Machine Models and Linear Time Complexity

Kenneth W. Regan[1]
Department of Computer Science, State University of New York at Buffalo
226 Bell Hall, Buffalo NY 14260-2000; regan@cs.buffalo.edu

Guest Column for Lane A. Hemaspaandra, SIGACT News Complexity Theory Column 2
To appear in *SIGACT News*, Vol. 24, No. 4, Fall 1993

1 The Problem of Linear Time

The first of several reasons Linear Time has received relatively little theoretical attention is that no one has given a *robust* notion of what "Linear Time" *is*. Whereas Polynomial Time is the same for just about every sequential model $\mathcal{M}$ that has been proposed, almost every two of these models seem to give different definitions of linear time. Two other reasons are that Linear Time seems not to have as many pleasant closure properties as Polynomial Time, and often depends on fussy details of how information is represented. (For instance: Given n integers in the range $1 \dots n^2$, should the length of the input be n or $\approx n \log n$? In linear time, a TM can add integers in standard binary notation but multiplication is considered unlikely; in prime-factor notation, a TM can multiply but likely not add.) However, one of several reasons for the importance of Linear Time is that the frontier of proving lower bounds for *natural* problems on *reasonable* models of general-purpose computation has only barely been pushed west of the $O(n)$ Meridian. This survey aims first to put the model-dependence issue into greater focus, and then to provide a new angle for work on lower bounds.

Machine models. Suppose that for every machine M_1 in model $\mathcal{M}_1$ running in time $t = t(n)$ there is a machine M_2 in $\mathcal{M}_2$ which computes the same partial function in time $g = g(t, n)$. If $g = O(t) + O(n)$ we say that model $\mathcal{M}_2$ *simulates* $\mathcal{M}_1$ *linearly*. If $g = O(t)$ the simulation has *constant-factor overhead*; if $g = O(t \log t)$ it has a *factor-of-*$O(\log t)$ *overhead*, and so on. The simulation is *on-line* if each step of M_1 is simulated by step(s) of M_2 in some transparent manner—see [vEB90, LL92b] for discussion of how to formalize this.

The most powerful sequential models in the survey by van Emde Boas [vEB90] are forms of the random access machine (RAM) with the *unit cost* time measure (RAM-TIME), which charges one time unit to execute any instruction. A standard RAM has addition and subtraction as arithmetic operations.[2] For our purpose, the arguments of Cook and Reckhow [CR73] that unit cost is *unreasonable* hold force. The SRAM model with arithmetic limited to increment and decrement lessens this objection, and here our interests begin. Cook and Reckhow instead advocated using the *log-cost* time measure (RAM-TIME$^{\log}$), which charges one time unit per bit of address and operand values in an instruction. Sources differ on how to represent an input $x \in \{0, 1\}^n$ in a RAM-based model: (i) as an integer in register R_0, (ii) bitwise in the first n registers, or (iii) on a separate read-only Turing tape; (iii) is standardized in [WW86] and used in [KvLP88, TLR92, LL92b].

We note the following "Large Cluster" of models known to simulate each other with overhead at most $O(\log t)$. The *storage modification machine* (SMM) of Schönhage [Sch80], the programming model of Jones [Jon93], and the SRAM all simulate each other linearly [Sch80, Jon93]. The *tree*

[1]The author's own work reported here was partly supported by NSF Grant CCR-9011248

[2]Having unit-cost multiplication yields the MRAM, which can accept QBF and other PSPACE-complete languages in polynomial time [HS74], and is regarded as a "parallel" model in [vEB90, TLR92].

computer (TC) can be described as a TM which has worktapes in the form of infinite binary trees, or which has standard tapes but can move a worktape head on some cell a to cell $\lfloor a/2 \rfloor$, $2a$, or $2a+1$ in one step. A TC can simulate a log-cost RAM with constant-factor overhead [PR81], and can be simulated by a TM in time $O(t^2/\log t)$ [PF79].[3] Loui and Luginbuhl [LL92b] give an $O(t \log t/\operatorname{loglog} t)$ simulation of a TC by a log-cost RAM M, and interestingly, prove that this is *optimal* for *on-line* simulation when $t(n) = n$. We note that their M has the property that every cell address a and register content i used is bounded by a polynomial in the running time t, which is the defining property of the "RAM with strongly polynomially compact memory" of Grandjean and Robson [GR91]. Gurevich and Shelah [GS89] show other models to belong (some with a small power of $\log t$), including the RAC model of [AV79] and "random-access Turing machines."

Below the "Large Cluster" are Turing machines with d-dimensional worktapes (d-TMs), whose time classes we denote by $\mathrm{DTIME}^d[t]$.[4] For $d = 1$ we have the standard multitape TM model, whose time classes are denoted by DTIME$[t]$, and for nondeterministic TMs, NTIME$[t(n)]$. DLIN and NLIN are short for DTIME$[O(n)]$ and NTIME$[O(n)]$. Turing machines with multiple heads per tape, even with head-to-head "jumps" allowed, can be simulated step-for-step by ordinary TMs (see [FMR72, PF79, Kos79, LS81]). TMs with only one worktape, or with some other kind of restricted storage such as a stack or queue or counters, are generally considered too weak to model general-purpose computation, though interesting nonlinear lower bounds have been proved for them [DGPR84, MS86, LV88, DMS91, LLV92].

Theorem 1.1 (a) $\mathrm{RAM\text{-}TIME}^{\log}[t] \subseteq \mathrm{DTIME}^d[t^{1+1/d}/\log t]$ [PF79].

(b) $\mathrm{DTIME}^d[t] \subseteq \mathrm{DTIME}[t^{2-1/d}]$ [Lou83].

Loui and Luginbuhl [LL92a] also prove that in the case $t(n) = O(n)$, (a) is close to optimal for *on-line* simulations. However, *no separations at all* have been proved for the language classes, not even for DLIN vs. SRAM-TIME$[O(n)]$.[5] Thus among the whole sweep of "reasonable" models, the problem of linear-time robustness is really wide open. The problem boils down to a matter of n vs. n^2. From reasonable RAMs to d-TMs it is a matter of n vs. $n^{1+1/d}$.

A focal point for the problem is the concept of *vicinity* of a machine model $\mathcal{M}$ raised by Feldman and Shapiro [FS92], which we formalize as follows. Let C be a data configuration, and let H_C stand for the finite set of memory locations [or data items] designated as "scanned" in C. For all $t > 0$, let I_t denote the set of locations [or items] i such that there exists an $\mathcal{M}$-program which, when started in configuration C, scans i within t time units. Then the model $\mathcal{M}$ has *vicinity* $v(t)$ if for all C and t, $|I_t|/|H_C| \leq v(t)$. Feldman and Shapiro contend that realistic models should have polynomial vicinity—perhaps cubic vicinity under physics as we know it. However the RAM, even under log-cost, has exponential vicinity. So do all the other models in the "Large Cluster," even that of [GR91] (consider t much less than the overall runtime). TMs with d-dimensional tapes have vicinity $O(t^d)$. The theorem of [PPST83] that NLIN $\neq$ DLIN is not known to carry over to any model of super-linear vicinity. Our "BM" model below has a parameter μ which calibrates its vicinity.

[3]This improves the $O(t^2)$ simulation of log-cost RAM by TM given in [CR73]; see also [Wie83, vEB90].

[4]A d-TM is simulated linearly by an SMM [Sch80], in time $O(t \cdot 5^{d \log^* t})$ by a TC [Rei82], and in time $O(t(\log t)^{1-1/d}(\operatorname{loglog} t)^{1/d})$ by a log-cost RAM [LL92a]. For $d = 1$, [KvLP88] give more detail: if the TM runs in time t, space s, and gives output length u, the log-cost RAM takes time $O(t + (n+u)\operatorname{loglog} s)$.

[5]The inclusion DTIME$[t] \subseteq$ RAM-TIME$[t/\log t]$ implies a $\log t$ separation, which is hardly felt to be the true power of the unit-cost RAM over the TM, but it was proved by [HPV75] only for $t(n) = \Omega(n \log n)$. Sudborough and Zalcberg [SZ76] prove DLIN $\subset$ RAM-TIME$[O(n)]$.

2 Linear-Time Problems

The following natural decision problems have received much attention in the literature. We may suppose that the list elements $x_i, y_j \in \{0,1\}^*$ all have the same length.

(a) Pattern matching: $L_{pat} := \{ p\#t : (\exists u, v \in \{0,1\}^*)\, t = upv \}$.

(b) Element (non)distinctness: $L_{dup} := \{ x_1\#x_2\#\dots\#x_m : (\exists i,j)\, i < j \wedge x_i = x_j \}$.

(c) List intersection: $L_{int} := \{ x_1\#x_2\#\dots\#x_m,\ y_1\#y_2\#\dots\#y_m : (\exists i,j)\, x_i = y_j \}$.

(d) Triangle: $L_\Delta := \{G : G$ is an undirected graph which contains a triangle$\}$.

L_{pat} belongs to DLIN [FP74] (cf. [FMR72, GS83]), and was recently shown not to be recognizable by a one-way multihead DFA [JL93]. L_{dup} and L_{int} can be solved in linear time by a RAM which treats list elements as cell addresses. L_Δ is not believed recognizable in linear time on a RAM at all—the best method known is to square the adjacency matrix of G.

Adachi and Iwata [AI84] showed that every language in DTIME$[O(n^k)]$ many-one reduces in $O(n \log n)$ time to a language L_k defined by a pebble game with $4k+1$ pebbles; thus L_k is not in DTIME$[n^{k-\epsilon}]$. Grandjean [Gra90] showed that every language in NLIN reduces to a natural NP-complete problem called "RISA" by DLIN-reductions; thus RISA is not in DLIN. However, we know of no language in NLIN $\cap$ P which is not in DLIN. For recent clear treatments of quadratic lower bounds on the *product* of space and time for sorting and other natural functions, see [Bea91, MNT93], and for related results on languages, [Kar86, Yao88, GS88]. For nonlinear lower bounds in other models see [KMW89, KW91, BS91, DM92].

3 A Game

The basic idea of our new model can be expressed as a one-person game in which the Player (P) seeks the least costly way to change given strings x, initially placed in locations $0\dots n{-}1$ of a "long" sequential file, into corresponding strings y. There is a monotone function $\mu : \mathbf{N} \to \mathbf{N}$ such that $\mu(a)$ represents the cost of accessing location a in the file. The intent is that low-numbered locations are like a fast-memory cache, while high locations figuratively reside on a large but slow disk drive. Typical μ functions, studied also in our major references [AACS87, ACS87], are $\mu_{\log}(a) := \lceil \log_2 a \rceil$ (basically the log-cost criterion), $\mu_d(a) := a^{1/d}$, which models the decay in response time for memory laid out on a d-dimensional grid, and step functions with finitely many values. (The cited papers write f for μ and α for $1/d$.) The highest "memory access cost function" we consider is $\mu_1(a) := a$.

A simple edit rule would allow P to change the character in some cell a, at a cost of $\mu(a)$ time units. The distinctive point of our game is that if several edits of a similar kind can be done in one block $[a\dots b]$ of consecutive locations in the file, P should profit from this *spatial locality* by being charged $\mu(a)$ or $\mu(b)$ only for the initial access, and unit time per edit thereafter. Here "similar kind" means that the edits can be done by a finite-state machine, for which we take the deterministic *generalized sequential machine* (GSM) formalized in [HU79]. (A GSM S is essentially a Mealy machine which can output not just one but zero, two, three, or more symbols in a given transition.)

Rule 1. In any move, P may mark locations a_1, b_1, a_2, b_2, with $b_1 < a_1$ and/or $b_2 < a_2$ allowed, and select a GSM S. Let z be the string held in locations $[a_1 \dots b_1]$. Then $S(z)$ is written to locations $[a_2 \dots b_2]$, at a charge of $\|S\| \cdot |z| + \mu(a)$, where $a = \max\{ a_1, b_1, a_2, b_2 \}$.

Here $\|S\|$ may be taken as the number of states in S; this discourages P from using GSMs which are too large. The output $S(z)$ "overtypes" any previous content of $[a_2 \ldots b_2]$. This editor does not have an insert/delete mode or allow P to "cut" z and/or "paste" $S(z)$ at location a_2. In practice, a large number of block insertions and deletions can "fragment" files and file systems. Willard [Wil92] gives efficient ways of simulating these operations on a realistic sequential file system model which lacks them. However, information in $[a_2 \ldots b_2]$ may be preserved under the following convention which seems to involve no loss of efficiency.

Rule 2. The blank B may be an output character of GSMs, and in Rule 1, every B in the output stream $S(z)$ leaves the previous symbol in its target cell in $[a_2 \ldots b_2]$ unchanged.

Under Rules 1 and 2 we speak of the *block move* $S[a_1 \ldots b_1]$ *into* $[a_2 \ldots b_2]$. By analogy with UNIX$^{(\mathrm{R})}$ text editing, the stream z is "piped" through the finite-state "filter" S. The move is *valid* if the two intervals are disjoint. The *strict boundary condition* is that the output $S(z)$ exactly fills $[a_2 \ldots b_2]$. (Here we can take for granted that P chooses b_2 to make this so, but in the uniform model below this is a nontrivial restriction.)

Complexity. Each block move adds 1 to the *pass count* $R(x)$, $|b_1 - a_1| \cdot \|S\|$ to the *work* $w(x)$, and $\mu(a)$ to the *memory access charges* μ-*acc*(x), under a given μ. The μ-*time* for the computation equals $w(x) + \mu$-*acc*(x). The *space* $s(x)$ equals the largest magnitude of an *access point* a in the computation. These complexity measures are extended to functions $R(n), w(n), \mu$-*acc*$(n), \mu$-*time*(n), and $s(n)$ of input lengths n in the usual manner.

4 The Block Move Model

The *Block Move* (BM) model can be regarded as an extension of the *Block Transfer* (BT) model of Aggarwal, Chandra, and Snir [ACS87]. The BT is a RAM with the special instruction *copy* $[a - m \ldots a]$ *into* $[b - m \ldots b]$, which is charged $m + \max\{\,\mu(a), \mu(b)\,\}$ time units. The BT likewise sets no fixed limit on the block-size m. Other RAM instructions involving some register a are charged $\mu(a)$ time units, and since this is the same as the time for *copy* $[a \ldots a]$ *into* $[0 \ldots 0]$, in the BT one may suppose that the other instructions involve registers 0 and 1 only. The BT uses RAM convention (ii): an input x of n symbols (or n integers) is placed in the first n registers. The authors of [ACS87] prove tight nonlinear bounds on the time required to copy every symbol to register 0, and hence to compute any function which depends on all of the input, viz. $\Theta[n \log n]$ for $\mu = \mu_1$, $\Theta[n \operatorname{loglog} n]$ for $\mu = \mu_d$ with $d > 1$, and $\Theta[n \log^* n]$ for $\mu = \mu_{\log}$. The same upper bounds apply, for the same μ, to the time for P in the game to change 0^n into any given $y \in \{\,0, 1\,\}^n$, using only *copy* and writing 0 or 1 to cell 0. By contrast, the other finite transductions allowed in Rule 1 can glean information about the input in a way that *copy* cannot, and enable a rich theory of linear time.

The BM model makes the above game *uniform* by fixing limits on the number of GSMs a given machine M can use, and on the number of locations which can be the *access points* a_1, b_1, a_2, b_2 in any one turn. The element of *robustness* is that it doesn't much matter how the game's basic idea is implemented, with regard to any μ function which gives polynomial vicinity. Some variants:

• *Pointer form:* M has four "pointers" labeled a_1, b_1, a_2, b_2, some number $m \geq 4$ of "pointer markers," and a finite control composed of "GSM states" and "move states." Initially one marker is in cell $n - 1$, the rest in cell 0. A GSM state S executes $S[a_1 \ldots b_1]$ *into* $[a_2 \ldots b_2]$, and passes control to a move state. In a move state, each pointer marker on some cell a may be moved to cell $\lfloor a/2 \rfloor$, $2a$, or $2a + 1$ or left where it is. Then the four pointers are redistributed among the m markers, and control branches according to the symbol in the cell now occupied by pointer a_1. The charge is $\mu(a)$ for the largest a involved in a pointer or marker move; markers which stay put are not charged in a move step.

• *Random-access form:* M has four *address tapes* labeled a_1, b_1, a_2, b_2 which can receive the output of block moves—here the output overwrites the address tape, and specifies an integer which becomes the value of the corresponding pointer for the next step. This form is closest to the BT, though based on a fixed-wordsize rather than integer RAM. *Richer forms* have any number k of main tapes, each with its own pointers and address tapes, and have block-move instructions which mark intervals on the k tapes and use GSMs which read a k-tuple of symbols at a time.

• *Reduced form:* M consists of a single GSM S, some of whose states r have no outarcs but instead have labels $l_1(r)$ and $l_2(r)$. M has just one tape with two heads; at the start of each *pass*, one is set to cell 0 and the other to cell a, where a is the *current address*. Initially $a = 0$ and the first pass has *mode* 'Ra,' meaning that the cell-a head reads z moving rightward from cell a, and the cell-0 head writes the output $S(z)$. In mode 'La' the cell-a head reads leftward instead of rightward. In mode '0R' the cell-0 head reads (necessarily moving rightward) and the cell-a head does the writing rightward from cell a; last, mode '0L' moves the latter leftward from cell a. If and when S enters some terminal state r, the label $l_1(r)$ alone determines whether the entire computation halts. If not, $l_1(r)$ states whether a is changed to $\lfloor a/2 \rfloor$, $2a$, or $2a+1$ or kept where it is for the next pass, and $l_2(r)$ sets one of the four *modes* for the next pass. (Note that here the boundaries b_1 of the read block and b_2 of the write block are not set before the pass, but are data-dependent.)

Theorem 4.1 ([Reg92]) *Under any memory cost function μ_d with $d \geq 1$ and rational, the above forms of the BM model, and further variants obtained by selectively enforcing the validity and strict boundary conditions and/or making block boundaries data-dependent, all simulate each other linearly in μ_d-time.*[6]

Hence all forms define the same complexity classes $\mathrm{D}\mu_d\mathrm{TIME}[O(t(n))]$, even for $t(n) = O(n)$. The smallest of these is TLIN $:= \mathrm{D}\mu_1\mathrm{TIME}[O(n)]$. Linear-time robustness almost holds under $\mu_{\log}$—all forms simulate each other within $O(\log t)$ overhead. To define $\mathrm{D}\mu_{\log}\mathrm{TIME}[t]$, and classes with a bound on $R(n)$, we refer to the pointer form.

Example 4.1. The language D_1 of balanced parentheses belongs to TLIN. Define S to work as follows on any $x \in \{\,(,)\,\}^*$: If $x = \lambda$, S goes to a terminal state r labeled ACCEPT; if x begins with ')', S goes to REJECT. Else S erases the leading '(' and thereafter takes bits in twos, translating

$$((\mapsto (\qquad)) \mapsto) \qquad () \mapsto \lambda \qquad)(\mapsto \lambda. \tag{1}$$

If x ends in '(' or $|x|$ is odd, S also signals REJECT. Then S has the property that for any $x \neq \lambda$ which it doesn't immediately reject, x is balanced iff $S(x)$ is balanced. Furthermore, $|S(x)| < |x|/2$. (We can think of the language D_1 as being self-reducible in a sharp sense.) Hence the reduced-form BM M which iterates S on its own output (Theorem 4.1 lets us ignore the validity condition) accepts D_1 in linear μ_1-time.

Since M also has $R(n) = O(\log n)$, we say D_1 belongs to *log-pass* TLIN. Some log-pass TLIN operations on lists $x_1\# \ldots \#x_m$ are: determining whether an item p appears in the list (and marking

[6]The reason d is stated to be rational is that some of the simulating machines M' actually calculate $\mu_d(a)$. The simulations do not necessarily preserve w and μ-*acc* individually up to constant factors, but do preserve their sum. If M is *memory-efficient*, meaning that μ-*time*$_M(n) = O(w_M(n))$, then M' is also memory-efficient. Always $s'(n) = O(s(n))$, but several simulations give $R'(n) = O(R(n) \log s(n))$.

all occurrences if so), finding the maximum element, and computing all prefix sums. Counting, and hence the set of palindromes, is in log-pass TLIN. Whether the language D_2 of balanced $(,), [,]$ is in TLIN at all is open. The reduced form is comparable to the "sweeping automata" and linear iterative arrays studied by Ibarra et al. (see [CIV88]).

5 Complexity Classes

Like the RAM but unlike what is known for the multitape TM (see [CR73, HU79]), the BM under any μ_d has only a constant-factor overhead for universal simulation.[7] One consequence is that the small fixed set $\mathcal{S}$ of GSMs used by the universal BM (same one for all μ_d) is also universal for the editing game; i.e. simulates any other set $\mathcal{S}'$ up to constant factors. Another consequence is the following "tight deterministic time hierarchy" theorem. A function t is *μ-time constructible* if $t(n)$ is computable from n in binary notation in μ-time $O(t(n))$.

Theorem 5.1 *With respect to μ_d ($d \geq 1$ and rational), let t_1 and t_2 be functions such that $t_1(n) \geq n$, $t_1 = o(t_2)$, and t_2 is μ_d-time constructible. Then* $\mathrm{D}\mu_d\mathrm{TIME}[t_1]$ *is properly contained in* $\mathrm{D}\mu_d\mathrm{TIME}[t_2]$.

The first open question this raises is whether there is any hierarchy at all when time is fixed and d varies, and in particular whether TLIN $\neq \mathrm{D}\mu_2\mathrm{TIME}[O(n)] \neq \mathrm{D}\mu_3\mathrm{TIME}[O(n)] \neq \ldots$ It seems reasonable to expect that separations by more than factors of $O(\log n)$ should be provable, but the following results point out the difficulty of obtaining even such a separation of TLIN from linear-time on a TC or SRAM:

Theorem 5.2 *For any time function $t(n) \geq n$,*

(a) $\mathrm{D}\mu_1\mathrm{TIME}[t] \subseteq \mathrm{DTIME}[t] \subseteq \mathrm{D}\mu_1\mathrm{TIME}[O(t \log t)]$.

(b) $\mathrm{D}\mu_{\log}\mathrm{TIME}[t] \subseteq \text{TC-TIME}[O(t)] \subseteq \mathrm{D}\mu_{\log}\mathrm{TIME}[O(t \log t)]$.

(c) *For any $d \geq 1$,* $\mathrm{D}\mu_d\mathrm{TIME}[t] \subseteq \mathrm{DTIME}^d[t]$.

Thus letting μ vary from $\mu_{\log}$ to μ_1 spans the whole range of models in Section 1. The second inclusion in (a) follows because the *Hennie-Stearns simulation* [HS66, HU79, WW86] is memory-efficient under μ_1. We suspect that for $d > 1$ the converse simulation in (c) requires notably more than the $O(\log t)$ overhead of the $d = 1$ case (a); see [PSS81] for related matters. The intuitive reason is that a d-TM may often change its head direction, but in going to a BM this is a break in pipelining and subject to a memory-access charge.

Let us abbreviate *quasi-linear* time by $qlin := n(\log n)^{O(1)}$, and following [Sch78], DTIME[$qlin$] to DQL, NTIME[$qlin$] to NQL. All models in the "Large Cluster" accept the same class—call it DNLT—of languages in time $qlin$. Gurevich and Shelah [GS89] also proved that the nondeterministic counterparts of these models accept exactly the languages in NQL. Finally, let $\mathrm{N}\mu\mathrm{TIME}[t(n)]$ be the class of languages accepted in μ-time $t(n)$ by *NBMs*, which may use nondeterministic GSM mappings in block moves.

Corollary 5.3 (a) $\mathrm{D}\mu_1\mathrm{TIME}[qlin] = \mathrm{DQL}$.

[7] TMs with a fixed number $k \geq 2$ of worktapes do have constant factor universal simulation [Für84]. The constant factor in Jones' universal simulation theorem is *independent* of the program being simulated [Jon93]—this yields a sharper result than Theorem 5.1 for his model. Sudborough and Zalcberg [SZ76] proved results similar to Jones' for the unit-cost RAM.

(b) $\mathrm{D}\mu_{\log}\mathrm{TIME}[qlin] = \mathrm{DNLT}$.

(c) *For all* $\mu \leq \mu_1$, $\mathrm{N}\mu\mathrm{TIME}[qlin] = \mathrm{NQL}$.

Thus if for some $d > e$ we could separate $\mathrm{D}\mu_d\mathrm{TIME}[O(n)]$ from $\mathrm{D}\mu_e\mathrm{TIME}[O(n)]$, or even TLIN from $\mathrm{D}\mu_{\log}\mathrm{TIME}[O(n)]$, by more than a polylog factor, then we would have proved NQL $\neq$ DQL. However, the question of NQL vs. DQL seems to have much the same "shape" as P vs. NP—see [Sch78, BG93]. Thus the problem of linear time robustness runs into old familiar issues of determinism versus nondeterminism. And by Theorem 5.2(a), even an $\omega(n \log n)$ bound on time under μ_1 implies a nonlinear lower bound for Turing machines.

6 Open Problems in Linear Time

The main open problem is to find nonlinear lower bounds on μ-time for natural problems. In particular, can the lower bounds for FFT, matrix transpose, and sorting in [ACS87] be established under the extension to the BT which Rules 1 and 2 represent? The evident idea is to exploit the fact that in order for a BM M to run in linear μ-time, the set of access points used in the computation must thin out at the high end of memory. In particular for $\mu = \mu_1$, there are long segments between access points which can be visited only a constant number of times. Can this idea be used to separate $\mathrm{D}\mu_d\mathrm{TIME}[O(n)]$ from $\mathrm{N}\mu_d\mathrm{TIME}[O(n)]$? Languages (a-d) in Section 2 all belong to NTLIN; indeed, the NBM need only guess a_1, b_1, a_2, b_2 for a single pass, making $O(\log n)$-many deterministic passes otherwise. Proving non-membership in *log-pass* TLIN, or generally with $R(n) = o(n)$, may be a leg up on the problem stressed by Valiant [Val92] of finding natural problems which don't have simultaneously linear-size, log-depth circuits. Some related open problems have come up in this work:

- Does every language in TLIN have linear-sized circuits? (Compare Theorem 5.2(a) and the fact that every $L \in$ DLIN has $O(n \log n)$-sized circuits.)
- Is log-pass TLIN $\subseteq$ NC^1? (Generally, NC^k $\subseteq$ BM poly work, $O(\log^k n)$ passes $\subseteq$ NC^{k+1} $(k \geq 1)$, and some restrictions on GSMs give equality in the former [Reg93].)
- Consider sorted lists which have $\sqrt{n}$ elements, each of size $\sqrt{n}$. For any $d > 1$, two such lists can be merged in linear μ_d-time. But can this be done in linear μ_1-time?
- Does every language accepted by a k-head DFA (see [JL93]) belong to TLIN?
- Can nonlinear lower bounds be proven for a TM with one worktape, initially holding the input x, and one pushdown store with the restriction that after any POP, the entire store must be emptied before the next PUSH?

Because μ is a parameter we have admittedly not given a single answer to the question "What is Linear Time?" However, we nominate $\mathrm{D}\mu_3\mathrm{TIME}[O(n)]$, $\mathrm{D}\mu_2\mathrm{TIME}[O(n)]$, and TLIN as leading candidates. We offer the BM as providing useful new structure for the study of linear-time computation and the quest for nonlinear lower bounds.

Acknowledgments I am grateful to Michael C. Loui and Lane A. Hemaspaandra for helpful comments on earlier drafts.

References

[AACS87] A. Aggarwal, B. Alpern, A. Chandra, and M. Snir. A model for hierarchical memory. In *Proc. 19th STOC*, pages 305–314, 1987.

[ACS87] A. Aggarwal, A. Chandra, and M. Snir. Hierarchical memory with block transfer. In *Proc. 28th FOCS*, pages 204–216, 1987.

[AI84] A. Adachi and S. Iwata. Some combinatorial game problems require $\Omega(n^k)$ time. *J. ACM*, 31:361–376, 1984.

[AV79] D. Angluin and L. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. Comp. Sys. Sci.*, 18:155–193, 1979.

[Bea91] P. Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM J. Comp.*, 20:270–277, 1991.

[BG93] J. Buss and J. Goldsmith. Nondeterminism within P. *SIAM J. Comp.*, 22:560–572, 1993.

[BS91] D. Mix Barrington and H. Straubing. Superlinear lower bounds for bounded-width branching programs. In *Proc. 6th Structures*, pages 305–313, 1991.

[CIV88] J. Chang, O. Ibarra, and A. Vergis. On the power of one-way communication. *J. ACM*, 35:697–726, 1988.

[CR73] S. Cook and R. Reckhow. Time bounded random access machines. *J. Comp. Sys. Sci.*, 7:354–375, 1973.

[DGPR84] P. Dūriš, Z. Galil, W. Paul, and R. Reischuk. Two nonlinear lower bounds for on-line computations. *Info. Control*, 60:1–11, 1984.

[DM92] C. Damm and C. Meinel. Separating complexity classes related to Ω-decision trees. *Theor. Comp. Sci.*, 106:351–360, 1992.

[DMS91] M. Dietzfelbinger, W. Maass, and G. Schnitger. The complexity of matrix transposition on one-tape off-line Turing machines. *Theor. Comp. Sci.*, 82:113–129, 1991.

[FMR72] P. Fischer, A. Meyer, and A. Rosenberg. Real-time simulations of multihead tape units. *J. ACM*, 19:590–607, 1972.

[FP74] M. Fischer and M. Paterson. String matching and other products. In R. Karp, editor, *Complexity of Computation*, volume 7 of *SIAM-AMS Proceedings*, pages 113–125. Amer. Math. Soc., 1974.

[FS92] Y. Feldman and E. Shapiro. Spatial machines: A more-realistic approach to parallel computation. *Comm. ACM*, 35:60–73, October 1992.

[Für84] M. Fürer. Data structures for distributed counting. *J. Comp. Sys. Sci.*, 29:231–243, 1984.

[GR91] E. Grandjean and J. Robson. RAM with compact memory: a robust and realistic model of computation. In *CSL '90: Proceedings of the 4th Annual Workshop in Computer Science Logic*, LNCS. Springer-Verlag, 1991.

[Gra90] E. Grandjean. A nontrivial lower bound for an NP problem on automata. *SIAM J. Comp.*, 19:438–451, 1990.

[GS83] Z. Galil and J. Seiferas. Time-space optimal string matching. *J. Comp. Sys. Sci.*, 26:280–294, 1983.

[GS88] Y. Gurevich and S. Shelah. Nondeterministic linear-time tasks may require substantially nonlinear deterministic time in the case of sublinear work space. In *Proc. 20th STOC*, pages 281–289, 1988.

[GS89] Y. Gurevich and S. Shelah. Nearly-linear time. In *Proceedings, Logic at Botik '89*, volume 363 of *LNCS*, pages 108–118. Springer-Verlag, 1989.

[HPV75] J. Hopcroft, W. Paul, and L. Valiant. On time versus space and related problems. In *Proc. 16th FOCS*, pages 57–64, 1975.

[HS66] F. Hennie and R. Stearns. Two–way simulation of multitape Turing machines. *J. ACM*, 13:533–546, 1966.

[HS74] J. Hartmanis and J. Simon. On the power of multiplication in random-access machines. In *Proc. 15th Annual IEEE Symposium on Switching and Automata Th eory (now FOCS)*, pages 13–23, 1974.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison–Wesley, Reading, MA, 1979.

[JL93] T. Jiang and M. Li. K one-way heads cannot do string-matching. In *Proc. 25th STOC*, pages 62–70, 1993.

[Jon93] N. Jones. Constant factors do matter. In *Proc. 25th STOC*, pages 602–611, 1993.

[Kar86] M. Karchmer. Two time-space tradeoffs for element distinctness. *Theor. Comp. Sci.*, 47:237–246, 1986.

[KMW89] M. Krause, C. Meinel, and S. Waack. Separating complexity classes related to certain input-oblivious logarithmic space-bounded Turing machines. In *Proc. 4th Structures*, pages 240–249, 1989.

[Kos79] R. Kosaraju. Real-time simulation of concatenable double-ended queues by double-ended queues. In *Proc. 11th STOC*, pages 346–351, 1979.

[KvLP88] J. Katajainen, J. van Leeuwen, and M. Penttonen. Fast simulation of Turing machines by random access machines. *SIAM J. Comp.*, 17:77–88, 1988.

[KW91] M. Krause and S. Waack. On oblivious branching programs of linear length. *Info. Comp.*, 94:232–249, 1991.

[LL92a] M. Loui and D. Luginbuhl. The complexity of on-line simulation between multidimensional Turing machines and random-access machines. *Math. Sys. Thy.*, 25:293–308, 1992.

[LL92b] M. Loui and D. Luginbuhl. Optimal on–line simulations of tree machines by random access machines. *SIAM J. Comp.*, 21:959–971, 1992.

[LLV92] M. Li, L. Longpré, and P. Vitányi. The power of the queue. *SIAM J. Comp.*, 21:697–712, 1992.

[Lou83] M. Loui. Optimal dynamic embedding of trees into arrays. *SIAM J. Comp.*, 12:463–472, 1983.

[LS81] B. Leong and J. Seiferas. New real-time simulations of multihead tape units. *J. ACM*, 28:166–180, 1981.

[LV88] M. Li and P. Vitányi. Tape versus queue and stacks: the lower bounds. *Info. Comp.*, 78:56–85, 1988.

[MNT93] Y. Mansour, N. Nisan, and P. Tiwari. The computational complexity of universal hashing. *Theor. Comp. Sci.*, 107:121–133, 1993.

[MS86] W. Maass and G. Schnitger. An optimal lower bound for Turing machines with one work tape and a two–way input tape. In *Proc. 1st Annual Conference on Structure in Complexity Theory*, volume 223 of *LNCS*, pages 249–264. Springer-Verlag, 1986.

[PF79] N. Pippenger and M. Fischer. Relations among complexity measures. *J. ACM*, 26:361–381, 1979.

[PPST83] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On determinism versus nondeterminism and related problems. In *Proc. 24th FOCS*, pages 429–438, 1983.

[PR81] W. Paul and R. Reischuk. On time versus space II. *J. Comp. Sys. Sci.*, 22:312–327, 1981.

[PSS81] W. Paul, J. Seiferas, and J. Simon. An information-theoretic approach to time bounds for on-line computation. *J. Comp. Sys. Sci.*, 23:108–126, 1981.

[Reg92] K. Regan. Linear time and memory efficient computation. Technical report, Computer Science Department, State University of NY at Buffalo, TR 92-28, 1992. Revised version submitted to *SIAM J. Comput.*

[Reg93] K. Regan. A new parallel vector model, with exact characterizations of NC^k, 1993. In preparation.

[Rei82] R. Reischuk. A fast implementation of multidimensional storage into a tree storage. *Theor. Comp. Sci.*, 19:253–266, 1982.

[Sch78] C. Schnorr. Satisfiability is quasilinear complete in NQL. *J. ACM*, 25:136–145, 1978.

[Sch80] A. Schönhage. Storage modification machines. *SIAM J. Comp.*, 9:490–508, 1980.

[SZ76] I.H. Sudborough and A. Zalcberg. On families of languages defined by time-bounded random-access machines. *SIAM J. Comp.*, 5:217–230, 1976.

[TLR92] J. Trahan, M. Loui, and V. Ramachandran. Multiplication, division, and shift instructions in parallel random access machines. *Theor. Comp. Sci.*, 100:1–44, 1992.

[Val92] L. Valiant. Why is Boolean complexity theory so difficult? In M. Paterson, editor, *Boolean Function Complexity*, volume 169 of *LMS Lecture Notes*, pages 84–94. Cambridge University Press, 1992. Meeting held July 1990.

[vEB90] P. van Emde Boas. Machine models and simulations. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 1–66. Elsevier and MIT Press, 1990.

[Wie83] J. Wiedermann. Deterministic and nondeterministic simulations of the RAM by the Turing machine. In R.E.A. Mason, editor, *Information Processing '83: Proc. 9th World Computer Congress*, pages 163–168. North-Holland, 1983.

[Wil92] D. Willard. A density control algorithm for doing insertions and deletions in a sequentially ordered file in a good worst-case time. *Info. Comp.*, 97:150–204, 1992.

[WW86] K. Wagner and G. Wechsung. *Computational Complexity.* D. Reidel, 1986.

[Yao88] A. Yao. Near-optimal time-space tradeoff for element distinctness. In *Proc. 29th FOCS*, pages 91–97, 1988.