Hyperloops Do Not Threaten the Notion of an Effective Procedure

Tim Button*

Cambridge University, button@cantab.net

This paper was published in 2009 by Springer-Verlag. The authoritative version should be obtained from SpringerLink. The paper should be cited as follows.
Button, Tim. 2009. "Hyperloops do not threaten the notion of an effective procedure". *CiE 2009 LNCS*, 5635, pp.68–78.

Abstract. This paper develops my (forthcoming) criticisms of the philosophical significance of a certain sort of infinitary computational process, a *hyperloop*. I start by considering whether hyperloops suggest that "effectively computable" is vague (in some sense). I then consider and criticise two arguments by Hogarth, who maintains that hyperloops undermine the very idea of effective computability. I conclude that hyperloops, on their own, cannot threaten the notion of an effective procedure.

It has been suggested that a particular kind of infinitary hypercomputer threatens the notion of an effective procedure. The infinitary computer in question is able to perform a *hyperloop*. I start by explaining what hyperloops are, and how they might be implemented in a general-relativistic spacetime (Sect. 1).

I consider whether hyperloops suggest that "effectively computable" is *polyvague*. I give reason to doubt this, but I also suggest that this kind of vagueness cannot seriously threaten the notion of an effective procedure (Sect. 2).

Next, I consider Hogarth's (2009b) claim that ordinary computers and hyperloopers, running the same algorithm, will give different outputs. Hogarth concludes that the output of an algorithm is intrinsically relativised to the machine that implements it. However, I show that hyperloopers only give different outputs because they implement *different* algorithms (Sect. 3).

Hogarth (2004, 2009a, 2009b) also maintains that there is a deep analogy between computing and geometry. He thinks that this shows that the very idea of an effective procedure is bankrupt. I suggest that Hogarth's attack on the notion of an effective procedure rests on a premise that is independent from considerations concerning hyperloops. This premise is independently interesting, but hyperloops, as such, tell us nothing about effective procedures (Sect. 4).

1 Hyperloops: from Python to Ouroboros

Let `fun()` be an arbitrary computable function, which takes a single natural number as an argument, and returns a boolean. If we ignore the fact that every

* Thanks to Mark Hogarth, Philip Welch, Sharon Berry, Michael Potter, Peter Smith, and five anonymous referees for this journal.

actual computer only has a only finite storage capacity, then, for any value of `x`, an ordinary computer can test whether `fun(x)`. But an ordinary computer cannot be expected to determine whether there *is* a value of `x` such that `fun(x)`. We can set an ordinary computer searching for a value of `x` such that `fun(x)`; but if `fun(x)` is not true for any value of `x`, the computer will loop through an ω-sequence of calculations, without ever terminating.

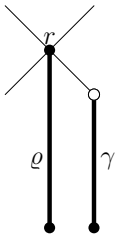
But suppose we had a computer which could *complete* ω-sequences of calculations. We could use this *hyperlooper* (a hyperlooping-computer) to test whether there is some value of `x` such that `fun(x)`. Programme 1 captures the idea:

Programme 1

```
xFound = 0                                                    1.1
hyperloop with x = 0:                                         1.2
      if fun(x):                                              1.3
            xFound = 1                                        1.4
            break                                             1.5
      x = x+1                                                 1.6
if xFound: print "Yes"                                        1.7
else: print "No"                                              1.8
```

This programme is written in a hypothetical programming language, Ouroboros. Ouroboros extends the Python language by adding the command "`hyperloop with...`". Executing Programme 1, the computer initialises a new variable, `x`, to `0`, and then starts looping through lines 1.3–1.6. If there is a value of `x` such that `fun(x)`, the computer discovers this, sets `xFound` to `1`, and `break`s out of the hyperloop (to line 1.7). If there is no value of `x` such that `fun(x)`, the computer loops through all possible values of `x` without ever changing `xFound` from its initial value of `0`. Having checked every value of `x`, the computer proceeds to line 1.7. So the computer is guaranteed to reach line 1.7, and when it does, `xFound` has the value `1` iff there is a value of `x` such that `fun(x)`.

Hogarth has noted that general relativity may supply a physical framework in which to implement Programme 1. The idea is to find a spacetime in which a human user, Dave, can survey the entire *infinite* worldline of some ordinary computer, Hal.[1] In particular, in the diagram to the right, a finite part of Dave's wordline is ϱ, and Hal's entire infinite worldline is γ. On γ, Hal executes the following Python programme:



Programme 2a

```
x = 0                                                         a.1
while fun(x) == 0:                                            a.2
      x = x+1                                                 a.3
sendASignalToDave()                                           a.4
```

If Hal finds a value of `x` such that `fun(x)`, Hal sends a signal to Dave (perhaps a light-beam). Dave carries a receiving device with him, which stores a variable,

[1] The required spacetime, the spacetime diagram, and this implementation are discussed in Hogarth (1992, 1994, 2004, 2009a); Earman & Norton (1993); Etesi & Németi (2002); Németi & Dávid (2006); Welch (2008).

`xFound`, initialised to a default value of `0`; this receiver sets `xFound` to `1` on (and only on) receipt of a signal from Hal. Once Dave reaches point r, Hal's entire *infinite* history lies in Dave's past, and any signal that Hal might have sent to Dave will have arrived. So from Dave's perspective, by point r, Hal has completed a hyperloop. Dave then executes the following Python programme:

Programme 2b

```
if xFound: print "Yes"                                    b.1
else: print "No"                                         b.2
```

The overall effect is a complete implementation of Programme 1.

With the formal and physical notion of a hyperloop on the table, several questions concerning hyperloops arise.

First: *What functions can be computed with hyperloops?* Welch (2008) has answered this technical question.[2] For the purposes of this paper, the only result we require is that hyperloopers are much more powerful than Turing computers.

Second: *Are hyperloopers genuine physical possibilities?* Much interesting foundational work has addressed this question, focussing primarily on the relativistically-realised-hyperloops just discussed.[3] But in this paper, I shall avoid all questions about physical computability.

The question I wish to consider is: *What consequences do hyperloops have for the notion of effective computability?* Note that, although some authors identify "effectively computable" with "physically computable",[4] this is *not* what I have in mind. I equate the *semi-formal* notion of an algorithm with that of an effective procedure, and I say that a function is effectively computable if and only if there is some effective procedure (algorithm) for computing it.[5]

2 Polyvagueness

I want to start by considering whether the discussion of hyperloops shows that "effectively computable" exhibits a certain kind of vagueness. I shall suggest that it probably does not, but that even if it does, there is no reason to worry.

[2] A computer which executes a single hyperloop can complete an ω-sequence of computations, and so can decide membership of any Π_1 or Σ_1 set. By nesting n-many hyperloops inside each other, we can write Ouroboros programmes for computers that complete ω^n-sequences of computations, and so decide any Π_n or Σ_n set. Beyond the programming capabilities of Ouroboros, but retaining the basic idea of a hyperloop, we can define hyperloopers to decide membership up through the hyperarithmetical sets. This is the limit of computation with hyperloops.

[3] e.g. Etesi & Németi (2002); Németi & Dávid (2006); and consult references in their papers. Even if relativistically-realised-hyperloops ultimately fail, some *other* physical model of a hyperloop may be feasible. For instance, Davies (2001) describes another realisation of hyperloops (which requires the infinite divisibility of matter).

[4] e.g. Etesi & Németi (2002 p.348).

[5] See Smith (2007 pp.315–23), Button (forthcoming Sect. 3.1).

Smith (2007 p.327) calls a predicate *polyvague* "if it ambiguously locates more than one mathematical kind." To illustrate the idea of polyvagueness, consider Euler's conjecture: for all polyhedra, Vertices − Edges + Faces = 2. Lakatos (1976) investigates various potential counterexamples to the conjecture, including: a picture frame; a cube with a hollow cubic interior; and a cylinder. But before admitting these *as* counterexamples to the conjecture, we must determine whether these objects are even polyhedra. Perhaps that is (or was) not clear in every case; in which case "polyhedron" is (or was) polyvague.

Shapiro (2006) has argued that "effectively computable" began life as a polyvague predicate, but that it has been gradually sharpened over time.[6] To show this, he gives examples of procedures for which there was no unanimous consensus (at one time or other) as to whether they were effective. I want to ask whether relativistically-realised-hyperloops, as described in Sect. 1, constitute a further example to show that "effectively computable" was polyvague.

An affirmative answer might be motivated thus. In early discussions of effective procedures, we might have suggested that an effective procedure is one which takes only a finite duration of our time to complete. Alternatively, we might have suggested that an effective procedure is one which takes only a finite number of discrete stages to complete. These characterisations might have seemed to be necessarily co-extensive. However, thinking about relativistically-realised-hyperloops shows that they are not. On the former characterisation (finite duration), relativistically-realised-hyperloops are effective procedures. On the latter characterisation (finite stages), any hyperloop fails to count as effective. Since there are at least two "natural kinds" here, maybe this shows that "effectively computable" was polyvague.

For now, I will concede that this shows that "effectively computable" was polyvague. It does *not* follow that the notion of an effective procedure is bankrupt. Discovering that a predicate is polyvague rarely undermines the notion at hand. For instance, even if Lakatos is right that "polyhedron" is polyvague, the notion of a polyhedron may well remain useful; but we may want to sharpen it, for some purposes. And generally, discovering that a predicate is polyvague forces us to ask two questions. First: why do we need to use the predicate? Second: which way(s) of precisifying the predicate best serves those needs?

It is relatively easy to state why we need the predicate "effectively computable". We need to think about effective computability and effective procedures whenever we talk about what can be done using algorithms (recall my explicit characterisation of "effective computability" in Sect. 1).

So, which way of sharpening "effectively computable" best serve our need to talk about algorithms? In the present case, we must choose whether or not to count relativistically-realised-hyperloops as effective procedures. Once we focus on the question of *physical realisation*, the choice is quite clear. Suppose we decide to treat relativistically-realised-hyperloops as effective procedures, on the grounds that they take only a finite duration (from our perspective). Other

[6] More accurately, Shapiro (2006) says that "idealized (human) computable" is "open textured". Both Shapiro and Smith invoke Lakatos to illustrate the idea at hand.

physical realisations of hyperloops will require infinite periods of time (from our perspective). So given a single procedure involving a hyperloop, we will have to treat some physical realisations of it as effective, and treat other realisations as ineffective. Physicists may be happy to draw a distinction on these lines, but this is not the kind of distinction that seem relevant to logical and mathematical reasoning.[7] From the latter perspective – which is the home of algorithms – it would be better to abstract the notion of an effective procedure from specifics relating to the various *physical realisations* of those procedures. This abstraction favours the definition of an effective procedure in terms of its number of stages, rather than in terms of its duration.

Accordingly, I think it is clear that relativistically-realised-hyperloops ought not (now) to count as effective procedures. But, more importantly: even if it was once an open question whether relativistically-realised-hyperloops ought to count as effective procedures, that does not by itself undermine the very idea of an effective procedure.[8]

3 The Absolute Nature of Algorithms

With this background fixed, I wish to consider Hogarth's (2009b) argument that the output of an algorithm is relative to the machine that implements the algorithm. To fix ideas, Hogarth considers a simple algorithm:

Algorithm 3

```
counter = 0                                    3.1
loop:                                          3.2
      print counter                            3.3
      counter = counter + 1                    3.4
```

We are to consider implementing Algorithm 3 on two different machines, with a standard realisation of the `print` function:[9]

A. *An ordinary computer.* This machine will output an ascending sequence of arabic numerals, "0","1","2",... representing finite numbers.
B. *A relativistically-realised-hyperlooper.* In this case, Hal (on γ) is to send a signal to Dave's receiver (on ϱ) every time Hal increases the value of `counter`. Dave's receiver decodes the signal so as to output all the arabic numerals, "0","1","2",.... After this (at point r) it will output "ω", representing ω, and will then continue, "ω+1","ω+2",....

[7] cf. Hamkins & Lewis (2000 p.568). I revisit this, emphasising *reasoning*, in Sect. 4.3.

[8] In fact, I suspect that it never was a genuinely open question. The founders of computability theory were well aware of supertasks and oracles, and so I conjecture that they would have declared that relativistically-realised-hyperloops are not effective procedures. But I shall not push this. Note also that I have no objection to our describing *hyper-effective* procedures, if we so wish. (See my discussion (forthcoming) of "effective SAD procedures".)

[9] Hogarth also considers implementing Algorithm 3 on a computer that nests a hyperloop within a hyperloop, and so completes an $\omega^2 + n$-sequence of computations.

The two implementations give different outputs, and Hogarth concludes that the output of an algorithm is generally relative to the machine that implements the algorithm.[10] As Hogarth notes, if this is correct, it is extremely significant: if algorithms are essentially relative, then *all* those areas of logic and mathematics that utilise effective procedures or algorithms are likewise essentially relative.

But before discussing Hogarth's claim, it will help to recall a point from the literature on supertasks. We shall focus on Thomson's Lamp (1954). At time $t = 0$, Thomson's Lamp is off. At $t = \frac{1}{2}$, it is flicked on. At $t = \frac{3}{4}$, it is flicked off. At $t = \frac{7}{8}$, it is flicked on.... Is the lamp on or off at $t = 1$? No answer seems warranted, and this is sometimes thought paradoxical. But it need not be so conceived. As Benacerraf (1962) notes, we have described the state of the lamp *only* for times $0 \leq t < 1$, so *any* state of the lamp at $t = 1$ is compatible with all its previous states. Paradox dissolved![11]

Let us now return to Hogarth's argument. By analogy with Thomson's Lamp, we ask the following question: in Implementation B, what is the value of the stored variable `counter` at stage ω? As for Thomson's Lamp, no answer seems warranted. Hogarth wants to say that the value is `ω`, but why not `0`, or `783`, or simply `Error`? Given a value of `counter` at some stage in the computation, we know the value of `counter` at the next stage: Algorithm 3 tells us that its value increases by 1 at each successor stage. But nothing in Algorithm 3 determines `counter`'s value at limit stages (i.e. at stages numbered by some limit ordinal). Any value of `counter` at stage ω is consistent with all its previous states.

One might think that Algorithm 3's output is essentially relative *precisely because* Algorithm 3 does not define the computer's behaviour at limit stages, and so because anything could happen at limit stages. This observation saves Implementation B from *paradox*, if it saves Thomson's Lamp from the same charge. But it does not save Hogarth's argument. Consider an "algorithm" which merely initialises a variable, then prints it. Should we say that the output of this algorithm is interestingly relative to its implementation, because anything could happen at successor stages? Hardly; but then we should not say that the output of Algorithm 3 is interestingly relative, just because it leaves a computer's behaviour undefined at limit stages.

The point can be put starkly as follows. Algorithm 3 defines a function by recursion. Hogarth wants to extend that definition into the transfinite. This

[10] Hogarth also maintains that this raises a "new problem for rule-following" since, when someone asks us to count and never stop, we do not know whether they want us to keep counting *through* ω and beyond.

[11] Allegedly; some philosophers are not content with this resolution. For example, Dummett (2005 pp.143–4) maintains that a complete description of a lamp's states for $0 \leq t < 1$ *ought* to entail its state at $t = 1$.

plainly requires an explicit transfinite recursion clause.[12] It is easy to provide one within the framework of Ouroboros:

Programme 4

```
hyperloop with counterA = 0:                          4.1
      print counterA                                  4.2
      counterA = counterA + 1                         4.3
counterB = 0                                          4.4
while counterB > -1:                                  4.5
      print "ω+" + str(counterB)                      4.6
      counterB = counterB + 1                         4.7
```

An implementation of Programme 4 seems to be what Hogarth has in mind in Implementation B. But line 4.4 explicitly defines the machine's behaviour at stage ω. So a hyperlooper running Programme 4 has a different output from an ordinary computer implementing Algorithm 3, precisely because they implement *different* algorithms. In short, hyperloops do not suggest that the output of an algorithm is essentially relativised.[13]

4 Hyperloops, Computability, and Geometry

Hogarth (2004 pp.689–690, 2009a, 2009b Sect.2) presents a rather different argument against the idea of an effective procedure, which draws a connection between computability and geometry.

Consider a geometrical question, such as "what is the sum of a triangle's interior angles?" This might be a question about the spatiotemporal structure of our world. Or it might be a purely formal question within a particular formal geometric theory. But, Hogarth claims, there are *only* physical and formal questions here. In particular, there is no "intuitively true" geometry to ask about.

Hogarth believes that the same is true of computing. When we ask whether a function is computable, we might be asking whether some physical machine could compute that function. Or we might be asking a purely formal question within a purely formal theory of computability. But we cannot ask any *other* question. Just as there is no "intuitively true" geometry, there is no room for a notion of effective computability that is neither formal nor physical.

In what follows, I shall consider three ways to read this analogy. My main aim is to show that hyperloops should play no part in it.

---

[12] Indeed, Hogarth provides one himself, saying that "ω... is the interpretation of the absence of a signal" from Hal (2009b Appendix). This is essentially an *extension* of the algorithm into the transfinite. It is probably worth noting that this extension is not quite right. ω is the interpretation of the absence of a signal *at point* r; indeed, ω is just the interpretation of *reaching point* r.

[13] Hogarth's "new problem for rule-following" (see fn. 10, above) dissolves in the same breath. Having added explicit rules to govern limit stages, we cannot pretend that we are "going on exactly as before".

4.1 A Circular Argument

In Button (forthcoming Sect. 3.3), I reconstruct the analogy thus:

1. Which functions can be computed by a (configuration of) machine(s) depends upon whether the machine(s) can execute hyperloops.
2. Whether machines can execute hyperloops can only either:
   (a) depend upon the laws of physics (and so concerns physical machines); or
   (b) be stipulated within some formal theory (and so concerns the formal definition of a "machine").
3. So which functions can be computed by a (configuration of) machine(s) is either a purely formal or a purely physical question. There are no interesting questions about effective computability.

Strikingly, any non-Turing theory of computability can be used in place of hyperlooping in this argument. To demonstrate this, consider a parallel argument. The tape of a Turing machine can be in any of infinitely many possible states. By contrast, a *finite-state-machine* has only finitely many possible states, with transition rules between them. Finite-state-machines are formally much weaker than Turing machines. Now:

1*. Which functions can be computed by a (configuration of) machine(s) depends upon whether there are infinitely many, or only finitely many, possible total-machine-states.[14]
2*. The number of possible total-machine-states can only either depend upon the laws of physics, or be stipulated within some formal theory.
3. So, as above, there are no interesting questions about effective computability.

This parallel argument is obviously irrelevant: when we think about effective computability, finite-state-machines are simply too weak to concern us. But this parallel argument and the original argument have exactly the same structure. So the argument structure must *itself* be too strong. We must determine why.

To deliver an analogue of premise 1, we need two technical results. First, we must describe a formal computer and prove that it is not equivalent to a Turing computer. Second, we must describe a system of physical laws that favours this non-Turing computer (we may do this by describing spatiotemporal laws, but we could in principle describe laws of any sort). These two technical results may arise from, or give rise to, important work in the foundations of computability and the physical sciences. However, we have a philosophical argument against effective computability only when we assert, in an analogue of premise 2, that *only* physical or formal issues could decide between a Turing computer and this non-Turing computer. This assertion carries all the philosophical burden of the argument, and it is obviously contentious. So what could justify it? It is certainly entailed by what I call "Hogarth's Thesis":

[14] A *total-machine-state* is meant to be fully encompassing; e.g., in the case of a Turing machine, it incorporates both the machine's *internal* state and the state of its *tape*.

Hogarth's Thesis: There are questions about what is physically computable; there are questions about what is formally computable (in various different formal systems); but there is no third side to computability.

However, Hogarth's Thesis is just a restatement of the desired *conclusion* of the original argument. What we need is an *independent* argument for Hogarth's Thesis, which arises from considering hyperloops. But none is forthcoming from the reading of Hogarth's analogy that we have been considering.[15]

4.2 The (Non-)Spatiotemporality of Computation

There is a much stronger reading of Hogarth's analogy between geometry and computability. According to the strong reading, "intuitive" computability is "intuitive" physical computing in some "intuitive" spacetime. But there is no such thing as the "intuitively true" geometry or spacetime. So we cannot dismiss as "unintuitive" the (relativistic) geometries which make hyperloops possible. So we must give up on the idea of "intuitive" computability.

This strong reading draws a tight link between geometry and computability, and thereby reinstates the role of hyperloops in the attack on effective computability. Indeed, it is misleading to call this link between geometry and computability a mere "analogy". On this reading, computation is intrinsically spatiotemporal, so computation and geometry *must* share the same fate.[16]

However, as stressed in Sect. 2, there is a good notion of effective computability which is *not* spatiotemporal. That notion does not concern computation in some "intuitive" spacetime. Rather, it concerns the *abstract* notion of a procedure of arbitrary finite length, where length is measured by number of stages and not temporal duration. Hyperloops – procedures of infinite length, in these terms – cannot speak much against this notion. Indeed, *hyperloops can threaten the atemporal notion of an algorithm only if they can threaten the semi-formal notion of an arbitrary finite sequence of stages.*

So, *can* hyperloops threaten the notion of an arbitrary finite sequence? Their only shot is as follows. Suppose we define a *finite number* as one that can be counted to in a finite duration of our time. Then there is undoubtedly a sense in which we can count through ω using a relativistically-realised-hyperloop (see Sect. 3). So this might show that what counts as a finite number is relativised to a physical machine, or to a formal system. By association, the notion of a finite sequence would also be relativised.

We already know how to respond to this argument. First, the procedure for counting through ω is obviously a *new* procedure, so cannot threaten our grasp of the old procedure (compare Sect. 3). Second, we should contest any definition of "finite number" in terms of what can be counted to in a particular *duration*: finite numbers are rather more abstract (compare Sect. 2).

[15] To be fair to Hogarth, note again that this was *my* reconstruction of his analogy!
[16] This reading is suggested by Hogarth (2009a p.277, 2009b fn.2).

4.3 An Invitation to Accept Hogarth's Thesis

Consequently, hyperloops simply cannot threaten the non-spatiotemporal notion of effective computability. But this observation is likely to bring us simply to an impasse. One side of the impasse accepts Hogarth's Thesis and regards the very idea of effective computability as an outdated "relic". The other side rejects Hogarth's Thesis, so regards relativistically-realised-hyperloops as "obviously irrelevant". Both sides will be equally frustrated with the other.

The only way way to break the impasse is to consider Hogarth's Thesis directly, without talking about hyperloops. And this suggests a final, weaker reading of Hogarth's analogy. On this weak reading, Hogarth is not attempting to *argue* against the notion of effective computability. He is simply *inviting* us to let go of effective computability, and accept his Thesis.

I am grateful for the invitation, but I am also wary of it. We can all agree that actual reasoning takes place in spacetime. But I doubt that *what follows from something by reason alone* is either a spatiotemporal or purely formal question. Similarly, we can all agree that actual computing takes place in spacetime. But I doubt that *what may be effectively computed – what follows by "unimaginative" reasoning alone* – is either a spatiotemporal or a formal question. To accept Hogarth's Thesis would be to abandon this doubt.

This is not to say that Hogarth's Thesis is false. Indeed, everything I have said is compatible with the veracity of Hogarth's Thesis. I have simply attempted to establish the following: Hogarth's Thesis cannot be established by considering hyperloops. Hyperloops do threaten the notion of an effective procedure.

References

[1962]Benacerraf, P.: Tasks, Super-Tasks, and the Modern Eleatics. Journal of Philosophy 59, 765–784 (1962)

[forthcoming]Button, T.: SAD Computers and two versions of the Church-Turing Thesis. British Journal for the Philosophy of Science, (forthcoming)

[2001]Davies, E.B.: Building Infinite Machines. British Journal for the Philosophy of Science 52, 671–682 (2001)

[2005]Dummett, M. Hume's Atomism about Events: a response to Ulrich Meyer. Philosophy 80, 141–144 (2005)

[1993]Earman, J., Norton, J.D.: Forever is a Day: Supertasks in Pitowsky and Malament-Hogarth Spacetimes. Philosophy of Science 60, 22–42 (1993)

[2002]Etesi, G., Németi, I.: Non-Turing Computations via Malament-Hogarth spacetimes. International Journal of Theoretical Physics 41, 341–70 (2002)

[2000]Hamkins, J.D., Lewis, A.: Infinite Time Turing Machines. Journal of Symbolic Logic 65, 567–604 (2000)

[1992]Hogarth, M.: Does General Relativity Allow an Observer to View an Eternity in a Finite Time? Foundations of Physics Letters 5, 173–181 (1992)

[1994]Hogarth, M.: Non-Turing Computers and Non-Turing Computability. Proceedings of the Philosophy of Science Association 1994.1, 126–138 (1994)

[2004]Hogarth, M.: Deciding Arithmetic using SAD computers. British Journal for the Philosophy of Science 55, 681–691 (2004)

[2009a]Hogarth, M.: Non-Turing Computers are the New Non-Euclidean Geometries. International Journal of Unconventional Computing 5, 277–291 (2009a)
[2009b]Hogarth, M.: A New Problem for Rule Following. Natural Computing (2009b)
[1976]Lakatos, I.: Proofs and Refutations. Cambridge University Press (1976)
[2006]Németi, I., Dávid, G.: Relativistic Computers and the Turing Barrier. Journal of Applied Mathematics and Computation 178, 118–42 (2006)
[2007]Smith, P.: An Introduction to Gödel's Theorems. Cambridge University Press (2007)
[2006]Shapiro, S.: Computability, Proof, and Open-Texture. In Olszewski, A., Wolenski J., Janusz R. (eds.): Church's Thesis After 70 Years, Ontos Verlag, 420–455 (2006)
[1954]Thomson, J.: Tasks and Supertasks. Analysis 15, 1–10 (1954).
[2008]Welch, P.D.: The Extent of Computation in Malament-Hogarth Spacetimes. British Journal for the Philosophy of Science 59, 659–674 (2008).