stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 139/80 JULI
(DEPARTMENT OF COMPUTER SCIENCE)

P.M.B. VITÁNYI

TWO-TAPE REAL-TIME COMPUTATION

Preprint

Kruislaan 413, 1098 SJ Amsterdam,

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

1980 Mathematics subject classification: 68C40, 68C25, 68G10
ACM-Computing Reviews - catagories: 5.23, 5.25, 5.26.

Two-tape real-time computation[*]

by

Paul M.B. Vitányi

L

ABSTRACT

Two heads on a single tape are more powerful than two single head tape units, as a storage device for real-time Turing machine computations, subject to the validity of a weak conjecture.

KEY WORDS & PHRASES: *Multitape Turing machines, real-time computation, multihead Turing machines, storage-retrieval, two heads versus two tapes*

[*] This report will be submitted for publication elsewhere.

1. INTRODUCTION

Amongst all classes of time-limited (deterministic) computations, the real-time computations distinguish themselves by being intrinsically feasible. While other time complexity classes, even the lowly linear time class, suffer the defect that there are unspecified parameters which might prohibit the actual execution of an algorithm for a problem therein, real-time computations are (up to manageable size of the machine parameters like state set and work tape alphabet) of practical impact. Real-time computations arise in computer applications like parsing problems, real-time control and so on. In the following we shall attack a well-known problem in this area: are two heads on a single tape a more powerful storage device than two single head tapes for real-time Turing machines? Our strategy is to show that there is a simple storage-retrieval problem which is doable by the former device, but if it were also doable by the latter one then such a machine must be able to shift $\Theta(n)$ bits over $\Theta(n)$ squares on its tapes in $\Theta(n)$ time, under the condition that the $\Theta(n)$-bit word is real-time retrievable at all times during this process: this we conjecture cannot be done by a 2-tape real-time Turing machine.

The afore-mentioned storage-retrieval problem is to recognize $L = \{xy2x \mid x,y \in \{0,1\}^*, 2 \notin \{0,1\}\}$ in real-time. A Turing machine, with a read-only input tape and a storage tape on which two heads operate, can easily do so. It will appear that a Turing machine with a read-only input tape and two single-head work tapes claimed to recognize L in real-time would need the mentioned very unlikely capabilities.

Time-limited computations of Turing machines with more than one work tape, or more than one head on a work tape, have been considered extensively before. One of the standard, and reasonably well understood, machine models in use for defining the time- and storage-complexity of computational tasks is the multitape Turing machine. A k-tape Turing machine consists of an input tape with a single read-only head and k storage (or work) tapes, each with a single read-write head. The k+1 heads are attached to a finite state control, which governs their action. In using the machine as a language recognizer, we assume that at the start of the computation a candidate word is written on the input tape, from left to right and with the input tape head scanning the leftmost symbol. Initially, the k storage tapes contain only blank symbols. The candidate word is accepted, if the machine reaches an accepting state at the end of its computation on this word. Such a device loosely corresponds to a computer which reads its input from a serial access read-only storage medium, and uses k read-write serial access storage media as its long-term memory. A Turing machine, with a k-head tape unit as storage, consists of a read-only input tape with a single head and a single work tape on which k read-write heads operate. This would correspond to a computer, which uses as its long-term memory a magnetic tape on which several heads compute. Physical constraints may prevent such a device from being realized. Nevertheless, most algorithms are more naturally stated in terms of computing models which allow faster memory access than the multitape Turing machines. Hence it is of interest that we can, by clever programming, simulate models with extended memory access capabilities by the basic multitape Turing machines without time loss. (And that therefore the time complexity classes are invariant under changes amongst such models.) Below we indicate only the storage device used in the computation. We assume that all machines mentioned have a separate read-only input tape. Thus, e.g., a 1-tape Turing machine has a read-only input tape and a single work tape on which a single head computes.

Pioneer papers in the study of multitape and multihead Turing machines and complexity aspects of their computations are, e.g., YAMADA [1962], RABIN [1963], HARTMANIS and STEARNS [1965], and HENNIE [1966]. STOSS [1970] indicated how to simulate a 2-head Turing machine by a 2-tape machine in linear time. P. FISCHER, MEYER and ROSENBERG [1972] showed, that we can simulate a k-head tape unit (k-head Turing machine) in real-time by a (11k-9)-tape Turing machine. LEONG and SEIFERAS [1977] improved on this result, by reducing the number of tapes to achieve this feat to 4k-4. M. FISCHER and ROSENBERG [1968] introduced the single tape Turing machine with a fast rewind square, and constructed a real-time simulation thereof by an 18-tape Turing machine. (A Turing machine with a fast rewind square can mark once a square on its work tape, and can later, at each moment of the computation, shift its work tape head back to the marked square in a single machine step, regardless of the distance in between.) Such extended models might seem to be more powerful (i.e. in real-time) than the ordinary multitape models. The above results show that, at an incurred expense of extra tapes, this is not so. As an example of a problem, which at first glance seems undoable by multitape Turing machines, we mention the real-time recognition of unmarked palindromes. The FISCHER, MEYER and ROSENBERG result was used by SLISENKO [1973], to give a 200 page proof that this problem is doable. GALIL [1978] gave a shorter and improved proof for the same fact. Extending both the multihead and the fast rewind facilities, SAVITCH and VITÁNYI [1977] defined the jump Turing machine, equipped with a multihead tape unit where the heads are allowed to jump to each others position in one machine step, regardless of the distances in between. They showed, that a k-head jump Turing machine can be simulated by an (8k-8)-tape Turing machine in linear time. Recently, KOSARAJU [1979] has claimed a proof that jump Turing machines can be simulated in real-time by multitape Turing machines. His simulation would require something like 128k-128 tapes for a k-head jump unit. How optimal the above simulations are in the number of tapes used has been hitherto unknown. BEČVÁŘ [1965] and many of the cited references raised the question of how many tapes are actually required to simulate a multihead tape unit, a Turing machine with a fast rewind square, and so on. RABIN [1963] indicated that 2 tapes (and hence a 2-head tape unit) are more powerful in real-time than 1 tape. Later, AANDERAA [1974] demonstrated, that k+1 tapes are more powerful in real-time than k tapes, $k \geq 0$. A more comprehensible exposition of AANDERAA's proof was provided by PERRY [1979]. VITÁNYI [1979, 1980a] showed that AANDERAA's result implies that k+1 heads are more powerful in real-time than k heads, $k \geq 0$. These results indicate that a (k+1)-head tape unit cannot be simulated by k tapes in real-time, $k \geq 0$. More in particular, we know that at least 2 tapes, but not more than 4 tapes are required to simulate a 2-head tape unit in real-time. (Note that the converse problem, of whether a k-head tape unit can simulate k tapes in real-time, is answered easily in the affirmative, since the k heads can maintain k separate tracks on the work tape to simulate the k tapes.)

In the sequel we show that, under the condition that a certain weak conjecture is true,

- a 2-head tape unit is more powerful than 2 single head tapes;
- for the one-way and two-way storage-retrieval units, as defined in FISCHER, MEYER and ROSENBERG [1972], precisely 3 tapes are necessary and sufficient

for real-time simulation by a multitape Turing machine;
- 3 tapes are necessary and 4 sufficient for the real-time simulation of queues, deques, 2-head tape units;

etc.

Proofs of nonfeasibility of computations of one type of machine by another type with very similar capabilities are notoriously difficult, since the supposedly weaker type might encode the information in a very clever manner. If we want to prove that a machine of type A is more powerful than a machine of type B, we have traditionally two strategies at our disposal. In the first strategy, we try to prove that the class A is rich enough to contain a machine which in some sense is universal with respect to the class B, and this machine is used to diagonalize over all machines of class B. In the second strategy, we show that machines of class B cannot store enough information to perform a certain task, which task can be fulfilled by some machines of class A, cf. RABIN [1963]. The problem we have set ourselves does not lend itself to either approach. 2-head real-time tape units are not strong enough to diagonalize over the set of 2-tape real-time Turing machines. Also, the information storage capacity of a 2-head tape unit is equal to that of 2 tapes. Thus, neither method applies and we have to resort to a fine analysis of the computational procedures available, and it is difficult to survey all the possibilities. New techniques are needed for handling this type of problem. It might be noted, that many of the most interesting questions about nonfeasibility of computations fall in this in-between region. To the author's knowledge, only RABIN [1963] and AANDERAA [1974] have provided techniques and results about such problems before.

2. PRELIMINARIES, STORAGE-RETRIEVAL PROBLEMS AND THE RESULT

A k-tape Turing machine consists of a finite-state control attached to a read-only input tape, k read-write storage tapes and a write-only output tape. A single head operates on each of the k+2 tapes. At any point in time, each head will be scanning one square on its tape, and the finite-state control will be in one state. Depending on this state and the symbols scanned by the input- and storage-tape heads, the machine will, in one step, do all of the following:

(i) on each storage tape: overwrite the symbol in the scanned tape square by a (possibly the same) symbol;

(ii) on the input tape and each storage tape: shift the head one square left, one square right or not at all;

(iii) on the output tape: do or do not write a symbol on the output tape; if a symbol is written then the output tape head is advanced one square to the right so that it is ready to write the next output symbol;

(iv) change the state of the finite-state control.

(The actions taken at (i) and (ii) may be different for different tapes.) Initially, the input tape contains the input word written from left to right, with the input tape head scanning the leftmost symbol; the finite-state control is in a distinguished start state; all storage tapes and the output tape contain only blank squares. In this paper, all machines considered will be *real-time* (on-line) *deterministic language recognizers*, unless otherwise indicated. Therefore, if the input tape contains an input word $a_1a_2...a_ia_{i+1}...a_n$, $1 \leq i < n$, then the input tape head is scanning a_i after i-1 steps, and it prints a 0 or 1 on the output tape and advances the input head one square right at the i-th step. $a_1a_2...a_i$ is *accepted*, if the symbol printed on the output tape was a 1, and otherwise *rejected*. The language accepted consists of the set of accepted words over the input alphabet. We call the machine described a k-*tape real-time Turing machine* or k-RTTM. In a k-head Turing machine the k storage tapes are replaced by a single storage tape, on which k read-write heads operate. Similarly to the above, we define a k-*head real-time Turing machine* or k-head RTTM. Additionaly, we observe that in a k-head RTTM the storage tape is initially blank, and all k storage tape heads scan initially the same square. If several heads are scanning the same square at some point in the computation, they must all write the same symbol at the current step to avoid conflicts. Without loss of generality, this can be obtained by allowing the heads on the storage tape to detect coincidence. For more formal definitions of the above and related concepts we direct the reader to ROSENBERG [1967] and to FISCHER, MEYER and ROSENBERG [1972]. Note that a 1-tape RTTM is the same as a 1-head RTTM.

Before proceeding to the storage-retrieval problem which shall be the main concern in this paper, we discuss a very similar-looking one below, so as to get some feeling about real-time language recognition with restrictions on the number of storage tapes. Let L' be defined by

$$L' = \{x2x \mid x \in \{0,1\}^*,\ 2 \notin \{0,1\}\}.$$

It was shown by VALIEV [1970], using RABIN's [1963] technique, that L' cannot be recognized by a 1-RTTM. (MEYER [1972] claims, independently, the same result.) 2-RTTM's recognizing L' do exist. Below we state the construction of one. Let each of the two storage tapes of the 2-RTTM M recognizing L' contain 8 tracks. Assume that we have already processed the prefix $y \in \{0,1\}^{4k}$ of a candidate word w as follows. (Let $y = a_1a_2...a_{4k}$.)

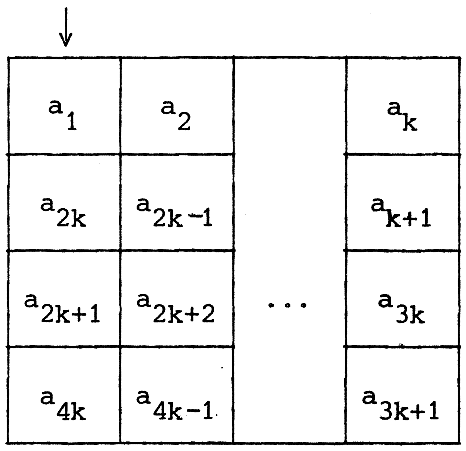
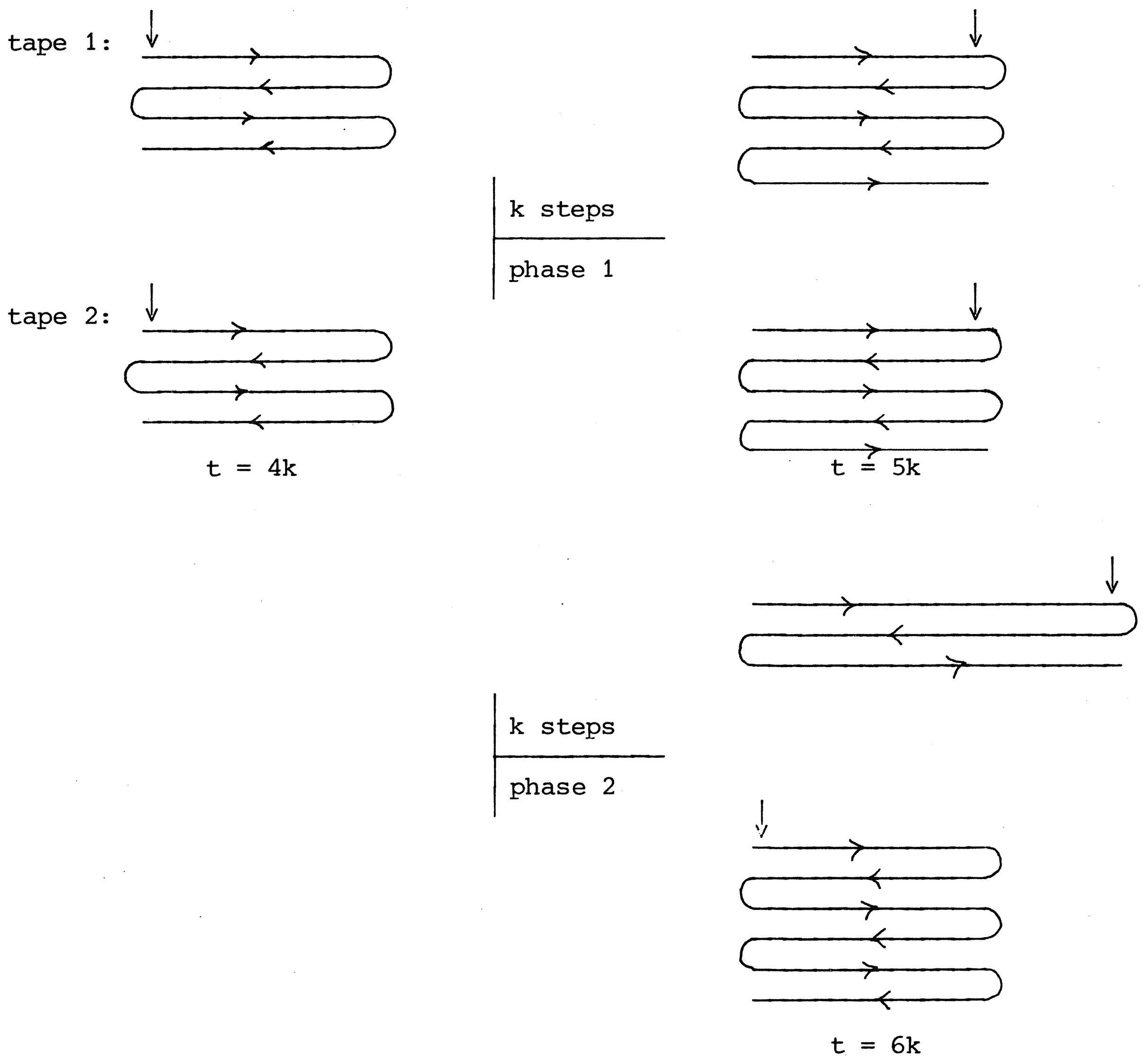


Figure 1: configuration on both storage tapes at time t = 4k.

According to Figure 1, we have recorded the prefix y of length 4k in 4 tracks on each storage tape, and the respective heads have returned to the start position. During the next 4k steps we record the additional 4k input symbols ($a_{4k+1}a_{4k+2}...a_{8k} \in \{0,1\}^*$) as indicated by Figure 2. It will be apparent from Figure 2, how we transcribe earlier recorded input from storage tape 1 to storage tape 2, and vice versa.

At the end of phase 3, i.e., 4k steps later we have regained the original situation (with a change of the direction in which $a_1a_2...a_{8k}$ is recorded on tape 2). Hence it suffices to show, that if the marker 2 is read at any time during the process described in Figure 2, we can correctly check the newly received input against the old input. That we can do this, depends on the fact that the length of the new input must be equal to the length of the old input, thus giving us enough leeway in time. We distinguish 4 cases.

Case 1. $|x| = 4 * 2^i$, $i \geq 0$. ($|x|$ denotes the length of x.) Then both heads are scanning a_1, and both tapes contain a continuously recorded copy of x. Hence, we can compare the newly received input with the recorded x without problems.



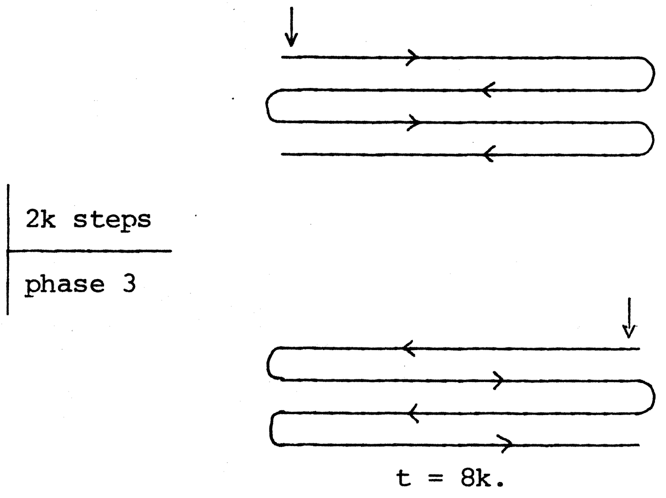


Figure 2: Reading $a_{4k+1}a_{4k+2}\dots a_{8k}$.

Case 2. We read the marker 2 during phase 1. The head on tape 1 marks its position, and, subsequently, moves at double speed to the origin and back to the marked position. In the same sweep it records (sparsely) the newly received input. Meanwhile, the head on tape 2 stays put from the time of reading the marker. When the head on tape 1 reaches the marked position anew, the head on tape 2 starts checking the remainder of the incoming input against the copy of x on its tape, from the position onwards at which it was stopped originally. If this position was i squares from the origin (start position), the head on tape 2 can check off the last $|x|-i$ symbols of the new input, against the last $|x|-i$ symbols of the recorded copy of x, by doing so. Meanwhile, now the head on tape 1 stays put, until the time the head on tape 2 reaches the origin for the first time. Then the head on tape 1 reads off the sparsely recorded first i symbols of the second copy of x at double speed, thus enabling the machine to check off also the first i symbols of the new input against the first i symbols of the old copy of x, which it will read in addition, while proceeding from the origin to the right. Since $i < |x|/4$, all this is finished before we receive the last symbol of the new copy of x.

Case 3. We read the marker 2 in phase 2. We proceed similarly to case 2, and note that we finish in time, since here $i < |x|/3$.

Case 4. We read the marker 2 in phase 3. Again, we proceed in a fashion similar to that of case 2, with the following difference. Initially the head on tape 2 moves in concert with the head on tape 1 to the (new) origin on its tape; meanwhile completing phase 3. When the head on tape 2 reaches the origin, it moves back again, in concert with the head on tape 1, to the position it was in when the interrupt came. Again, everything works out since i (i.e., distance of the head to the origin on its tape at the time the marker is read) $< |x|/3$.

The doubling of speed of the storage tape heads is achieved by using HARTMANIS and STEARNS' [1965] speed-up

THEOREM 2.1. L' = $\{x2x \mid x \in \{0,1\}^*, 2 \notin \{0,1\}\}$, *can be recognized by a* 2-RTTM, *but not by any* 1-RTTM.

In VITÁNYI [1980b] we defined the notion of relativized obliviousness. A Turing machine is *oblivious*, if the headmovements of the machine at step t depend on t only. We say that an on-line Turing machine M is $\{0,1\}$-*alphabet-oblivious*, if for each two words $w,w' \in \Sigma^*$ (Σ the input alphabet of M, and w, w' such that $h(w) = h(w')$, where h is a homomorphism such that $h(0) = h(1) = 1$ and $h(a) = a$ for $a \in \Sigma - \{0,1\}$) holds that M makes exactly the same sequence of head movements while processing w as it makes while processing w'.

COROLLARY 2.2. L' *can be recognized by a* $\{0,1\}$-*alphabet-oblivious* 2-RTTM.

A storage-retrieval problem very similar to the previous one is the problem of recognizing

$$L = \{xy2x \mid x,y \in \{0,1\}^*, 2 \notin \{0,1\}\}.$$

It is easy to see that L can be recognized by a $\{0,1\}$-alphabet-oblivious 2-head RTTM. Similarly, L can be recognized by a $\{0,1\}$-alphabet-oblivious 1-RTTM, which is allowed a single fast rewind square as in FISCHER and ROSENBERG [1968], and by a $\{0,1\}$-alphabet-oblivious one-way storage-retrieval unit as in FISCHER, MEYER and ROSENBERG [1972]. The latter, in its turn, can be simulated by a $\{0,1\}$-alphabet-oblivious 3-RTTM. In fact, the method to recognize L by a $\{0,1\}$-alphabet-oblivious 3-RTTM provides the basic step to obtain the main result in the above reference. The task we have set ourselves is, to prove that L cannot be recognized by a ($\{0,1\}$-alphabet-oblivious) 2-RTTM, thus demonstrating that 2 heads are more powerful than 2 tapes as a storage device for ($\{0,1\}$-alphabet-oblivious) real-time Turing machines. In this paper we succeed to reduce the problem as follows:

THEOREM 2.3. *If* L *is recognized by a* 2-RTTM M, *then for each integer* $s > 0$ *we can find a constant* $f_\varepsilon > 0$ $(\varepsilon = 1/s)$ *such that the following holds:*

- *For each word* $v \in \{0,1\}^*$, *we can find a word* $w \in \{0,1\}^*$, *such that, during the processing of* w *by* M, *there is a time* t, $f_\varepsilon |w| \leq t \leq |w|$, *at which time the heads on both tapes are simultaneously at least* $f_\varepsilon |w|$ *squares removed from the squares they scanned at time* 0.
- w *is obtained from* v *by inserting a* 0 *or a* 1 *after each* $k(s-1)$*-th position in* v, $k = 1,2,\ldots,\lfloor |v|/(s-1) \rfloor$. *Therefore,* $|w| = s\lfloor |v|/(s-1) \rfloor$ + *remainder* $(|v|/(s-1))$.
- *By construction there are* $2^{(1-\varepsilon)n}$ *distinct such words* w *of length* n *in* $\{0,1\}^*$.

Clearly, Theorem 2.3 implies:

COROLLARY 2.4. *If* L *is recognized by a* 2-RTTM M, *which behaves obliviously as long as it reads* 0s and 1s *(e.g., by being* $\{0,1\}$*-alphabet-oblivious), then we can find a constant* $f > 0$ *such that, during the processing of each initial word in* $\{0,1\}^n$ *by* M, *there is a time* t, $fn \leq t \leq n$, *at which time the heads on both tapes of* M *are simultaneously at least* fn *squares removed from the squares they scanned at time* 0.

These results imply, that a 2-RTTM M, claimed to recognize L, must have transported $\Theta(n)$ bits of information over $\Theta(n)$ distance on its tapes, *meanwhile* being able to recover the i-th bit of information at the i-th step subsequent to receiving the marker 2, which can happen at all times during this process. We conjecture that M cannot do so, and the proof of this conjecture is the subject of further research.

The problem with proving Theorem 2.3 in the next section is two-fold. The first aspect is to prove Corollary 2.4, while the second aspect is to prove that, while storing a word in $\{0,1\}^*$, subject to the restriction that the 2-RTTM M must be able to retrieve the stored word at any time during this process in real-time, M cannot have any real gain from being nonoblivious, i.e., the strengthening of Corollary 2.4 to Theorem 2.3. This is not so with respect to the whole problem, i.e. both storing and retrieving, since it can be shown (VITÁNYI [1980b]), that to recognize L or L' by a totally oblivious on-line Turing machine takes time order n log n.

3. PROOF OF THEOREM 2.3.

The line of reasoning we shall pursue to prove Theorem 2.3 is as follows:

(i) Since L' cannot be recognized by a 1-RTTM, it follows that L cannot be recognized by a 1-RTTM. We show that this fact implies, that a 2-RTTM M claimed to recognize L would have to use both of its tapes about equally intensive.

(ii) By folding both tapes of M around the start positions, we can represent the simultaneous head positions at each step by a pair of nonnegative integers. Using (i), and by exploiting the fact that M must store all information concerning the processed input in $\{0,1\}^*$ in some way or other, we show that, during the processing of an input of length n in $\{0,1\}^*$, the heads on both tapes of M must get simultaneously $\Theta(n)$ squares away from the starting positions.

The above would be the complete strategy followed if we could assume that M is oblivious with respect to the initial string in $\{0,1\}^*$, i.e., in the proving of Corollary 2.4. To overcome the problems attending the non-obliviousness of M, we resort to the trick of dividing $\{0,1\}^*$ in infinitely many subsets, on each of which the behaviour of M must be similar to M's behaviour on $\{0,1\}^*$. More in particular, for each $\varepsilon > 0$, $\varepsilon = 1/s$ and s a natural number, we obtain $2^{(1-\varepsilon)n}$ subsets of $\{0,1\}^n$, each of cardinality $2^{\varepsilon n}$, and on each of which the behaviour of M is similar to the behaviour of M on $\{0,1\}^n$. We now turn to the actual proof.

Assume by way of contradiction that L is recognized by a 2-RTTM M. Let the number of states of M be n_1, and the number of letters in its work tape alphabet be n_2.

DEFINITION. If M has input w then, for i = 1,2, the *work space* $t_i(w)$ of M on w is the sequence of tape squares on tape i, covered by the motion of M, while having the input sequence w.

If x is a sequence of squares on the tape, or a sequence of symbols, then $|x|$ will denote the *length*, i.e., the number of elements, of x. Let x be an input sequence. By the *coding* of x, we shall refer to the pair (code $(t_1(x))$, code$(t_2(x))$), where, for i = 1,2, code$(t_i(x))$ is the sequence of symbols in the squares of the work space $t_i(x)$ at the end of processing x. By the *instantaneous description* of M at x, we shall refer to code$(t_1(x))$, code$(t_2(x))$, the state of M, and the positions of the heads on M's two tapes, at the end of the processing of x.

LEMMA 3.1. *There exists a numerical constant* $e > 0$*, such that, for every integer* $n > 0$*, there exists a* $v \in \{0,1\}^n$ *and* $en \leq |t_1(v)| + |t_2(v)|$.

PROOF. There are 2^n sequences v in $\{0,1\}^*$ such that $|v| = n$. Since we must be able to distinguish between v and v', for $v \neq v'$, v and v' must lead to different instantaneous descriptions. Otherwise $v2v \in L(M) \Rightarrow v'2v \in L(M)$ and therefore $L(M) \neq L$. Let $|t_1(v)| + |t_2(v)| \leq k$ for all $v \in \{0,1\}^n$. Then there are at most $n_1 \cdot n_2^k \cdot k^2$ different instantaneous descriptions of M at inputs v. Hence $2^n \leq n_1 \cdot n_2^k \cdot k^2$. If n is large, this forces k to be large, so that we can assume that $k^2 n_1 \leq n_2^k$ (we assume that $n_2 \geq 2$). Thus $2^n \leq n_2^{2k}$ and hence

$$\frac{\log 2}{2 \log n_2} n \leq k.$$

Thus we may take $e_1 = 1/2\ ((\log 2)/(\log n_2))$. This e_1 will do for all n larger than some n_0; for suitably smaller e the Lemma will hold for all n. □

If M was $\{0,1\}$-alphabet-oblivious, Lemma 3.1 would read "for all $v \in \{0,1\}^n$". Since such statements are more or less what we need in the course of the proof, Lemma 3.1 is not sufficient. Intuitively, however, it is clear that Lemma 3.1 should read "for almost all $v \in \{0,1\}^n$", since it seems reasonable to assume that M can make only a very small fraction of exceptions on a general strategy for storing words in $\{0,1\}^n$. This intuition we make formal by the following division of $\{0,1\}^*$ in infinitely many subsets, for each of which Lemma 3.1, and later Lemmas, will be shown to hold.

DEFINITION. Let $\varepsilon > 0$ be any constant such that $\varepsilon = 1/s$, and s an integer. For every word $v \in \{0,1\}^*$, we define the set $[v]$ as follows:*)

$$[v] = \{w \in \{0,1\}^* \mid \text{by replacing each ks-th symbol, } k = 1,2,\ldots \ldots,\lfloor |w|/s \rfloor, \text{ in } w \text{ by } \varepsilon \text{ we obtain } v_1 \text{ and } v = v_1v_2 \text{ for some } v_2 \in \{0,1\}^*\}.$$

Define further the set $[[v]]$ as follows:

$$[[v]] = \{w \in \{0,1\}^* \mid \text{if } v = v_1a,\ a \in \{0,1\} \text{ then } w \in [v] - [v_1]\}.$$

From the definition several properties of $[v]$ and $[[v]]$ are readily ascertained.

PROPERTY 1. If $w \in [[v]]$ then $|w| = s\lfloor |v|/(s-1) \rfloor$ + remainder $(|v|/(s-1))$. Hence, for $|v|$ large enough, $|w| \approx (s/(s-1))|v|$.

PROPERTY 2. There are 2^n words v in $\{0,1\}^n$ and therefore 2^n sets $[[v]]$. If $v_1,v_2 \in \{0,1\}^n$ and $v_1 \neq v_2$ then $[[v_1]] \cap [[v_2]] = \emptyset$. Each set $[[v]]$, $v \in \{0,1\}^n$, has cardinality $2^{\lfloor n/(s-1) \rfloor}$.

PROPERTY 3. There are $2^{(1-\varepsilon)n}$ pairwise disjoint sets $[[v]]$ in $\{0,1\}^n$, all of which contain $2^{\varepsilon n}$ elements, if $|v|$ is a multiple of s-1.

PROPERTY 4. Let $v = a_1a_2\ldots a_n$. Then $[v] = \bigcup_{i=0}^{n} [[a_1a_2\ldots a_i]]$. Hence $v = v_1v_2$ iff $[v_1] \subseteq [v]$.

For clearness sake we can elucidate the way $\{0,1\}^*$ is divided into subsets as follows, cf. Figure 3. Let $j_i \in \{0,1\}^{s-1}$, $0 \leq i \leq 2^{s-1}-1$, such that j_i is a s-1 bit word and, but for the leading nonsignificant zeros, j_i is the

*) By ε we denote both a small constant and the empty word; confusion is avoided by the context.

binary representation of i.

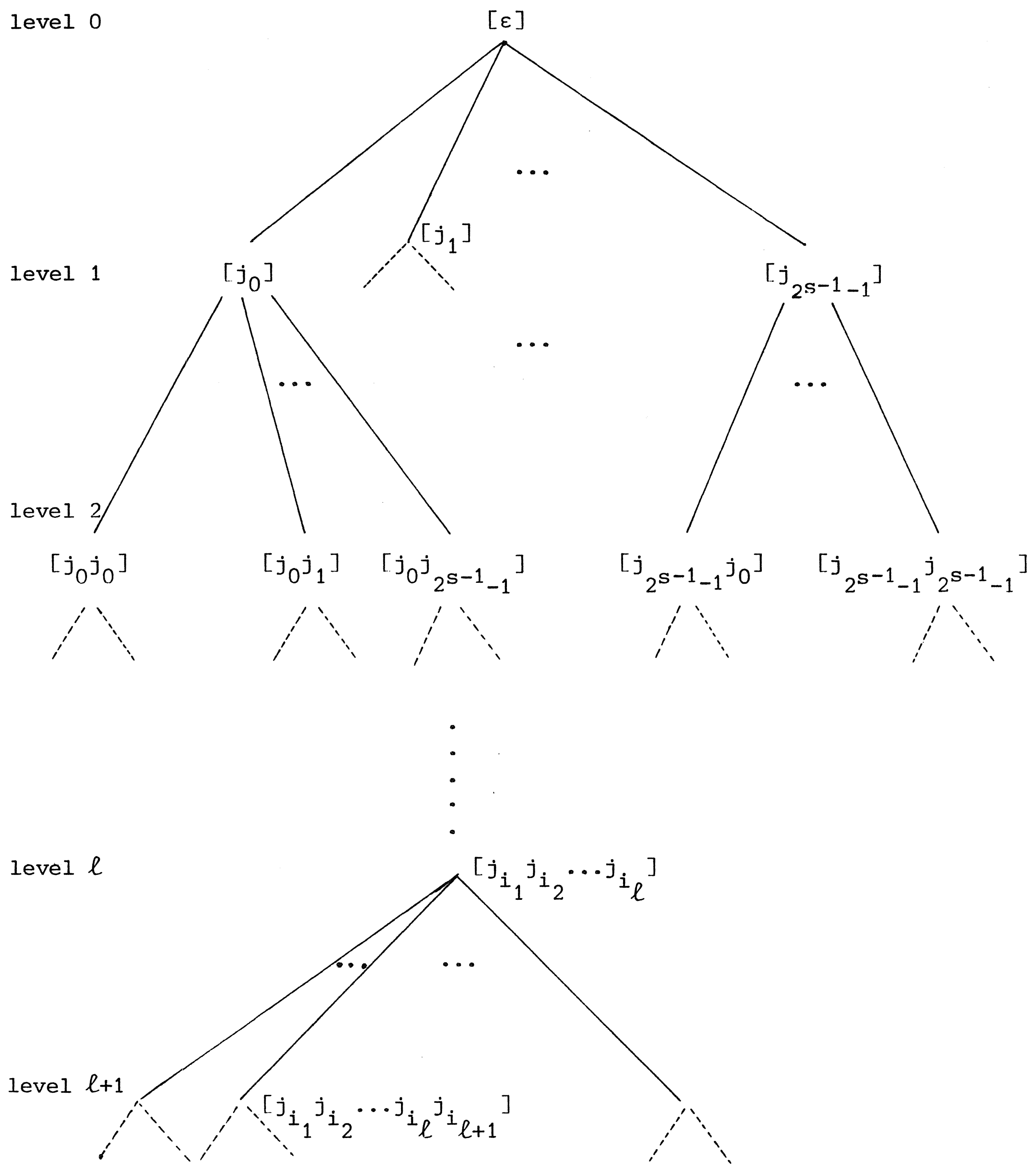


Figure 3. Tree representing division of $\{0,1\}^*$ into increasingly many subsets of increasingly many words of increasing lengths. (In the Figure, ε stands for the empty word.)

PROPERTY 5. Let $j_{\ell+1} = a_1a_2 \dots a_{s-1}$, $a_i \in \{0,1\}$, $1 \leq i \leq s-1$. Then the path from $[j_{i_1}j_{i_2} \dots j_{i_\ell}]$ to $[j_{i_1}j_{i_2} \dots j_{i_\ell}j_{i_{\ell+1}}]$ contains the sets $[j_{i_1}j_{i_2} \dots j_{i_\ell}a_1]$ up to $[j_{i_1}j_{i_2} \dots j_{i_\ell}a_1a_2 \dots a_{s-2}]$ in that order in between nodes $[j_{i_1}j_{i_2} \dots j_{i_\ell}]$ and $[j_{i_1}j_{i_2} \dots j_{i_\ell}j_{i_{\ell+1}}]$.

Hence we see that $[v_1]$ contains $[v_2]$ iff v_2 is a prefix of v_1. Furthermore, $[v_i] \cap [v_2] = [v_3]$ where v_3 is the greatest common prefix of v_1 and v_2.

If we substitute double brackets for the single brackets in Figure 3 we see that at each next level the sets consist of words which are s bits longer than the words in the sets of the previous level; each set at level ℓ has 2^{s-1} sons at level $\ell+1$, and each set at level $\ell+1$ has twice as many elements as its father at level ℓ.

Hence we see that for each of the $2^{(1-\varepsilon)n}$ disjoint subsets $[[v]]$ in $\{0,1\}^n$, $\#[[v]] = 2^{\varepsilon n}$, we obtain similarly to Lemma 3.1 that

$$(1) \qquad \frac{\varepsilon \log 2}{2 \log n_2} n \leq k$$

and therefore analogous to Lemma 3.1:

LEMMA 3.2. *For each* $\varepsilon > 0$ *there exists a numerical constant* $e_\varepsilon > 0$ *such that for every integer* n *and* $[[v]] \subseteq \{0,1\}^n$ *there exists a word* $w \in [[v]]$ *and* $e_\varepsilon n \leq |t_1(w)| + |t_2(w)|$. *We may take* $e'_\varepsilon = 1/2\ (\varepsilon(\log 2)/(\log n_2))$. *This* e'_ε *will do for all* n *larger than some* n_ε; *for suitably smaller* e_ε *the Lemma will hold for all* n.

N.B. We tacitly assume that n is a multiple of s. If n is not a multiple of s, the Lemma (and the following Lemma's) will hold under a trivial modification. In the sequel $\varepsilon > 0$ equals $1/s$, for some integer s.

Let u be an infinite word over $\{0,1\}$, that is, $u \in \{0,1\}^\infty$. Let $u(n)$ be the prefix of u of length $n - \lfloor n/s \rfloor$, so the words in $[[u(n)]]$ have length n. Define

$$(2) \qquad m^{(u)}(n) = \max_{w \in [u(n)]} \ \min_{i \in \{1,2\}} \{|t_i(w)|\},$$

where we drop the superscript on $m(.)$ when u is understood. Notice, that m is monotonic increasing.

LEMMA 3.3. *Let M be a 2-RTTM claimed to recognize* L. *For each* $\varepsilon > 0$ *($\varepsilon = 1/s$ and s an integer) there exists a numerical constant $c_\varepsilon > 0$ such that, for each $u \in \{0,1\}^\infty$ and each n, there exists a word $w \in [u(n)]$ for which $|t_1(w)|$, $|t_2(w)| \geq c_\varepsilon n$.*

PROOF. Let M, ε, $[[.]]$ and $[.]$ be as in the statement of the Lemma. Suppose, by way of contradiction, the Lemma is false. Then, for each numerical constant $c > 0$ there exists a word $u \in \{0,1\}^\infty$ and an integer n such that, for all words $w \in [u(n)]$ holds:

(3) $$|t_1(w)| < cn \quad \vee \quad |t_2(w)| < cn.$$

From definition (2) and (3) it follows

(4) $$m^{(u)}(n) < cn,$$

and therefore we can deduce the following:

for each integer $p > 0$ there exists an integer $n_{1/p}$ and a $u \in \{0,1\}^\infty$ such that, for $n = p.\ n_{1/p}$,

(5) $$m^{(u)}(i) < n_{1/p} \text{ for all } i \leq n.$$

Assume, without loss of generality, that in the processing of each word $w \in [u(n)]$ we have

(6) $$|t_2(w)| < n_{1/p} \ \vee\ |t_1(w)| < n_{1/p}.$$

(N.B. Displayed formula (7) is deleted). For ease of notation we denote, henceforth, $u(n)$ by v.

By folding both tapes of M around the start position, the infinite ends pointing right, we can, without loss of generality, assume that no head on a tape of M ever moves left of the start position. If the head would go left of its start position in the original 2-RTTM, then it now moves right on the upper track of its tape in the new 2-RTTM. We assume that the 2-RTTM M, claimed to recognize L, is in this normal form. Now construct from M a 1-RTTM M^* as follows. Let Q be the state set of M and let Γ be the work tape alphabet of M. Then $Q^* = Q \times \Gamma^{2\, n_{1/p}} \times \{0,1,\ldots,n_{1/p}-1\}^2$ is the state set of M^*, and $\Gamma^* = \Gamma$ is the work tape alphabet of M^*. That is, M^* is constructed from M by including the instantaneous description, of the initial tapesegments of length $n_{1/p}$ on both tape 1 and tape 2 of M, in the finite state control of M^*, see Figure 4.

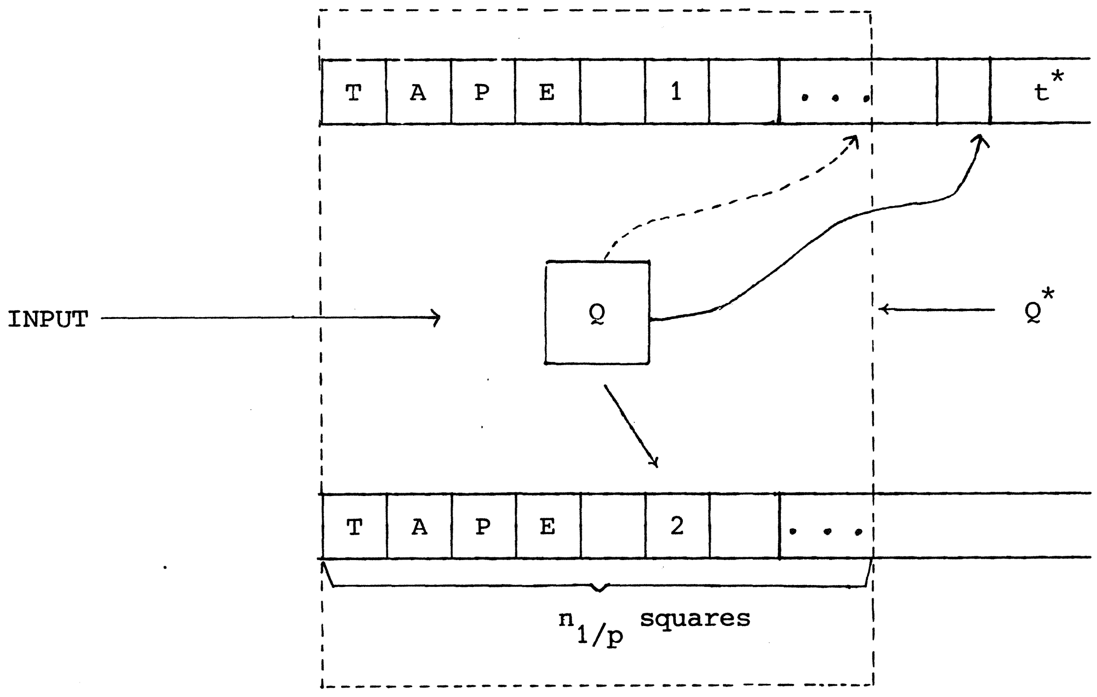


Figure 4. Construction of M^* from M.

Because of (6), we can process all words w in [v], $w \in [v] \Rightarrow |w| \le p \cdot n_{1/p}$, correctly by M^*. The idea is that if we meet the marker 2, we replace the current instantaneous description of M^* by the corresponding instantaneous description of M, to do the retrieval phase. In the sequel of proof, we show that M^* cannot process all input in [v] so as to enable M to recognize L. Therefore (6) is false, which implies that (5) is false. In its turn the falsehood of (5) implies that (3) is false, and that therefore the Lemma is true. To show that M^* cannot process all input w in [v], $|w| \leq p.n_{1/p}$, we use the idea of *bottlenecks* in 1-RTTM computations due to RABIN [1963].

Let M^* be as described with $n_1^* = \#Q^*$ and $n_2^* = \#\Gamma^* = n_2$. We proceed by a sequence of Claims. In the following $n = p \cdot n_{1/p}$ and $[[v]] \subseteq \{0,1\}^n$.

<u>CLAIM 1</u>. There exists a numerical constant $\gamma_\varepsilon > 0$ such that for all $u \in [v]$, $|u| \leq n/2$, there exists a word w, $|w| = n/2$ and $uw \in [v]$, and $\gamma_\varepsilon n \leq |t^*(uw)|$.

<u>PROOF OF CLAIM</u>. There are $2^{\varepsilon n/2}$ words w in $\{0,1\}^{n/2}$ such that $uw \in [v]$. If $w_1 \neq w$ then uw_1 and uw must be coded differently. Otherwise, uw2uw and $uw_1 2uw$ will both be accepted by M, since uw and uw_1 both lead to the same instantaneous description of M^*. Let $|t^*(uw)| \leq k$ for all such w. Then there are at most $n_1^* \cdot n_2^{*k} \cdot k$ different codings of inputs w. Hence,

(8) $$2^{\varepsilon n/2} \leq n_1^* \cdot n_2^{*k} \cdot k.$$

Substituting n_1^* and n_2^* we obtain:

(9) $$2^{\varepsilon n/2} \leq n_1 \cdot n_2^{2n_{1/p}} \cdot 2n_{1/p} \cdot n_2^k \cdot k.$$

By Lemma 3.2 we have $2n_{1/p} + k > e_\varepsilon n$, so that $k > (e_\varepsilon - 2/p)n$. Hence for p large enough, by $n = p.\ n_{1/p}$, n is large enough so that we can assume that $k.n_1.\ 2n_{1/p} < n_2^k$. Thus,

(10) $$2^{\varepsilon n/2} \leq n_2^{2(k+n_{1/p})},$$

and hence,

(11) $$((\varepsilon n/2)\log 2 - 2n_{1/p}\log n_2)/(2\log n_2) \leq k.$$

Since $n = p \cdot n_{1/p}$ we obtain by setting $p \geq (8\log n_2)/(\varepsilon \log 2)$:

(12) $$\frac{\varepsilon \log 2}{8 \log n_2}\, n_{1/p}\, p \le k.$$

Thus we may take $\gamma'_\varepsilon = (\varepsilon \log 2)/(8 \log n_2)$. This γ'_ε will do for all n greater than some n_0 (equivalently, all p greater than some p_0). For suitably smaller γ_ε, the claim will hold for all n (equivalently, for all p).
END OF PROOF OF CLAIM.

CLAIM 2. There exists an integer $d_\varepsilon > 0$ (depending only on M^* and hence on M and ε) such that for $u \in [v]$ and $n/2 \ge |u|$, there exists a word w, $uw \in [v]$ and $|w| = n/2$, such that:
a) $\gamma_\varepsilon n < |t^*(uw)|$;
b) no more than 1/3-rd of the squares of $t^*(uw)$ are covered by M^* more than d_ε times.

PROOF OF CLAIM. Let us choose a word $w \in \{0,1\}^{n/2}$, $uw \in [v]$, for which a) holds. Let d_1 be a number, such that more than 1/3-rd of the squares of $t^*(uw)$ are covered by M^* more than d_1 times. Then the total number of moves of M^* exceeds $1/3|t^*(uw)|d_1 \ge 1/3(d_1\gamma_\varepsilon n)$. But since M^* operates in real-time the number of moves of M^* by the input uw is exactly $|uw| \le n$. Thus, $1/3(d_1\gamma_\varepsilon n) \le n$ and $d_1 \le 3/\gamma_\varepsilon$. (The number $d_\varepsilon = \ulcorner(3/\gamma_\varepsilon) + 1\urcorner$ satisfies b).)
END OF PROOF OF CLAIM.

To derive a contradiction from the fact that M^* can correctly process all inputs in [v], we develop the idea that, in working on certain input sequences, the machine M^* develops bottleneck squares on its work tape, through which information cannot flow in sufficient quantity. The idea of a bottleneck is made precise in the following.

DEFINITION. Let u, $uw \in [v]$. A square B on $t^*(uw)$ is called a *bottleneck square* of $t^*(uw)$ if:
1) under input uw the machine passes through B no more than d_ε times (where d_ε is as in the previous Claim 2);
2) B lies outside the workspace $t^*(u)$;
3) the length of the section of $t^*(uw)$ determined by B, which does not contain $t^*(u)$, exceeds $|u| + 1$.

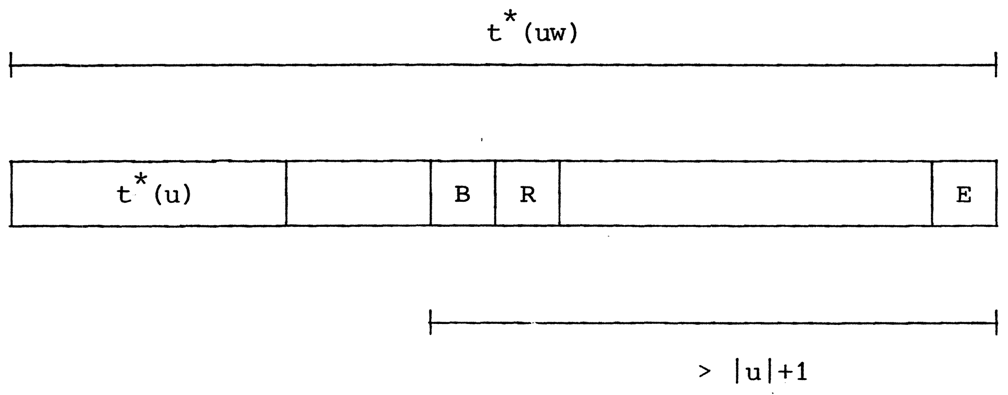


Figure 5. Bottleneck square on t^*.

Let j be an integer such that $3|u| + 3 < \gamma_\varepsilon j$ and also $|u| < j$, $u \in [v]$. By claim 2 there exists a word w, $uw \in [v]$ and $|uw| \le n$, such that $\gamma_\varepsilon j \le |t^*(uw)|$, and fewer than 1/3-rd of the squares of $t^*(uw)$ are covered more than d_ε times. Now $|t^*(u)| \le |u| + 1 < (\gamma_\varepsilon j)/3 \le |t^*(uw)|/3$. Dividing $t^*(uw)$ into 3 equal parts (there is a trivial modification of the argument if $|t^*(uw)|$ is not divisible by 3), we see that there is an interval of length $(2/3)|t^*(uw)|$ on the right end of $t^*(uw)$, which does not contain squares of $t^*(u)$. In this interval consider the 1/3-rd of $t^*(uw)$ which does not run to the end. Since fewer than 1/3-rd of the squares of $t^*(uw)$ are covered more than d_ε times by M^*, there is a square B in this 1/3-rd of $t^*(uw)$ which is covered at most d_ε times. There are at last $|t^*(uw)|/3 \ge (\gamma_\varepsilon j/3) > |u|+1$ squares between B and the end of $t^*(uw)$. Thus B is a bottleneck square, which gives us:

<u>CLAIM 3</u>. For each $u \in [v]$ such that $3|u| + 3 \le \gamma_\varepsilon n/2$ there exists a $w \in \{0,1\}^*$, $uw \in [v]$ and $|uw| \le n$, and the tape $t^*(uw)$ of M^* has a bottleneck square.

Now let u and w be as in Claim 3. As the input uw is coming in, there is a first time when M^* enters the rightmost square E of $t^*(uw)$, Figure 5. Let $z \in \{0,1\}^*$ be the initial section of w, such that uw is the sequence leading up to the first visit of M^* at E. Thus $t^*(uw)$ and $t^*(uz)$ have the same righthand end square E, and B is also a bottleneck square of $t^*(uz)$. Denote the square immediately to the right of B by R. By a *passage* of M^* *through* B, we mean either a move of M^* from B to R or a move from R to B.

The *state of M^* during a passage* is the state M^* has when it reaches R in the first case, and the state M^* has when it reaches B in the second case. Note, that the passages of M^* through B do not include atomic moves of M^* in which it starts on B and stays on B. Under the input uz the machine M^* will first cover the tape $t^*(u)$ and then, under the z portion of the input, move to square E. Let $p_1, p_2, \ldots, p_r$ be the consecutive passages through B ($r = 1$ is not excluded). The passage p_1 is a move from B to R, p_2 a move from R to B, etc. Let the state of M^* during the passage p_i be s_i, $1 \le i \le r$. The *scheme* of the bottleneck square B is the r-tuple $(s_1, s_2, \ldots, s_r)$; Now the number r of passages through B is at most d_ε. Thus, there are at most N,

$$N = n_1^* + n_1^{*2} + \ldots + n_1^{*d_\varepsilon} ,$$

different schemes of bottleneck squares, where n_1^* is the number of states in M^*. Let g be a number such that $N < 2^{\varepsilon g}$. For each $u \in [v]$, $|u| = g$, let $w \in \{0,1\}^*$ be such that $uw \in [v]$ and uw has a bottleneck square B_u, and let z denote the section of w leading to the first visit of M^* to the end E_u of $t^*(uw)$. There must be two different words $u_1, u_2 \in [v]$, $|u_1| = |u_2| = g$, such that the bottleneck squares B_{u_1} and B_{u_2} have the same scheme, say $(s_1, s_2, \ldots$ $\ldots, s_r)$. Let

$$u_1 z_1 = u_1 \tau_1 \ldots \tau_{n_1} \ldots \tau_{n_2} \ldots \tau_{n_r} \ldots \tau_{n_{r+1}}$$

$$u_2 z_2 = u_2 \sigma_1 \ldots \sigma_{m_1} \ldots \sigma_{m_2} \ldots \sigma_{m_r} \ldots \sigma_{m_{r+1}}$$

where $\tau, \sigma \in \{0,1\}$, τ_{n_1} is the input when M^* visits B_{u_1} during the first passage, τ_{n_2} is the input when M^* visits B_{u_1} during the second passage, and so on up to τ_{n_r}; similarly for $\sigma_{m_1}, \sigma_{m_2}, \ldots, \sigma_{m_r}$ in the second sequence $u_2 z_2$. After receiving the input $\tau_{n_{r+1}}$ ($\sigma_{m_{r+1}}$), M^* visits for the first time the right hand end square E_{u_1} (E_{u_2}).

We now come to the main point in the argument. In the sequence $u_1 z_1$ replace, for every odd i, $1 \le i \le r-2$, the segment $\tau_{n_i+1} \ldots \tau_{n_{i+1}-1}$ by the word $\sigma_{m_i+1} \ldots \sigma_{m_{i+1}-1}$. Furthermore, replace $\tau_{n_r+1} \ldots \tau_{n_{r+1}}$ by $\sigma_{m_r+1} \ldots \sigma_{m_{r+1}}$. Call the resulting sequence $u_1 z_1'$. (The fact that possibly $u_1 z_1' \notin [v]$ is of no concern to the argument that follows.) Note, that all the changes were made in the z_1 portion of u_1z_1. Now, u_1z_1 and u_2z_2 have the same scheme of states in the passages of M^* through B_{u_1} and B_{u_2}, respectively, and our changes in u_1z_1 were made only in the inputs *between* visits to B_{u_1}, while M^* was on the right of B_{u_1}, or after the last visit to B_{u_1}. One can see by finite induction over i, $1 \le i \le r+1$, that u_1z_1' again has the same scheme $(s_1,s_2,\ldots,s_r)$ and that at each input τ_{n_j}, j odd and $2 \le j \le r+1$, the portion of the tape right of B_{u_1} is identical with the portion of the tape $t^*(u_2z_2)$ right of B_{u_2} at input σ_{m_j}, and that the states of M^* at the corresponding inputs are the same.

The work spaces $t^*(u_1z_1')$ and $t^*(u_2z_2)$ have squares B_{u_1} and B_{u_2}, respectively, with the following properties. The workspace $t^*(u_i)$ is completely to the left of B_{u_i}, $i = 1,2$. The portions of $t^*(u_1z_1')$ and $t^*(u_2z_2)$, beyond B_{u_1} and B_{u_2}, are strictly longer than $|u_1| = |u_2| = g$. By the previous paragraph, at the end of the inputs u_1z_1' and u_2z_2, M^* is at the end squares E_{u_1} and E_{u_2} of the respective workspaces, and the portions of the tape from B_{u_1} to E_{u_1} and from B_{u_2} to E_{u_2}, as well as the states of M^* at E_{u_1} and E_{u_2}, are identical. Assume now that both u_1z_1' and u_2z_2 are followed by the input $2u_1$. We have, since $u_1 \ne u_2$,

$$u_1z_1'2u_1 \in L \quad \text{and} \quad u_2z_22u_1 \notin L.$$

Now replace M^* by the corresponding instantaneous description of M, in both cases of the processing of $u_1z_1'2u_1$, and $u_2z_22u_1$, at the instant we receive the input letter 2. We now want, in both cases, to check whether u_1 is a prefix of the previously processed input. But $|2u_1| = g + 1$ is less than the distance from E_{u_i} to B_{u_i}, $i = 1,2$. Since M operates in real-time and makes one move per input symbol, it will stay, throughout the input portion $2u_1$, to the right of B_{u_i}. Thus, M will start in both cases in the same state and will move through identically printed portions of its tapes. It will therefore be in the same states at the ends of $u_1z_1'2u_1$ and $u_2z_22u_1$, scanning the same symbols, and hence cannot accept one and reject the other. Therefore, the assumption that M^* correctly process all inputs in [v] of length $\le n$, leads to the fact that M either accepts both $u_1z_1'2u_1$ and $u_2z_22u_1$, $u_1 \ne u_2$, or rejects them both. Hence the assumption of (6) leads, by way of the construction of M^*, to the conclusion that M does not recognize L. Hence (6), leads to a contradiction. Since (6) is false, (3) must be false and therefore the Lemma true.

An analysis of the previous proof shows that we can derive from it explicitly, how large c_ε in the statement of the Lemma can be chosen for given M and ε.
Setting $\gamma_\varepsilon = (\varepsilon \log 2)/(8 \log n_2)$; $d_\varepsilon = \ulcorner(3/\gamma_\varepsilon)+1\urcorner$; $N = \Sigma_{i=1}^{d_\varepsilon} n_1^{*i} \leq n_1^{*\,(d_\varepsilon+1)}$ (assuming $n_1^* \geq 2$); g such that $N < 2^{\varepsilon g}$ and $3g+3 \leq \gamma_\varepsilon n/2$; $n = p\cdot n_{1/p}$; we choose p so large that $n_1 \cdot 2n_{1/p} < n_2^{n_{1/p}}$ and therefore $n_1^* < n_2^{3n_{1/p}}$. Then after some computation we see that, if we take logarithms in base 2:

$$p \geq 2^8 . 3^3 . \log^3 n_2 . \varepsilon^{-3} \qquad \square$$

suffices, under the condition that p is so large that $n_1 2n_{1/p} < n_2^{n_{1/p}}$. Setting c_ε to 1/p satisfies the Lemma, and therefore, all c_ε satisfying

$$(13) \qquad c_\varepsilon \leq \varepsilon^3 (2^8 . 3^3 . \log^3 n_2)^{-1} . \qquad \square$$

It would seem that the above technique can be used in general to show that if a given language is not accepted by a k-RTTM, then the assumption that it is accepted by a (k+1)-RTTM necessitates that the (k+1)-RTTM must use all of its tapes equally intensive (in the same order of magnitude $\Theta(n)$) if it can be shown that a k-RTTM claimed to recognize the given language would be fooled by some word of length p log(#Q), where Q is the state set of the k-RTTM, p is arbitrarily large and the dependencies are as described in the proof of the previous Lemma.

An argument similar to the one above would show that if there is a 2-RTTM M recognizing L, then, for some inputs of length n, the difference between the head positions on tape 1 and tape 2 (head position = distance from head on tape to its start position) must grow larger than cn, for some constant c > 0, during the processing of that input. The approach is sketched in Figure 6, where we reduce M to a 1-RTTM M^* which keeps the contents of the two tapes, on the segments of tape 1 and 2 in between the two head-positions, in its finite control. I.e., on a tape of length $n_{1/p}$ which is glued together at the ends so that square 1 follows square $n_{1/p}$.

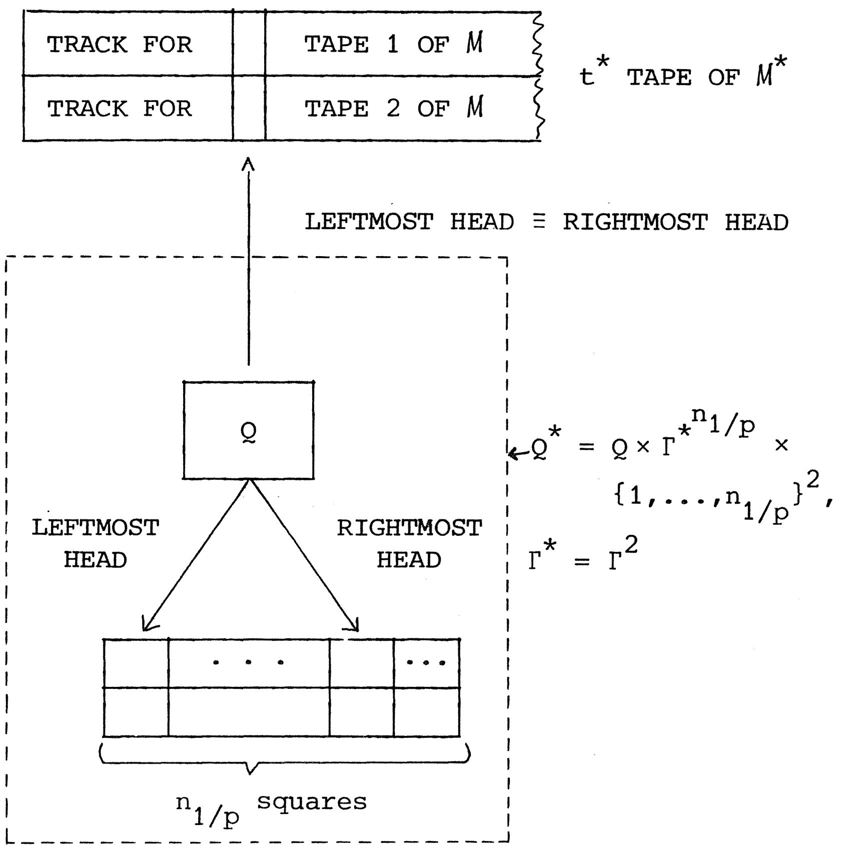


Figure 6. From both tapes of M a segment of $n_{1/p}$ squares has been cut out and put in the finite control of M^*.

The next step will be to prove that if L is recognized by a 2-RTTM M then the heads will get simultaneously far away from their starting positions. In a 1-RTTM, which has to store incoming input of length n, it is obvious that the work tape head has to move $\Theta(n)$ squares away from its starting position during this process. The analog for a 2-RTTM would be that both of its heads move $\Theta(n)$ squares away from their respective starting positions simultaneously during the processing of an input of length n. This, however, is not at all obvious. If we consider for a moment 3-RTTMs recognizing L, we see that the 3-RTTM recognizing L, reported by FISCHER, MEYER and ROSENBERG [1972], has at all times, up to reading the marker 2, at least one of its tape heads at the starting position. We shall show, that a 2-RTTM M recognizing L can, as a consequence of Lemma 3.3, not do this.

An analysis of the proof of Lemma 3.3 shows that it also supports:

LEMMA 3.4. *Let M be a 2-RTTM claimed to recognize* L. *For each* $\varepsilon > 0$ *there exists a numerical constant* c_ε *such that for all* $u \in [v]$, $|u| < \gamma_\varepsilon n/6$, *there exists a word* $w \in \{0,1\}^*$, $uw \in [v]$ *and* $|w| = n/2$, *such that* $c_\varepsilon n \le |t_i(uw)|$, $i = 1,2$. *(Here the constant* c_ε *can be chosen as the one in Lemma 3.3.)*

Recall that M is a 2-RTTM with one-way infinite tapes claimed to recognize L. Number the squares on the two tapes of M as 0,1,2,... from left to right. Let $p_i(w,t)$ be the number of the square under scan on tape i, $i = 1,2$, at step t during the processing of a word w. I.e., if $w = a_1a_2\ldots$ $\ldots a_t \ldots$ and the input head is scanning a_t, then the head on storage tape i, $i = 1,2$, is scanning square $p_i(w,t)$. As argued before, we can assume, without loss of generality, that M has one-way infinite tapes, by imagining the two-way infinite tapes of a given 2-RTTM M as being folded around their starting positions.

LEMMA 3.5. *Let M be a 2-RTTM claimed to recognize* L *and M has one-way infinite storage tapes. For each* $\varepsilon > 0$ *there is a numerical constant* $f_\varepsilon > 0$, *such that, for each* $v \in \{0,1\}^*$, *there is a word* $w \in [v]$ *and* $f_\varepsilon|v| \le p_i(w,t)$, $i = 1,2$, *for some* $t \le |w|$.

PROOF. Since in the proof we will use two different ε's to obtain our result, we index the square brackets by the ε they correspond to. Let $\varepsilon_1 = 1/(2s) > 0$ and s an integer. Let m, r and n be integers such that $\Sigma_{j=0}^{2r} c_{\varepsilon_1}^{-j} m = n$. Let $v = v_0v_1\ldots v_{2r}$ be a word in $\{0,1\}^*$ and $|v_j| = ((2s-1)/(2s))c_{\varepsilon_1}^{-j} m$ for $0 \le j \le 2r$. Now form the set $[v]_{\varepsilon_1}$. Assume without loss of generality that $c_{\varepsilon_1} < \gamma_{\varepsilon_1}/4$. Then we have, by Lemma 3.4, that we can find a n-bit word $w_0w_1\ldots w_{2r}$ with $w_0w_1\ldots w_j \in [[v_0v_1\ldots v_j]]_{\varepsilon_1}$ for all j, $0 \le j \le 2r$, such that

(14) $$c_{\varepsilon_1}^{-j+1} m \le |t_i(w_0w_1\ldots w_j)| < c_{\varepsilon_1}^{-j-1} m, \qquad i = 1,2,$$

where $t_i(z)$ is, as usual, the workspace used on tape i of the 2-RTTM M during processing of a word $z \in \{0,1\}^*$. The *lower bounds* in (14) are argued as follows. $|w_j| = c_{\varepsilon_1}^{-j}m$ since $|w_j| = ((2s)/(2s-1))|v_j|$. $|w_0w_1\ldots w_{j-1}| = \Sigma_{i=0}^{j-1} c_{\varepsilon_1}^{-i}m = ((c_{\varepsilon_1}^{-j}-1)/(c_{\varepsilon_1}^{-1}-1))m$. Now, if $c_{\varepsilon_1} < \gamma_{\varepsilon_1}/(3+\gamma_{\varepsilon_1})$, which is certainly the case if $c_{\varepsilon_1} < \gamma_{\varepsilon_1}/4$, then $|w_0w_1\ldots w_{j-1}| < \gamma_{\varepsilon_1}|w_j|/3$ and the lower bound follows by setting $|w_j| = n/2$ in Lemma 3.4. The *upper bounds* in (14) follow from the fact that $|t_i(w_0w_1\ldots w_j)| \le |w_0w_1\ldots w_j| = \Sigma_{i=0}^{j} c_{\varepsilon_1}^{-i}m < c_{\varepsilon_1}^{-(j+1)}m$.

Now let $\varepsilon = 1/s$ and suppose the Lemma does not hold for any $f_\varepsilon \ge (c_{\varepsilon_1}/z)(m/n)$, where z is some integer greater than 1. Let S_i be the starting position, B_i be the $c_{\varepsilon_1}m/z$-th square, and E_i be the $c_{\varepsilon_1}m$-th square on tape i, i = 1,2, of M, Figure 7.

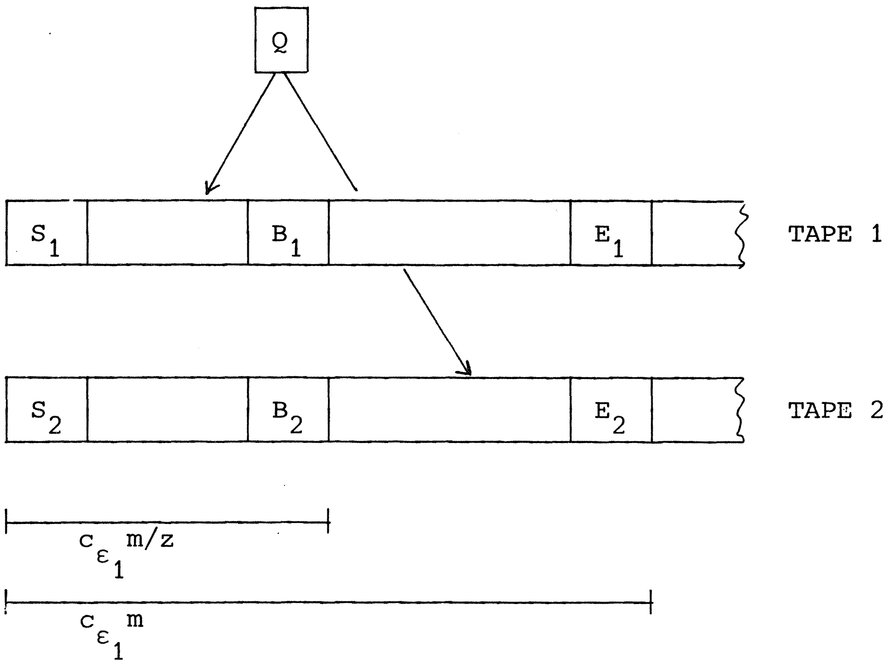


Figure 7

By assumption the heads on the storage tapes cannot be simultaneously to the right of the respective squares $B_{1,2}$ during the processing of a word $w \in [v]_{\varepsilon_1}$. By (14) we have that, during the processing of w_0 of the chosen word $w_0w_1\ldots w_{2r} \in [v]_{\varepsilon_1}$, the workspace used on both tapes is at least $c_{\varepsilon_1}m$ and therefore the head on tape 1 must cross E_1 and the head on tape 2 must cross E_2. Since by assumption not both heads can be simultaneously to the right of the B squares, this can only happen when the head on, say tape 1, proceeds from B_1 to E_1 (meanwhile the head on tape 2 is on the segment

$S_2 - B_2$), returns later from E_1 to B_1 and stays on the segment $S_1 - B_1$ while the head on tape 2 moves across B_2 to E_2. All this during the processing of w_0.

Since by (14) we have

(15) $$|t_i(w_0w_1\ldots w_{2j})| < |t_i(w_0w_1\ldots w_{2j+2})|,$$

for $i = 1,2$ and all j, $0 \le j < r$, by the same reasoning we have that during the processing of $w_{2j+1}w_{2j+2}$, the head on one tape must cover the segment $B - E$ on its tape at least once, while the head on the other tape stays on the segment $S - B$, and vice versa. Hence, during the processing of $w = w_0w_1\ldots w_{2r}$, the segment $B_1 - E_1$ must be covered at least r times and also the segment $B_2 - E_2$ must be covered at least r times, in at least 2r disjoint time intervals. Therefore, we can divide the word w in subwords as follows:

(16) $$w = u_1v_1u_2v_2\ldots u_rv_ry$$

such that: (i) during the processing of u_j the head on both tapes is on E or to the left of E; (ii) during the processing of v_j the head on one tape is to the right of the square E on that tape; (iii) at all times the head on at least one tape is on the B square or to the left of the B square on that tape; (iv) $|u_j| \ge ((z-1)/z)c_{\varepsilon_1}m$; for all j, $1 \le j \le r$.

By the assumption that the heads on the storage tapes of M cannot be simultaneously to the right of the respective B squares, we can, similarly to the proof of Lemma 3.3, construct a 1-RTTM M^* which during the processing of any word in $[v]_{\varepsilon_1}$, maintains the appropriate instantaneous descriptions M would have during the processing of that word, as follows. M^* stores the initial segments $S_i - B_i$, $i = 1,2$, in its finite control Q^*. Hence,

$$Q^* = Q \times \Gamma^{2c_{\varepsilon_1}m/z} \times \{0,1,\ldots,(c_{\varepsilon_1}m/z)-1\}^2.$$

The tape t^* of M^* will maintain on two tracks the tape contents of the one-way infinite tape segments $B_i - \infty_i$, $i = 1,2$, of the tapes of M. Since at all times only the head on a single tape of M can be to the right of a B square, we can maintain the contents of the $B_1 - \infty_1$ and $B_2 - \infty_2$ segments, on the two tracks of t^*, by the single storage head of M^* on t^*. Note, that if the head on, say, tape 1 is to the right of B_1 we can, under our assumptions, move the head on tape 2 to the right of B_2 only if first the head on tape 1 returns to the segment $S_1 - B_1$. Hence we have,

$$\Gamma^* = \Gamma^2,$$

and we depict M^* in Figure 8.

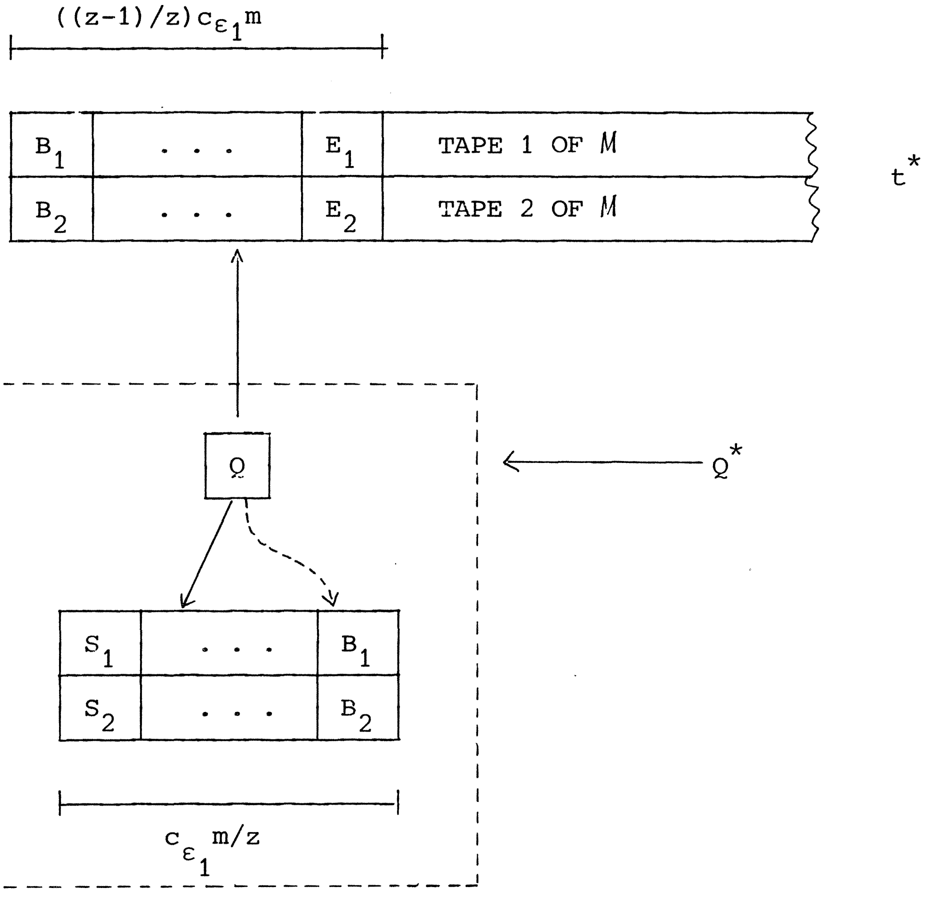


Figure 8. Construction of M^*.

We will now argue that M^* spends so much time on the $B - E$ segment of t^*, and receives so much input symbols during that time, that it cannot store all words so that it can distinguish later between all different ones. However, up until now we are only sure that a single word, viz. $w \in [v]_{\varepsilon_1}$, causes M^* to spend $r((z-1)/z)c_{\varepsilon_1}m$ time on the segment B - E of t^*, according to (iv). Now recall that $\varepsilon_1 = 1/2s$, s an integer, and therefore if

(17) $$v = a_1a_2\ldots a_{((2s-1)/(2s))n} \quad ,$$

then the words $w \in [v]_{\varepsilon_1}$ have the following form

(18) $$w = v_1v_2\ldots v_i\ldots v_{n'} \quad , \qquad 1 \leq n' \leq n/2s,$$

and each

$$v_{i+1} = a_{i(2s-1)+1}a_{i(2s-1)+2}\ldots a_{(i+1)(2s-1)}b_{i+1}, \qquad 0 \leq i < n/2s,$$

where the a's and b's are in {0,1}, the a_j's are as in (17), $1 \leq j \leq ((2s-1)/(2s))n$. By our previous reasoning, for every v as in (17) we can find a sequence of values $b_1,b_2,\ldots,b_{n/2s}$ giving us a word w as in (18) for which (14) holds. By varying the s-th letter of each v_{i+1}, $0 \leq i < n/2s$, over 0 and 1 we obtain $2^{n/2s}$ distinct v's. Hence for each such v there exists a $w \in [v]_{\varepsilon_1}$ for which (14) holds. Let V be the set of these words w. Now form the word $\bar{v}$ as follows:

(19) $$\bar{v} = \bar{v}_1\bar{v}_2\ldots\bar{v}_{n/(2s-1)}$$

such that each

$$\bar{v}_{i+1} = a_{i(2s-1)+1}a_{i(2s-1)+2}\ldots a_{i(2s-1)+s-1}a_{i(2s-1)+s+1}\ldots$$
$$\ldots a_{(i+1)(2s-1)}, \qquad 0 \leq i < n/2s,$$

where the a's are as in (17). I.e., by deleting in v all letters at positions $j \equiv s \bmod (2s-1)$.

We now have $|\bar{v}| = ((s-1)/s)n$, and for $\varepsilon = 1/s$ it holds that $V \subseteq [\bar{v}]_\varepsilon$. If we now restrict our attention to the set of input words in V, we see that all words in V cause M^* to follow the behaviour of M as described in (16) and points (i) - (iv) after that. Moreover, at each j-th input symbol of a word $w \in V$, $j \equiv s \bmod (2s)$ we can choose each one of 0 and 1 as input and still stay in V.

Hence, since in (16) $|u_1u_2...u_r| \geq r((z-1)/z)c_{\varepsilon_1}m$, there are

(20) $$2^{(r((z-1)/z)c_{\varepsilon_1}m)/2s}$$

pairwise different words w in V which differ only in $u_1u_2...u_r$, cf. (16), i.e., which differ only when the head of M^* on t^* is on E or to the left of E. Call the *crossing scheme* of a word $w \in V$ the sequence of 2r states in which M^* enters (leaves) the segment B - E of t^* at the first letter (last letter) of the subwords $u_1,u_2,...,u_r$ of w, together with the contents of B - E of t^*, when M^* crosses E on its way to the right at the end of u_r. (Note that the first state of each crossing scheme here must be the start state of M^*.) Hence there are at most

(21) $$n_1^{*2r} \cdot n_2^{*((z-1)/z)c_{\varepsilon_1}m}$$

different crossing schemes. Now let W be the set of pairwise different words w in V which differ only in the intervals while the head on t^* is on segment B - E. Now suppose that two words $w,w' \in W$, $w \neq w'$,

$$w = u_1v_1u_2v_2...u_rv_ry$$

$$w' = u'_1v_1u'_2v_2...u'_rv_ry$$

have the same crossing scheme. Then clearly, subsequent to the processing of u_r of w and u'_r of w', M^* cannot distinguish between w and w' since in both cases it has the same instantaneous description. Hence in this case M can also not distinguish between w and w', after the processing of these words, and M must accept both w2w and w'2w or reject them both. Hence M cannot accept L: contradiction. Since the size of W is given by (20) and the number of different crossing schemes by (21), we must have that

(22) $$n_1^{*2r} \cdot n_2^{*((z-1)/z)c_{\varepsilon_1}m} \geq 2^{(r((z-1)/z)c_{\varepsilon_1}m)/2s} .$$

Since $n_1^* = n_1 \cdot n_2^{2c_{\varepsilon_1} m/z} \cdot (c_{\varepsilon_1} m/z)^2$ and $n_2^* = n_2^2$, we obtain by substitution (and by noting that $n_1 \cdot (c_{\varepsilon_1} m/z)^2 < n_2^{c_{\varepsilon_1} m/z}$ for m large enough) that

$$n_2^{6r+2(z-1)} \geq 2^{(r(z-1))/(2s)} \tag{23}$$

Setting $\mu = (\log 2)/(2s \log n_2)$ we obtain

$$6r + 2(z-1) \geq \mu r(z-1) \tag{24}$$

Now the inequality (24) is not satisfied if we set $r = 3/\mu$ and $z = (18/\mu)+2$. That is, if we choose

$$r = (6s \log n_2)/(\log 2)$$

and

$$z = (36s \log n_2)/(\log 2),$$

M^* must have the same crossing scheme for 2 different words in W, and therefore M cannot accept L.

Analysing the previous proof we can compute $f_\varepsilon > 0$. The lengths of the words in W is given by $n = \Sigma_{j=0}^{2r} c_{\varepsilon_1}^{-j} m$ and therefore (for $c_{\varepsilon_1} < 1/2$)

$$mc_{\varepsilon_1}^{-2r} < n < mc_{\varepsilon_1}^{-(2r+1)}.$$

Setting $f_\varepsilon = (c_{\varepsilon_1} m)/(zn)$, we obtained for the given values of r and z a contradiction, which therefore certainly will hold for $n = mc_{\varepsilon_1}^{-(2r+1)}$. Hence we have

$$f_\varepsilon \geq c_{\varepsilon_1}^{2r+2}/z.$$

Since $\varepsilon_1 = \varepsilon/2$ we obtain by substitution of r and z as above and c_{ε_1} as in (13), while recalling that $\varepsilon = 1/s$ and logarithms are in base 2, that

$$(23) \qquad f_\varepsilon \geq \frac{(\varepsilon^3(2^{11}.3^3.\log^3 n_2)^{-1})^{12\varepsilon^{-1}\log n_2+2}}{36\varepsilon^{-1}\log n_2}$$

N.B. under assumption that n is large enough. □

PROOF OF THEOREM 2.3. By Lemma 3.5 and the properties of [] and [[]]. □

REFERENCES

[1] AANDERAA, S.O. [1974], *On* k-*tape versus* (k-1)-*tape real-time computation,* SIAM-AMS Proceedings, Vol. 7 (Complexity of Computation), 75-96.

[2] BEČVÁŘ, J. [1965], *Real-time and complexity problems in automata theory,* Kybernetica 1, 475-498.

[3] FISCHER, M.J. & A.L. ROSENBERG [1968], *Limited random access Turing machines,* Proceedings 9-th IEEE Switching and Automata Theory Conference, 356-367.

[4] FISCHER, P.C., A.R. MEYER & A.L. ROSENBERG [1972], *Real-time simulation of multihead tape units,* JACM 19, 590-607.

[5] GALIL, Z. [1978], *Palindrome recognition in real-time on a multitape Turing machine,* J. Comp. Syst. Sci. 16, 140-157.

[6] HARTMANIS, J. & R.E. STEARNS [1965], *On the computational complexity of algorithms,* Trans. Am. Math. Soc. 117, 285-306.

[7] HENNIE, F.C. [1966], *On-line Turing machine computations,* IEEE Trans. EC 15, 35-44.

[8] KOSARAJU, R. [1979], *Real-time simulation of concatenable double-ended queues by double-ended queues,* Proceedings 11-th ACM Symp. Theory Comp., 346-351.

[9] LEONG, B. & J. SEIFERAS [1977], *New real-time simulations of multihead tape units,* Proceedings 9-th ACM Symp. Theory. Comp. , 239-248.

[10] MEYER, A.R. [1972], *An optimal time bound for a one-tape on-line Turing machine computation,* (unpublished; cited in [4].)

[11] PERRY, H.M. [1979], *An improved proof of the Rabin-Hartmanis-Stearns conjecture,* MIT, Tech. Rept. MIT/LCS/TM-123.

[12] RABIN, M.O. [1963], *Real-time computation,* Israel Journal of Mathematics 1, 203-211.

[13] ROSENBERG, A.L. [1967], *Real-time definable languages,* JACM 14, 645-662.

[14] SAVITCH, W.J. & P.M.B. VITÁNYI [1977], *Linear time simulation of multihead Turing machines with head-to-head jumps,* Springer Lecture Notes in Computer Science (ICALP 4) 52, 453-464.

[15] SLISENKO, A.O. [1973], *Recognition of palindromes by multihead Turing machines,* in: "*Problems in the Constructive Trend in Mathematics VI*", 30-202 (Proceedings of the Steklov Institute of Mathematics 129; V.P. Orverkov and N.A. Sonin, eds.), Academy of Sciences USSR; English translation by R.H. Silverman, 25-208, Amer. Math. Soc., Providence, R.I., 1976.

[16] STOSS, H.J. [1970], k-*Band Simulation von* k-*Kopf Turing Maschinen,* Computing 6, 309-317.

[17] VALIEV, M.K. [1970], *Certain estimates of the time of computations on Turing machines with an input,* Cybernetics 6(1972) 734-741. Translated from Kibernetica 6 (1970) 26-32.

[18] VITÁNYI, P.M.B. [1979], *Multihead and multitape real-time Turing machines.* Mathematisch Centrum, Tech. Report IW 111, June 1979.

[19] VITÁNYI, P.M.B. [1980a], *On the power of real-time Turing machines under varying specifications,* Springer Lecture Notes in Computer Science (ICALP 7), to appear.

[20] VITÁNYI, P.M.B. [1980b], *Relativized obliviousness.* Springer Lecture Notes in Computer Science (MFCS '80), to appear.

[21] YAMADA, H. [1962], *Real-time computation and recursive functions not real-time computable,* IRE-Trans. EC 11, 753-760.