Annals of Pure and Applied Logic 165 (2014) 1501–1511

Contents lists available at ScienceDirect

Annals of Pure and Applied Logic

www.elsevier.com/locate/apal

The computational strengths of α-tape infinite time Turing machines

Benjamin Rin

University of California, Irvine, Logic and Philosophy of Science, Irvine, CA, USA

ARTICLE INFO

Article history:
Available online 15 May 2014

MSC:
03D10
03D60
68Q05
03D78

Keywords:
Infinite time Turing machine
Ordinal computability
Supertask
Transfinite computation

ABSTRACT

In [7], open questions are raised regarding the computational strengths of so-called ∞-α-Turing machines, a family of models of computation resembling the infinite-time Turing machine (ITTM) model of [2], except with α-length tape (for $\alpha \geq \omega$). Let T_α denote the machine model of tape length α (so T_ω is just the ITTM model). Define that T_α is *computationally stronger* than T_β (abbreviated $T_\alpha \succ T_\beta$) precisely when T_α can compute all T_β-computable functions $f\colon {}^{\min(\alpha,\beta)}2 \to {}^{\min(\alpha,\beta)}2$, plus more. The following results are found: (1) $T_{\omega_1} \succ T_\omega$. (2) There are countable ordinals α such that $T_\alpha \succ T_\omega$, the smallest of which is precisely γ, the supremum of ordinals clockable by T_ω. In fact, there is a hierarchy of countable T_αs of increasing strength corresponding to the transfinite (weak) Turing-jump operator $^\nabla$. (3) There is a countable ordinal μ such that neither $T_\mu \succeq T_{\omega_1}$ nor $T_\mu \preceq T_{\omega_1}$—that is, the models T_μ and T_{ω_1} are computation-strength incommensurable (and the same holds if countable $\mu' > \mu$ replaces μ). A similar fact holds for any larger uncountable device replacing T_{ω_1}. (4) Further observations are made about countable T_α.

1. Background

Over the past few years, interest has grown in the study of models of transfinite computation—theoretical machines that extend classical computibility theory into an infinitary context. A number of distinct models have been devised, typically resembling one of the classical machines (Turing machines, register machines, etc.) but with some modification that takes the operations into the transfinite. For a detailed survey, see [14] or [15]. Of the machines so defined during this recent wave, particular attention has been given to the infinite time Turing machines of [2], which were the first to see print.

Infinite time Turing machines, or ITTMs, are a set of theoretical computing machines similar to classical Turing machines, with the exception that halting computations are not assumed to run for only finitely many steps. Instead, computations are permitted to run for transfinitely many steps before halting. Whereas a classical finitary Turing machine is considered to "fail" in some sense if it does not successfully halt within

E-mail address: brin@uci.edu.

http://dx.doi.org/10.1016/j.apal.2014.04.016
0168-0072/

a finite number of steps, an ITTM may compute through steps $1, 2, \ldots, \omega, (\omega+1), (\omega+2), \ldots, (\omega+\omega), \ldots$ and halt after, say, $\omega^4 + \omega + 17$ steps. It can even halt after a non-recursive ordinal number of steps.

In [2], the machines are conceived as having three tapes: an *input tape*, an *output tape*, and a *scratch tape* intended for calculations. Each machine has a read–write *head* that at any given time occupies the nth cell C_n of each of these tapes simultaneously (the leftmost cell is C_0, and the tapes have ω-many cells extending infinitely to the right). In the present work, we instead imagine just a single tape for input and output, with a parallel tape for scratch work. This type of machine is computationally equivalent to the three-tape machines.[1]

It will be convenient to use the fact from [3, Corollary 2.4] that it is possible to use the single scratch tape of a machine to simulate having a machine with multiple parallel tapes. As described in [6], there are "canonical but tedious" translations of programs on machines with n tapes to programs on machines with one tape. We also recall from [6] that one can use finite binary strings as codes for 'symbols', and thus at times may treat the tape as containing a sequence of arbitrary symbols from an alphabet rather than binary bits.

Since an ITTM may run for transfinitely many steps, it has time to read and process infinite-length input, and to write infinite-length output. Hence ITTMs can be understood as computing (partial) functions on the reals ${}^{\omega}2$. (We can still represent a natural number k by a string consisting of all 0s except C_k.) The functions computable by an ITTM form a strict superset of the Turing-computable functions.

To define the machine's behavior fully, [2] stipulates that on successor ordinal steps, cell values are calculated identically to the way they are determined in a Turing machine, but at limit ordinal steps, cell values take on the *lim sup* of their values at previous stages.[2] In the present paper, as in [6] and elsewhere, the *lim inf* is used instead. The difference is immaterial as the resulting models of computation are equivalent. We also set the machine's *head position* and *program state* at the limit stage to their respective *lim inf*s from prior stages.[3]

For ITTMs there are both computable and noncomputable functions $f\colon {}^{\omega}2 \to {}^{\omega}2$. Any recursive function is computable, as is the classical halting problem. An ITTM can solve the latter by running any Turing machine computation in parallel with some program known to halt after ω steps, checking whether the simulated Turing computation has halted or not by that stage. The model's computational power goes far beyond this, however. It is capable of deciding membership in any set of reals up to complexity Π^1_1—in particular, it is able to determine whether a give real $a \in {}^{\omega}2$ codes[4] a well-ordered relation on $\mathbb{N}$ (see [2]). However, all ITTM-decidable sets of reals (indeed, all ITTM semi-decidable sets of reals) fall below complexity Δ^1_2. For an example of non-computable functions, there are ITTM analogs of the classical halting problem, defined below.

While it is clear that the ITTM model is strictly stronger than the finite Turing model, it is natural to ask whether it can be made even stronger. In [2], there is some analysis of extensions of the model using oracles. In [6], the model is extended by lengthening the tape from having ω-many cells to having a proper class of cells (indexed by ordinals $\alpha \in \mathbf{Ord}$). In [1] and [8], machines with fixed α-length tape are considered, with α assumed to be admissible. For these, the assumption is also made that computation duration is limited to be below α-many steps. In analogy with Turing computation, which can be thought of as running on ω-length tapes for some duration below ω-many steps, these machines run with α-length

[1] Indeed, by [3], it is sufficient to have just *one cell* for scratch work accompanying the input/output tape to guarantee the same capabilities as the three-tape ITTM. Without this, a machine can compute most, though curiously not all, of the functions computable by three-tape machines.

[2] That is, the value of cell n at limit ordinal step α is 0 (respectively, 1) if it had stabilized at value 0 (respectively, 1) by some step $\beta < \alpha$ (in other words, it had that value at step β and did not change after), but if the value of cell n did not stabilize (i.e., it changed its value infinitely often cofinally before step α), then at the limit α it defaults to value 1. If [2] had used *lim inf* instead (as we do here), then it would behave the same except default instead to value 0.

[3] Rather than these stipulations, in [2] there is a special program state designated for limit steps called the *limit state*, and the head position is automatically set to C_0 at limit steps, but it again makes no material difference.

[4] Every real r is considered to *code* the relation $\lhd$ on the set $\mathbb{N}$ given by: $a \lhd b$ iff the $\langle a, b\rangle$th digit of r is 1, where $\langle \cdot, \cdot \rangle$ refers to the Gödel pairing function.