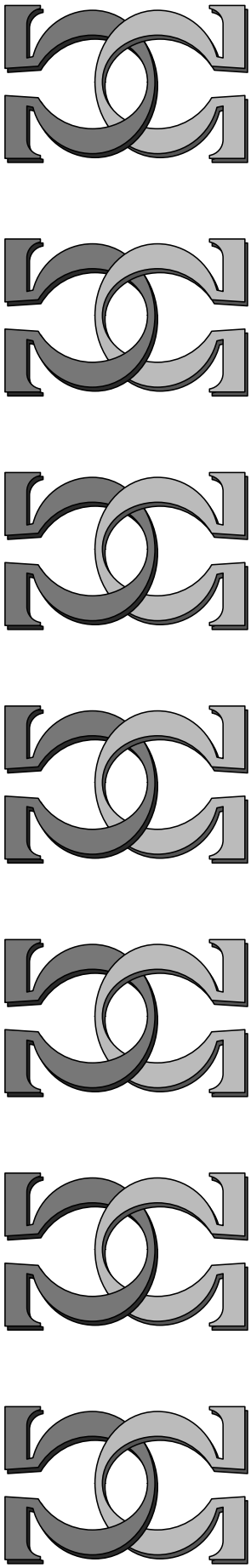

CDMTCS
Research
Report
Series

A Note on Accelerated Turing Machines

C. S. Calude[1] and L. Staiger[2]
[1]University of Auckland, NZ
[2]Martin-Luther-Universität, Germany

CDMTCS-350
February 2009

Centre for Discrete Mathematics and
Theoretical Computer Science

A NOTE ON ACCELERATED TURING MACHINES

CRISTIAN S. CALUDE, LUDWIG STAIGER

Abstract. In this note we prove that any Turing machine which uses only a finite computational space for every input cannot solve an uncomputable problem even in case it runs in accelerated mode.

1. Accelerated Turing machines

An accelerated Turing machine (sometimes called Zeno machine) is a Turing machine that takes 2^{-n} units of time (say seconds) to perform its nth step; we assume that steps are in some sense identical except for the time taken for their execution. Such a machine can run an infinite number of steps in one unit of time. Accelerated Turing machines have been discovered by Weyl [15] in 1927 and studied by various authors including Boolos and Jeffrey [1], Calude and Păun [3], Copeland [2], Ord [7], Potgieter [8], Shagrir [9, 10], Stewart [12], Svozil [13]; accelerated computation appears in various contexts, for example in relativistic computation, see Hogarth [6] and Etesi and Németi [4].

The main feature of an accelerated Turing machine consists in its capability of computing in a finite time an infinite sequence of steps, thus allowing it to solve uncomputable problems. For example, the following (informal) accelerated Turing machine can solve the halting problem of an arbitrarily given Turing machine T and input w in finite time:

```
begin program
write 0 on the first position of the output tape;
set i = 1;
begin loop
simulate the first i steps of T on w;
if T(w) has halted, then write 1 on the first position
of the output tape;
i = i + 1;
end loop
end program
```

Calude: Department of Computer Science, The University of Auckland, Private Bag 92019, Auckland, New Zealand, cristian@cs.auckland.ac.nz.

Staiger: Martin-Luther-Universität Halle-Wittenberg, Institut für Informatik, D - 06099 Halle, Germany, staiger@informatik.uni-halle.de.

Work done during L. Staiger's visit to CDMTCS in January 2009.

By inspecting the first position of the output tape we need one unit of time to run the above machine in order to decide whether $T(w)$ stops or not. Alternatively, we can clock the computation and see whether the machine itself has halted before one unit of time or not. Note that Svozil [13] proved that the halting problem of accelerated Turing machines is not decidable by any accelerated Turing machine.

Are accelerated Turing machines physically possible? This is a challenging problem discussed by various authors. We contribute with a small result to this discussion by examining the computational space required by an accelerated Turing machine running an infinite computation: *is it finite or not?* This question was posed to the first author by Fearnley [5].

2. Is the space used by an accelerated Turing Machine always finite?

Let us start with the following informal example:

```
set i=0;
begin loop
i=i+1;
end loop
```

It is clear that the accelerated Turing machine executing the above set of instructions needs an infinite computational space. Is this just an accident or do we have a more general situation?

Before being tempted by a hasty answer let us note that the following set of instructions is infinite, but requires only a finite amount of space:

```
set i=1;
while (i > 0) do
       i=1;
end while
```

To be able to answer the above question we will introduce a formal model of Turing machine and we state a few general facts. We assume that reader is familiar with the basics of Turing computability, e.g. [11, 14].

Let $(X, \Gamma, S, s_0, \Box, \delta)$ be a Turing machine in which X is the input alphabet, $\Gamma \supset X$ is the work tape alphabet, S is the set of states, s_0 is the initial state, $\Box \in \Gamma \setminus X$ is the blank symbol[1], and δ is the (partial) transition function. We assume that the Turing machine has one tape where initially the input is written on, and the machine starts its processing in state s_0 by scanning the first symbol of the input word.

A configuration of the Turing machine is a word $k \in \Gamma^* S \Gamma^*$, where the first and last symbols are different from the blank symbol $\Box$. Here states (in S) are considered also to be symbols, so for uniqueness we assume that $S \cap \Gamma = \emptyset$. A configuration $w_1 s w_2$, with $w_1, w_2 \in \Gamma^*, s \in S$ describes the machine with $w_1 w_2$ on the tape, in state s with the head

[1]We explicitly exclude the blank symbol from the input alphabet.

scanning the first symbol of w_2. If w_2 is the empty word then the right part of the (infinite) tape is assumed to contain only finitely many blank symbols. If w_1 is the empty word then the infinite tape left to the cell scanned by the machine is assumed to contain only finitely many blank symbols.

The successor configuration of a configuration $k = w_1 s w_2$ is obtained by a simple local re-writing process derived only from δ, s and its left and right neighbour letters in k.

Let $M = (X, \Gamma, S, s_0, \square, \delta)$ be a Turing machine and x an input word. We define the computational space used by M on x, $space_M(x)$, to be the number—finite or infinite—of cells used by M on x. The function $time_M(x)$ denotes the number of steps executed by M on input x (see [14]). By $M(x) < \infty$ we denote the fact that M stops on x.

Clearly, $space_M(x) < \infty$ whenever $M(x) < \infty$, and $M(x) = \infty$ iff $time_M(x) = \infty$. The halting problem for M is the problem of deciding given x whether $M(x) < \infty$. It is well known that for most Turing machines M, the halting problem for M is undecidable.

A simple counting[2] argument justifies the following (see [14]):

Lemma 1. *If $M(x) < \infty$, then*

$$time_M(x) \leq |\Gamma|^{space_M(x)} \cdot |S| \cdot space_M(x).$$

Assume that for a specific input x we know that $space_M(x) < \infty$ (the halting problem for M may be undecidable): Can we decide in finite time whether $M(x)$ halts or not?

Theorem 2. *For every Turing machine M, the set $\{x \in X^* : M(x) < \infty\}$ is decidable relative to the (oracle) set $\{\in X^* : space_M(x) < \infty\}$.*

Proof. Consider the Turing machine M, input x and let $space_M(x) = c_x < \infty$. If M does not stop on x then the list of all configurations is infinite, but there are at most $(|\Gamma| + |S|)^{c_x+1}$ different configurations of length $c_x + 1$. Thus one configuration appears twice in that list. Otherwise, if M stops on x then no repetition can occur.

Having in mind this observation one constructs an observer Turing machine O that lists all configurations of M generated by the computation $M(x)$ and continues as follows:

(1) If M stops on x then O stops too and declares $M(x) < \infty$.
(2) If M does not stop on x then on the first repetition in the list of configurations generated by $M(x)$ the machine O stops and declares that $M(x) = \infty$.

□

Corollary 3. *If for every x, $space_M(x) < \infty$, then the halting problem for M is decidable.*

Corollary 4. *If the halting problem for M is undecidable then $\{x \in X^* : space_M(x) = \infty\} \neq \emptyset$.*

[2] $|Y|$ denotes the cardinality of the set Y.

Corollary 5. *The set* $\{(M,x) : M \text{ Turing machine}, x \in X^*, space_M(x) < \infty\}$ *is computably enumerable but not computable.*

A Turing machine M running in 'accelerated mode' is denoted by A_M. In other words, M and A_M have the same description, but M runs in normal mode, i.e. each instruction is executed in a fixed unit of time, while A_M runs in an accelerated mode. Observe that $M(x) = \infty$ iff $time_M(x) = \infty$ iff $time_{A_M}(x) = 1$.

There is a similarity between computational time and space; however, this parallel is not perfect. For example, it is not true that an accelerated Turing machine which uses unbounded space has to use an infinite space for some input (as it seems to be claimed in Ord [7, p. 24]). The reason is that every reasonable computable problem requires at least the input data x to be scanned, so it needs at least a space greater than the length of x.

Let $\chi_M : X^* \to \{0,1\}$ be the function defined by

$$\chi_M(x) = \begin{cases} 1, & \text{if } M(x) < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

This function can always be computed by an accelerated Turing machine $A_{M'}$ in finite time.[3] If the computational space is finite for every input, then acceleration does not add computational power:

Corollary 6. *Let* A_M *be an accelerated Turing machine with* $space_{A_M}(x) < \infty$ *for all inputs* x*. Then the function* χ_M *is Turing computable. Again, the Turing machine computing* χ_M *is not necessarily* M*.*

Acknowledgment

We thank L. Fearnley for posing the problem discussed in this note and P. Potgieter for illuminating discussions.

References

[1] G. Boolos, R. C. Jeffrey. *Computability and Logic*, Cambridge University Press, Cambridge, 1980.
[2] B. Copeland. Accelerating Turing machines, *Minds and Machines* 12 (2) (2002), 281–300.
[3] C. S. Calude, G. Păun. Bio-steps beyond Turing, *Biosystems* 77 (1–3) (2004), 175–194.
[4] G. Etesi, I. Németi. Non-Turing computations via Malament-Hogarth space-times, *International Journal of Theoretical Physics* 41 (2002), 341–370.
[5] L. Fearnley. Email to (and discussions with) C. Calude, 3 December 2008.
[6] M. Hogarth. Does general relativity allow an observer to view eternity in a finite time? *Foundations of Physics Letters* 5 (1992), 173–181.
[7] T. Ord. *Hypercomputation: Computing More than the Turing Machine*, Honours Thesis, Computer Science Department, University of Melbourne, Australia, 2002; `arxiv.org/ftp/math/papers/0209/0209332.pdf`.
[8] P. H. Potgieter. Zeno machines and hypercomputation, *Theoretical Computer Science* 358 (2006), 23–33. `arXiv:cs.CC/0412022`

[3] $A_{M'}$ is not necessarily equal to A_M.

[9] O. Shagrir. Accelerating Turing machines, in Friedrich Stadler and Michael Stltzner (eds.). *Time and History (Papers of the 28th International Wittgenstein Symposium)*, Austrian Ludwig Wittgenstein Society, 2005, pp. 276–278.
[10] O. Shagrir. Super-tasks, accelerating Turing machines and uncomputability, *Theoretical Computer Science*, 317 (2004), 105–114.
[11] M. Sipser. *Introduction to the Theory of Computation*, PWS, Boston, 2006, second edition.
[12] I. Stewart. Deciding the undecidable, *Nature* 352 (1991), 664–665.
[13] K. Svozil. The Church-Turing thesis as a guiding principle for physics, in: C. Calude, J. Casti, M. Dinneen (eds.). *Unconventional Models of Computation*, Springer, 1998, Berlin, pp. 371–385.
[14] K. Wagner, G. Wechsung. *Computational Complexity*, Deutscher Verlag der Wissenschaften, Berlin, 1986.
[15] H. Weyl. *Philosophy of Mathematics and Natural Science*, Princeton University Press, Princeton, 1949.