Technical Report 2010-001
Sublinear Space Real-Time Turing Machines Cannot Count[*]

Stefan D. Bruda

Department of Computer Science, Bishop's University
Sherbrooke, Quebec J1M 1Z7, Canada
email: stefan@bruda.ca
6 August 2010

Abstract

We show that sublinear-space bounded real-time Turing machines (be they deterministic or nondeterministic) are equivalent to finite automata. Non-regular real-time definable languages (and also quasi-real-time languages, their nondeterministic extension) can thus only be accepted with $\Omega(n)$ work space.

1 Introduction

The oldest model of real time is probably the *real-time Turing machine* [14]. These machines (and their languages, called real-time definable) have been studied extensively [1, 5, 7, 9, 10, 11]. The real-time Turing machine is deterministic, though nondeterministic extensions were also studied [2, 6].

Questions about multiple tapes, multiple heads, and other extensions were thus addressed. By contrast, little attention has been given to the space bounds of real-time Turing machines. We address this issue partially in this paper, by finding a significant gap in the space hierarchy of real-time Turing machines: We show that any language that is accepted by such a machine using sublinear space is equally acceptable by a finite automaton, and that nondeterminism does not influence this gap.

We believe that this gap shows the limitation of the Turing machine model in the area of real time.

2 Preliminaries

Given some alphabet A, the set A^k is defined recursively by $A^1 = A$, and $A^i = A \times A^{i-1}$ for $i > 1$. The empty word is denoted by ϵ. We assume familiarity with Turing machines; the following is only a sketch definition given for completeness (and to summarize notation).

For some constant k, $k \geq 1$, a (k-tape) *deterministic Turing machine* is a tuple $M = (K, \Sigma, \delta, s_0)$, where K is the set of states, not containing the halt state h, $s_0 \in K$ is the initial state, Σ is the tape alphabet containing the blank symbol #, and δ is the transition function, $\delta : (K \times \Sigma^k) \longrightarrow (K \cup \{h\}) \times (\{R, L, N\}^k \times \Sigma^k)$; R, L, N govern the head move, with L standing for "left," R for "right," and N for "no move." A configuration of M is a tuple $(q, x_1\underline{a_1}y_1, \ldots, x_k\underline{a_k}y_k)$, where q is the current state, $x_i a_i y_i$ is the content of the i-th tape, and a_i is the symbol that is currently scanned by the head of tape i, $1 \leq i \leq k$. A configuration C_1 yields in one step another configuration C_2, written $C_1 \vdash_M C_2$ iff C_2 can be obtained from C_1 in one application of δ. As usual,

1

$\vdash_M^*$ is the transitive and reflexive closure of $\vdash_M$ (with the subscript often omitted when there is no ambiguity).

A nondeterministic Turing machine is identical to a deterministic Turing machine, except that $\delta \subseteq (K \times \Sigma^k) \times ((K \cup \{h\}) \times (\{R, L, N\}^k \times \Sigma^k))$ (so that a configuration can nondeterministically yield in one step more than one configuration).

A Turing machine M is said to accept some input w iff M stops (that is, M reaches the halt state h) on w; that is, M accepts w iff $(s_0, w, \epsilon, \dots, \epsilon) \vdash_M^* (h, w_1, \dots, w_k)$ for some $w_i \in \Sigma^*$, $1 \leq i \leq n$. M accepts a language L whenever M accepts w iff $w \in L$.

A Turing machine is f-space bounded [8] iff the first tape is read-only and for any configuration $C = (q, w_1, w_2 \dots, w_k)$ such that $(s_0, w, \epsilon, \dots, \epsilon) \vdash^* C$ for some word $w \in \Sigma^*$, we have $|w_i| \leq f(|w|)$ for every $2 \leq i \leq k$. In other words, the size of any possible configuration of a machine working on w is bounded by $f(|w|)$ (save for the first tape which however becomes read-only; this allows the definition of sublinear space bounds).

We use one [10] of the several equivalent definitions of real-time Turing machines:

For some constant k, $k \geq 1$, an *on-line Turing machine* is a deterministic $(k + 1)$-tape Turing machine (with k working tapes and one input tape) $M = (K_p \uplus K_a, \Sigma, \delta, s_0)$, where $\delta : (K_p \times \Sigma \times \Sigma^k) \cup (K_a \times \Sigma^k) \longrightarrow (K_p \cup K_a \cup \{h\}) \times (\{R, L, N\}^k \times \Sigma^k)$. The head on the input tape is allowed to move only to the right. A configuration of an on-line k-tape Turing machine becomes $C = (q, t, x_1\underline{a_1}y_1, \dots, x_k\underline{a_k}y_k)$, where q is a state, $t \in \Sigma^*$ is the (not yet considered) content of the input tape, $x_i a_i y_i$ is the content of the i-th working tape, and a_i is the symbol that is currently scanned by the head of tape i, $1 \leq i \leq k$. The set of states is divided into two subsets: the set of *polling* states K_p and the set of *autonomous* states K_a. All the states that lead to h in one step are polling states, and the initial state is a polling state. In addition, the relation $\vdash_M$ has the following property: if $q \in K_p$, $q'' \in K_a$, and $q' \in K_p \cup K_a$, then $(q, abv, x_1, \dots, x_k) \vdash_M (q', bv, x_1', \dots, x_k')$, $(q'', abv, x_1, \dots, x_k) \vdash_M (q', abv, x_1', \dots, x_k')$, and $(q, \epsilon, x_1, \dots, x_k) \vdash_M (h, \epsilon, x_1', \dots, x_k')$. M accepts the input w as usual.

A *real-time Turing machine* is an on-line Turing machine for which $K_a = \emptyset$. A language accepted by such a machine is called *real-time definable*. The languages accepted by nondeterministic real-time Turing machines (defined identically to the real-time Turing machine except for the nondeterminism) are called *quasi-real-time* languages [2].

In summary, an on-line Turing machine has a unidirectional (and read-only) first tape (called input tape). It has therefore no knowledge about further input data. Between reading two input symbols, such a machine is allowed to go into a number of autonomous states, where it performs some work without considering any input. In addition to these requirements, a real-time Turing machine has no autonomous state, it being forced to consume an input datum at every step.

A (nondeterministic) finite automaton is an on-line Turing machine with no work tapes (that is, no memory). For a deterministic finite automaton the set of autonomous states is furthermore empty[1]. It is known however that deterministic and nondeterministic finite automata accept the same class of languages (called regular languages) [8].

3 Small Space Real-Time Turing Machines Cannot Count

The purpose of the work tape(s) of a Turing machine is to keep some information about the input; such an information is updated as the machine runs and becomes the basis of the final

---

[1]Finite automata are typically defined differently [8], but we find this definition convenient given that we already introduced on-line and real-time Turing machines.

decision of the machine to accept (or not) the input. Given the treatment of input in on-line (and thus real-time) Turing machines we refer to the process of managing such information as *counting*. Indeed, such a machine sees the input one symbol at a time; it can then update (or not) the extra information stored on its work tapes. Such an update consists at the very least in the incrementation or decrementation of a counter. The processing could be of course more sophisticated, but we are going to show that not even incrementing/decrementing can be achieved in real time on sublinear space, so taking the simplest processing into account will suffice.

Lemma 1. *A sublinear-space bounded real-time Turing machine (deterministic or nondeterministic) cannot count any quantity in $\omega(1)$.*

Proof. Let the input string be w of length $|w| = n$.

Consider a real-time Turing machine with one counter that needs to count some quantity $g(n) \in \omega(1)$. This counter can only be represented in unary notation, for indeed any other encoding requires in the worst case a non-constant time for incrementation (and decrementation)— this is caused by the carry that is generated by an incrementation and will eventually need to propagate throughout the number in any other encoding than linear. It follows that $g(n)$ can only be sublinear.

Note further that a real-time Turing machine cannot know n in advance of reaching the end of the input[2] and cannot therefore anticipate—not even by a nondeterministic guess, since such a guess implies trying all the possible variants—any bound for $g(n)$ (so that it can stop on exceeding the bound). An input that does not belong to the language can then bring $g(n)$ to linear, thus causing the Turing machine to exceed its allotted space (that $g(n)$ can be linear for a not necessarily acceptable input w is immediate using an information-theoretic argument). Counting beyond $\omega(1)$ is therefore impossible.   □

Theorem 2. *The languages accepted by sublinear-space bounded real-time Turing machines (deterministic or nondeterministic) include exactly all the regular languages.*

Proof. Since sublinear-space bounded real-time Turing machines cannot count beyond $O(1)$ (Lemma 1) then their work memory is useless. Indeed, counting up to $O(1)$ can be easily done using the finite state control, as commonly done for finite automata.   □

Corollary 3. *Non-regular real-time definable languages exist and can only be accepted using $\Omega(n)$ space.*

Proof. This is a direct consequence of Theorem 2. The only thing that needs consideration is the existence of non-regular real-time definable languages. Such languages can be easily found, one of the most immediate being $\{a^n b^n : n \geq 0\}$ (known non-regular and acceptable by a real-time Turing machine that uses a single counter).   □

4   Conclusions

We identified quite easily a gap in the space complexity of real time. This gap is considerably wider than for classical Turing machines (from constant to linear in real time as opposed to from constant to $\log \log n$ in a non-real-time setting [13]).

---

[2]And not even then if it does not have access to linear space on the working tape, as seen here; this is however forward looking and therefore a circular argument.

We regard such a wide gap as an emphasis of what has been more or less subconsciously the belief in the computing field, namely that the real-time Turing machine is too weak a model to capture the actual phenomena that take place in the real world; not even the nondeterminism helps in this respect. The extremely simple computation engine of a Turing machine is particularly well suited to analyze computations in general, but it is apparently too simple to be really useful in real time. The random access machine (RAM) is a marginally richer model (that performs elementary arithmetic operations in constant time) and can easily accomplish in real time and logarithmic space all the processing implied in Lemma 1 (that is unavailable to the real-time Turing machine with sublinear space).

Overall, it appears that the real-time Turing machine does not have a niche. Indeed, the model of choice for real time in formal methods is either the finite automaton [12] or the very powerful timed transition system [3, 4]. As we saw (to some degree) here, such choices are not accidental.

Acknowledgments

References

[1] S. O. AANDERAA, *On k-tape versus $(k-1)$-tape real time computation*, in Complexity of Computation, R. Karp, ed., SIAM-AMS Proceedings, volume 7, 1974, pp. 75–96.

[2] R. V. BOOK AND S. A. GREIBACH, *Quasy–realtime languages*, Mathematical Systems Theory, 4 (1970), pp. 97–111.

[3] L. B. BRIONES AND E. BRINKSMA, *A test generation framework for quiescent real-time systems*, in Formal Approaches to Testing of Software, 2004, pp. 71–85.

[4] S. D. BRUDA AND C. DAI, *A testing theory for real-time systems*, International Journal of Computers, 4 (2010), pp. 97–106.

[5] P. C. FISCHER, *Turing machines with a schedule to keep*, Information and control, 11 (1967), pp. 138–146.

[6] P. C. FISCHER AND C. M. R. KINTALA, *Real-time computations with restricted nondeterminism*, Mathematical Systems Theory, 12 (1979), pp. 219–231.

[7] A. KLEIN AND M. KUTRIB, *Deterministic turing machines in the range between real-time and linear-time*, Theoretical Computer Science, 289 (2002), pp. 253–275.

[8] H. R. LEWIS AND C. H. PAPADIMITRIOU, *Elements of the Theory of Computation*, Prentice-Hall, 2nd ed., 1998.

[9] M. O. RABIN, *Real time computations*, Israel Journal of Mathematics, 1 (1963), pp. 203–211.

[10] A. L. ROSENBERG, *Real-time definable languages*, Journal of the ACM, 14 (1967), pp. 645–662.

[11] ——, *On the independence of real-time definability and certain structural properties of context-free languages*, Journal of the ACM, 15 (1968), pp. 672–679.

[12] J. SPRINGINTVELD, F. VAANDRAGER, AND P. R. D'ARGENIO, *Testing timed automata*, Theoretical Computer Science, 254 (2001), pp. 225–257.

[13] A. SZEPIETOWSKI, *Turing Machines with Sublogarithmic Space*, Springer Lecture Notes in Computer Science 843, 1994.

[14] H. YAMADA, *Real-time computation and recursive functions not real-time computable*, IRE Transactions on Electronic Computers, EC-11 (1962), pp. 753–760.