Output Concepts for Accelerated Turing Machines

P. H. Potgieter[1] and E. E. Rosinger[2]

[1]University of South Africa (Pretoria)
[2]University of Pretoria

CDMTCS-351
February 2009

Output concepts for accelerated Turing machines

Petrus H Potgieter
Department of Decision Sciences
University of South Africa (Pretoria)
PO Box 392, Unisa, 0003
php@member.ams.org, potgiph@unisa.ac.za

Elemér E Rosinger
Department of Mathematics and Applied Mathematics
University of Pretoria
eerosinger@hotmail.com

2009-02-16

Abstract

The accelerated Turing machine (ATM) is the work-horse of hypercomputation. In certain cases, a machine having run through a countably infinite number of steps is supposed to have decided some interesting question such as the Twin Prime conjecture. One is, however, careful to avoid unnecessary discussion of either the possible actual use by such a machine of an infinite amount of space, or the difficulty (even if only a finite amount of space is used) of defining an outcome for machines acting like Thomson's lamp. It is the authors' impression that insufficient attention has been paid to introducing a clearly defined counterpart for ATMs of the halting/non-halting dichotomy for classical Turing computation. This paper tackles the problem of defining the output, or final message, of a machine which has run for a countably infinite number of steps. Non-standard integers appear quite useful in this regard and we describe several models of computation using filters.

Introduction

In this paper, we shall consider the behaviour of ordinary Turing machines with a single input/working tape and an output tape. A Turing machine (TM) will therefore map each input to a sequence of configurations of the machine, each instantaneous description of the machine consisting of a pair (v, y) where

- v is a description of the machine state, position of the heads and content of the input/working tape; and
- y is a description of the content of the output tape.

A conventional description of a specific machine T therefore gives rise to a mapping

$$T : x \mapsto (v_n^x, y_n^x)_{n=0}^{\infty}$$

subject to the usual rules for operation of a TM, including that v_1^x has the machine in the initial state and x on the input/working tape etc. The values x, y and v will always be natural numbers, of course.

We dispense, without loss of generality, with the designation of a specific set of accepting states of the machine since a single accepting state can always be introduced artificially without changing the input-output behaviour of the machine for conventionally halting computations. Every sequence of instantaneous descriptions

$$(v_n^x, y_n^x)_{n=0}^{\infty}$$

which is eventually constant, corresponds to a classically accepting computation giving output

$$\lim_{n\to\infty} y_n^x$$

on the input x. Such a sequence of instantaneous descriptions is also referred to as a *run* of the machine.

Accelerated Turing machines, or "Zeno machines", represent an attempt to attribute properties to those sequence of instantaneous descriptions that are not eventually constant. In this contribution we try to classify these sequences in a simple and coherent way and also to extend the definitions to non-standard natural numbers. We do not intent to present an extensive survey of the field of hypercomputation, of which there are very many (see Ord, 2006), nor a position on the status vis-à-vis the Church-Turing thesis, of non-finitary computation (see Davis, 2006).

This exposition does not specify any new feature of the machine, but rather proposes a definition of the *output* of non-halting computations. In this sense, it is akin to the infinite time Turing machines introduced by Hamkins and Lewis (2000) and Hamkins and Seabold (2001) although our approach differs – as we shall see – substantially from theirs. Our approach attempts to put the halting of classical Turing machines and the defined output of non-halting machine (a kind of machine in infinite time) within similar frameworks – based on filters and ultrafilters, respectively. This is *hypercomputation* in the same sense that any supertask machine (e.g. the infinite-time machines of Hamkins and Seabold) is. Again, like Hamkins and Seabold, we do not propose a physical mechanism for realising such machines. Obviously any of the speculative models for concretising supertask machines, e.g. based on Relativity Theory (Hogarth, 1992a), could be used to propose a (pseudo-)physical model of our machines.

Hypercomputation after the fact

Suppose that a Turing machine were somehow able to complete an infinite number of steps (in what is, for us, a finite amount of time) and then send a signal to us, containing – for example – the answer to a yes/no question. Such an accelerated Turing machine (ATM) would be able to answer many interesting questions and can be entertained in certain models of relativistic space-time (Hogarth, 1992b). An ATM could easily solve the Goldbach conjecture, that

> every odd number greater than 2 can be written as the sum of two prime numbers.

This ATM would simply run through all the natural numbers and send a signal whenever an exception is found. The operator of the machine would conclude, if no such signal is received, that the Goldbach conjecture holds.

This simple 'Goldbach machine' illustrates some of the difficulties with ATMs. First, the operator vitiates the principle *tertium non datur* by drawing a concrete conclusion from the absence of a signal. This is hard to justify in the context of computation since the two outcomes from running the 'Goldbach machine' cannot (as far as we know) be verified in the same way – the presence of a signal being verifiable in finite time using an ordinary TM, but not the absence of a signal. In conventional computation, it is axiomatic that if a $\{0,1\}$-valued function f is computable, then $1-f$ is computable by essentially the same steps. This correspondence fails for Goldbach-type machines.

The second conceptual difficulty is that confirmation of Goldbach's conjecture, being the absence of a signal from this machine, requires the machine to use an infinite amount of space on the working tape since a Goldbach decomposition has to be found for every integer. This conundrum has been discussed elsewhere and will not be pursued here but it is worth recalling that any TM will eventually

1. halt,
2. go into a loop, or

3. use an infinite amount of space.

Of course, the last two possibilities are not mutually exclusive but this second difficulty is pervasive for ATMs (Calude and Staiger, 2009).

We have implicitly allowed the ATM under discussion in this section to give only one signal to the operator. Depending on the context, we may interpret the presence of the signal as the answer *true* to a question, or as *false*, depending on the exigencies of the problem that the machine is supposed to solve. Indeed Calude et al. (2006) have described how even seemingly more complex problems such as the Collatz conjecture can be solved by a signalling machine but this is non-constructive in the sense that it depends on a problem of which, though easier, we do not know the solution.

The third dilemma contains an echo the first difficulty. Consider the signal given by an ATM of the kind described above. Each ATM of this kind corresponds in a quite simple way to an ordinary TM where the ordinary TM halts if and only if the ATM gives the signal and does not halt otherwise. The solution of a problem by the ATM then corresponds to deciding whether the corresponding, almost identical, TM halts or not. Therefore, the ATM does nothing, in principle, except for querying an oracle for the Halting Problem (HP).

Below, we shall consider how one may attempt to interpret non-finite runs of a Turing machine in a way that goes beyond these signalling machines that query an oracle for HP. This will remind us of some of the difficulties related to supertasks.

Supertasks

Our model will be essentially logical – as it has to be – since we are most concerned about defining the output of the machine for certain inputs for which it does not halt in finite time. We are implicitly assuming that the interesting forms of hypercomputation are related to infinite computational time. However, the difficulties – à la Zeno – do not arise from the identification of logical with computational time but from the problem of associating some finite value with an infinite sequence of events.

Oddities of this kind, appearing in infinite time computation have also been discussed elsewhere, e.g. (Cotogno, 2003; Cohen and Gold, 1977). These difficulties arise from much the same source as the cigars in the folkloric Hilbert Non-Smoking Hotel where all guests are searched for, and relieved of, all tobacco on arrival but where each guest can enjoy cigar in their room – as Guest 1 got his from Guest 2, who got two cigars from Guest 3, who got three cigars from Guest 4 etc.

Perhaps the canonical supertask is Thomson's lamp (Thomson, 1955), a bulb which is switched on at time 0, off at time $\frac{1}{2}$, off again at $\frac{3}{4}$, on at $\frac{7}{8}$ and so on. What is, might one ask, the state of the lamp at time 1? Some have argued that both the on and off states at time $t = 1$ are inconsistent with the operation at times $t < 1$ of the lamp. Our approach (we, as they, disregard the *tertium* of a broken lamp) suggests that *both* the on and off states at time $t = 1$ are consistent. For, suppose it is impossible to observe the lamp at an infinite number of instants, and suppose the last instant $t < 1$ at which an observer looked at the lamp was $t = 1 - \frac{1}{n}$. Without loss of generality, suppose the lamp had been switched on at time $1 - \frac{1}{n}$ and the observer sees the lamp *on* at $t = 1$. This would *not* appear inconsistent as this observer would simply believe that nothing happened for

$$1 - \frac{1}{n} < t < 1$$

whereas an observer seeing the lamp *off* at $t = 1$ would simply believe that the lamp had been switched off one more time, thereby attributing the observed final state of the lamp to one more event than the first observer would.

Ordinary accepting computations by Fréchet filter $\mathcal{F}$

As above, a conventional halting computation on a Turing machine T, on input x, can be fully described by a sequence

$$(v^x_n, y^x_n)^\infty_{n=0}$$

which is eventually constant. It is this concept of *eventually constant* which we shall try to generalise in this paper. The sequence is, of course, fully determined by x and the rules that determine the machine T.

In mathematics, the concept of a *filter* has proved a good model of the notion of *large set.* Recall that a *filter* on a set X is a non-void family of subsets of X, not containing the empty set, and closed with respect to the taking of finite intersections and the forming of supersets. Consider, for example, the Fréchet filter

$$\mathcal{F} = \{L \subseteq \mathbb{N} \mid \mathbb{N} \setminus L \text{ is finite}\}$$

consisting of all co-finite subsets of the natural numbers $\mathbb{N}$, where $0 \in \mathbb{N}$, as customary in computer science.

Definition 1. *A sequence* $(v^x_n, y^x_n)^\infty_{n=0}$ *is an* accepting computation *whenever it is eventually constant, i.e. whenever it is constant on some element of the Fréchet filter* $\mathcal{F}$.

As above, we have discarded the notion of explicitly defined accepting state, without loss of generality. For, a Turing machine with an explicitly defined accepting state obviously satisfies Definition 1 and a Turing machine with an accepting computation by the same definition, can be rewritten with an explicitly defined accepting state. Definition 1 allows us to directly extend the notion of an accepting computation by looking at filters strictly larger than $\mathcal{F}$.

Accepting computations by $\mathcal{G} \supseteq \mathcal{F}$

Suppose, now, $\mathcal{G} \supseteq \mathcal{F}$ is an arbitrary collection of (necessarily) infinite subsets of $\mathbb{N}$, including all the cofinite sets. We attempt to redefine an accepting computation using the elements of $\mathcal{G}$ instead of $\mathcal{F}$ as earlier, so as to – prospectively – enlarge the number of runs of a Turing machine T that could be considered to actually compute something.

Definition 2. *A sequence* $(v^x_n, y^x_n)^\infty_{n=0}$ *is a* $\mathcal{G}$-accepting computation *whenever it is constant on some element of* $\mathcal{G}$.

When $\mathcal{G}$ is equal to the Fréchet filter $\mathcal{F}$, a $\mathcal{G}$-accepting computation is an accepting computation in the usual sense. Normally, it makes sense for $\mathcal{G}$ to be a filter. Closure w.r.t. supersets is an obvious requirement, as is $\mathcal{G} \not\ni \emptyset$. It is desirable that $\mathcal{G}$ be closed under the taking of intersections derives from the common-sense notions that

- the complement of a large set should be considered small and vice versa; and
- the union of two small sets should still be small.

Therefore, if A and B are considered large, then

$$A^C \cup B^C$$

being the union of two small sets, is small, and its complement

$$A \cap B = \left(A^C \cup B^C\right)^C$$

is, of course, large.

Accepting computations by ultrafilter $\mathcal{U} \supseteq \mathcal{F}$

Definition 2 is especially interesting when we consider $\mathcal{U}$-accepting computations, where $\mathcal{U}$ is an ultrafilter. Recall that an *ultrafilter* on X is a filter with the property that for each $A \subseteq X$, either A or its complement belongs to the filter. The Axiom of Choice guarantees the existence of *ultrafilters* $\mathcal{U}$ on $\mathbb{N}$ which contain $\mathcal{F}$. Fix one such $\mathcal{U}$. The subsets belonging to a specific filter are often seen as the *large* sets. The $\mathcal{F}$-large sets are the cofinite subsets of $\mathbb{N}$ and the natural generalisation of the cofinite sets is the collection of $\mathcal{U}$-large sets. To summarise the key definitions so far:

Accepting computation – when there exists an $\mathcal{F}$-large set of points in time where the output tape content as well as the machine state remain constant.

$\mathcal{U}$-accepting computation – when there exists a $\mathcal{U}$-large set of points in time where the output tape content as well as the machine state remain constant.

To see how a Thomson's lamp by Turing machine is avoided in this approach, consider a machine T_{TL} with alphabet $\{0,1\}$ which at time n writes

$$\frac{1+(-1)^n}{2}$$

to the first position of the output tape and has a minimal number of states. The output tape of the machine is now a Thomson's lamp in the usual sense but any run of this machine is a $\mathcal{U}$-accepting computation since, by the properties of an ultrafilter, either the set of odd points in time, or the set of even points, belongs to $\mathcal{U}$ and therefore T_{TL} is, w.r.t. $\mathcal{U}$-accepting computation, equivalent to a machine outputting a constant bit on the tape. That bit is either 1 or 0, depending on the filter $\mathcal{U}$.

Using the ultrafilter therefore allows us to assign, admittedly not very constructively, an output to the machine T_{TL}. We are convinced that in case machines such as the Goldbach conjecture machine are deemed to have a valid output, so should the Thomson lamp machine. It is not extraordinarily liberal to consider a run of T_{TL} to be an accepting computation since this machine simply oscillates between two global states. However, as the proposition below demonstrates, the notion of $\mathcal{G}$-accepting computations does not, unfortunately, go much beyond machines of the type T_{TL}.

Proposition 1. *Given a filter $\mathcal{G} \supset \mathcal{F}$, every $\mathcal{G}$-accepting computation is either*

(i) an $\mathcal{F}$-accepting computation, i.e. a usual accepting computation; or

(ii) a computation that ends in a finite cycle of global states of the machine, in the fashion of Thomson's lamp.

Proof. Suppose that (i) does not hold, i.e. that the machine in question has a run

$$(v_n^x, y_n^x)_{n=0}^{\infty}$$

which is constant on some

$$A \in \mathcal{G} \setminus \mathcal{F}.$$

A is infinite since if it were not, we would have $A^c \in \mathcal{F} \subset \mathcal{G}$ by definition and hence

$$\emptyset = A \cap A^c \in \mathcal{G}$$

which would contradict the assumption that $\mathcal{G}$ is a filter. Since A is infinite, it has at least two distinct members, $m_1, m_2 \in A$ with $m_1 \neq m_2$ and since

$$\left(v_{m_1}^x, y_{m_1}^x\right) = \left(v_{m_2}^x, y_{m_2}^x\right)$$

we have

$$\left(v_{m_1+1}^x, y_{m_1+1}^x\right) = \left(v_{m_2+1}^x, y_{m_2+1}^x\right)$$

etc. which proves the statement. □

This remark shows that the brute-force Goldbach conjecture machine will not amoun to a $\mathcal{U}$-accepting computation and that this notion is therefore of use only in dealing with the logical difficulties arising from simpler devices. In the next section we shall therefore abandon stability of the sequence $(y_n^x)_{n=0}^{\infty}$ as a goal.

Contrast this interpretation with the treatment of T_{TL} in the infinite machine framework of Hamkins e.a. (Hamkins and Lewis, 2000). In their framework, the state of the machine is defined at the first limit ordinal ω and – if $\limsup$ has been chosen over $\liminf$ as the limiting operator for each cell of the tape – the content of the tape 'at' the ordinal ω is just $+1$. In our approach, we do not know whether a run of T_{TL}, being a $\mathcal{U}$-accepting computation, will 'output' 1 or 0. For each choice of $\mathcal{U}$ it will be one of the two but we do not know which because the ultrafilter $\mathcal{U}$ is not constructively given.

Consider a slightly accelerated T_{ATL} which at time n writes

$$\frac{1+(-1)^{n+1}}{2}$$

to the first position of the output tape and has a minimal number of states. This machine is the 'evil twin' of the Turing lamp machine T_{TL}, always have the exact opposite on the tape if the machines are run concurrently. Clearly the 'output' of T_{ATL} is either 1 or 0, depending on the choice of $\mathcal{U}$.

Furthermore, considered as $\mathcal{U}$-accepting computations, the machines T_{TL} and T_{ATL} actually do have opposite outputs. In the infinite-time machines of Hamkins and Seabold, however, T_{TL} and T_{ATL} are in the same state at ordinal ω. The present authors find it perhaps more intuitive that T_{TL} and T_{ATL} compute different outputs. For a start, the two machines – if started at the same time – are always in opposite states. On the other hand, if they are started one unit of time apart then they would always seem to be in the *same* state. However, it could very well be that the question – whether T_{TL} and T_{ATL} compute the same or opposite thing, or neither – is akin to speculation about the virginity of Hamlet.[1]

An ultrafilter-based acceptance notion using the output tape

The proposition in the previous section shows that an acceptance notion, based on stability on a large set in time, for interesting machines needs to ignore part of the machine.[2] Perhaps the most usual notion of this kind is the classical notion of a limit-computable function, which we rephrase here in terms of the Fréchet filter.

Definition 3. *A function f is* limit computable *provided there exists a Turing machine T such that for each x, the run*

$$(v_n^x, y_n^x)_{n=0}^{\infty}$$

of T on x has $(y_n^x)_{n=0}^{\infty}$ constant on a set $F \in \mathcal{F}$. That is, $f(x)$ is the limit of the content of the output tape, if T is run on input x. Here one does not require any kind of stability of $(v_n^x)_{n=0}^{\infty}$.

Clearly the brute-force approach to deciding the Goldbach conjecture is limit computable, as is every set which is Turing-reducible to HP (Soare, 2007). An interesting notion of hypercomputation might therefore require a slightly different notion of accepting computation and we propose using the ultrafilter $\mathcal{U} \supset F$ of the previous section.

Definition 4. *A given run of a Turing machine T is called a $\mathcal{U}$-*limiting computation *whenever the output tape is stable for a subset of time which belongs to $\mathcal{U}$.*

Naturally, the $\mathcal{F}$-limiting computations correspond the ordinary limit computation. Furthermore, every run of the Thomson's lamp machine is a $\mathcal{U}$-limiting computation which is not an $\mathcal{F}$-limiting computation. One might, if one likes, consider the content of the output tape at those points in time belonging to filter element $A \in \mathcal{U}$ on which the tape is stable, as the output of a specific computation. Since the intersection of the sets in the filter cannot be empty, this notion of output is well-defined. In the next section, we shall see how this can be identified with the finite non-standard generalised outputs. Note that there are TM runs that are not $\mathcal{U}$-limiting computations, e.g. that of the machine that fills the tape in the simplest possible way.

Let us examine one further toy example. Consider a Turing machine T_{d} which operates with the alphabet $\{0, 1\}$ and as follows.

```
write "0" on the tape up to position 98;
n = 0;
while 1 > 0 do
     write "1" on position 99;
     go back and write "0" on position 99;
     write "0" in the 2^n positions to the right of 99;
     move back to position 99;
     n = n + 1;
end while;
```

Clearly the machine has neither an accepting computation, nor a $\mathcal{U}$-accepting computation (as defined in the previous sections) since the output tape is in any specific state only for a finite time. This machine also does not describe any limit-computable function because there is no $\mathcal{F}$-large set in time for which position 99 is unchanged. However, it is conceivable that – for the right choice of $\mathcal{U}$ – it represents a $\mathcal{U}$-limiting computation. If one only considers the output tape, then this is not so unreasonable – the tape is actually mainly filled with 0, with the exception of a very occasional appearance of 1 in position 99. If one assumes that the tape is originally filled by the symbol 0 then the output tape of T_{d} (apart from the movement of the head) behaves exactly like the output tape of the machine executing the following actions.

```
write "0" on the tape up to position 98;
n = 0;
while 1 > 0 do
     write "+1" on position 99;
     go back and write "0" on position 99;
     wait for as long as it take to write "0" in the 2^n
       positions to the right of 99;
     n = n + 1;
end while;
```

Whether this latter machine would be thought to output a tape filled with zeroes or a tape with a single 1 on it, would depend on the ultrafilter. This is an ambiguity which, we find, reflects the true nature of the situation.

Now, in the ordinal computation of Hamkins and Seabold where one simply takes the lim sup of the tape, the machine T_{d} has the same state at ω as a device executing the following program.

```
write "0" on the tape up to position 98;
write "+1" on position 99;
```

This seems mildly counter-intuitive to us, just as the Hamkins-Seabold treatment of the Thomson lamp machine does.

Turing machines with non-standard output

In this final section, we give an obvious example of how an output can be defined for any run of a Turing machine. The motivation is to handle the output – even for non-halting, in the classical sense – for Turing-type machines in a completely unified framework. The basic idea is elementary: to simply define the sequence of output tape content, at each discrete moment in time, as the *output* of the machine.

Given a run

$$(v_n^x, y_n^x)_{n=0}^{\infty}$$

of a Turing machine with input x, we shall therefore consider the output of the machine associated with the input x to be an equivalence class of the sequence

$$(y_n^x)_{n=0}^{\infty}$$

of contents of the output tape. In order to faithfully preserve the ordinary notion of accepting computation, the equivalence class of a sequence with finite limit should contain all other sequences with the same limit. If $\mathcal{U} \supset F$ is the ultrafilter discussed earlier, we proceed to use the equivalence classes used to define the non-standard natural numbers in Robinson's development of the non-standard analysis.

Definition 5. *For each sequence of natural numbers (a_n) we set*

$$(a_n)_{\mathcal{U}} = \{(b_n) | \{m | a_m = b_m\} \in \mathcal{U}\}$$

which is the equivalence class of all sequences that agree with (a_n) on a $\mathcal{U}$-large set of indices.

We write k^* for the equivalence class of the sequence which has only one value, k. The following observation results trivially from the preceding discussion. If $(a_n)_{\mathcal{U}} = k^*$ for some $k \in \mathbb{N}$ then $(a_n)_{\mathcal{U}}$ is said to be *finite.*

Definition 6. *Given a run $(v_n^x, y_n^x)_{n=0}^{\infty}$ of some Turing machine, we define*

1. *the* non-standard output *of the machine, corresponding to input x to be the equivalence class $(y_n^x)_{\mathcal{U}}$; and*
2. *the* non-standard terminal internal configuration *of the machine, corresponding to input x, to be $(v_n^x)_{\mathcal{U}}$.*

This allows the following straight-forward observation.

Proposition 2. *Any classically accepting run of a Turing machine has finite output and has a finite terminal state.*

The converse is, of course, not true – the Turing lamp machine T_{TL} being a simple counter-example. In terms of Abraham Robinson's non-standard analysis (Robinson, 1996) the output and final states are now non-standard natural numbers. Furthermore, we can define the accepting computations in a coherent way also for non-standard inputs, i.e. inputs of the form $(a_i)_{\mathcal{U}}$.

Definition 7. *Let a Turing machine T be given as well as a non-standard input $(a_i)_{\mathcal{U}}$. We define a* run *of the machine on this input by considering the classical sequence of instantaneous descriptions, or runs $(v_n^{a_i}, y_n^{a_i})_{n=0}^{\infty}$, and letting the new, non-standard run, be*

$$((v_n^{a_n})_{\mathcal{U}}, (y_n^{a_n})_{\mathcal{U}}) .$$

Now, $(v_n^{a_n})_{\mathcal{U}}$ will be the terminal internal configuration of the machine and $(y_n^{a_n})_{\mathcal{U}}$ will be the output of the computation, which is always defined but not necessarily finite.

Note that the terminal internal configuration contains information about the working tape and the machine state as well as positions of the heads. It follows directly from the definition that if

$$(a_n)_{\mathcal{U}} = (b_n)_{\mathcal{U}}$$

then

$$(v_n^{a_n})_{\mathcal{U}} = (v_n^{b_n})$$

and

$$(y_n^{a_n})_{\mathcal{U}} = (y_n^{b_n})_{\mathcal{U}}$$

so the non-standard output and final internal configuration are well-defined.

One can easily see that within this framework the halting problem for ordinary Turing machines can be solved by a machine that outputs $(1)_{\mathcal{U}}$ if the machine halts, and $(0)_{\mathcal{U}}$ otherwise. Let us call ordinary Turing machine descriptions, equipped with the scheme of operation described above, non-standard Turing machines (NSTMs or NST machines). It is clear, of course, what the concatenation of two NST machines would compute as the output of an NST is *always* a valid input for another NSTM. However, exactly how this concatenation would be implemented on an NSTM – and whether this would be possible at all – is not clear. It is possible, for example, that certain additional conditions on the ultra-filter $\mathcal{U}$ will be required or that this definition will not allow concatenation at all. Concatenations is, of course, an important topic in hypercomputation (Cotogno, 2003).

The non-standard approach to computability has been investigated before, i.a. by Richter and Szabo (1988) and Andréka et al. (1982a,b). It is absolutely conceivable that an approach via functions (which would largely eliminate the time-dynamics of the output tape) is more sensible.

Conclusion

The paper has explored how the filter and ultrafilter concepts can be used to characterise the behaviour of certain non-classical computation schemes based on Turing machines. A fully non-standard scheme w.r.t. the input, output and run length – in which, however, the machine still has a classically finite description – is proposed as one way to overcome the problem of defining the output or final global state of the machine. The authors regard this as a tentative proposal with some promise – at least for extending the vocabulary of hypercomputation by accelerated Turing machines.

References

Andréka, H., Németi, I., and Sain, I. (1982a). A complete logic for reasoning about programs via nonstandard model theory I. *Theoretical Computer Science*, 17(2):193–212.

Andréka, H., Németi, I., and Sain, I. (1982b). A complete logic for reasoning about programs via nonstandard model theory II. *Theoretical Computer Science*, 17(3):259–278.

Calude, C. S., Calude, E., and Dinneen, M. J. (2006). A new measure of the difficulty of problems. *Journal of Multiple-Valued Logic and Soft Computing*, 12(3-4):285–307.

Calude, C. S. and Staiger, L. (2009). A Note on Accelerated Turing Machines. Technical Report CDMTCS-350, Centre for Discrete Mathematics and Theoretical Computer Science.

Cohen, R. S. and Gold, A. Y. (1977). Theory of ω-languages. I. characterizations of ω-context-free languages. *Journal of Computer and System Sciences*, 15:169–184.

Cotogno, P. (2003). Hypercomputation and the physical Church-Turing thesis. *The British Journal for the Philosophy of Science*, 54(2):181–223.

Davis, M. (2006). Why there is no such discipline as hypercomputation. *Appl. Math. Comput.*, 178(1):4–7.

Hamkins, J. D. and Lewis, A. (2000). Infinite time Turing machines. *The Journal of Symbolic Logic*, 65:567–604.

Hamkins, J. D. and Seabold, D. E. (2001). Infinite time Turing machines with only one tape. *MLQ. Mathematical Logic Quarterly*, 47:271–287.

Hogarth, M. (1992a). Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, 5(2):173–181.

Hogarth, M. L. (1992b). Does general relativity allow an observer to view an eternity in a finite time? *Found. Phys. Lett.*, 5(2):173–181.

Ord, T. (2006). The many forms of hypercomputation. *Appl. Math. Comput.*, 178(1):143–153.

Richter, M. M. and Szabo, M. E. (1988). Nonstandard methods in combinatorics and theoretical computer science. *Polish Academy of Sciences. Institute of Philosophy and Sociology. Studia Logica. An International Journal for Symbolic Logic*, 47:181–191.

Robinson, A. (1996). *Non-standard analysis.* Princeton Landmarks in Mathematics and Physics. Princeton University Press, Princeton, N.J.

Soare, R. (2007). *Computability and Incomputability*, pages 705–715.

Thomson, J. (1954–1955). Tasks and Super-Tasks. *Analysis*, 15:1–13.

Notes

[1]Simile used by Martin Davis in the Notices of the AMS, May 2008, with regard to the continuum hypothesis.

[2]This point is also made clearly by Calude and Staiger (2009) who discuss how any interesting accelerated machine will use an infinite amount of tape.