stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 246/83 DECEMBER
(DEPARTMENT OF COMPUTER SCIENCE)

P.M.B. VITÁNYI

ON TWO-TAPE REAL-TIME COMPUTATION AND QUEUES

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, Kruislaan 413, Amsterdam, The Netherlands.

The Mathematical Centre, founded 11 February 1946, is a non-profit institution for the promotion of pure and applied mathematics and computer science. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics subject classification: 68C40, 68C25, 68G10
1982 CR. Categories F.1, F.2.2, F.2.3

ON TWO-TAPE REAL-TIME COMPUTATION AND QUEUES*

by

Paul M.B. Vitányi

ABSTRACT

A Turing machine with two storage tapes can not simulate a queue in both real-time and with at least one storage tape head always within $o(n)$ squares from the start square. This fact may be useful for showing that a two-head tape unit is more powerful in real-time than two one-head tape units, as is commonly conjectured.

Keywords & Phrases: multitape Turing machine, multihead Turing machine, real-time computation, two heads versus two tapes, storage retrieval, queue, incompressible string, Kolmogorov complexity

* This work is a revised and up-to-date version of [Vi2]. It will be published elsewhere.

1. Introduction.

In the world of Turing machine like devices real-time computation is the nearest analogon to real-time computation in concrete computer systems. To compare the relative computation power of two storage devices a fine distinction can be made by the capabilities in real-time. Thus, [Ra] showed that two one-head tape units are more powerful in real-time than one such unit. Later [Aa] generalized this by demonstrating superiority of $k+1$ one-head tape units over k such units, in real-time. In [PSS] a new information-theoretic argument was introduced to strip the proof in [Aa] down to its essentials, while [Pa2] strenghtened the result by exploiting the techniques further. Moreover, it was shown that considering multihead tape units resulted in an similar real-time hierarchy [PSS, Vi1]. Thus, it appeared, adding a head in general increases computing power. But does it also make a difference whether the heads reside on the same tape? The advantage obtained by placing the heads on the same tape is that they can read each others writing. Nonetheless, showing that it is impossible to overcome this advantage by clever programming, as seems very likely at this time, appears to be a hard nut to crack. The present paper tries to provide an initial step for showing that a two-head tape unit is more powerful in real-time than two one-head tape units. It is shown that two one-head tape units can not both simulate a queue in real-time and keep the minimum of the distances of the scanned tape squares to the start squares of size $o(n)$. That is, information which has initially been stored near the start squares may have to be transported to a disjoint set of tape squares (while also new incoming data has to be stored) to be real-time reproducible in first-in-first-out order. A two-head tape unit can trivially simulate a queue in real-time while not having to shift data at all. In one form or the other this subject has received attention before. It has been shown that k-head tape units can be simulated by k one-head tape units in linear time [St] and by $(4k-4)$ one-head tape units in real-time [LS] (improving an earlier result of [FMR]). For $d>1$ a two-head d-dimensional tape unit can be simulated in real-time by 3 d-dimensional and some 1-dimensional tape units, all with one head. For $d>1$ and $k>2$ a k-head d-dimensional tape unit can be simulated in real-time by $O(h^3)$ d-dimensional and $O(h^3 d)$ 1-dimensional tape units, all with one head [LS]. For $d>1$, the fastest on-line simulation of a $(k+1)$-head d-dimensional tape unit by a k-head d-dimensional tape unit requires nonlinear time [PSS] and similarly for $d=1$ by [Pa2]. For 2-dimensional tape units it was shown in [Pa1] that 2 heads on a single tape are more powerful in real-time than 2 tapes with a single head each. The same question for 1-dimensional tapes has remained unresolved. For relevant definitions of the used concepts as Turing machines, on-line simulation, real-time (simulation), and so on, see e.g. [FMR]. We consider only the storage structure of the machines. So a k-tape Turing machine has k single-head storage tapes attached to its finite control, apart from input- and output tape (-terminal). A one-head tape unit is a 1-tape Turing machine. Two one-head tape units form a 2-tape Turing machine. A two-head tape unit is a Turing machine with a single two-headed storage tape. We use the terminology interchangeably.

The present account relates how far I got with the two-heads versus two tapes real-time problem by 1980. A preliminary and cumbersome version appeared as [Vi2]. Meanwhile, new techniques for such problems, based on Kolmogorov complexity, were introduced in [PSS] and further exploited in [Pa1, Pa2] and elsewhere. Here we give a streamlined account of the contents of [Vi2] using the "incompressible string" arguments of [PSS].

2. Kolmogorov Complexity

The ideas on descriptional complexity below were developed independently by Kolmogorov [Ko] and Chaitin [Ch]. We closely follow the discussion in [PSS]. Consider the problem of describing a vector $\bar{x}$ of strings x_i over 0's and 1's. The string entries of the vector can be separated by ¢'s so that the vector is a string too. That is, $\bar{x} \in \{0,1,¢\}^*$. Any computable function f from vectors of strings over 0's and 1's to such vectors, together with a vector $\bar{y}$, such that $f(\bar{y}) = \bar{x}$, is such a description.

A descriptional complexity K_f of $\bar{x}$, relative to f and $\bar{y}$, is defined by

$$K_f(\bar{x} \mid \bar{y}) = \min\{ \mid d \mid \ \mid d \in \{0,1\}^* \ \& \ f(d ¢ \bar{y}) = \bar{x} \} \ .$$

For the *universal* computable partial function f_0 we have that, for all vectors $\bar{x},\bar{y}$, $K_{f_0}(\bar{x} \mid \bar{y}) \leqslant K_f(\bar{x} \mid \bar{y}) + c_f$, for all f with appropriate constant c_f. So the canonical relative descriptional complexity $K(\bar{x},\bar{y})$ can be set equal to $K_{f_0}(\bar{x} \mid \bar{y})$. Define the *descriptional complexity* of $\bar{x}$ as $K(\bar{x}) = K(\bar{x} \mid \lambda)$. ($\lambda$ denotes the empty string, so as not to overuse the symbol ϵ in the sequel). Since there are 2^n binary strings of length n, but only $2^n - 1$ possible shorter descriptions d, it follows that $K(x) \geqslant \mid x \mid$ for some binary string x of each length. Following [PSS] further, we call such strings *incompressible*. It also follows that $K(x \mid y) \geqslant \mid x \mid$ for some binary string x of each length. Since similarly $K(x) \geqslant (1-\delta) \mid x \mid$ for $2^{\delta \mid x \mid}$ strings over $\{0,1\}$, which thus cannot be compressed to less than $(1-\delta) \mid x \mid$ bits, such "nearly" incompressible strings are abundant. Note that a string $x = uvw$ can be specified by v, $\mid x \mid$, $\mid u \mid$ and the bits of uw. Thus,

$$K(x) \leqslant K(v) + O(\log \mid x \mid) + \mid uw \mid \ ,$$

so that with $K(x) \geqslant (1-\delta) \mid x \mid$ we obtain

$$K(v) \geqslant \mid v \mid - \delta \mid x \mid - O(\log \mid x \mid) \ .$$

3. Outline of the approach

Without loss of generality, we assume that all tape units below have semi-infinite tapes. That is, the squares of the tapes can be enumerated from left to right by the natural numbers. The 0th square is called the *start* square. Assume further, also without loss of generality, that the tape units write only 0's and 1's in the storage squares and relax the *real-time* requirement to *constant delay*. A computation is of *constant delay* if there is a fixed constant c such that there are at most c computation steps in between processing the nth and the $(n+1)$th input symbol, for all n. Thus, constant delay with $c=1$ is the same as real-time, and it is not difficult to see that each computation of constant delay can be sped up to a real-time computation by expanding the storage alphabet and the size of the finite control. In order to show that two single head tape units, which real-time simulate a first-in-first-out storage device like a queue, will be forced to continuously transport earlier stored data, we concentrate on a subproblem. Consider the real-time recognition of

$$L = \{ x ?y \mid x,y,z \in \{0,1\}^* \& \ x = yz \} \ .$$

The subset L' of L, defined as L with $y = x$ and z the empty string, can not be recognized in real-time by a one-head tape unit [Va]. We show that, although L' itself can be recognized by two one-head tape units in real-time [FMR, Vi2], if two one-head tape units accept L in real-time then the worst-case space on both tapes must be linear. Exploiting that fact it is shown that two one-head tape units, recognizing L in both real-time and sublinear worst-case closest (to the start square) head position, are forced to traverse a linear length tape segment on one or the other tape so often, while keeping the other head relatively immobile, that they can be fooled. This fooling of the machine is due to the fact that the description it has to record, while on the subject tape segment, may not be compressible to fit the sum of the available storage and the maximal amount of data which can have been exported out. The idea here is not the bottleneck strategy as in [Ra, Va] but rather an extended form of crossing sequence argument. We use the mnemonic Order-of-Magnitude symbols as follows.

$f(n) \in O(g(n))$ if there exist positive constants c and n_0 such that $\mid f(n) \mid \leqslant cg(n)$ for *all* $n > n_0$.

$f(n) \in \Omega(g(n))$ if there are positive constants c and n_0 such that $f(n) \geqslant cg(n)$ for *all* $n > n_0$.

$f(n) \in o(g(n))$ if $\lim_{n \to \infty} f(n)/g(n) = 0$.

4. Driving both heads simultaneously far away

First we show that both tape units have to be used equally extensive in the process of real-time recognizing L as defined above.

Definition. If a two-tape Turing machine M has input x then the *work space* $t_i(x)$ of M on x is the segment of tape squares on tape i $(i=1,2)$, covered by the motion of M, while having input sequence x. The *worst-case best-tape* space $m(n)$ of M on $\{0,1\}^*$ is defined as

$$m(n) = \max_{|x|=n} \min_{i=1,2} \{t_i(x) \mid x \in \{0,1\}^*\} .$$

Below we make extensive use of *crossing* sequences. For each one-head tape unit, contained in a larger machine, we assume that, when it makes a move, it first overprints the symbol scanned and performs all necessary changes in the remainder of the machine and only then moves its storage head. Thus, for any pair of adjacent tape squares, we can list the sequence of machine descriptions, apart from the tape unit concerned, in which the unit *crosses* from one square to the other. The first crossing is from left to right; after that crossings alternate in direction. The sequence of partial machine descriptions so related to an intersquare boundary, or square, is called a crossing sequence. Early use of such sequences can be found in [Ra].

Lemma 1. *A two-tape Turing machine accepting L in real-time has a worst-case best-tape space $m(n) \in \Omega(n)$ on $\{0,1\}^*$.*

Proof. Let M be a real-time two-tape Turing machine accepting L and contradicting the Lemma. Without loss of generality M has semi-infinite tapes, writes only 0's and 1's on its storage tapes and operates with finite delay c. Now consider a sufficiently long incompressible string $x \in \{0,1\}^*$ of length n. Let n be divisible by $9c$ to simplify the calculation. By the contradictory assumption $m(n) \notin \Omega(n)$, and let for the chosen x and n the value $m(n)$ be a very small fraction of n. We use the machine M to obtain an impossibly compressed description of x. To store x, M has to use $n-c_M$ work tape squares, with c_M a fixed constant depending only on M, by the incompressibility of x. So M will need to write on at least $c+n/3$ distinct work tape squares on the tape of which it uses most squares. Let this be the first tape. Let $x=uvw$, where $|u|=n/(9c)$ and uv is the smallest prefix which drives M's first work tape head at least $n/3$ squares from the start square.

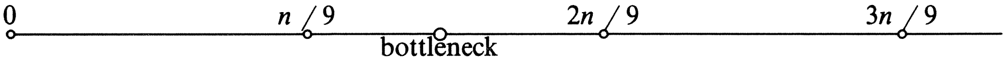

Figure 1. The most heavily used (e.g. first) worktape with the initial three $n/9$-size consecutive blocks. The bottleneck square is indicated in the middle block.

Consider M's computation on uv. Since we can divide the first worktape in at least 3 consecutive blocks of $n/9$ squares, justified from the start square, we can reason as follows. Before the head on the first worktape reaches the second block of $n/9$ squares, all of u has been read. The head on the first worktape reaches the end of the third block of $n/9$ squares at the end of reading uv. Since the total computation on x takes no more than cn steps, not all squares of the second block of $n/9$ squares on the first work tape have been crossed more than $9c$ times. Call one least crossed such square the *bottleneck*. By the contradictory assumption only $m(n)$ squares on the second work tape are used. So the following must constitute a description of x. (Logarithms are taken with base 2 unless otherwise indicated.)

u. A description of u in terms of M's operation:

- a description of M of binary length c_M;
- a description of this discussion of binary length c_D;
- the binary $\log n$ length value of n;
- the location of the *bottleneck* square in the second block of $n/9$ squares on the first work tape in no more than $\log n$ bits;
- the crossing sequence at the bottleneck.

vw. The literal description of vw, that is, $n - |u|$ bits.

The crossing sequence at the bottleneck consists of a sequence of crossing states with attached times of crossing. A crossing state consists of the state of the finite control of M (M has s states) together with the contents of the second work tape and the head position on the second work tape. Thus there are not more than

$$(s 2^{m(n)} \log m(n) \log cn)^{9c}$$

such crossing sequences, which can be represented by

$$9c(m(n) + \log\log m(n) + \log\log cn + \log s)$$

bits. To recover u, determine enough of M's instantaneous description after reading uv to try each input continuation ?y with $|y| = n/(9c)$. The tapecontents on the first storage tape left of the bottleneck will not be needed, since they are not scanned in processing ?y. Hence, summing the total binary length of the obtained description of x we have

$$n - n/(9c) + 9cm(n) + o(n) \ ,$$

contradicting the choice of x as an incompressible string. Consequently, the contradictory assumption is shown false and the Lemma true. □ □

It is not difficult to see that the above proof also supports:

Lemma 2. *For a two-tape Turing machine which accepts L in real-time there exists a constants $\epsilon, \delta > 0$ such that for each "nearly" incompressible string $x \in \{0,1\}^*$ with $K(x) \geq (1-\delta)|x|$ as initial input, each head must range farther than $\epsilon |x|$.*

Definition. The *position* of a head on a semi-infinite tape is the number of the scanned square. In a two-tape Turing machine M the *closest-head* position is the least of the two. The *worst-case* closest-head position of M on $\{0,1\}^*$ is

$$p(n) = \max_{|x|=n} \{\text{closest}-\text{head position while processing } x \in \{0,1\}^* \} \ .$$

Theorem. *A two-tape Turing machine accepting L in real-time has a worst-case closest-head position $p(n) \in \Omega(n)$ on $\{0,1\}^*$.*

Proof. Let M be a real-time two-tape Turing machine accepting L and contradicting the Lemma. Without loss of generality M has semi-infinite tapes, writes only 0's and 1's on its storage tapes and operates with finite delay c. Let ϵ be as in Lemma 2, and choose $0 < \epsilon_1 << \epsilon_2 << \epsilon$. Let x be a particular incompressible string over $\{0,1\}$ with $|x| = n$ large and $p(n) < \epsilon_1 n$. That is, M has always a head within distance $\epsilon_1 n$ of a start square while processing x. Nonetheless, for each t ($t_0 < t \leq n$ for some $t_0 > 0$) each head must have ranged farther than ϵt by the time M processes the tth input bit, by Lemma 2. By choice of the values of the ϵ's this combination of requirements forces repeated traversals of the tape segments $[\epsilon_1 n, \epsilon_2 n]$ on both tapes. Call these segments S_1 and S_2. By input symbol $\epsilon_2 n / \epsilon$ both S_1 and S_2 have been traversed from left to right. Yet no head has ranged farther than $c\epsilon_2 n / \epsilon$ squares from its start square. Call the corresponding limit squares L_1. One head, say on tape i_1, must now be within $\epsilon_1 n$ squares of its start square, that is, left of its S-segment.

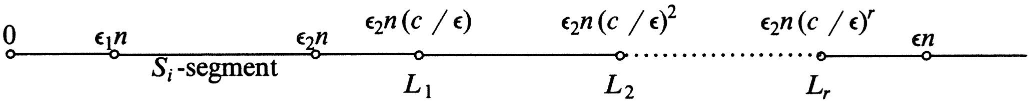
0
$\epsilon_1 n$
$\epsilon_2 n$
$\epsilon_2 n(c/\epsilon)$
$\epsilon_2 n(c/\epsilon)^2$
$\epsilon_2 n(c/\epsilon)^r$
ϵn
S_i-segment
L_1
L_2
L_r

Figure 2. Tape i $(i=1,2)$ with S_i-segment and range limits $L_1, \ldots, L_r$. Processing x the head eventually crosses square ϵn.

By input symbol $(c/\epsilon)(\epsilon_2 n/\epsilon)$ both heads must have ranged farther than L_1, and therefore the head on tape i_1 must have traversed its S-segment once more. Yet no head has ranged farther than $(c^2/\epsilon)(\epsilon_2 n/\epsilon)$ from its start square. Call the corresponding limit square L_2. One head, say on tape i_2, must be again within $\epsilon_1 n$ squares of its start square, left of the S-segment. Similarly, we can be sure of another traversal of an S-segment by input symbol $(c/\epsilon)^2(\epsilon_2 n/\epsilon)$, and so on. Therefore we can be sure of at least

$$r = \log_{c/\epsilon}(\epsilon/\epsilon_2)$$

complete traversals of the S-segments. Note, that while one head is *on* or *right* of its S-segment, the other one must be *left* of its S-segment by the contradictory assumption.

In our description of x below we give the symbols of x, read during r complete traversals of S-segments by M, in terms of M's operation. The remaining symbols are given literally in a suffix. Thus, $r(\epsilon_2-\epsilon_1)n/c$ literal bits are replaced by the following operational description.

- A description of this discussion of binary length O(1).
- A description of M of binary length O(1).
- The value of n of binary length $\log n$.
- The *final* contents of the S-segments, upon completion of processing x, in $2(\epsilon_2-\epsilon_1)n$ bits.
- The time, state of M and opposite tape's left-of-S contents and headposition at the start and finish of each complete traversal of an S-segment. Altogether this takes no more than $O(r\log n) + O(r) + O(r\epsilon_1 n)$ bits.

Thus, we have compressed the description of x to no more than

$$n - \frac{r(\epsilon_2-\epsilon_1)n}{c} + 2(\epsilon_2-\epsilon_1)n + O(r\epsilon_1 n + r\log n)$$

bits. By appropriate choice of $\epsilon_{1,2}$ we can find a constant $\delta>0$ such that the description of large enough x takes $(1-\delta)|x|$ bits, contradicting incompressibility (or "near" incompressibility for $0<\delta'<\delta$ such that $K(x) \geqslant (1-\delta')|x|$) of x. Hence the Theorem. □

Corollary. The proof of the Theorem supports the stronger assertion that there are constants $\epsilon, \delta > 0$ such that for $2^{\delta|x|}$ ("nearly" incompressible) words $x \in \{0,1\}^*$ with $K(x) \geqslant (1-\delta)|x|$ both heads of M will be simultaneously forced at least $\epsilon|x|$ squares away from the start square, at some time during the processing of x.

ACKNOWLEDGEMENT. The present presentation of the arguments in terms of Kolmogorov complexity is entirely due to the insistence of the referee. So is the present conciseness. The courtesy of the anonymous referee, in making available his 1981 seminar notes along these lines, on the original results in [Vi2], is gratefully noted. The more elaborate but clumsy [Vi2] is available on request.

REFERENCES

Aa Aanderaa, S.O., On k-tape versus $(k-1)$-tape real-time computation. In SIAM-AMS Proceedings Vol. 7 "Complexity of Computation", 1974, 75 - 96.

Ch Chaitin, G.J., Agorithmic Information Theory, *IBM J. Res. Dev.* **21** (1977) 350 - 359.

FMR Fisher, P.C., A.R. Meyer and A.L. Rosenberg, Real-time simulation of multihead tape units, *J. Ass. Comp. Mach.* **19** (1972) 590 - 607.

Ko Kolmogorov, A.N., Three approaches to the quantitative definition of information, *Problèms in Information Transmission, Vol. 1, No. 1,* (1965) 1 - 7.

LS Leong, B.L., and J.I. Seiferas, New real-time simulations of multihead tape units, *J. Ass. Comp. Mach.*

PSS Paul, W.J., J.I. Seiferas and J. Simon, An information theoretic approach to time bounds for on-line computation. Proc. 12th ACM Symposium on Theory of Computing, 1980, 357 - 367.

Pa1 Paul, W.J., On heads versus tapes. Proc. 22nd IEEE Conf. on Foundations of Computer Science, 1981, 68 - 73.

Pa2 Paul, W.J., On-line simulation of $k+1$ tapes by k tapes requires nonlinear time. Proc. 23rd IEEE Conf. on Foundations of Computer Science, 1982, 53 - 56.

Ra Rabin, M.O., Real-time computation, *Israel J. of Mathematics* **1** (1963) 203 - 211.

St Stoss, H.-J., k-Band Simulation von k-Kopf Turing Machinen, *Computing* **6** (1970) 309 - 317.

Va Valiev, M.K., Certain estimates of the time of computations on Turing machines with an input, *Cybernetica* **6** (1972) 734 - 741. Translated from *Kibernetica* **6** (1970) 26 - 32.

Vi1 Vitányi, P.M.B., On the power of real-time Turing machines under varying specifications. Proceedings of the 7th Int. Coll. on Automata, Languages and Programming, *Lecture Notes in Computer Science* **85**, Springer Verlag, Berlin, 1980, 658 - 671.

Vi2 Vitányi, P.M.B., Two-tape real-time computation. Tech. Rept. IW 139, Mathematisch Centrum, Amsterdam, The Netherlands, Juli 1980.

69F20
69F21
69F22