ON TWO-TAPE REAL-TIME COMPUTATION AND QUEUES*

Paul M.B. Vitányi
Centre for Mathematics and Computer Science (C.W.I.)
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

ABSTRACT

This is a draft version of [J. Computer and System Sciences, 29 (1984) 303 - 311].

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

Proposed Running Head: TWO-TAPE REAL-TIME COMPUTATION

* This work was supported by the Stichting Mathematisch Centrum.

1. Introduction.

Real-time computation in the world of Turing machine like devices is the 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, [9] showed that two one-head tape units are more powerful in real-time than one such unit. Later [1] generalized this by demonstrating superiority of $k+1$ one-head tape units over k such units, in real-time. In [6] a new information-theoretic argument was introduced to strip the proof in [1] down to its essentials, while [8] strengthened the result by exploiting the techniques further. Moreover, it was shown that considering multihead tape units resulted in an analogous real-time hierarchy [6, 12]. 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 $o(n)$. Thus, in order that an n-length string $z = xy$ be real-time reproducible in first-in-first-out order, the following has to hold for each small enough c $(0 < c \leq 1)$. A description of a prefix string x of length cn, which has initially been stored on the cn-length initial tapesegments, has to be transported (while storing y) to a disjoint set of tape squares all of which are at least $2cn$ squares away from the start squares. At all times during this transport x (in fact as much of the prefix of z yet received) has to be real-time reproducible on call too. One would think that this is too much to ask of the two single-head tape units. Obviously, a queue can be implemented on a real-time two-head tape unit which does not have 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 [10] and by $(4k-4)$ one-head tape units in real-time [5] (improving an earlier result of [3]). 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 [5]. 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 [6] and similarly for $d = 1$ by [8]. For 2-dimensional tape units it was shown by

[7] 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. [3]. We consider only the storage structure of the machines. So a k-tape Turing machine has k single-head storage tapes connected 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 [13]. Meanwhile, new techniques for such problems, based on Kolmogorov complexity, were introduced in [6] and further exploited in [7, 8] and elsewhere. Here we give a streamlined account of the contents of [13] using the ‘‘incompressible string’’ arguments of [6].

2. Kolmogorov Complexity

The ideas on descriptional complexity below were developed independently by Kolmogorov [4] and Chaitin [2]. We closely follow the discussion in [6]. 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

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

For the *universal* computable partial function f_0 we have that, for all f with appropriate constant c_f, for all vectors $\bar{x},\bar{y}$, $K_{f_0}(\bar{x} \,|\, \bar{y}) \le K_f(\bar{x} \,|\, \bar{y}) + c_f$. So the canonical relative descriptional complexity $K(\bar{x},\bar{y})$ can be set equal to $K_{f_0}(\bar{x} \,|\, \bar{y})$. Define the *descriptional complexity* of $\bar{x}$ as $K(\bar{x}) = K(\bar{x} \,|\, \lambda)$. ($\lambda$ 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 $K(x) \ge |x|$ for some binary string x of each length. Following [6] further, we call such strings *incompressible*. It also follows that $K(x \,|\, y) \ge |x|$ for some binary string x of each length. Since similarly $K(x) \geq (1-\delta)|x|$ for $2^{\delta|x|}$ strings over $\{0, 1\}$, which thus cannot be compressed to less than $(1-\delta)|x|$ bits, such ''nearly'' incompressible strings are abundant. Note that a string $x = uvw$ can be specified by v, $|x|$, $|u|$ and the bits of uw. Thus,

$$K(x) \leq K(v) + O(\log |x|) + |uw| \ ,$$

so that with $K(x) \geq (1-\delta)|x|$ we obtain

$$K(v) \geq |v| - \delta|x| - O(\log |x|) \ .$$

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 [11]. We show that, although L' itself can be recognized by two one-head tape units in real-time [3, 13], if two one-head tape units accept L in real-time then the worst-case work 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 [9, 11] 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 $|f(n)| \leq cg(n)$ for *all* $n > n_0$.

$f(n) \in \Omega(g(n))$ if there are positive constants c and n_0 such that $f(n) > cg(n)$ 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 of M on $\{0, 1\}^*$ is

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

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 descriptions, of the remainder of the machine, 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 [9].

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 $m(n)$ $\Omega(n)$, 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*. See Figure 1. 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 O(1);
- a description of this discussion of binary length O(1);
- 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. The times of crossing are all in between time 0 and time cn. 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

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

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* $\varepsilon, \delta > 0$ *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* $\varepsilon |x|$.

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

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

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 $0<\varepsilon_1 \ll \varepsilon_2 \ll \varepsilon$. Let x be a particular incompressible string over $\{0, 1\}$ with $|x|=n$ large and $p(n) < \varepsilon_1 n$. That is, M has always a head within distance $\varepsilon_1 n$ of a start square while processing x. Nonetheless, for each t ($t_0<t\le n$ for some $t_0>0$) each head must have ranged farther than εt by the time M processes the tth input bit. By choice of the values of the ε's this combination of requirements forces repeated traversals of the tape segments $[\varepsilon_1 n, \varepsilon_2 n]$ on both tapes. Call these segments S_1 and S_2. By input symbol $\varepsilon_2 n/\varepsilon$ both S_1 and S_2 have been traversed from left to right. Yet no head has ranged farther than $c\varepsilon_2 n/\varepsilon$ squares from its start square. Call the corresponding limit squares L_1. See Figure 2. One head, say on tape i_1, must now be within $\varepsilon_1 n$ squares of its start square, that is, left of its S-segment.

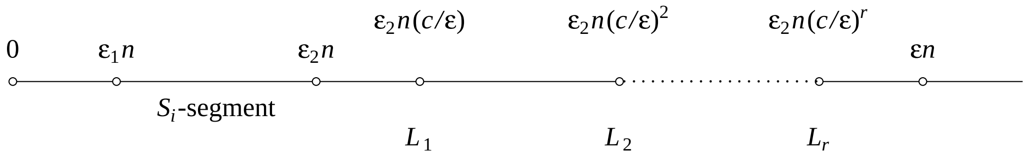

Figure 2. Tape i ($i=1, 2$) with S_i-segment and range limits $L_1, \ldots, L_r$. While processing x the head eventually crosses square εn.

By input symbol $(c/\varepsilon)(\varepsilon_2 n/\varepsilon)$ 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 $(c^2/\varepsilon)(\varepsilon_2 n/\varepsilon)$ from its start square. Call the corresponding limit square L_2. One head, say on tape i_2, must be again within $\varepsilon_1 n$ 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 $(c/\varepsilon)^2(\varepsilon_2 n/\varepsilon)$, and so on. Therefore we can be sure of at least

$$r = \log_{c/\varepsilon}(\varepsilon/\varepsilon_2)$$

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 We now determine a *subsegment* S' on each S-segment, of at least half the length of an S-segment (that is, length S' is at least $(\varepsilon_2-\varepsilon_1)n/2$) such that S' has *short* crossing sequences for *both* ends. There exists such an S'-segment on each tape of M with not more than $R = 4c/\varepsilon_2$ crossings of either end. For if not both such S'-segments could be found then the total time taken by M exceeds the assumed time bound cn, since the total time majorizes the sum of the lengths of the crossing sequences for the squares of the $S-S'$ subsegments on the two tapes. Viz., at least $(\varepsilon_2-\varepsilon_1)n\, 2c/\varepsilon_2 > cn$ crossings happen on these squares.

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, at least $r(\varepsilon_2-\varepsilon_1)n/2c$ literal bits are replaced by the operational description below. Having fixed all parameters we finally give the description of x as follows.

- 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 positions of the end squares of the S'-segments in $O(\log n)$ bits.
- The *final* contents of the S'-segments, upon completion of processing x, in at most $2(\varepsilon_2-\varepsilon_1)n$ bits.
- The time, state of M and both tapes' left-of-S' contents and headpositions at the start and finish of each excursion on an S'-segment, while processing x. Altogether this takes no more than $O(R \log n) + O(R) + O(R\varepsilon_1 n)$ bits.
- The literal remainder of x.

To retrieve x from this description, try all n-length strings over 0's and 1's until one matches the description above. That is, start with the literal suffix until an S'-excursion is due. Try all continuations until the one is found whose entrance- and exit conditions match that in the operational description. Then continue with the literal suffix until the next S'-excursion is due, and so on. Finally, match the S'-segments upon completion. If anything does not fit try the next string. By design the description matches x. If it also matches a string $y \neq x$, then both y and x drive M into the same instantaneous description, as is easily ascertained. Therefore, they can not be distinguished afterwards, contradicting acceptance of L. Thus, we have compressed the description of x to no more than

$$n - \frac{r(\varepsilon_2 - \varepsilon_1)n}{2c} + 2(\varepsilon_2 - \varepsilon_1)n + O(R\varepsilon_1 n + R\log n)$$

bits. By substitution of r, R and by appropriate choice of ε_1, ε_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) of x. Hence the Theorem. □

The proof of the Theorem supports the stronger assertion that there are constants $\varepsilon, \delta > 0$ such that for $2^{\delta |x|}$ (‘‘nearly’’ incompressible) words $x \in \{0, 1\}^*$ with $K(x) \geq (1-\delta)|x|$ both heads of M will be simultaneously forced at least $\varepsilon |x|$ squares away from the start square, 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 [13], is gratefully noted. Similar thanks are due to Joel Seiferas who moreover introduced the R-size crossing sequences in the proof of the Theorem.

REFERENCES

1. S.O. Aanderaa, On k-tape versus $(k-1)$-tape real-time computation, *in* ‘‘Complexity of Computation’’ (R.M. Karp, Ed.), SIAM-AMS Proceedings Vol. 7, pp. 75 - 96, Amer. Math. Soc., Providence, R.I., 1974.

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

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

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

5. B.L. Leong, and J.I. Seiferas, New real-time simulations of multihead tape units, *J. Ass. Comp. Mach.* **28** (1981), 166 - 180.

6. W.J. Paul, J.I. Seiferas and J. Simon, An information theoretic approach to time bounds for on-line computation, *J. Computer and System Sciences* **23** (1981), 108 - 126.

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

8. W.J. Paul, 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.

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

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

11. M.K. Valiev, 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.

12. P.M.B. Vitányi, 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.

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

FIGURES

Figure 1.

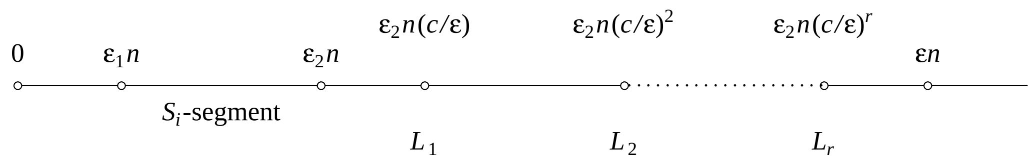

Figure 2.

FIGURE LEGENDS

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.

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