An Information-Theoretic Approach
to Time Bounds for On-Line Computation

W.J. Paul, J.I. Seiferas,  
and J. Simon*

TR64
February 1980

* Wolfgang J. Paul  
Fakultät für Mathematik  
Universität Bielefeld  
4800 Bielefeld  
West Germany

J.I. Seiferas  
Computer Science Department  
University of Rochester  
Rochester, NY 14627

Janos Simon  
Computer Science Department  
The Pennsylvania State University  
University Park, PA 16802

This paper is to be presented on April 30, 1980 at the 
12th Annual ACM Symposium on Theory of Computing, in 
Los Angeles, CA.

This research was supported in part by the National 
Science Foundation under Grants MCS77-06613 and MCS79-
05006. It was facilitated by the use of Theory Net  
(NSF Grant MCS78-01689).
AN INFORMATION-THEORETIC APPROACH TO TIME BOUNDS FOR ON-LINE COMPUTATION

(Wolfgang J. Paul
Fakultät für Mathematik
Universität Bielefeld
4800 Bielefeld, WEST GERMANY

Joel I. Seiferas
Computer Science Department
University of Rochester
Rochester, New York 14627

Janos Simon
Computer Science Department
The Pennsylvania State University
University Park, Pennsylvania 16802

Introduction

Static, descriptional complexity (program size) [16, 9] can be used to obtain lower bounds on dynamic, computational complexity (such as running time). We describe and discuss this "information-theoretic approach" in the following section. Paul introduced it in [13], to obtain restricted lower bounds on the time complexity of sorting. We use the approach here to obtain lower time bounds for on-line simulation of one abstract storage unit by another. A major goal of our work is to promote the approach.

Our main results show that more points of access into storage can save significant time. The storage units we consider are multthead multidimensional variants of the ordinary one-dimensional "Turing machine tape", and the points of access are the tape heads. Our bounds complement earlier results by Rabin [15], Aanderaa [1], Hennie and Stearns [8], Hennie and Pippenger and Fischer [14], and Grigoriev [6].

Aanderaa's result considers on-line simulation of $h$ single-head one-dimensional tapes (or even pushdown stores) by $h'$ such tapes, for $h' < h$ (inadequate number of tapes). Although his argument does not yield any superlinear lower bound for the worst-case time $T(n)$ to handle $n$ commands, it does show that such a simulation is impossible in real time. This generalizes Rabin's earlier result for $h = 2$, $h' = 1$. Our main result, Theorem 1 below, includes a multidimensional version of Aanderaa's result. The proof is entirely different, however, and does yield a superlinear lower bound.

In our corollary to Theorem 2 below, we show how to strengthen Aanderaa's result about tapes to a result about heads. In fact, both Aanderaa's result and our higher-dimensional version hold even if the $h'$ simulating heads can be on the same tape and are allowed to jump to each others' positions. Even if the $h'$ simulating heads can be on a $d'$-dimensional tape for $d < d' < d^2$, Theorem 1 still gives a superlinear lower bound for the time to simulate an $h$-head $d$-dimensional tape unit. All this supports the general conjecture that nothing else can make up for an inadequate number of points of access into storage.

Now suppose that every $d$-dimensional tape can be simulated on-line in time $T(n)$ by a similar tape with just $h'$ heads ($h'$ fixed). By the corollary to Theorem 1 below, $T(n)$ has to be $\Theta(n^{1+1/d} - \delta)$ for a relatively small (just barely greater than $2/(d^2 + d)$). In contrast, Hennie and Stearns' analogous upper bound for $d = 1$ is only $T(n) = O(n \log(n))$ [8, 14]. In Theorem 3, we show that every $d$-dimensional tape unit can be simulated on-line by a similar unit with just two heads in time $T(n) = O(n^{1+1/d - \delta})$, where $\delta$ is very small ($\delta = 1/(d^2 + d)$). (In fact, one of the two heads need only access a separate one-dimensional tape.) This shows that the lower
bound in Theorem 1 is a rather good one.

Hennie's result considers on-line simulation in time $T(n)$ of an $h$-head $d$-dimensional tape unit by an $h'$-head $d'$-dimensional tape unit, for $d'<d$ (inadequate dimension). His argument yields a superlinear lower bound proportional to $n^{1-1/d+1/d'}$, regardless of $h$ and $h'$. For $d'=1$, Pippenger and Fischer have a matching upper bound, even for $h'$ fixed. By the corollary to Theorem 1 below, there is no such matching upper bound for $d'>1$ and $h'$ fixed; in this sense, the Pippenger-Fischer result cannot be generalized. For $d'=d-1$ and $h'$ not fixed, Grigor'ev has a looser upper bound, $O(n^{1+1/d'})$.

The Information-Theoretic Approach

Lower bounds on inherent worst-case computational complexity are notoriously easy to conjecture but hard to prove. One reason is the difficulty in finding sufficiently hard inputs for each different algorithm. The worst case for one algorithm might be expedited as a "special case" by another. If we can find inputs not susceptible to handling as special cases, then we might be able to convert our intuitions to proofs more easily.

One way to handle an input efficiently as a special case is to find and work with a much smaller description of the same input. We can prevent this sort of special handling if we provide inputs which are suitably incompressible. The work of Kolmogorov [9] and Solomonoff [16] shows how to make this precise in a robust way and that suitably incompressible streams of input data are abundant.

We have discovered that such an information-theoretic approach is particularly useful for proving lower bounds on the complexity of simulating abstract storage units on-line. The incompressibility forces simulators to use a lot of space and hence to spend a lot of time retrieving distant information. The approach is responsible for our main new result, Theorem 1 below.

The information-theoretic approach also serves as a rigorous, yet natural, tool equivalent to vague intuitions already in limited use. (In this sense, its value is analogous to that of nonstandard analysis.) These potentially valuable intuitions have not been cultivated much in the past, because conversion to rigorous proofs seemed so difficult. Rare successful conversions of this sort were performed by Hennie [7] and Aanderaa [1].

At certain points in their proofs, it seems that the argument should be able to proceed for any "typical" or "random" input sequence, but their proofs capture this intuition only with great effort, by counting aggregates of input sequences, essentially to show that not all can fail to be sufficiently "typical". Following Kolmogorov, the new approach in such situations is to look at a particular sequence which is random in the rigorous, domain-independent sense that it is incompressible. The effect is to remove obscuring domain-dependent counting from such proofs, and instead simply to cite the result of the one simple counting argument which shows that there are incompressible strings. The resulting simplification of Aanderaa's proof is presented as the proof of Theorem 2 below.

Descriptional Complexity [9]

We wish to define the descriptional complexity of a tuple $x$ of binary strings given another tuple $y$ of binary strings. We will use the symbol $\#$ to separate the components of our tuples. Any computable partial function $F: \{0,1,\#\}^* \rightarrow \{0,1,\#\}^*$ can be viewed as a relative description scheme, in terms of which one can define a relative descriptional complexity $K_F: \{0,1,\#\}^* \times \{0,1,\#\}^* \rightarrow \mathbb{N}$ by

$$K_F(x|y) = \min\{|d| : d \in \{0,1,\#\}^* \land F(dy) = x\}.$$

Because there is a "universal" computable partial function, there is some $F_0$ for which

$$\forall x \forall y K_{F_0}(x|y) \leq K_F(x|y) + c_F.$$

Except for an additive constant, therefore, $F_0$ is as succinct a relative description scheme as any; so we define the relative descriptional complexity $K: \{0,1,\#\}^* \times \{0,1,\#\}^* \rightarrow \mathbb{N}$ by $K(x|y) = K_{F_0}(x|y)$. We define $K(x)$ to be simply $K(x|\lambda)$, where $\lambda$ is the null string.

Since there are $2^n$ binary strings of length $n$ but only $2^n-1$ possible shorter descriptions $d$, we can be sure that $K(x) \leq |x|$ for some binary string $x$ of each length. Such strings are incompressible. For each $y$, similarly, $K(x|y) \leq |x|$ holds for some binary string $x$ of each length.
Abstract Storage Units

An abstract storage unit is an (infinite-state) "sequential machine" $M : \Sigma^* \times Q \rightarrow Q$ with finite input alphabet $\Sigma$ (commands), finite output alphabet $\Delta$ (responses), and internal state set $Q$ (command histories). A deterministic automaton (finite-state machine with access to some storage unit of its own) simulates an abstract storage unit (the "virtual" unit) if its input-output behavior matches that of the simulated storage unit. (Because each command's response must precede the next command, such simulations are said to be on-line. We consider only on-line simulations, so we choose to shorten "on-line simulation" to just "simulation".) If the simulating automaton requires at most $T(u;n)$ steps in the worst case to handle a sequence of $n$ commands following the initial command sequence $u$, then it simulates the storage unit in time $T(u;n)$. Real time means $T(u;1) = O(1)$ (bounded by a constant), and time $T(n)$ means time $T(n;1)$. We use the rule $S(u,vw) = S(u,v)S(uv,w)$ to extend storage unit $S$ to a function on $Q^* \times Q^*$. Two command histories $u, v \in Q^*$ are equivalent for $S$ if $S(u, v) = S(v, w)$ for every $w \in Q^*$. Let us note that the tape units under consideration here do qualify as abstract storage units. For a pushdown store, each input command is either "push 0", "push 1", "test for emptiness", or "pop", The corresponding output responses are 0, 1, whether the store is empty (0 for yes, 1 for no), and what symbol (0 or 1) gets popped (0 if there is nothing to pop), respectively. A counter is a pushdown store without the command "push 1". In either case, the corresponding output response indicates what symbol tape head number $i$ is left scanning after the command is "executed". We assume that initially such a tape unit has all heads coincident and the symbol 0 ("blank") written at every tape location and that there is at least one other symbol, 1, in the tape alphabet.

Note that several abstract storage units can be combined into one. The composite command alphabet is a disjoint union of the individual command alphabets.

An abstract storage unit with sufficiently atomic commands needs only a binary response alphabet $\Delta = \{0, 1\}$. Note that simulation of such a storage unit amounts to what is usually called an on-line language recognition problem, with 1 signalling "acceptance so far" and 0 signalling "rejection so far".

Inadequate Access to Multidimensional Tapes

Our first lemma demonstrates the effect of inadequate redundancy in a relatively limited-access representation of multiple-access data. For this lemma, it is enough to consider string data.

**Lemma 1.** Let $X$ be a set of $k$ strings $x(1), \ldots, x(k)$, each of length $m$, and let $x = x(1) \ldots x(k)$. Let $Y$ be a set of $k'$ strings, each of length $m'$. If $x$ is incompressible, then there is an $h$-tuple $(x_1, \ldots, x_h)$ of strings from $X$ such that $K(x_1, \ldots, x_h) > m/2$ for every $h$-tuple $(y_1, \ldots, y_h)$ of strings from $Y$, provided $m', m/(\log(k), \text{ and } (mk)/(m'k,h'/h))$ are large enough in terms of $h$ and $h'$.

**Corollary.** In addition, let the superset $Z$ of $Y$ be the set of all strings $z$ for which $K(z|y)$ is a small enough fraction of $m$ for some $y$ in $Y$. For the same $h$-tuple $(x_1, \ldots, x_h)$ as above, we still get $K(z_1, \ldots, z_h) > m/3$ for every $h$-tuple $(z_1, \ldots, z_h)$ of strings from $Z$.

**Proof of Lemma 1.** Suppose, to the contrary, that for every such $h$-tuple, there is such an $h'$-tuple such that $K(x_1, \ldots, x_h|y_1, \ldots, y_{h'}) < m'/4$. To reach a contradiction, we show that $K(x) < |x|$ if the parameters satisfy their constraints.

The number of $h$-tuples from $X$ is $k^h$, while the number of $h'$-tuples from $Y$ is only $k^{h'}$. Hence, there must be some one such $h$-tuple $(y_1, \ldots, y_{h'})$ which works for at least $p = k^{h'/h} > 1$ distinct such $h$-tuples. The number of distinct components of these $h$-tuples must be at least $q = \ldots$
For each sufficiently long command sequence \( u \), consider the balls of radius \( \tau \) centered at the simulator heads. Choose a parameter \( s \) and pick out \( k = \Theta(d'/s^d) \) disjoint subballs, each of radius \( s \). In each of these subballs, we store (in some canonical manner) a string of length \( m = \Theta(s^d) \), chosen so that some concatenation of these \( k \) strings is incompressible. Let \( X \) be the set of these \( k \) strings.

Lemma 2. For \( h' < h \), suppose an \( h' \)-head \( d' \)-dimensional tape unit with head-to-head jumps can simulate an \( h \)-single-head-tape \( d \)-dimensional storage unit in time \( T(u;r) \). For each sufficiently large \( r \), there is a command sequence \( u_0 \) of length \( r^d \) such that the following holds for every longer command sequence \( u \) equivalent to \( u_0 \):

Either

\[
T(\{u\}) \log T(\{u\}) = \Theta(\tau^{ch'/h'})
\]

or

\[
T(u;r) = O(\tau^{d(1-h'/h+d/d')/(d+1)(1-h'/h)})
\]

\[
(\log T(\{u\}))^d(\tau^{ch'/h})/(d+1)(1-h'/h))^{d'}
\]

The implicit multiplicative constants here depend only on \( d, h, d', h' \), and the size of the simulator's tape alphabet.

Proof. A similar lemma with conclusion \( T(u;r) = \Theta(d'/d') \) can be used in Hennie's lower bound argument for inadequate dimension \( (d' < d) \). In both cases, we use the initial sequence \( u_0 \) to write a "sufficiently incompressible ball" \( B \) of radius \( \tau/2 \) and to send all the virtual heads to its center. (We use the "head shift metric": \( B \) consists of those virtual tape positions within \( \tau/2 \) head shifts of its center.) In Hennie's case \( (d' < d) \), a simple volume argument suffices:

\[
T(u;r') \cdot \Theta(d') \quad \text{(the accessible representation volume)}
\]

has to be \( \Theta(d) \) \( (\text{the volume of } B) \). In the case of adequate dimension \( (d' = d) \), the argument has to be more subtle, since small radius in adequate dimension does give enough volume for a representation. On the other hand, small radius prevents much redundancy, provided \( d' \) is not too much larger than \( d \). It is this lack of redundancy that will create headaches for a simulator with an inadequate number of heads \( (h' < h) \). Lemma 1 above was designed to capture the effect of inadequate redundancy in a relatively limited-access representation of multiple-access data. Our proof will exploit that lemma.

To select our ball \( B \) of radius \( \tau/2 \), we choose a parameter \( s \) and pick out \( k = \Theta(d'/d') \) disjoint subballs, each of radius \( s \). In each of these subballs, we store (in some canonical manner) a string of length \( m = \Theta(s^d) \), chosen so that some concatenation of these \( k \) strings is incompressible. Let \( X \) be the set of these \( k \) strings.

Let \( u \) be any longer command sequence equivalent to \( u_0 \). In the representation by the simulator at the end of the initial command sequence \( u \), consider the balls of radius \( T(u;r) \) centered at the simulator heads. Choose a parameter \( t \) and cover these balls with \( \Theta(T(u;r)^{d'/d'}) \) balls of radius \( \Theta(t) \) such that each subball of radius \( t \) lies entirely within a member of the cover, and such that each tape square lies in at most \( O(1) \) different members of the cover. Combine pairs of cover members with nonblank volume less than \( t^{d'/d'} \) until at most one such cover member is left, reducing to
k' = \Theta(T(\|u\|) \log T(\|u\|)/d') cover members. (We were led to this economy by a weaker suggestion from M. Loui, A. Meyer, and M. Sipser.) For each uncombined member, select a depth-first listing (including shifts) of its contents. For combined members, list only the nonblank contents, with addresses. Let Y be the set of these k' strings, each unambiguously padded out to length m' = \Theta(t d'). As in the corollary to Lemma 1, let Z be the set of strings z for which, for some y in Y, K(z|y) is a small fraction of m. Note that this set includes a description of the contents (even including any head positions) of each member of the cover at any possible time within the next T(\|u\|) steps, provided T(\|u\|) is a sufficiently small fraction of m. (Include the sequence of at most T(\|u\|) writes, shifts, and jumps performed, and the relative location of each simulator head within distance T(\|u\|) of the member.)

We show now that T(\|u\|) = \Omega(\|u\|/s), provided s is a large enough multiple of T(\|u\|)^{1/4} (which does imply that T(\|u\|) is a small fraction of m = \Theta(s^d)) and t is a small enough fraction of \((r^d/(T(\|u\|) \log T(\|u\|))^h/h))\cdot\left(1-\frac{h'}{h}\right)). For there to be no such t \leq s, we would have to have T(\|u\|) \log T(\|u\|) = \Omega(t d^{h'/h}) already, so it is safe to assume there are such t.) If, in addition, t/s is large enough (no loss of generality, since the assertion T(\|u\|) = \Omega(t d^{h'/h}) is trivial otherwise), then it works out that m', m/\log(k), and \((mk)/(m'^{h'/h})\) are large enough for Lemma 1 and its corollary. In this case, let \(x_1, \ldots, x_m\) be the guaranteed h-tuple of strings from X.

Consider the following r commands:

\[ \lfloor r/2 \rfloor \text{ commands: Send the virtual heads to the subballs where } x_1, \ldots, x_{r/2} \text{ are stored.} \]
\[ \lceil r/2 \rceil \text{ commands: Repeatedly, in } O(s) \text{ commands, make an inquiry requiring more than } s \text{ simulator steps. If there were ever no such inquiry, then we could construct } x_1, \ldots, x_r \text{ from the simulator and its "radius-t instantaneous description" at that time. For some } h'-tuple (z_1, \ldots, z_{h'}), \text{ then, an upper bound for } K(x_1, \ldots, x_r|z_1, \ldots, z_{h'}) \text{ would depend only on the simulator. But this would contradict the corollary's assertion that } K(x_1, \ldots, x_r|z_1, \ldots, z_{h'}) \text{ exceeds } \]

\[ m/5, \text{ provided } t \text{ is so large that } m = \Theta(s^d) = \Omega(T(\|u\|)) = \Omega(r) \text{ is sufficiently large. It follows that } T(\|u\|) = \Omega(r/s). \]

We get our strongest conclusion above if we choose s as small as permitted (s = \Theta(T(\|u\|)^{1/d})) and t as large as permitted (t = \Theta(c^d/(T(\|u\|) \log T(\|u\|))^h/h)/\left(1-\frac{h'}{h}\right))). Solving T(\|u\|) = \Theta(r/s) for T(\|u\|) gives the desired lower bound. \(\square\)

**Theorem 1.** For \(h' < h\), suppose an \(h'\)-head d'-dimensional tape unit with head-to-head jumps can simulate an \(h\)-single-head-tape d-dimensional storage unit in time \(T(n)\). For each sufficiently large \(n\), \(T(n) = \Omega(n^{1+\epsilon}/\log(n^{1+\epsilon}))\), where \(\alpha = (d/d'-1)(1-h'/h)\) and \(\beta = (d/d')(h'/h)\).

In particular, \(T(n) = \Omega(n^{1+\epsilon})\) for some \(\epsilon > 0\) if \(d'<2\); and if \(d'\geq2\) and \(h' = h-1\), then \(T(n) = \Omega(n^{2+\epsilon})\) for any \(\epsilon < (1-1/d)/(d+h)\). The implicit multiplicative constants here depend only on \(d, h, d', h', \) and the size of the simulator's tape alphabet.

**Corollary.** Suppose that every d-dimensional tape unit can be simulated in time \(T(n)\) by an \(h'\)-head d'-dimensional tape unit with head-to-head jumps (\(h'\) fixed). Then \(T(n) = \Omega(n^{1+1/d'-\gamma})\) for every \(\gamma > 1/(d'+1/d)\). In particular, \(\gamma\) can be smaller than \(1/d\) if \(d' = 1\); and \(T(n) = \Omega(n^{1+\gamma})\) for every \(\gamma > 2/(d'+2)\) if \(d' = d\).

**Proof of Theorem 1.** For convenience, let \(F(r, T(\|u\|) \log T(\|u\|))\) be the long expression in Lemma 2. Without loss of generality, assume \(T(n)\) is polynomial in \(n\), so that the expression is \(\Theta(F(r, T(\|u\|) \log T(\|u\|)))\).

For any large enough \(n\), take \(r = \Theta(n^{1/d})\) with \(r^d < n/2\). Choose \(u_0\) as in Lemma 2, and inductively cite that lemma to obtain \(u_{l+1} = \Theta(r)\) long such that either \(T(u_0, \ldots, u_l) \log T(u_0, \ldots, u_l) = \Omega(r d^{h'/h}) = \Omega(n^{h'/h})\) or the simulator requires \(\Omega(F(r, T(\|u_0\|, \ldots, u_l| \log |u_0\|, \ldots, u_l|))\) steps to handle \(u_{l+1}\) following the initial command sequence.
While \( T(u_0 \cdots u_1) \geq T(n) \). Since \( h/h' > 1 + c \), we need only consider the case that the second alternative above always holds. Therefore, \( T(n) = T(u_0n|u_0|) \geq \Omega(n^{-1/d} F(n^{-1/d}, T(n) \log n)) \). Solving for \( T(n) \) gives the theorem. 

**Inadequate Access to One-Dimensional Tapes**

On a higher-dimensional tape, any part of a ball can be reached and queried in time which is small compared to the volume of the ball. This allows an allegedly efficient simulator little time to revise its representation of the ball, so that a nearly static representation will have to suffice. This simplification leads to the nonlinear lower bounds derived above, but it does not yield any nontrivial lower bounds for simulation of one-dimensional tapes. Our general information-theoretic approach, however, can be used to give a simplified derivation of Aanderaa's result for one-dimensional tapes.

Our argument needs only a specialized version of Aanderaa's "Overlap Lemma". Our version, Lemma 3 below, deals with one particular sequence rather than with many, so no averaging is involved. Although our proof of Lemma 3 amounts to a specialization of Aanderaa's more general proof, we include it here for completeness.

**Definition** [2, 12, 1]. Consider any sequence \( \ell_1, \ldots, \ell_n \). An overlap event in the subinterval \( I = [i, n] \) is a pair \( (i, j) \) with \( i, j \in I, i < j \), and \( \ell_i = \ell_j \notin [\ell_{i+1}, \ldots, \ell_{j-1}] \). If \( \omega_c(I) \) is the number of overlap events \( (1, j) \) in \( I \) with \( i \leq t < j \), then the internal overlap in \( I \), \( \alpha(I) \), is \( \max_{t \in I} \omega_c(I) \).

**Lemma 3.** For every \( N, r, s > 1 \) and every sequence of length \( r^N \), there is some subinterval \( I = [i, r^N] \) of length \( |I| = r^{s-1}N \). Solving for \( T(n) \) gives the theorem.

**Proof** [1]. We use the simple fact that the length of each subinterval exceeds the number of overlap events in it.

Parse the interval \([1, r^N]\) into \( r \) subintervals \( I_1, I_2, \ldots \) of length \( r^{s-1}N \), \( r^2 \) subintervals \( I_{11}, I_{12}, \ldots \) of length \( r^{s-2}N \), 

\[
\vdots
\]

\( r^s \) subintervals \( I_{11}, I_{22}, \ldots \) of length \( N \).

(Each parse is an \( r \)-fold refinement of the preceding one.) If the lemma fails, then the following holds for all \( i, j \):

\[
\omega(I_{ij}) > (1/s + 1/r) |I_{ij}| = (1/s + 1/r) r^{s-1}N.
\]

It follows that

\[
\sum_{i,j} \omega(I_{ij}) > \sum_{i,j} (1/s + 1/r) r^{s-1}N = (1/s + 1/r) r^N.
\]

Below we obtain a contradictory upper bound. For each \( i, j \), select \( t_{ij} \in I_{ij} \) so that \( \omega_c(I_{ij}) = \alpha(I_{ij}) \), thereby "distinguishing" the overlap events counted in \( \omega_c(I_{ij}) \) from \( \alpha(I_{ij}) \). By induction on \( i \), select a subset \( A \) of the subintervals \( I_{ij} \):

\( I_{ij} \in A \Leftrightarrow \) there is no subinterval \( I_{i',j} \in A \) with \( i' < i \) and \( t_{i',j} \in I_{ij} \). By design, all the distinguished overlap in the subintervals in \( A \) is disjoint; so \( \sum_{i,j} \omega(I_{ij}) \leq \sum_{i,j} \omega_c(I_{ij}) < r^N \). By induction on \( i \), \( A \) contains at most \( r_i \) subintervals \( I_{i,j} \), with \( i' < i \); so \( A \) contains at most \( r_i \) of the \( r_i \) subintervals \( I_{ij} \).

Noting that \( \alpha(I_{ij}) < |I_{ij}| = r^{s-1}N \), we conclude

\[
\sum_{i,j} \omega_c(I_{ij}) < \sum_{i,j} \omega_c(I_{ij}) < r^N \sum_{i,j} \omega_c(I_{ij}) < s r^N = N \cdot r^{s-1}N.
\]

Summing our two upper bounds, we get

\[
\sum_{i,j} \omega(I_{ij}) < r^N + sr^{s-1}N = (1/s + 1/r) r^N,
\]

the desired contradiction. 

Aanderaa's stated result is that fewer than \( h \) single-head one-dimensional tapes cannot simulate \( h \) such tapes, or even \( h \) pushdown stores, in real time. His argument actually proves the slightly stronger assertion of our Theorem 2 below. Without loss of generality, that theorem will consider only real-time simulators with three convenient constraints:

- binary tape alphabets,
- separate commands for reads, writes, and shifts,
- exactly the same number of steps to handle each virtual command (call this number the \textit{delay} of
Theorem 2. Consider simulation of \( h \) pushdown stores by single-head one-dimensional tapes. For each prospective real-time (\( h-1 \))-tape simulator \( M \) with delay \( c \), there is a virtual command sequence \( w \) on which \( M \) errs. Moreover, \( w \) need be no longer than some bound \( N_{x,c} \) depending only on \( h \) and \( c \).

Proof. The idea is to push incompressible data at \( h \) very different virtual rates, to find a virtual rate the prospective simulator neglects, and to use this neglect to get either an error or too short a description of the commands to the corresponding virtual store.

To make it clear which parameters below can depend on which others, we carefully order the assignment steps in our argument:

1. Consider delay-\( c \) real-time simulation of \( h \) pushdown stores.
2. Choose relative-push-density factor \( d \) large enough for the analysis below. Set \( S = \{ x_1, x_2, \ldots, x_h \} \), where \( x_1 \) is a string of \( d \) push commands to virtual store number 1 \( i \), and let \( \theta = d + d^2 + \ldots + d^h \) be the length of each string in \( S \).
3. Choose "overlap fraction" \( \varepsilon > 0 \) small enough for the analysis below.
4. Choose \( N \) large enough for the analysis below, and divisible by \( ch \theta \) for convenience, intending to take \( N_{x,c} = 2N^2 + 1 \).
5. Consider any prospective real-time (\( h-1 \))-tape simulator \( M \) with delay \( c \).
6. Take \( x \in S^* \) with \( \kappa(x | \mathcal{H}) \leq |x| = N^2 \).

Assuming for the sake of argument that no prefix of \( x \) is vulnerable to a sequence of subsequent pop commands to some virtual store, we will contradict the incompressibility of \( x \). This will let us conclude that \( M \) errs on some command sequence \( w = yz \), where \( y \) is a prefix of \( x \) and \( z \) is a sequence of pop commands to some virtual store, with \( |z| \leq |y| \), so that \( |yz| \leq 2|y| \leq 2|x| \leq 2N^2 < N_{x,c} \).

On virtual command sequence \( x \), \( M \) computes for \( cN^2 \) steps. Consider the corresponding sequence \( i_1, i_2, \ldots, i_{cN^2} \) of most recently accessed storage locations. Choose a nonnull subinterval \( I = [1, ch^2] \) with \( |I| \) divisible by \( N \) and \( m(I) \leq \varepsilon |I| \). By Lemma 3, this is possible if \( ch \geq r^3 \) for \( r = s = 2/\varepsilon \). (Recall that we are allowed to choose \( N \) large in terms of \( \varepsilon \).) It is within time interval \( I \) that we will consider the rates at which the prospective simulator's heads move. Low overlap on the prospective simulator's one-dimensional tapes will force the heads to go farther and farther from the data they have recorded.

Once we have \( I \), we parse it into subintervals \( I_1, \ldots, I_n \) of length \( |I|/h \). In each of these subintervals, the number of commands to virtual store number \( i \) is \( n_i = (d/\theta)|I|/(ch) \). Because all the data pushed onto virtual store number \( i \) in \( I \) can be retrieved in \( n_i \) subsequent pop commands to that store, \( M \) will have to be able to retrieve that same data without access to any tape square farther than \( chn_i \) from the head positions at the end of \( I \). Say that \( M \) "neglects" virtual store number \( i \) in \( I \) if each head that ranges farther than \( (ch/d)n_i \) in \( I_i \) ranges farther than \( chn_i \) in the concatenation of the subsequent subintervals (call this concatenation \( I_{>i} \)).

To see that \( M \) does neglect some virtual store in \( I \), suppose it does not. For each \( i' \) \( (1 \leq i \leq h) \), then, let \( j(i) \) be a head that ranges farther than \( (ch/d)n_i \) in \( I_i \) but no farther than \( chn_i \) in \( I_{>i} \). One of \( M \)'s \( h-1 \) heads must serve as both \( j(i) \) and \( j(i') \) for \( i < i' \). But then that overworked head must range no farther than \( chn_i \) in \( I_{>i} \), yet farther than \( (ch/d)n_i \), \( \geq (ch/d)n_i \) \( \leq chn_i \) in \( I_{>i} \) alone, a contradiction.

Assume that \( M \) neglects virtual store number \( i \) in \( I \). Because \( m(I) \leq \varepsilon |I| \), each head that visits more than \( (ch/d)n_i \) tape squares in \( I_i \) revisits at most \( \varepsilon |I| \) of them in \( I_{>i} \) and ends up at least \( chn_i - 26|I| \) tape squares away from them. (It is safe to assume \( chn_i > (ch/d)n_i \geq (d-1/\theta)|I| \geq (1/\theta)|I| > 26|I| \), since we are allowed to choose \( \varepsilon \) small in terms of \( \theta \).) Even in the next \( chn_i \) steps by \( M \), therefore, no more than \( 26|I| \) of the tape squares can be revisited. It follows that the \( n_i \) bits pushed onto virtual store number \( i \) in \( I_i \) can be recovered from \( M \), \( O(h((ch/d)n_i + \varepsilon |I|)) \) bits of its instantaneous description at the end of \( I_i \), and the rest of \( x \).
If we provide the sequence of bits pushed by the latter literally, along with \( i \), the location and length of \( I \), and an appropriate formalization of this whole discussion (much as in the proof of Lemma 1), then we get

\[
K(x|M) \leq O((\log |x|) + O(1 + h + c + d)) + (|x| - n_i) + O(1 + h + c + d).
\]

Substituting \( |x| = \log |x| \), we get

\[
K(x|M) \leq |x| + O(\log |x|) + O(1 + h + c + d)
\]

since we are allowed to choose \( \delta \) small in terms of \( \delta \), and \( d \) large in terms of \( c \) and \( h \). Substituting \( n_i = (\delta^4/\delta)|I|/(\log N/\delta) = d|x|^1/(\log \delta) \), we finally get

\[
K(x|M) \leq |x| + O(\log |x|) + O(1 + h + c + d)
\]

since we are allowed to choose \( |x| = N^2 \) large in terms of \( c \), \( h \), \( d \), and \( \delta \).  

**Corollary.** Even an \((h-1)\)-head tape with head-to-head jumps cannot simulate \( h \) pushdown stores in real time.

**Proof.** By induction on \( h' < h \), we prove that an \( h' \)-head tape with head-to-head jumps cannot simulate \( h \) pushdown stores in real time. The base case, \( h' = 1 \), follows trivially from Theorem 2.

In the induction case, the idea is to drive the \( h' \) heads of any simulated \( M \) with delay \( c \) farther apart than \( N_{h,c} \), with the next head-to-head jump at least \( N_{h,c} \) virtual commands (\( cN_{h,c} \) steps) away, and then to cite Theorem 2 as if the \( h' \) heads were on separate tapes. Note that the initial tape contents within distance \( cN_{h,c} \) of each head can be managed in finite-state control, and that this does not change the delay \( c \).

In the only case omitted above, every virtual command sequence leaves a pair of \( M \)'s heads within distance \( cN_{h,c} \) or has an extension shorter than \( N_{h,c} \) which causes a head-to-head jump. But then there is a real-time machine \( M' \), described below, which correctly simulates the \( h \) pushdown stores if \( M \) does, and which has only \( h' + 1 \) heads. By the induction hypothesis \( M' \) errs, so \( M \) errs too.

We design \( M' \) to simulate \( M \) if \( M \) correctly simulates the \( h \) pushdown stores. While \( M \) has a pair of heads within distance \( c + cN_{h,c} \), \( M' \) stations a head at one of the positions and commits the relative position of the other to finite-state memory. In this case \( M' \) is able to handle the next virtual command just as \( M \) would, but with a longer (though still bounded) delay. When the pair of heads gets too far apart and some other pair of heads is within distance \( c + cN_{h,c} \) (checkable within bounded delay), \( M' \) orders one of the heads in the new pair to jump to the head which was serving the old pair and then to shift to its new post nearby. When the pair of heads gets too far apart and no other pair of heads is within distance \( c + cN_{h,c} \), there must be a sequence of at most \( N_{h,c} \) additional commands which causes a head-to-head jump by \( M \). Still within real-time and temporarily suppressing all output, \( M' \) can find a shortest such sequence, simulate \( M \) on it, and then simulate \( M \) on an equally long sequence of commands which undoes the virtual damage, pushing what the first sequence popped and popping what it pushed. Before the repair, this leaves a pair of heads within distance \( c \). Even after the repair, therefore, it leaves a pair of heads within distance \( c + cN_{h,c} \), so that the simulation can continue.

**Two-Tape Simulation of Multiple Tapes**

The simulations we use and design in this section are particularly well-structured in the following sense: The simulators' tape heads return to the origins on their tapes at the completion of work on each input command. Call this sort of simulation \( H \)-simulation ("H" for "homing"). The value of \( H \)-simulation is that several \( H \)-simulations can interleave use of (different tracks on) the same tape units without interference or time loss. Although real-time \( H \)-simulation is just finite-state transduction, there are nontrivial \( H \)-simulations which do not run in real time.

**Lemma 4** [4, pp. 276-277]. A counter can be \( H \)-simulated in linear \( (O(n)) \) time by a single-head one-dimensional tape.

**Corollary.** Any fixed number of counters can be \( H \)-simulated in linear time by a single-head one-dimensional tape.

**Lemma 5** [8]. A single-head one-dimensional tape can be \( H \)-simulated in time \( O(n \log n) \) by a pair of single-head one-dimensional tapes. (This
Corollary. Any fixed number of one-dimensional tapes, even with multiple heads \([5, 11]\) and head-to-head jumps \([10]\), can be H-simulated in time \(O(n \log n)\) by a pair of single-head one-dimensional tapes.

Theorem 3. For \(d=2\), any multihead \(d\)-dimensional tape, even with head-to-head jumps, can be H-simulated by a pair of single-head tapes, one \(d\)-dimensional and the other one-dimensional, in time \(O(n^{1+1/d-\alpha})\), where \(\alpha = 1/(d(d-1)+1)\).

Proof. To get a representation in small radius, the H-simulator strategy will be to pack non-blank "pages" of virtual storage compactly onto a "secondary storage" track of the \(d\)-dimensional tape. To get by efficiently with a single head on that tape, the strategy will be to copy "active" pages into the vicinity of the origin on a "primary storage" track of the same tape. The major problem will be to find the right pages in secondary storage fast enough. Our solution will involve bounded-depth recursion of the entire H-simulation.

Inductively, we will describe a sequence of H-simulation procedures, each of which "recurses one level deeper". Each procedure will use a pair of single-head tapes, one \(d\)-dimensional and the other one-dimensional, to H-simulate the virtual \(d\)-dimensional tape. For each \(k\), there will be a constant \(c_k\) such that the time \(T_k(n)\) for H-simulation procedure \(k\) will satisfy \(T_k(n) \leq c_k U_k(n)\) for \(U_k(n) = n^{1+1/d-\alpha} + (n^2)^k\) and \(\delta = 1-1/d^2\). It will follow that \(T_k(n) = O(n^{1+1/d-\alpha})\) for any sufficiently large \(k\), as desired.

Let H-simulation procedure 0 be a naive one, with \(T_0(n) = O(n^3)\), say. (A pair of one-dimensional tapes suffices.) For \(k > 0\), the key to H-simulation procedure \(k\) will be a procedure \(SIM_k(n)\) to H-simulate the first \(n\) virtual commands in total time \(O(U_k(n))\) when \(n\) is provided as an auxiliary read-only off-line input (in unary notation, say). H-simulation procedure \(k\) will then be something like

\[
\text{for } i = 1, 2, 3, \ldots \text{ do } [SIM_k(2^i); \text{ erase tapes}].
\]

To provide the repetitious input this would require, and to properly screen the repetitious output it would generate, we include an input manager and an output manager. To make erasing easy, we modify \(SIM_k(2^i)\) to specially mark each square it visits. Erasure can then be achieved during depth-first traversal of the connected graph formed by the marked squares on each tape; each marked tape square is erased the last time it is visited.

Through virtual command \(n\), the input and output managers and the loop control require only \(O(2 + 2^2 + \ldots + 2^{[\log_2 n]} = O(n)\) commands to a fixed finite set of one-dimensional tape units. These commands can be H-simulated by the Hennie-Stearns procedure (Lemma 5 and its corollary) in time \(O(n \log n)\) on tracks of the two tapes actually available. The time for \(SIM_k(2^i)\) and the subsequent erasure is \(O(U_k(2^i))\), so the total time to H-simulate the first \(n\) virtual commands (for \(n\) now) is

\[
O(n \log n + U_k(2) + U_k(2^2) + \ldots + U_k(2^{[\log_2 n]})) = O(U_k(n))\), as desired.
\]

It remains only to describe and analyze \(SIM_k(n)\). It is in this procedure that the main tracks on the \(d\)-dimensional tape will be one for "primary storage" and one for "secondary storage". Tracks of the one-dimensional tape will be used for (linear-time H-simulations of) counters (Lemma 4 and its corollary) and for assistance in copying "pages" of virtual storage between primary and secondary storage on the \(d\)-dimensional tape. In addition, on separate tracks of its tapes, \(SIM_k(n)\) will make use of the inductively available H-simulation procedure \(k-1\).

Each page of virtual storage will be a \(d\)-dimensional cube of \(b^d\) tape squares, where \(b = \sqrt[4]{n(\log n)/(dd-1)+1}\). To H-simulate the first \(n\) virtual commands, \(SIM_k(n)\) will Hsimulate \(\lceil n/b \rceil\) time intervals, each one (except possibly the last) \(b\) virtual commands long. At the beginning of each such time interval, the \(3^d\) pages nearest each virtual head position will be found and loaded into primary storage, around the origin. The next \(b\) virtual commands can then be H-simulated directly in primary storage without any virtual head leaving the \(O(3^d)\) pages there. At the end of the time interval, the \(O(3^d)\) pages will be copied back to their locations in secondary storage. Those pages not yet assigned locations will be assigned vacant locations as
close as possible to the origin in secondary storage, regardless of their virtual neighbors' assigned locations. Only pages which have ever been loaded into primary storage will be assigned locations and stored in secondary storage, all other virtual pages being implicitly blank. When a page is loaded into primary storage, it is copied from secondary storage if it has a location there, and set up entirely blank otherwise.

So that pages' locations in secondary storage can be found, some sort of an "index" will have to be maintained. For this purpose, we use procedure \(k - 1\) to \(H\)-simulate a scaled down "map" of the \(d\)-dimensional virtual tape unit. Each virtual page (on a side) is represented by a \(d\)-dimensional cube just big enough to hold the location in secondary storage of the virtual page. The map heads are kept near the representatives of the pages currently scanned by the virtual heads.

If the components of each virtual page's location are stored in binary, then each representative above has to be only \(O(\log(n/b)^{1/d})\) on a side. In time proportional to the components' values, \(O(n/b)^{1/d}\), they can be converted from and to their unary representations in the counters \(H\)-simulated on the one-dimensional tape.

Note that some care is required to copy a page between primary and secondary storage. The page has to be copied onto and from the one-dimensional tape (in row major order, say). Since both tape heads are involved, the linear-time counter \(H\)-simulations on the one-dimensional tape are not available to signal the ends of rows, etc. A simple solution is to use those counters ahead of time to prepare a "form" to copy onto and from on the one-dimensional tape.

The following activities account for all \(H\)-simulator time:

1. \(O(1)\) commands to counters for each other \(H\)-simulator step counted below.

2. Initial calculation of such constants as \(b, n/b, \) and \(\log(n/b)\) (\(O(n)\) \(H\)-simulator steps).

3. \(O((n/b) \log(n/b)) = O(n^{1-1/(d(d-1)+1)})\) \(\log n\) = \(O(n^{\delta})\) commands to \(H\)-simulation procedure \(k - 1\) (\(T_{k-1}(O(n^{\delta}))\) \(H\)-simulator steps).

4. Conversion of page location components to and from binary (\(O((n/b)(n/b)^{1/d})\) \(H\)-simulator steps).

(5) Shifting to and from pages in secondary storage (\(O((n/b)(n/b)^{1/d}b)\) \(H\)-simulator steps).

(6) Copying between primary and secondary storage (\(O((n/b)b^{\delta})\) \(H\)-simulator steps).

(7) Direct \(H\)-simulation of virtual commands in primary storage (\(O((n/b)b^{\delta})\) \(H\)-simulator steps). Therefore, the total time for \(SIM(n)\) is

\[
0(\sum_{k=1}^{n} 1 + 1/d - \alpha + \delta + \sum_{k=1}^{n} 1 + \log n)
\]

as required.

Remaining Questions

Consider simulation of \(h\) single-head \(d\)-dimensional tapes by an \(h'\)-head \(d'\)-dimensional tape. Probably only for \(h' \geq h\) and \(d' \geq d\) is simulation possible in real time. Hennie's result [7, 6] leaves open only cases with \(h' < h\). We handle those cases with \(h' < h\) and \(d' < d\) (Theorem 1 above), and Aanderaa handles those cases with \(h' < h\) and \(d' = d' = 1\) (Theorem 2 above). The remaining, still open cases have \(h' < h\) but \(d' \geq max(d^2, 2)\).

Handling cases with \(h' < h\) and \(d' = d' = 1\) above might involve generalizing Aanderaa's proof (Theorem 2). The problem is that low overlap seems less helpful on a higher-dimensional tape. It might help, however, to consider cases with \(h'\) much smaller than \(h\), so that some simulator head has to handle many different virtual head rates. Perhaps it would help to obtain a time interval in which all simulator head motion is "essentially linear", with each overlap event spanning only a short time interval.

To further pursue the question of what is needed to compensate for inadequate access into storage, one can consider even more exotic tapes (tree-shaped, for example) for the simulator heads, or even more restricted tapes for the virtual heads. For the proof of Theorem 2, the pushdown stores' \(h\) virtual heads need only one "turn" (switch from pushing to popping, or vice versa) among them; but we do not know whether this is enough for the corollary. A more drastic restriction would replace the \(h\) pushdown stores by \(h\) counters. In this case, a one-head simulation is possible in linear time [4]; so the remaining ques-
tion concerns real-time simulation. Note that the related "origin-crossing problem" is solvable in real time, by a one-head Turing machine [3].

Aanderaa's proof shows that the prospective simulator gets "caught off base" at some time when there is low overlap. Our intuition suggests that the prospective simulator can be caught off base at practically any point in the incompressible command sequence we use, and that the overlap lemma (Lemma 3 above) is not even needed. If the proof can thus be freed from reliance on that lemma, it might become easier to adapt it for other purposes. For example, we might be able to consider virtual queues (first in, first out) rather than just pushdown stores (last in, first out), or we might be able to show that simulation is not possible even in linear time. (Without the real-time assumption, there might not happen to be enough virtual commands in the low-overlap time interval.)

References