputation, namely the use of slow-Kerr black holes. However, the wider literature discusses numerous computational models of super-Turing computation (analogue recursive neural networks, trial-and-error machines, and the like [Sta06]). To what extent can each of these models be extended or re-interpreted in the context of membrane systems?

– Can the models in question be formalised, and their properties verified mechanistically via a theorem prover or proof assistant? Providing concrete formal analyses can be expected to add support to our claims that hypercomputation in the context of P systems is physically meaningful. My colleagues and I have recently started investigating this area, but much work remains to be done.

References

[AK00] C. J. Ash and J. F. Knight. *Computable Structures and the Hyperarithmetical Hierarchy*. Elsevier, Amsterdam, 2000.

[ANS12] H. Andréka, I. Németi, and G. Székely. Closed Timelike Curves in Relativistic Computation, 2012. `arXiv:1105.0047[gr-qc]`.

[BCLT08] E. J. Beggs, J. F. Costa, B. Loff, and J. V. Tucker. Computational complexity with experiments as oracles. *Proc. Royal Society, Series A*, 464:2777–2801, 2008.

[BG10] F. Bernardini and M. Gheorghe. Tissue and Population P Systems. In G. Păun, G. Rozenberg, and A. Salomaa, editors, *The Oxford Handbook of Membrane Computing*, pages 227–250. OUP, Oxford, 2010.

[CHRW10] C. W. Chou, D. B. Hume, T. Rosenband, and D. J. Wineland. Optical Clocks and Relativity. *Science*, pages 1630–1633, 24 September 2010.

[CP04] C. S. Calude and Gh. Păun. Bio-steps beyond turing. *BioSystems*, 77:175–194, 2004.

[Ein20] A. Einstein. *Relativity: The Special and General Theory*. Henry Holt, New York, 1920.

[EN93] J. Earman and J. Norton. Forever is a Day: Supertasks in Pitowsky and Malament-Hogarth Spacetimes. *Philosophy of Science*, 5:22–42, 1993.

[EN02] G. Etesi and I. Németi. Non-Turing computations via Malament-Hogarth space-times. *Int. J. Theoretical Physics*, 41:341–370, 2002. `arXiv:gr-qc/0104023v2`.

[GET$^+$09] S. Gillessen, F. Eisenhauer, S. Trippe, T. Alexander, R. Genzel, F. Martins, and T. Ott. Monitoring stellar orbits around the Massive Black Hole in the Galactic Center. *The Astrophysical Journal*, 692:1075–1109, 23 February 2009.

[GS12] M. Gheorghe and M. Stannett. Membrane system models for super-Turing paradigms. *Natural Computing*, 11:253–259, 2012.

[Hod12] S. Hod. On the instability regime of the rotating Kerr spacetime to massive scalar perturbations, 2012. `arXiv:1205.1872v1[gr-qc]`.

[Lap51] P. S. Laplace. *A Philosophical Essay on Probabilities*. Dover Publications, New York, 1951. Translated into English from the original French 6th ed. by F. W. Truscott and F. L. Emory.

[Maj10] D. Majaess. Concerning the Distance to the Center of the Milky Way and its Structure. *Acta Astronomica*, 60(1):55–74, 2010.

[Pen68] R. Penrose. Structure of spacetime. In C. M. DeWitt and J. A. Wheeler, editors, *Battelle rencontres*, pages 121–235. W. A. Benjamin, New York, 1968.

[Ric05] J. Richard. Les Principes des Mathématiques et le Problème des Ensembles. *Revue Générale des Sciences Pures et Appliquées*, 30 June 1905.

[Sta06] M. Stannett. The case for hypercomputation. *Applied Mathematics and Computation*, 178:8–24, 2006.

[Sta13] M. Stannett. Computation and Spacetime Structure. *Int. J. Unconventional Computing*, in press, 2013. Special Issue on New Worlds of Computation 2011.

[Tho54] J. F. Thomson. Tasks and Super-Tasks. *Analysis*, 15(1):1–13, October 1954.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc., Series 2*, 42:230–265, 1937, submitted May 1936.

[Tur39] A. M. Turing. Systems of Logic Based on Ordinals. *Proc. London Math. Soc., Series 2*, 45:161–228, 1939.

[Tur50] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[vH77] J. van Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931.* Harvard University Press, Cambridge, MA, 1977.

[Xia92] Z. Xia. The existence of noncollision singularies in Newtonian systems. *Annals of Mathematics*, 135:411–468, 1992.