AFDELING INFORMATICA   IW 111/79   JUNI
(DEPARTMENT OF COMPUTER SCIENCE)

P.M.B. VITÁNYI

MULTIHEAD AND MULTITAPE REAL-TIME
TURING MACHINES

Preprint

2e boerhaavestraat 49 amsterdam

Multihead and multitape real-time Turing machines *)

by

Paul M.B. Vitányi

ABSTRACT

It is shown that (k+1)-head tape units are more powerful in real-time than k-head tape units. Closure properties are investigated of classes of languages accepted by real-time Turing machines with k one-head tapes or one k-head tape.

---

1. INTRODUCTION

Real-time computations (of Turing machines) are especially interesting
within the class of time-limited computations because of their intrinsic
feasibility. The usual Turing machine model we meet in complexity theory is
the multitape Turing machine. A k-tape Turing machine consists of a read-only
input tape and a finite control attached to k storage tapes. On each storage
tape a single read-write head can, according to the input symbol read to-
gether with the state of the finite-state control and the symbols under
scan on the k storage tapes, modify the scanned symbol and move one
square left, right or not at all. Most algorithms, however, are more natural-
ly stated in terms of computing models which allow faster memory access.
In a multihead Turing machine several read-write heads may compute on a
single storage tape. A k-head tape unit consists of a Turing machine with
a read-only input tape, a finite-state control and a single storage tape on
which k read-write heads operate. FISCHER, MEYER and ROSENBERG [1972] proved
that one can simulate a k-head tape unit by a multitape Turing machine in
real-time. LEONG and SEIFERAS [1977] improved this result by showing that a
k-head tape unit can be simulated in real-time by a 4k-4 tape Turing machine.
With respect to the converse question: it is trivial to show that a k-head
tape unit can simulate a k-tape Turing machine in real-time. RABIN [1963]
has observed that 2-tape Turing machines are more powerful in real-time than
1-tape Turing machines. (Recall that a 1-tape Turing machine has one input
tape and one storage tape with a single head.) Later, AANDERAA [1974] demon-
strated that k+1 tapes are more powerful in real-time than k tapes, k ≥ 1.
Together with the LEONG and SEIFERAS' result this shows that more heads will
yield additional power in real-time. Specifically, it follows that a 4k-3
head tape unit is more powerful than a k head tape unit in real-time. We
will show that AANDERAA's result implies that a k+1 head tape unit is more
powerful than a k head tape unit in real time.

In ROSENBERG [1967] several closure properties of the class R of real-
time Turing machine languages are investigated. We will investigate such
questions for the classes R(k) (languages recognized by k-tape real-time
Turing machines or k-RTTM's) and $R^H(k)$ (languages recognized by k-head
real-time tape units or k-RTTU's). Furthermore, we consider the relations

between R(k) and $R^H(k)$.

For formal definitions and so on concerning multitape- and multihead
Turing machines, real-time computations, etc. we refer to ROSENBERG [1967],
FISCHER, MEYER and ROSENBERG [1972] and LEONG and SEIFERAS [1977].

2. k+1 HEADS ARE BETTER THAN k HEADS IN REAL-TIME

AANDERAA [1974] proved by a very complicated argument that there is,
for each $k \geq 1$, a language A_{k+1} which can be recognized by a (k+1)-RTTM but
not by a k-RTTM. For completeness we define A_{k+1} below by a real-time
algorithm which accepts it using k+1 pushdown stores. The input alphabet
is $\Sigma_{k+1} = \{0_i, 1_i, P_i \mid 1 \leq i \leq k+1\}$. The algorithm is as follows:

```
"ACCEPTENABLED := TRUE;
Initialize k+1 stacks to empty;
REPEAT FOREVER
   CASE NEXTINPUTLETTER OF
   0 : Push 0 in stack i
    i
   1 : Push 1 on stack i
    i
   P : IF stack i empty
    i
       THEN ACCEPTENABLED := FALSE and reject input
       ELSE BEGIN
            pop stack i;
            IF element popped was 1
            AND ACCEPTENABLED
            THEN accept input
            ELSE reject input
            END
   ENDCASE"
```

The strategy used to prove that k+1 heads are more powerful in real-
time than k heads (on a single tape) is, by a judicious choice of input,
to force the heads so far apart that for a given recognition problem the
k-head unit must act like a k-tape Turing machine since the heads will never

read each others writing.

THEOREM 2.1. *There is a language which is recognized by a k+1 head real-time Turing machine but not by any k head real-time Turing machine.*

PROOF. By induction on the number of heads.

k=1. The language A_2 cannot be recognized by a 1-tape (= 1-head) real-time Turing machine, but can be recognized by a 2-tape (and hence by a 2-head) RTTM. Set $H_2 = A_2$.

k > 1. Suppose the theorem is true for all j < k. Hence, in particular there is a language H_k such that H_k is recognized by a k-head RTTM but not by a (k-1)-head RTTM. Define H_{k+1} as follows:

$$H_{k+1} = H_k \cup H_k * A_{k+1},$$

where * is a special symbol not in the alphabet of A_i, $i \geq 2$.

Let M_k be a k-head RTTM claimed to recognize H_{k+1}. Present M_k with a string of the form

$$w = a_1^{(2)} a_2^{(2)} \ldots a_{n_2}^{(2)} * a_1^{(3)} a_2^{(3)} \ldots a_{n_3}^{(3)} * \ldots * a_1^{(k+1)} a_2^{(k+1)} \ldots a_{n_{k+1}}^{(k+1)}$$

$$= w_2 * w_3 * \ldots * w_{k+1}$$

such that w_i is over the alphabet of A_i, $2 \leq i \leq k+1$. During the processing of w_2, M_k must recognize A_2. Since A_2 cannot be recognized by a 1-head RTTM, the distance between the outermost heads on the storage tape of M_k must grow larger than any given constant c_2 for a suitable choice of w_2. Hence, after the processing of this w_2 we can single out a head h_1 on the storage tape of M_k which is at least c_2/k tape squares removed from every other head. Choose c_2 later so that $c_2/k > 2 \sum_{i=3}^{k+1} (n_i + 1)$. Hence, for the remainder of the computation on w, M_k consists in effect of at best a single head tape and a (k-1)-head tape unit. Now M_k is presented with w_3. Since $w_3 \in A_3$ cannot be done in real-time by 2 single-headed tapes, M_k must use its remaining (k-1)-head tape unit in an essential way during the processing of w_3. I.e., the distance between the outermost heads of the remaining (k-1)-head tape unit must grow larger than any constant c_3 for a suitable choice

of w_3. Hence, we can single out a head h_2 ($h_2 \neq h_1$) such that the distance
of h_2 to every other head h_i ($h_i \neq h_2$ and $h_i \neq h_1$) is greater than $c_3/(k-1)$
after the processing of w_3. Now take c_3 so large, that $c_3/(k-1) >$
$> 2 \sum_{i=4}^{k+1} (n_i+1)$. For the remainder of the computation on w, M_k consists
now in effect of 2 single head tapes and one $(k-2)$-head tape unit. Repeating
the argument we can choose $w_4, \ldots, w_k$ such that after the processing of w_k
we are left in effect with a k-tape RTTM which is required to determine
whether $w_{k+1} \in A_{k+1}$. According to AANDERAA [1974], for each k-tape RTTM
claimed to recognize A_{k+1} we can construct a word v which fools the machine.
Let w_{k+1} be such a word, and choose $c_k, w_k, c_{k-1}, w_{k-1}, \ldots, c_2, w_2$ so that the
above inequalities and conditions are satisfied. Hence w is accepted by M_k
iff $w \notin H_{k+1}$ which contradicts the assumption that M_k recognizes H_{k+1}. It
is easy to see that k+1 pushdown stores can recognize H_{k+1} in real-time. □

Surprisingly, an argument like "H_k is not accepted by a $(k-1)$-head
RTTM and hence $H_{k+1} = H_k \cup H_k * A_{k+1}$ is not accepted by a k-head RTTM"
does not work, since we cannot assume a priori that in a k-head RTTM recog-
nizing H_k all heads get pairwise arbitrarily far apart for some input. We
could only conclude that all k heads are necessary, but it might very well
be that for each time t some heads are near to each other. Then we could
be stuck with a set of tape units, one of which is a multihead one, for
which AANDERAA's proof might not work. By the above argument we precluded
that possibility. Due to the form of A_{k+1}, the above line of reasoning
works also for A_{k+1} itself. Hence, $A_{k+1} \in R(k+1) - R^H(k)$ and we have

COROLLARY 2.2. *There is a language which can be recognized by* k+1 *pushdown
stores in real-time (and hence by a (k+1)-RTTM) but not by any k-head RTTM.*

The relation between tapes and pushdown stores is direct; clearly 2k
pushdown stores can simulate k tapes in real-time. Hence from AANDERAA's
result we have: (if $R^P(k)$ denotes the class of languages recognizable by
k pushdown stores in real-time)

$$R^P(k+1) - R(k) \neq \emptyset;$$
$$R^P(k) \subset R^P(k+1) \quad ;$$
$$R(k) \subset R(k+1) \quad ;$$
$$R(k) \subset R^P(2k).$$

By the result above it appears that we can replace R by R^H in these formulae. By using LEONG and SEIFERAS' [1977] result, it follows from the above that

LEMMA 1.3.

(i) $R(k) \subseteq R^H(k) \subset R(4k-4)$

(ii) $R(k+1) - R^H(k) \neq \emptyset$

(iii) $R^H(k+1) - R^H(k) \neq \emptyset$.

From the proof of Theorem 2.1 it will be readily ascertained that for any language $L \in R(k+1) - R(k)$ it holds that

$$L^k \notin R^H(k) \qquad (\text{assume } \varepsilon \in L)$$

and

$$\bigcup_{i=1}^{k} (L\{*\})^i \in R(k+1) - R^H(k).$$

In the diagram below we depict the present state of affairs with regard to the inclusion relations between the families $R(k)$ and $R^H(k)$.
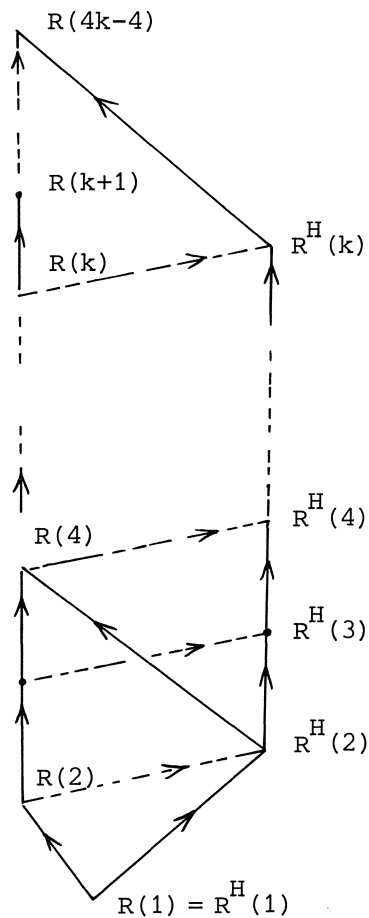

Figure 1

Connection by a solid arrow from X to Y means that X is strictly included in Y. Connection by a dotted arrow from X to Y means that X is included in Y but that it is not yet known whether inclusion is strict. The main open problem here is whether $R(k)$ is strictly included in $R^H(k)$, $k \geq 2$.

3. CLOSURE PROPERTIES OF $R(k)$

In ROSENBERG [1967] several closure properties of the class R of languages accepted by real-time Turing machines were investigated. It appeared that R is closed under union as well as intersection with regular sets, complementation, suffixing with a regular set, inverse real-time transducer mapping, and minimization. R is not closed under concatenation, Kleene star, reversal, (nonerasing) homomorphism, inverse nondeterministic sequential machine mapping, quotient with a regular set, maximization and prefixing with a regular set.

When we restrict the number of tapes the picture gets different: $R(k)$ is closed under complementation, union as well as intersection with regular sets, suffixing with regular sets, inverse gsm mapping and minimization. $R(1)$ is not closed under union or intersection, nor under inverse real-time transducer mapping.

In this section we will investigate some more closure properties of (number of) tape restricted real-time languages. It will e.g. appear that $R(k)$ is closed under several marked operations; furthermore it often happens that the closure under certain operations of $R(k)$ is in $R(2k)$ but not in $R(2k-1)$.

LEMMA 3.1. $R(k)$ *is closed under marked union, marked concatenation and marked Kleene star.*

PROOF. Marked union is obvious. We prove marked Kleene star.
If $L \in R(k)$ then so does $(L\{¢\})^*$, where $¢$ is a symbol not occurring in a word in L. Viz. let M be a k-RTTM accepting L. We construct a k-RTTM M' as follows. Upon reading a marker $¢$, the machine remembers that all previous input segments between markers were words in L. It creates clean storage by maintaining markers on each storage tape delineating the workspace used

for the computation segment in between reading two markers. Similarly we prove closure under marked concatenation. $\square$

LEMMA 3.2. R(k) *is not closed under union or intersection, for* $k > 0$.
If we take $A \in R(k_1)$ *and* $B \in R(k_2)$ *then* $A \cup B, A \cap B \in R(k_1+k_2)$, *but not necessarily* $A \cup B, A \cap B \in R(k_1+k_2 - 1)$.

PROOF. Let A_k denote AANDERAA's language over k generators. Then $A_{k_1} \in R(k_1)$ and $A_{k_2} \in R(k_2)$. Let Σ_{k_i} be the alphabet of A_{k_i}, $i = 1,2$, and let $\Sigma_{k_1} \cap \Sigma_{k_2} = \emptyset$. Then it is easy to see that $L_1 \in R(k_1)$ and $L_2 \in R(k_2)$, where L_1 and L_2 are defined as:

$$L_1 = \text{shuffle } (A_{k_1}, \Sigma_{k_2}^*) \cap (\Sigma_{k_1} \cup \Sigma_{k_2})^* \{P_i \mid P_i \in \Sigma_{k_1}\}$$

$$L_2 = \text{shuffle } (A_{k_2}, \Sigma_{k_1}^*) \cap (\Sigma_{k_1} \cup \Sigma_{k_2})^* \{P_i \mid P_i \in \Sigma_{k_2}\}.$$

Now $L_1 \cup L_2 = A_{k_1+k_2}$ and hence belongs to $R(k_1+k_2) - R(k_1+k_2 - 1)$. It follows, since our Turing machines are deterministic, that $\overline{A_{k_1+k_2}} \in$
$\in R(k_1+k_2) - R(k_1+k_2 - 1)$, $\overline{L_1} \in R_{k_1}$ and $\overline{L_2} \in R_{k_2}$. Hence $\overline{L_1} \cap \overline{L_2} = \overline{A_{k_1+k_2}} \in$
$\in R(k_1+k_2) - R(k_1+k_2 - 1)$. It remains to be proven that for $A \in R(k_2)$ and $B \in R(k_2)$ it holds that $A \cup B, A \cap B \in R(k_1+k_2)$. But it is easy to construct a (k_1+k_2)-RTTM which checks for inclusion in A with k_1 tapes and for inclusion in B with the remaining k_2 tapes. $\square$

Since R is closed under the Boolean operations (as follows also from the above lemma) the lemma creates infinite hierarchies of language families, which are all included in R.

LEMMA 3.3. R *is not closed under shuffle.*

PROOF. In ROSENBERG [1967] it is proved that the language

$$L = \{\Sigma^* x \Sigma^* 2 x^R \mid \Sigma = \{0,1\}, x \in \Sigma^*\}$$

is not in R. The same proof applies to

$$L' = \{\Sigma^* x \Sigma^* 2 h(x^R) \mid \Sigma = \{0,1\}, x \in \Sigma^*, h(0) = a \text{ and } h(1) = b\}.$$

But

$$L' = \text{shuffle}(\{x2h(x^R) \mid x \in \{0,1\}^*, h(0) = a \text{ and } h(1) = b\}, \Sigma^*) \cap$$

$$\cap \; \Sigma^* 2\{a,b\}^*,$$

with $\{x2h(x^R) \mid x \in \{0,1\}^*, h(0) = a \text{ and } h(1) = b\} \in R(1)$ and $\Sigma^*2\{a,b\}^* \in R(0)$. Hence since $L' \notin R$ also the first (shuffle) component of L' does not belong to R. □

Hence the shuffle of a language in $R(1)$ and a language in $R(0)$ (even Σ^*) does not need to belong to R. If, however, the languages which are shuffled are over disjoint alphabets, and the first one is in $R(k_1)$ and the second one in $R(k_2)$, then their shuffle is clearly in $R(k_1+k_2)$. Let L_1 and L_2 be the languages defined in the proof of Lemma 3.2. Then $L_1 \in R(k_1)$ and $L_2 \in R(k_2)$. Now take L_1 and L_2 over disjoint alphabets, say $\Sigma_{k_1} \cup \Sigma_{k_2}$ and $\Sigma'_{k_1} \cup \Sigma'_{k_2}$ but interpret the primed and unprimed symbols as being the same. Then, to recognize shuffle (L_1, L_2) is exactly the same problem as to recognize $A_{k_1+k_2}$. Hence we have

COROLLARY 3.4. *If* $A \in R(k_1)$ *and* $B \in R(k_2)$ *and the alphabets of* A *and* B *are disjoint, then shuffle* $(A,B) \in R(k_1+k_2)$ *but shuffle* (A,B) *does not need to belong to* $R(k_1+k_2 - 1)$.

LEMMA 3.5. $R(k)$ *is not closed under inverse real-time transducer mapping. The closure of* $R(k_1)$ *under inverse* k_2-RTTM *mapping is contained in* $R(k_1+k_2)$ *but not in* $R(k_1+k_2 - 1)$.

PROOF. That the closure of $R(k_1)$ under inverse k_2-RTTM mapping is contained in $R(k_1+k_2)$ was demonstrated by ROSENBERG [1967]. If we transduce $A_{k_1+k_2}$ by a k_2-RTTM M which works as described below we obtain a language A_{k_1} in $R(k_1)$ of which the inverse k_2-RTTM mapping is contained in $R(k_1+k_2) -$ $- R(k_1+k_2 - 1)$. Let Σ_{k_1} be the alphabet of A_{k_1} and let Σ_{k_2} be the alphabet of A_{k_2}. If M gets an input symbol $\in \Sigma_{k_2}$ which drives it into an accepting state for A_{k_2}, M outputs $1_i P_i$ $(1_i, P_i \in \Sigma_{k_1})$. If M gets an input symbol $\in \Sigma_{k_2}$ which drives it into a nonaccepting state it outputs $0_i P_i$ $(0_i, P_i \in \Sigma_{k_1})$. If M gets an input symbol $\in \Sigma_{k_1}$ it outputs that symbol. Hence, clearly a string $w \in (\Sigma_{k_1} \cup \Sigma_{k_2})^*$ is mapped to a string in A_{k_1} (if M is an A_{k_2} recognizer) iff $w \in A_{k_1+k_2}$. □

4. CLOSURE PROPERTIES OF MULTIHEAD RTTM LANGUAGES

According to FISCHER, MEYER and ROSENBERG [1972], the family of multi-head RTTM languages equals R and hence the (non) closure properties mention-ed before apply. If we look at multihead RTTM languages in $R^H(k)$ the situa-tion is different. Here not more is known than we can readily deduce from the results on R(k) and simulations like LEONG and SEIFERAS [1977]. With the results of the previous section we can deduce something more. Clearly, $R^H(k)$ is closed under complementation, union and intersection with regular sets, suffixing with regular sets, inverse gsm mapping and minimization. If $R^H(k) = R(k)$, which is a well known open problem, then all results in Sec-tions 2 and 3 hold even if we denote by k only the total number of heads on the storage tapes, and don't take into account the way in which the heads are distributed.

Clearly, $R^H(k)$ is closed under marked union.

LEMMA 4.1. $R^H(k)$ *is closed under marked concatenation iff* $R^H(k)$ *is closed under marked Kleene star iff* $R^H(k) = R(k)$.

PROOF. Suppose $R^H(k)$ is closed under marked concatenation and $L_k \in R(k) -$
$- R^H(k-1)$. Then for each language $L \in R^H(k)$ we have that
$L' = (L_k\{*\} \cup \{\varepsilon\})^k$ L belongs to $R^H(k)$. However, any k-head RTTM recogniz-ing $(L_k\{*\} \cup \{\varepsilon\})^k$ L gets reduced to essentially a k-tape RTTM by the time it starts recognizing L. Hence the closure of $R^H(k)$ under marked concatena-tion implies $R^H(k) = R(k)$. By Lemma 3.1, $R^H(k) = R(k)$ implies that $R^H(k)$ is closed under marked concatenation.

By setting $L' = (L\{*\} \cup \{\varepsilon\})^{k+1}$ for each language L in $R^H(k)$ we prove in a similar fashion that closure of $R^H(k)$ under marked Kleene star is equivalent to $R^H(k) = R(k)$. □

Note that by the real-time multitape simulation result the closure of $R^H(k)$ under marked concatenation (marked Kleene star) is contained in R(4k-4) and hence in $R^H(4k-4)$.

Lemma 3.2, Corollary 3.4 and Lemma 3.5 hold if we replace R(k) every-where by $R^H(k)$. The proofs are completely analogous, with an additional application of Theorem 2.1.

5. REAL-TIME JUMP TURING MACHINES

A k-head jump Turing machine (cf. SAVITCH and VITÁNYI [1977]) is a k-head Turing machine where at each step the k heads may be redistributed over the scanned tape squares. In SAVITCH and VITÁNYI [1977] it was shown that a k-head jump Turing machine can be simulated in linear time by a 2k-head Turing machine and hence by a (8k-8)-tape Turing machine. KOSARAJU [1979] has claimed that, by a complicated simulation, a k-head jump Turing machine can be simulated in real-time by a multitape Turing machine. It is at present unresolved whether k heads are more powerful than k tapes in real-time. A possibly easier problem is to show that k heads with jumps are more powerful than k tapes in real-time. We will show that these matters are related.

It is easy to see that $R^J(k)$ (the class of languages accepted in real-time by k-head jump Turing machines) is closed under marked concatenation and marked Kleene star. By feeding the k-fold marked concatenation of a language in $R(k) - R(k-1)$ we can always reduce a k-head RTTM to a k-tape RTTM. This, however, is not the case for a k-head jump RTTM. Hence k jump heads are more powerful than k tapes iff k jump heads are more powerful than k heads. Similarly, k jump heads are more powerful than k heads if k heads are more powerful than k tapes. Hence we have

LEMMA 5.1.

(i) $R(k) \subset R^J(k)$ *iff* $R^H(k) \subset R^J(k)$;

(ii) *if* $R(k) \subset R^H(k)$ *then* $R^H(k) \subset R^J(k)$.

REFERENCES

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

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

KOSARAJU (1979), *Real-time simulation of concatenable double-ended queues by double-ended queues,* Proceedings 11-th STOC.

LEONG, B. & J. SEIFERAS (1977), *New real-time simulations of multihead tape units,* Proceedings 9-th STOC.

RABIN, M.O. (1963), *Real-time computation,* Israel Journal of Mathematics 1 (1963), 203-211.

ROSENBERG, A.L. (1967), *Real-time definable languages,* JACM 14 (1967), 645-662.

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