stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 140/80 JULI

PAUL M.B. VITÁNYI

REAL-TIME TURING MACHINES UNDER VARYING SPECIFICATIONS

Preprint

Kruislaan 413, 1098 SJ Amsterdam,

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

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

1980 Mathematics subject classification: 68C40, 68C25, 68F10

ACM-Computing Reviews - categories: 5.23, 5.25, 5.26

Real-time Turing machines under varying specifications *)

by

Paul M.B. Vitányi

ABSTRACT

We investigate the relative computing power of Turing machines with differences in the number of tapes, heads pro tape, instruction repertoire etc. We concentrate on the k-tape, k-head and k-head jump models as well as the 2-way multihead (writing) finite automata with and without jumps. Differences in computing power between machines of unlike specifications emerge under the real-time restriction. In particular it is shown that a (k+1)-head tape unit is more powerful than a k-head tape unit as a storage device for real-time Turing machines, and that jumps add power to multihead 2-way real-time finite automata.

KEY WORDS & PHRASES: *complexity, real-time computations, multitape Turing machines, multihead Turing machines, jump Turing machines, multihead finite automata, multihead writing automata*

---

*) An extended abstract of this paper was presented at the seventh International Colloquium on Automata, Languages and Programming, Noordwijkerhout, The Netherlands, July 1980. The material in sections 2 and 3 appeared earlier in VITANYI [1979].

This report will be submitted for publication elsewhere.

1. INTRODUCTION

Since the first Turing machine appeared in 1936, there have been many advances in the field. In the late 1950's the *multitape* Turing machine was introduced, often equiped with a separate read-only input tape. Since then we saw the arrival of the *multihead* Turing machine, Turing machines with a *fast rewind square* (also called limited random-access machines), Turing machines with *head-to-head jumps*, and many others. One common feature in this abundance of models is that they all have a finite control and an unrestricted read-write storage facility. This allows each model, whatever its specification, to compute all recursive functions. Differences in capabilities become apparent if we impose time limitations, and in particular when we demand the machines to operate in *real-time*. As a standard in this area we may take the class of *real-time definable languages* R, which is the class of all languages accepted by multitape Turing machines in real-time, ROSENBERG [1967]. It has been shown that all of the above mentioned variations of Turing machines accept in real-time precisely R. Hence we observe that, within the world of real-time Turing machine-like devices, R plays somewhat the same role as the class of recursively enumerable languages in the world of computability at large. Like in this wider setting, we shall impose restrictions on the machines and observe what happens. In the province of real-time computations, differences in computing power amongst unlike Turing machines may come out under variations in instruction repertoire, amount or type of storage devices, in short, under different *specifications*.

The class of real-time definable languages is remarkably extensive (e.g. the set of unmarked palindromes is in R, GALIL [1978]). To prove that a given language is not in R is often hard. Proofs usually rely on an information-capacity argument, see HARTMANIS & STEARNS [1965] and ROSENBERG [1967]. To prove that a language is not accepted by a class of machines A, whereas it is accepted by a class of machines B with very similar capabilities, e.g., A is the class of k-tape real-time Turing machines and B is the class of (k+1)-tape real-time Turing machines, is harder still, and not many techniques have been developed for addressing such problems. In this paper we shall be concerned with this type of question.

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.

Originally real-time computations were defined relative to the multitape Turing machines. Most algorithms, however, are more naturally stated in terms of computing models which allow faster memory access. In a multihead machine several read-write heads may compute on a single storage tape. A k-head tape unit consists of a Turing machine with a single storage tape on which k read-write heads operate. P. FISCHER, MEYER & ROSENBERG [1972] proved that one can simulate a k-head tape unit in real-time by a multitape Turing machine with 11k-9 tapes. Later, LEONG & SEIFERAS [1977] improved this to 4k-4 tapes. 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.) AANDERAA [1974] demonstrated that k+1 tapes are more powerful than k tapes in real-time. Together with the LEONG & 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 in real-time than a k-head tape unit. We shall 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, section 2.

In ROSENBERG [1967] several closure properties of R are investigated. We investigate such questions for the classes R(k) (languages recognized by k-tape real-time Turing machines), $R^H(k)$ (languages recognized by k-head real-time Turing machines) and $R^J(k)$ (languages recognized by k-head real-time Turing machines with head-to-head jumps). Furthermore, we shall consider the relations between R(k), $R^H(k)$ and $R^J(k)$, sections 3 and 5.

In SAVITCH & VITÁNYI [1977] it was shown that a k-head jump Turing machine can be simulated in linear time by an (8k-8)-tape Turing machine. KOSARAJU [1979] has claimed a proof that jump Turing machines can be

simulated in real-time by multitape Turing machines at the cost of many tapes in the latter pro head in the former machine. In section 4 we show that the analog of this result does not hold if we restrict ourselves to 2-way multihead finite automata. The sample languages we use to prove this result are interesting in their own right, since they give once more an indication how wrong our intuition can be with respect to which languages belong to R and which languages do not. In general we prove that for real-time multihead finite automata the jump option cannot be compensated for by adding heads, nondeterminism and bidirectionality; an extra head cannot be compensated for by adding jumps, nondeterminism and bidirectionality; nondeterminism cannot be compensated for by adding jumps, extra heads and bidirectionality; and, more obvious, bidirectionality cannot be compensated for by adding extra heads, jumps and nondeterminism. With respect to real-time 2-way multihead writing finite automata it is shown that k+1 heads are better than k, and that the k-head version of the machine is less powerful than the k-head real-time Turing machine.

But for RABIN's and AANDERAA's results, all results in the area of models of real-time Turing machines are about feasibility of simulating one type of machine by another one. Virtually nothing is known about the non-feasibility of certain computations, which are possible on a machine of specification A, by a machine of specification B. Obvious open problems in this area of specified Turing machines are, for instance:
$R(2) \subset R^H(2)$; $R^H(k) \subset R^H(k+1)$; $R^J(k) \subset R^J(k+1)$; $R(k) \subset R^H(k)$; $R(k) \subset R^J(k)$; $R^H(k) \subset R^J(k)$? Some of these questions we shall decide, or alternatively, show some interdependence among seemingly unrelated questions.

For formal definitions and so on concerning multitape- and multihead Turing machines, real-time computations, etc. we refer to ROSENBERG [1967], FISCHER, MEYER & ROSENBERG [1972] and LEONG & 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 FOREOVER
  CASE NEXTINPUTLETTER OF
  0_i: Push 0 in stack i
  1_i: Push 1 on stack i
  P_i: IF stack i empty
       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.

Therefore, subsequent to the processing of this w_2, we can single out a nonscanned segment s_1 of the storage tape of M_k in between the outermost heads, such that the length of s_1 is greater than or equal to c_2/k tape squares. Denote the middle square of s_1 by M_1, see Figure 1.

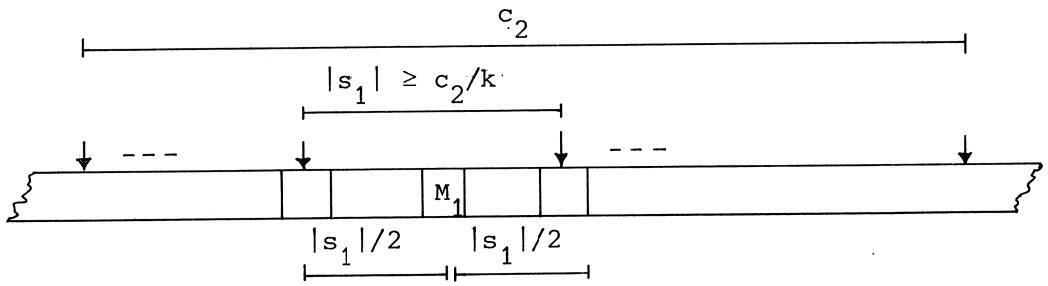

Figure 1. M_k's storage tape at time $t = |w_2|$.

Now assume that we have chosen c_2 such that $c_2/k > 2 \sum_{i=3}^{k+1} (n_i+1)$. Then, for the remainder of the computation on w, no head will cross square M_1, and therefore, from time $t = |w_2|$ onwards M_k will consist in effect of a $k_1^{(1)}$-head tape unit and a $k_2^{(1)}$-head tape unit, when $k_1^{(1)}$ is the number of heads left of M_1 and $k_2^{(1)}$ is the number of heads right of M_1 at time $t = |w_2|$, $k_1^{(1)}, k_2^{(1)} \geq 1$ and $k_1^{(1)} + k_2^{(1)} = k$. Now M_k is presented with w_3.

Since $w_3 \in A_3$ cannot be decided by 2 single headed tapes in real-time, M_k must use its remaining $k_1^{(1)}$- and $k_2^{(1)}$-head tape units in an essential way during the processing of w_3. I.e., the distance between the outermost heads on at least one of the $k_1^{(1)}$-head and $k_2^{(1)}$-head tape units must grow larger than any given constant c_3 for a suitable choice of w_3 (and the multihead unit concerned must have at least 2 heads). Without loss of generality, we assume this is the case for the $k_1^{(1)}$-head tape unit. Similar to before, we can, subsequent to the processing of w_3, single out a nonscanned tape segment s_2 of the $k_1^{(1)}$-head tape unit, in between the outermost heads on this unit, such that the length of s_2 is greater than or equal to $c_3/k_1^{(1)}$ tape squares. Denote the middle square of s_2 by M_2, see figure 2.

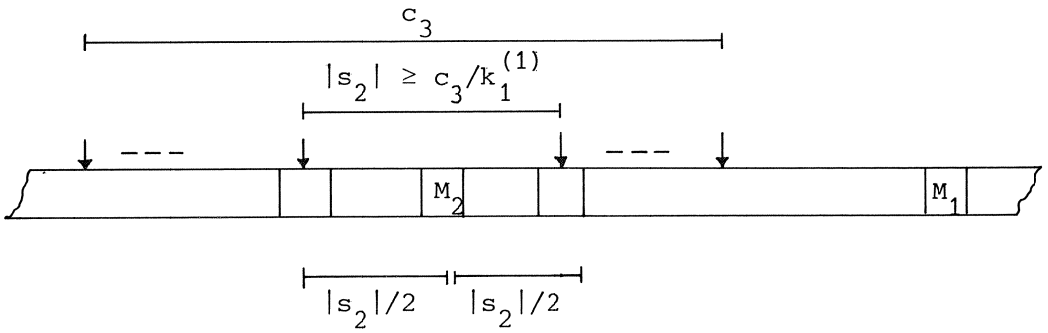

Figure 2. $k_1^{(1)}$-head tape-unit at time $t = |w_2 * w_3|$.

Now assume that we have chosen c_3 such that $c_3/k_1^{(1)} > 2\Sigma_{i=4}^{k+1}(n_i+1)$. Then, for the remainder of the computation on w no head will cross M_2, and therefore, from time $t = |w_2 * w_3|$ onwards M_k will consist in effect of a $k_1^{(2)}$-head, a $k_2^{(2)}$-head and a $k_3^{(2)}$-head tape unit, where $k_1^{(2)}$ is the number of heads left of M_2, $k_2^{(2)}$ is the number of heads in between M_2 and M_1 and $k_3^{(2)}$ is the number of heads right of M_1 at time $t = |w_2 * w_3|$, $k_1^{(2)}, k_2^{(2)}, k_3^{(2)} \geq 1$, $k_1^{(2)} + k_2^{(2)} = k_1^{(1)}$ and $k_3^{(2)} = k_2^{(1)}$. Repeating the argument we can choose $w_4, \ldots, w_k$ such that subsequent to 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},...,c_2, w_2, in that order, so that the above inequalities and conditions are satisfied after each such choice. Hence w is accepted by M_k iff $w \notin H_{k+1}$ which contradicts the assumption that M_k recognizes H_{k+1}. (The above argument seemingly contains a circularity which might invalidate it. The word v which fools the machine trying to recognize A_{k+1} does not only depend on the finite control but *also* on the initial tape contents. Thus the argument seems to become circular: w_{k+1} depends on $w_2*w_3*...*w_k*$, while $w_2,w_3,...,w_k$ depend on the length of w_{k+1}. As it happens, AANDERAA's argument does not need to make any assumptions about the initial tape contents of the k-RTTM assumed, by way of contradiction, to accept A_{k+1}. Hence he proves in fact that for all k-RTTM M there exists a positive integer n such that for all initial tape contents of M there exists a word v of at most length n which fools M. The existence of such a bound n eliminates the apparent circularity from the above argument.) 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 recognizing 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.

The situation we have in mind is exemplified by, e.g., the languages E_k, $k \geq 4$, in section 5 (although AANDERAA's proof technique fails there for another reason, as shall be pointed out). As an example of a language which can be recognized by a 4-head RTTM in which there are always 2 heads together, and which probably cannot be recognized by a 4-RTTM, or a 3-head RTTM, we give the language L below. Clearly, we cannot conclude from $L \notin R^H(3)$ (if that is the case) that $L \cup L * A_5 \notin R^H(4)$ just because $A_5 \notin R(4)$. We would need to show at least that A_5 cannot be recognized by a RTTM with one 2-head tape and two 1-head tapes as storage.

$$L' = \{u_1 w w^R u_2 v v^R u_3\ 2\ 0^{|u_1 w|}\ 2\ 0^{2|w|}\ 2\ 0^{|u_3 v|}\ 2\ 0^{|v|} \mid u_1 w u_2 v u_3 \in \{0,1\}^*\}$$

$$L = \{x \in \{0,1,2\}^* \mid x \text{ is a prefix of a word in } L'\}.$$

For suppose we want to recognize L by a 4-head RTTM. During the initial input over {0,1}, it seems that we can do nothing more than record the incoming bit stream on the storage tape. Supposing this to be the case, if we take $|w|, |v| \in \theta(n^{2/3})$, $|u_2| \in \theta(n)$, $|u_1|, |u_3| \in \theta(n^{2/3})$, where n is the length of the input word, we need 2 heads to check ww^R (since to check ww^R with 1 head takes time $\theta(n^{4/3})$) and 2 heads to check vv^R (for the same reason). To cross u_2 with some head takes time $\theta(n)$, but upon meeting the first letter 2 we have only time $\theta(n^{2/3})$ left. Hence all 4 heads seem necessary, although there always are 2 together. (We leave it to the reader to show how a 4-head RTTM, or even a 4-head 2-way real-time deterministic finite automaton, can recognize L such that at all times during this recognition process 2 heads scan the same square.)

If this conjecture is true, then $L \in R^H(4) - R^H(3)$. But in this case, $L \in R^H(4) - R^H(3)$ together with $A_5 \notin R(4)$ does not, without additional considerations, imply $L \cup L * A_5 \notin R^H(4)$.

By the proof technique of Theorem 2.1 we precluded such a flaw in our argument. Due to the form of A_{k+1}, the above line of reasoning works also for A_{k+1} itself. Hence, $A_{k+1} \in R(k+1) - R^H(k)$.

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):

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

By the result above it follows that we can replace R by R^H in the first formula above. It also follows that

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

$$R^H(k) \subset R^H(k+1).$$

By using LEONG & SEIFERAS' [1977] result we obtain

LEMMA 2.3. $R(k) \subseteq R^H(k) \subset R(4k-4)$.

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

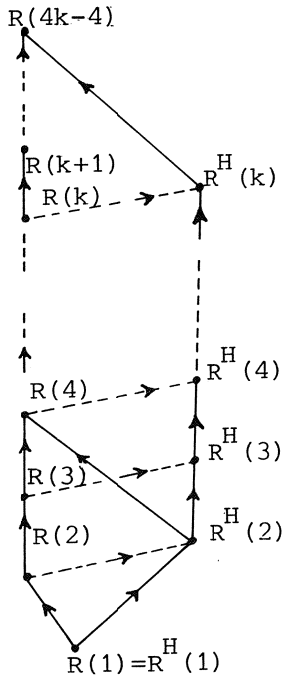

Figure 3

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

In ROSENBERG [1967] several closure properties of the class R of languages accepted by real-time Turing machines were investigated. He showed

that R is closed under union, intersection, complementation, suffixing with a regular set, inverse real-time transducer mapping and minimization. R is not closed under concatenation (even with regular sets), Kleene star, reversal, (nonerasing) homomorphism, inverse nondeterministic sequential machine mapping, quotient with a regular set, maximization and prefixing with a regular set. R is incomparable with the (deterministic) context free languages, contains ambiguous context free languages and is contained in the deterministic context sensitive languages.

With respect to restrictions on the number of tapes, ROSENBERG noted that R(k) is closed under complementation, union with regular sets, 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. He furthermore conjectured that the union (intersection) of A,B with $A \in R(k_1)$ and $B \in R(k_2)$, can yield a language in $R(k_1+k_2) - R(k_1+k_2-1)$.

Here we shall show that this conjecture is true, and also investigate some other closure properties of (number of) tapes restricted real-time languages. It will, e.g., appear that R(k) is closed under several marked operations; and that the closure of $R^H(k)$ under these marked operations is equivalent to the equality of R(k) and $R^H(k)$ (modulo a trivial restriction).

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. Let M be a k-RTTM recognizing L. We construct a k-RTTM M^* recognizing $(L\{\not\epsilon\})^*$ as follows. M^* works just like M with the following modifications. Upon reading a marker $\not\epsilon$, the machine remembers whether or not all previous input segments in between 2 consecutive markers were words in L. It creates clean storage by maintaining markers on each storage tape delineating the work space in use of the current computation segment in between reading markers. Similarly we prove closure under marked concatenation. □

According to FISCHER, MEYER & ROSENBERG [1972], the family of multi-head RTTM languages equals R and hence the (non) closure properties

mentioned before apply. If we look at multihead RTTM languages in $R^H(k)$ the situation is different. Here not more is known than we can readily deduce from the results on R(k) and simulations like LEONG & SEIFERAS [1977]. With the preceding results 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 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 3.2.[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. (i) Suppose that $R^H(k)$ is closed under marked concatenation. Then, for each language L ($\varepsilon \in L$) in $R^H(k)$ we have that A_k^*L belongs to $R^H(k)$. However, every k-head RTTM recognizing A_k^*L gets reduced to essentially a k-tape RTTM, in the manner described in the proof of Theorem 2.1, by the time it starts recognizing L. Hence the closure of $R^H(k)$ under marked concatenation 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.
(ii) Now let L be any language in $R^H(k)$. It is easy to see that $L \cup L * A_k \in R^H(k)$. Consider the language $L' = (L \cup L*A_k)^*$, and apply a similar argument as in (i). □

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 $R^H(4k-4)$. The following result settles a conjecture by ROSENBERG [1967].

LEMMA 3.3. R(k) *is not closed under union or intersection, for* $k > 0$. *If* $k_1 + k_2 \geq 1$ *and 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

1) The markers in an input, due to marked concatenation or marked Kleene star, serve to indicate the beginning of a new task. Accordingly, it seems reasonable to assume that recognizing RTTMs *ignore*, subsequent to reading such a marker, the garbage left on the storage tapes by the preceding computation segment. Under these conditions the proofs of Lemma's 3.2 and 3.9 hold.

defined as:

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

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

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 $\overline{A_{k_1+k_2}} \in$ $\in R(k_1+k_2) - R(k_1+k_2-1)$, $\bar{L}_1 \in R(k_1)$ and $\bar{L}_2 \in R(k_2)$. Hence $\bar{L}_1 \cap \bar{L}_2 = \overline{A_{k_1+k_2}} \in$ $\in R(k_1+k_2) - R(k_1+k_2-1)$. It remains to be proven that for $A \in R(k_1)$ 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. □

Since R is closed under the Boolean operations (which also follows from the above Lemma) we can generate infinite proper hierarchies of language families by taking closures of $R(k_1)$ and $R(k_2)$ with respect to $\cap$ and $\cup$, all of which are included in R.

Since $A_{k+1} \notin R^H(k)$ for all $k \geq 0$, we also obtain the analogue for multihead tape units.

COROLLARY 3.4. *If* $k_1 + k_2 \geq 1$ *and we take* $A \in R^H(k_1)$ *and* $B \in R^H(k_2)$ *then* $A \cup B$, $A \cap B \in R^H(k_1+k_2)$, *but not necessarily* $A \cup B$, $A \cap B \in R^H(k_1+k_2-1)$.

The only remaining operation, investigated by ROSENBERG, under which R is closed, and with respect to which the status of R(k) is open, is the inverse real-time transducer mapping.

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 $w \in A_{k_1+k_2}$. □

COROLLARY 3.6. *The closure of* $R^H(k_1)$ *under inverse* k_2*-head* RTTM *mapping is contained in* $R^H(k_1+k_2)$ *but not in* $R^H(k_1+k_2-1)$.

An important operation, not treated in ROSENBERG [1967], is the shuffle operation.

LEMMA 3.7. 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,

$$L' = \text{shuffle}\ (\{x2h(x^R) \mid x \in \{0,1\}^*, h(0)=a \text{ and } h(1)=b\}, \Sigma^*) \cap \Sigma^* 2 \{a,b\}^*,$$

with

$$\{x2h(x^R) \mid x \in \{0,1\}^*,\ h(0)=a \text{ and } h(1)=b\} \in R(1) \text{ and } \Sigma^* 2 \{a,b\}^* \in R(0).$$

Since $L' \notin R$ and $\Sigma^* 2 \{a,b\}^*$ is regular, the shuffle component of L' does not belong to R either. □

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 3.3. 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

LEMMA 3.8. *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* $R(k_1+k_2-1)$. *Analogously, the Lemma holds for the corresponding multihead* RTTM's.

There is a deterministic context free language not in R, cf. ROSENBERG [1967]. It is easy to see that $\{a^n b^n c^n \mid n \geq 1\}$ is in R(1) and that furthermore $R \subset$ DLBA. Hence Figure 4 gives the inclusion diagram.

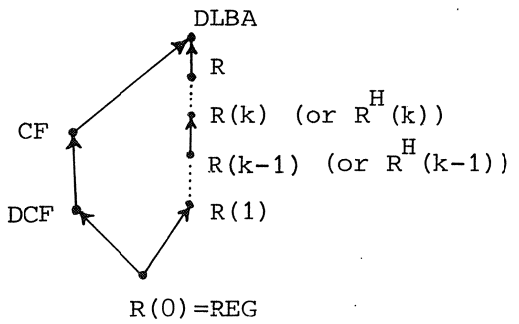

Figure 4. The position of R(k) (or $R^H(k)$), $k \geq 1$, in the linguistic hierarchy. (No connection by a directed sequence of arrows means incomparable).

It is not difficult to show that all nonclosure results for R, derived by ROSENBERG, hold by the same counterexamples for R(1), and therefore for each R(k), $k \geq 1$. We also mention that already R(1) contains inherently ambiguous context free languages, viz., $\{a^i b^i c^j \mid i,j \geq 1\} \cup \{a^i b^j c^j \mid i,j \geq 1\}$ is inherently ambiguous context free and easily recognized by a 1-RTTM.

In SAVITCH and VITÁNYI [1977] the jump Turing machine was introduced. A k-head jump Turing machine is a k-head Turing machine, where at each step

the k heads may be redistributed over the currently scanned tape squares, i.e., instantaneous head-to-head jumps are allowed, irrespective of the distances in between the heads concerned. It was shown that a k-head jump Turing machine can be simulated in linear time 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. The next Lemma shows that these matters are related.

LEMMA 3.9.[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* $R^H(k) \subset R^J(k)$.

PROOF. (i) The "if" part is obvious. Now suppose that $R(k) \subset R^J(k)$ and $R^H(k) = R^J(k)$. Let L be a language in $R^H(k) - R(k)$. By first feeding A_k we can always reduce a k-head RTTM to a k-tape RTTM. Hence $A_k \cup A_k * L \notin R^H(k)$. But clearly $A_k \cup A_k * L \in R^J(k)$, since the heads may jump together when the machine reads the marker. Therefore, the assumption leads to a contradiction and the "only if" part holds.
(ii) is proved similarly. □

The above Lemma is clearly due to the fact 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, as is R(k), whereas the closure of $R^H(k)$ under these operations is equivalent to the equality of $R^H(k)$ and R(k).[1)]

4. REAL-TIME 2-WAY MULTIHEAD FINITE AUTOMATA WITH AND WITHOUT JUMPS

Recall that we saw before that KOSARAJU [1979] has shown that the jump Turing machine as defined in SAVITCH & VITÁNYI [1977] may be simulated in real-time by multitape Turing machines. Hence $R^J = R$ (where $R^J = \bigcup_{k=1}^{\infty} R^J(k)$). In this section we show that for 2-way multihead finite automata the head-to-head jump facility does extend the class of languages accepted in real-

time. Incidentally, this shows also that the class of languages accepted by real-time 2-way multihead finite automata is strictly included in R. To obtain the result, we give several example languages which are acceptable in real-time by 2-way 2-head finite automata with jumps, but not by any real-time 2-way multihead finite automaton without jumps. Hence these languages belong to R, and constitute nontrivial examples of the power of the head-to-head jump option. Let in the following $h: \{0,1,\bar{0},\bar{1}\}^* \to \{0,1\}^*$ be a homomorphism which is defined by $h(\bar{a}) = h(a) = a$ for $a \in \{0,1\}$.

$$L_1 = \{\bar{\bar{w}}\bar{\bar{v}}aav^R \mid \bar{\bar{w}}\bar{v} \in \{0,1,\bar{0},\bar{1}\}^*, v \in \{0,1\}^*, a \in \{0,1\}, h(\bar{v}) = v\};$$

$$L_2 = \{\bar{w}b\bar{\bar{u}}cva \mid \bar{\bar{w}}\bar{u} \in \{0,1,\bar{0},\bar{1}\}^*, v \in \{0,1\}^*, \bar{c} \in \{\bar{0},\bar{1}\}, |\bar{u}| = |v|,$$

$$a \in \{0,1\}, b \in \{0,1,\bar{0},\bar{1}\}, h(b) = a\}.$$

The reader will easily figure out more complicated examples along these lines. Note that L_1, L_2 are linear context free but not deterministic context free.

LEMMA 4.1. *L_1, L_2 are accepted by real-time 2-way 2-head finite automata with jumps.*

PROOF. Let M be a 2-way 2-head finite automaton with jumps as follows. The front head reads from left to right one letter at a time. Whenever this first head reads a barred letter it calls the second head to its present position. This second head starts reading from right to left one letter at a time. So M is able to recognize L_1. A minor variation of M can recognize L_2. □

LEMMA 4.2. *L_1,L_2 are not accepted by any real-time 2-way multihead finite automation without jumps.*

PROOF. We prove the Lemma for L_1. Suppose L_1 is recognized by a k-head real-time 2-way finite automaton M_k but not by any (k-1)-head one. Since L_1 is not regular, such a k must be greater than 1. Since M_k is real-time, there must be at least one head which moves right at each step. For each

constant c we can find an input word w such that, during the processing of w by M_k, some head lags behind the vanguard head more than c squares. If this were not so, then all heads are at all times within c squares of the vanguard head, and we could replace M_k by an ordinary finite automaton with a finite-state control $Q^* = Q \times \{0,1,\bar{0},\bar{1}\}^c \times \{0,1,\ldots,c-1\}^{k-1}$, where Q is the finite-state control of M_k, which keeps track of the symbols under the k-1 nonvanguard heads of the simulated machine. This would imply that L_1 is regular: contradiction. Since by assumption L_1 is not recognizable by a (k-1)-head real-time 2-way finite automaton, for each constant c we can find an input word w such that, during the processing of w by M_k, all k-1 heads lag behind the vanguard head more than c squares. For suppose this were not the case. Since the vanguard head moves right at each step, at least one particular head must be at all times within c squares of the vanguard head, and similarly to above, we would be able to replace M_k by an (k-1)-head machine M^*_{k-1} with a finite-state control $Q^* = Q \times \{0,1,\bar{0},\bar{1}\}^c \times \{0,1,\ldots,c-1\}$ which keeps also track of the symbol under the neighboring head of the vanguard head. Contrary to the assumption, this would imply the falsehood of the Lemma for k-1. So suppose that, subsequent to processing an input prefix, all other heads of M_k lag behind the vanguard head more than c squares, and the vanguard head now starts to read suffix $w \in \{0,1,\bar{0},\bar{1}\}^*$, $|w| \leq c+1$. Hence, no other head of M_k will ever scan a symbol from w. Let the input prefix, which forces the k-1 nonvanguard heads more than c squares behind the vanguard head, be v. At time $|v| + 1$, all these k-1 heads scan a particular element of v. Now consider a suffix ensemble $W = \{0,1\}^{c/2}\{\bar{0}\}\{0,1\}^{c/2}$. The number of distinct positions on v of these k-1 heads, multiplied by the number of distinct states of the finite control M_k can attain when the vanguard head crosses $\bar{0}$, is bounded above by $(c/2)^{k-1} \times \#Q$. The number of prefixes in $\{0,1\}^{c/2}$ is $2^{c/2}$. If $2^{c/2} \geq (c/2)^{k-1} \times \#Q$, which happens for c large enough, two distinct such prefixes, say u_1 and u_2, lead to the same instantaneous description of M_k after processing vu_1 and vu_2. Therefore, M_k accepts either both $vu_1\bar{0}u_1^R$ and $vu_2\bar{0}u_1^R$ or rejects them both. Since $u_1 \neq u_2$ it follows that M_k does not accept L_1. The proof that L_2 is not accepted by any real-time 2-way multihead finite automaton proceeds similarly. □

Hence we have:

THEOREM 4.3. (i). *There are languages recognized by real-time 2-way 2-head finite automata with jumps which are not recognized by any real-time 2-way multihead finite automaton without jumps.*
(ii) *The class of languages accepted by real-time 2-way k-head finite automata with jumps properly includes the class of languages accepted by such automata without jumps.*

Computations of 1-way multihead finite automata have been considered by YAO & RIVEST [1978]. They show that k+1 heads are better than k heads for both the deterministic and the nondeterministic versions of the machine. Furthermore, they show that the k-head nondeterministic variety is strictly more powerful than the k-head deterministic one. Recently, JANIGA [1979] studied the analog questions for 2-way real-time multihead deterministic (resp. nondeterministic) finite automata, from now on called 2DRTFA and 2NRTFA, respectively. He obtained, mutatis mutandis, the same results for the 2-way real-time machines as did YAO and RIVEST for the 1-way (no time limit) variety. Whereas the latter used "palindromes" of $\binom{k}{2}$ strings to obtain their result, for the 2-way real-time case the former employed strings of k palindromes. E.g., let PALM be the set of palindromes in $\{0,1\}^*\{2\}\{0,1\}^*$. Let $P_k = (\text{PALM}\{*\})^k$. Then P_k is recognized by a (k+1)-head 2DRTFA but not by any k-head 2NRTFA. $\{0,1,2,*\}^* - P_k$ is accepted by a 2-head 2NRTFA but not by any k-head 2DRTFA. Now consider the language $P = \bigcup_{k=1}^{\infty} P_k$. It is easy to see that P is recognized by a 2-head 2DRTFA with jumps, but that P is not accepted by any multihead 2NRTFA without jumps because of JANIGA's result. Therefore we have:

THEOREM 4.4. *The class of languages accepted by* k*-head* 2NRTFA *with jumps properly includes the class of languages accepted by* k*-head* 2NRTFA *without jumps,* $k \geq 2$. *The same holds for* 2DRTFA*'s (i.e. Theorem* 4.3*).*

Another matter which we would like to decide is the power of jumps versus nondeterminism for the machines.

THEOREM 4.5. *There is a language acceptable by a* 2*-head* 2NRTFA *which is*

not acceptable by any multihead 2DRTFA *with jumps.*

PROOF. The language L in the proof of Lemma 3.7 was not in R, and hence, by KOSARAJU's [1979] result, is not acceptable by any multihead 2DRTFA with jumps. It is easy to see how L can be accepted by a 2-head 2NRTFA. □

The only question remaining seems to be whether (k+1)-head 2DRTFA's with jumps are more powerful than k-head 2DRTFA's with jumps, and the same matter for the nondeterministic versions. For a proof we might use the language J_k over the alphabet

$$\Sigma = \{0,1\} \times F \times M \times Q,$$

where

$$F = \{f \mid f \text{ is a total function } f\colon \{0,1\}^k \times Q \to \{0,1\}\},$$

$$M = \{m \mid m \text{ is a total function } m\colon \{1,2,\ldots,k\} \times Q \to$$

$$\to \{\text{left, right, no move}\} \text{ and } m(1,q) = \text{right}$$

$$\text{for all } q \in Q\}.$$

The interpretation is as follows. J_k is recognized by a k-head 2DRTFA M with state set Q. Suppose M has an input $s_1 s_2 \ldots s_i s_{i+1} \ldots s_n$ on its tape, $s_i = (a_i, f_i, m_i, q_i) \in \Sigma$, $1 \le i \le n$. At the i-th step the vanguard head 1 of M reads s_i in state $q_{i-1} \in Q$ and outputs $f_i(a_{j1}, a_{j2}, \ldots, a_{jk}, q_{i-1})$ where a_{jh} is the first element of the symbol read by the head h at that moment, $1 \le h \le k$. Subsequently, M repositions head h according to $m_i(h,q_i)$, $1 \le h \le k$, and enters state q_i.

THEOREM 4.6. J_{k+1} *is accepted by a* (k+1)*-head* 2DRTFA *but not by any* k*-head* 2NRTFA *with jumps. Hence* (k+1)*-head* 2DRTFA (2NRTFA) *with jumps are strictly more powerful than* k*-head* 2DRTFA (2NRTFA) *with jumps.*

PROOF. k=1. 1-head 2NRTFA accept only regular sets, and J_2 is not regular.

(It is easy to find a regular restriction on J_2 which yields a language isomorphic with $\{0^n1^n \mid n \geq 1\}$.)

$k > 1$. By using the m-element of the input symbols we can always necessitate the comparison of k+1 pieces of input which are arbitrary far apart and, say, of length x. Hence we would have to compare k+1 nonoverlapping words over {0,1} of length x, using k heads, in x steps. However, in x steps the k-head 2NRTFA with jumps can access at most $kx + \lceil \log|Q| \rceil$ bits of relevant information which is smaller than (k+1)x if we choose x large enough. □

If we take J'_k equal to J_k but without "left" in the range of $m \in M$ we can similarly prove:

COROLLARY 4.7. J'_{k+1} *is accepted by a* (k+1)*-head* 1DRTFA *but not by any* k*-head* 1NRTFA *with jumps. This implies that all inclusions according to the number of heads in the* 1XRTFA *are proper, where* $X \in$ {D,N,D *with jumps*, N *with jumps*}.

All results above hold whether or not we assume end markers, or that the heads can detect coincidence.

We think that Theorem 4.3 also holds for the corresponding Turing machine versions which are allowed to modify the contents of each square on the storage tapes but a bounded number of times, for some fixed constant bound.

Resuming, we obtain, for each $k \geq 1$, the inclusion graph of Figure 5. All inclusions are proper. Classes which are not connected by a sequence of directed arrows are incomparable. Hence we see that there are 3 distinct parameters: determinism-nondeterminism, no jumps-jumps, and the number of heads. We observe that jumps + nondeterminism cannot make up for an additional head; additional heads + nondeterminism cannot make up for jumps; and jumps + additional heads cannot make up for nondeterminism. The same picture holds for the one-directional variant. If we consider one-direction versus two-direction as an additional parameter, we observe that, similar to before, a gain in power according to one of the four of these parameters, cannot be compensated for by gain in power according to the remaining three.

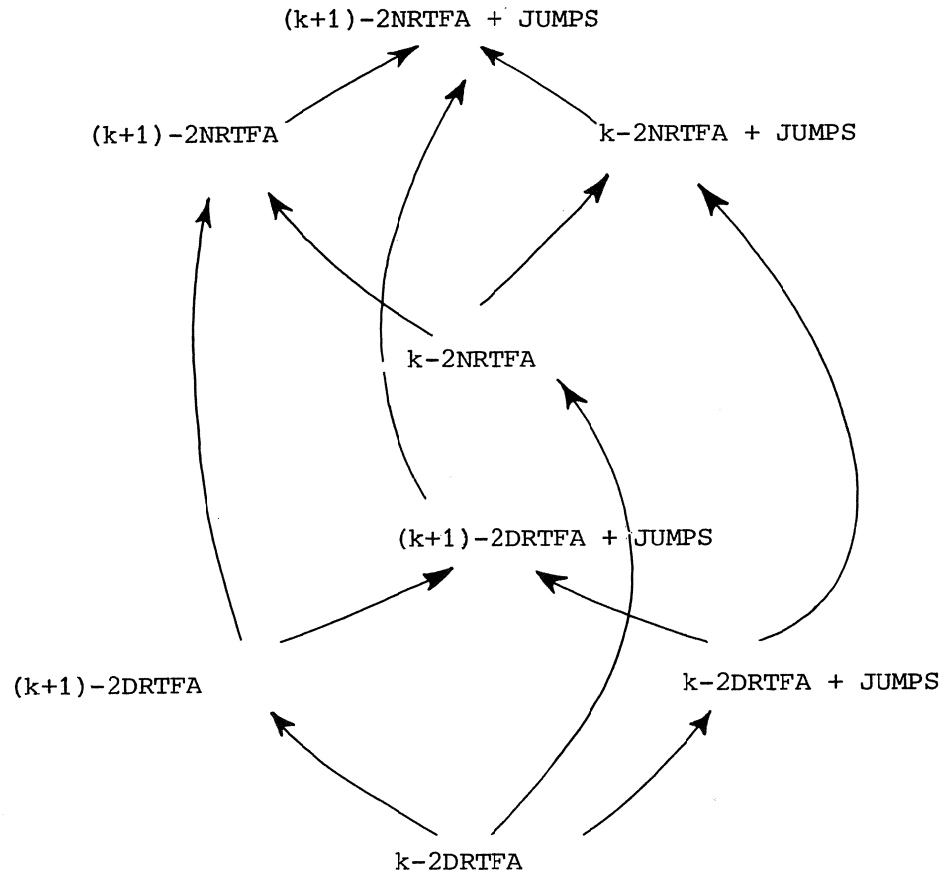

Figure 5. Inclusion diagram for the computing power of real-time 2-way multihead finite automata according to number of heads, determinism and jump option.

Below we shall briefly consider the *writing* deterministic variants of the discussed automata. We denote these devices by putting a "W" before "FA" in the used acronym. It is clear that such devices are very near in computing power to multihead RTTM's, and in fact (where $<(\leq)$ means "less (or equally) powerful"):

(*) (k-1)-head RTTM $\leq$ k-head 2DRTWFA $\leq$ k-head RTTM,

and at least one of the inclusions in (*) is proper by Theorem 2.1. Since a k-head RTTM is, but for the initial inscription of the input on the storage tape, the same as a (k+1)-head 2DRTWFA with a read-only vanguard head, it is to be assumed that both inclusions above are proper.

LEMMA 4.8. k-*head* 2DRTWFA $<$ k-*head* RTTM, *for all* $k \geq 1$. (*For* $k = 0$ *the Lemma has no meaning.*)

PROOF. The base case $k = 1$ is obvious, since 1-head 2DRTWFA's accept only regular sets and there is a 1-RTTM which accepts the nonregular language A_1. We prove the Lemma by induction on the number of heads, viz., by showing $A_k \in$ k-head RTTM - k-head 2DRTWFA. Suppose the Lemma holds for $1,2,\ldots,k-1$ but not for k. Then A_k is accepted by a k-head 2DRTWFA M_k but, according to Theorem 2.1 and (*) not by any (k-1)-head 2DRTWFA. Therefore, like in the proof of Lemma 4.2, we can find, for each constant c, an input word v_c such that at the v_c-th step all k-1 nonvanguard heads lag behind the vanguard head at least c squares in the computation of M_k. Hence, for the input ensemble $\{v_c\}$ W, with

$$W = \{w \in \Sigma_k^* \mid |w| \leq c \quad \text{and for all } i,\ 1 \leq i \leq k,\ |w|_i \geq 0\},$$

where $|w|_i$ denotes the number of 0_i's and 1_i's subtracted by the number of P_i's in w, the following holds. M_k becomes in effect a (k-1)-head RTTM with a particular initial instantaneous description, viz, the state of the finite control, the contents of the first v_c squares and the distribution of the k-1 nonvanguard heads, subsequent to the processing of v_c, since on the input ensemble $\{v_c\}$ W the nonvanguard heads shall never read the vanguard heads writing on the W segment of the input. For the input ensemble $\{v_c\}$ W the task for M_k is to recognize $\{v_c\}$ W $\cap$ A_k, which set is equal to $\{v_c\}$ (W $\cap$ A_k). Hence this task for M_k entails the recognition of W $\cap$ A_k by a (k-1)-head RTTM M with the described initial instantaneous description. Observing that AANDERAA [1974] shows in fact that no (k-1)-RTTM M^* can distinguish between $A_k \cap W$ and $W-A_k$ (instead of, more weakly, between $A_k \cap \Sigma_k^c$ and $\Sigma_k^c - A_k$) for some large enough c depending only on M^*, while M^* may be assumed to have initially inscribed storage tapes, we can, similarly to the proof of Theorem 2.1, show that there is a word v in W which fools M^* and therefore M_k. Hence our assumption that M_k recognizes A_k leads to a contradiction and the Lemma is proven. □

From (*) together with Theorem 2.1 it follows that k-head 2DRTWFA $<$ (k+2)-head 2DRTWFA. Lemma 4.8 and (*) yield:

COROLLARY 4.9. k-*head* 2DRTWFA < (k+1)-*head* 2DRTWFA *for all* k ≥ 1, *the case for* k=1 *being obvious.*

For the lefthand inclusion of (*) it holds that for k=1 both classes are equal in recognition power. For k=2, however, the inclusion is proper.

LEMMA 4.10. (i) 0-RTTM = 1-head 2DRTWFA
(ii) 1-RTTM < 2-head 2DRTWFA.

PROOF. (i) is obvious.
(ii) VALIEV [1970] has shown, using RABIN's [1963] techniques, that $L = \{x2x \mid x \in \{0,1\}^* \text{ and } 2 \notin \{0,1\}\}$ cannot be recognized by 1-RTTMs. It is easy to see how L can be recognized by a 2-head 2DRTWFA (or by a 2-head 1DRTFA for that matter). □

The improvement of Lemma 4.10 to the general case does not seem easy. In fact, it seems about as hard as proving that k-head RTTMs are more powerful than k-RTTMs. A possible candidate to distinguish (k-1)-head RTTMs from k-head 2DRTWFA's is the following language:

$$V_{k-1} = A_{k-1} \cup \{w*v \mid w = a_1a_2\ldots a_m \in \Sigma_{k-1}^*,\ v \in \{0,1\}^*,\ |v| \leq |w| \text{ and for } v = b_1b_2\ldots b_\ell \text{ it holds that for } 1 \leq i \leq \ell,\ b_i = 1 \text{ if } a_1\ldots a_i \in A_{k-1} \text{ and } b_i = 0 \text{ if } a_1\ldots a_i \notin A_{k-1};\ \text{furthermore, for some } i,\ 1 \leq i \leq k-1,\ \text{the number of } P_i\text{'s in } w \text{ exceeds the number of } 0_i\text{'s and } 1_i\text{'s in } w\}$$

V_{k-1} can be recognized by a k-head DRTWFA as follows. At each step the vanguard head notes on an extra track whether the processed prefix belongs to A_{k-1}, while the remaining k-1 heads simulate the required k-1 pushdown stores. When the vanguard head reaches the marker * one of the stores must be empty, i.e., one head must be at the start square. This head now starts to compare the acceptance record against v, which is read by the vanguard head. Likely candidates for showing a difference between machine classes which are very close in recognition power will be the subject matter for the entire next section.

5. ON THE RELATIVE POWER OF TAPES, HEADS AND JUMP HEADS IN REAL-TIME TURING MACHINES

One of the major drawbacks in the game of showing a difference in power between two very similar machine types A and B such as considered in this paper, apart from the difficulties involved in giving a proof, is to find some likely candidates for showing a difference between type A and type B. RABIN's [1963] language in R(2)-R(1) did not generalize in an obvious way to show a difference between R(k+1) and R(k), $k > 1$. AANDERAA [1974] provided a uniform construction for a language in R(k+1)-R(k), $k \geq 1$. No likely candidates for showing the difference between, e.g., R(k) and $R^H(k)$ or $R^H(k)$ and $R^J(k)$ have been proposed, except possibly $\{xy2x \mid xy \in \{0,1\}^*\}$ for showing a difference between $R^H(2)$ and R(2). In the present section we propose to fill this gap, besides proving some facts about the candidates. The only languages known to be in R-R(k) are the languages A_{k+i}, $i \geq 1$, but unfortunately these languages are not in $R^H(k)$ either. PAUL, SEIFERAS and SIMON [1980] have, independently, by a method similar to the one introduced in VITÁNYI [1979] (cf. Theorem 2.1), shown that A_{k+1} is not recognized by a k-head RTTM with head-to-head jumps either, and therefore $R^J(k) \subset R^J(k+1)$, for all k, and $A_{k+1} \notin R^J(k)$. (This, by the way, implies the analogs of Lemma 4.8 and Corollary 4.9 for the jump versions of the machines occurring there). Hence the only candidates of which we have negative results are not acceptable either by placing all heads on the same tape nor by adding the jump option. From the existing simulation results it is also clear that there cannot be a single language L which is acceptable by some k-head (jump) RTTM but not by any multitape (multihead) RTTM, thus proving the required results by a single example as in section 4. Now consider a language which is like A_k but with the extra requirement that at all times during the processing of the input w by a k stack machine at least 2 of the stacks are of equal length for w to be accepted. More formally, if $|v|_i$ denotes the number of 0_i's and 1_i's subtracted by the number of P_i's in v, then:

$$E_k = \{w \in \Sigma_k^* \mid w \in A_k \;\&\; \forall v \in \text{prefix}(w)\; \exists i,j \in \{1,2,\ldots,k\} \; [i \neq j \;\&\; ||v|_i - |v|_j| \leq 1]\}$$

LEMMA 5.1. $E_k \notin R(k-2), R^H(k-2), R^J(k-2)$.

PROOF. Suppose, by way of contradiction, that the (k-2)-RTTM M accepts E_k. Now change M to a (k-2)-RTTM M^* which accepts A_{k-1} by having the finite control of M, for every letter $0_{k-1}, 1_{k-1}, P_{k-1}$ read $0_{k-1}0_k, 1_{k-1}1_k, P_{k-1}P_k$, respectively, and speed up the storage handling as much as required. Then A_{k-1} is accepted by the (k-2)-RTTM M^* contradicting known results. $E_k \notin R^H(k-2)$ then follows by Theorem 2.1 and for $E_k \notin R^J(k-2)$ see the introduction of this section. □

(The case k = 2 above is obvious since E_2 is not regular.) Note that AANDERAA's proof does not show that $E_k \notin R(k-1)$ since the subset $S\Sigma_k^*$ used in AANDERAA's proof (which in fact shows that no (k-1)-RTTM can distinguish between $S\Sigma_k^* \cap A_k$ and $S\Sigma_k^* \cap (\Sigma_k^* - A_k)$) is disjoint from E_k.

LEMMA 5.2. $E_2 \in R(1)$, $E_3 \in R^H(2)$.

PROOF. $E_2 \in R(1)$ is obvious. $E_3 \in R^H(2)$: keep the 3 stacks on different tracks of the recognizing 2-head RTTM M. Whenever there is a change in pairs of equal size stacks, all 3 stacks must be of equal length, otherwise we reject the input. Both heads of M therefore come together with everything to the right of them blank, and therefore the role of the "fat" head, maintaining 2 tracks, can change. □

We conjecture that $E_3 \notin R(2)$. To prove this conjecture also would prove that $R(2) \subset R^H(2)$, a well-known open problem. In general we conjecture that $E_k \notin R(k)$, $k \geq 2$, which for the case k = 3 would show that the LEONG-SEIFERAS simulation is optimal for 2 heads. By Lemma 5.1 and the fact that a multihead machine can detect coincidence we have that

LEMMA 5.3. $E_k \in R^H(k) - R^H(k-2)$.

LEMMA 5.4. $E_k \in R^J(k-1)$ *for all* $k > 1$.

PROOF. Keep the k stacks concerned on k different tracks of the (k-1)-head jump RTTM M recognizing E_k. Since at all times, during the processing of a candidate word $w \in \Sigma_k^*$, 2 of the tracks must contain stacks of equal height,

at any time the pair of tracks concerned changes somewhere a pair of equal height tracks must be formed, thus freeing a head which can jump to the place where it is needed. If at any time all stack heights diverge M rejects the prefix processed and also every addition to it. □

COROLLARY 5.5. $E_k \in R^J(k-1) - R^J(k-2)$.

We conjecture that E_k cannot be recognized by a (k-1)-head RTTM for $k \geq 4$. A proof of this fact would show that $R^H(k) \subset R^J(k)$ for $k \geq 3$, leaving open the case $k = 2$. Although we have an upper bound on the recognition of E_k by multihead RTTM's (with respect to the number of heads needed) we have not yet a good upper bound for recognition by multitape RTTM's except by the crude $E_k \in R(4k-4)$ offered by Lemma 5.3 and the LEONG-SEIFERAS' result.

LEMMA 5.6. $E_2 \in R(1)$; $E_3 \in R(4)$; $E_k \in R(2k-2)$, $k \geq 3$.

PROOF. $E_2 \in R(1)$ is obvious. $E_3 \in R(4)$: we can check for inclusion in A_3 using 3 stacks and determine whether the difference between the lowest and the highest stack is less than 2 by a 4-th stack. The identity of the current pair of equal-size stacks can be maintained in the finite control. Similarly, we can keep track of the k-2 consecutive differences in height concerned in the acceptance of E_k by k-2 stacks, thus assuring $E_k \in R(2k-2)$, $k \geq 3$. □

To use k-2 stacks for keeping track of k-2 counters seems somewhat extravagant. It is known, P. FISCHER, MEYER and ROSENBERG [1968], that an arbitrary number of counters can be linear time simulated by a 1-tape Turing machine, but the real-time simulation is still an open problem. (Note, that the related origin crossing problem is solvable in real-time by a 1-tape Turing machine, M. FISCHER and ROSENBERG [1968b]). Hence, for the time being, Lemma 5.6 appears to be the best we can do.

We can generalize the above approach in several directions. For instance, by requiring that i of the k stacks have the same height at all times during the processing of the input. Formally,

$$E_{\binom{k}{i}} = \Big\{ w \in \Sigma_k^* \mid w \in A_k \;\&\; \forall v \in \text{prefix}(w)\ \exists j_1,j_2,\ldots,j_i \in \{1,\ldots,k\},\ j_1<j_2<\ldots<j_i \ [\,||v|_{j_\ell} - |v|_{j_m}| \le 1 \text{ for all } j_\ell, j_m \in \{j_1,j_2,\ldots,j_i\}\,]\Big\}.$$

These languages are especially suited to jump Turing machines since it is easily seen that:

LEMMA 5.8. $E_{\binom{k}{i}} \in R^J(k-i+1)$.

Furthermore,

LEMMA 5.9.

(i) $E_{\binom{k}{i}} \in R^J(k-i+1) - R^J(k-i)$

(ii) $E_{\binom{k}{i}} \in R^H(k-i+1) - R^H(k-i)$ *for* $i > k/2$

and

$E_{\binom{k}{i}} \in R^H(k) - R^H(k-i)$ *for* $i \le k/2$.

(iii) $E_{\binom{k}{i}} \in R(2k-i) - R(k-i)$.

The latter two Lemmas are proven similar to the previous Lemmas concerning E_k $(= E_{\binom{k}{2}})$. Some border cases with $i \ge k/2$ yield: $E_{\binom{5}{3}} \in R^H(3)$, $E_{\binom{5}{4}} \in R^H(2) \subset R(4)$, and, because of the real-time simulation of heads by tapes in LEONG and SEIFERAS [1977], it follows from Lemma 5.9(ii) that $E_{\binom{k}{i}} \in R(4(k-i))$ for $i > k/2$ and therefore, e.g., $E_{\binom{k}{3k/4}} \in R(k)$ which is better than what we obtain from Lemma 5.9(iii).

Looking at the above we see there is a relation between the optimality of the real-time simulations of jump heads by heads, and heads by tapes, and how many tapes or heads are needed to recognize $E_{\binom{k}{i}}$. Let f(k) be the minimum number of tapes (heads) needed for simulating k jump heads in real-time. Then, if we need at least k tapes (heads) for accepting $E_{\binom{k}{i}}$, $i < k/2$,

then

$$f(k-i+1) \geq k.$$

Hence the conjecture that we need k or more tapes (heads) to recognize $E_i^{(k)}$ for $i < k/2$ can be dissolved if we can improve KOSARAJU's result to "less than 2 k tapes (heads) are necessary for the real-time simulation of k jump heads"

Yet another language sequence we might consider is $A_k - E_k$, $k \geq 1$. Since $A_k - E_k$ contains AANDERAA's subset $A_k \cap S\Sigma_k^*$, it follows that $A_k - E_k \notin R(k-1), R^H(k-1), R^J(k-1)$. We also see that $A_k - E_k \in R^H(k), R^J(k)$. With respect to acceptance by k-RTTM's the same upper bounds apply as argued for E_k. This is not so for the languages $A_k - E_k'$, where E_k' is like E_k but the condition of two stack heights being equal only holds at the end of the processing of the input word, i.e.,

$$E_k' = \{w \in \Sigma_k^* \mid w \in A_k \;\&\; \exists_{i \neq j} i,j \in \{1,\ldots,k\}[\,||w|_i - |w|_j| \leq 1]\}.$$

Here we have that $A_2 - E_2' \in R(3)$ but, presumably, that $A_2 - E_2' \notin R(2)$. By the now familiar reasoning, if the latter case is affirmative then $A_2*(A_2-E_2') \in R^J(2) - R^H(2)$, settling the question whether or not $R^H(2) \subset R^J(2)$.

Some of the candidates to try for solving the various questions met are given in the table below.

	$R(k) \subset R^H(k)$?	$R^H(k) \subset R^J(k)$?
k=2:	$L = \{xy2x \mid xy \in \{0,1\}^*\}$ E_3, $A_2 - E_2'$	$A_2*(A_2-E_2')$
arbitrary $k \geq 3$:	E_k, $A_k - E_k'$	E_{k+1}

Acknowledgements. *J. SEIFERAS pointed out to me that the earlier version of the proof of Theorem* 2.1 *in VITÁNYI* [1979] *may have been prone to (but wasn't)*

circularity of the argument. Discussions with W. SAVITCH were valuable for section 4.

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, M.J. & A.L. ROSENBERG (1968a), *Limited random acces Turing machines,* Proceedings 9-th Annual IEEE-SWAT Conference on Switching and Automata Theory, 356-367.

FISCHER, M.J. & A.L. ROSENBERG (1968b), *Real-time solutions to the origin-crossing problem,* Mathematical Systems Theory, 2, 257-263.

FISCHER, P.C., A.R. MEYER & A.L. ROSENBERG (1968), *Counter machines and counter languages,* Mathematical Systems Theory 2, 265-283.

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

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

HARTMANIS, J. & R.E. STEARNS (1965), *On the computational complexity of algorithms,* Trans. AMS 117, 285-306.

JANIGA, L. (1979), *Real-time computations of two-way multihead finite automata,* in: Fundamentals of Computation Theory (FCT '79) (L. Budach ed.), Akademie Verlag, Berlin, 214-218.

KOSARAJU, R. (1979), *Real-time simulation of concatenable double-ended queues by double-ended queues,* Proceedings of the 11-th Annual ACM Symposium on Theory of Computing, 346-351.

LEONG, B.L. & J.I. SEIFERAS (1977), *New real-time simulations of multihead tape units,* Proceedings of the 9-th Annual ACM Symposium on Theory of Computing, 239-248.

PAUL, W.J., J.I. SEIFERAS & J. SIMON (1980), *An information-theoretic approach to time bounds for on-line computation*. Techn. Rept. TR 64, Department of Computer Science, University of Rochester, Rochester, New York, February 1980.

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

ROSENBERG, A.L. (1967), *Real-time definable languages*, JACM 14, 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, 453-464.

VALIEV, M.K. (1970), *Certain estimates of the time of computations on Turing machines with an input*, Cybernetics 6 (1972), 737-741. Translated from Kibernetica 6 (1970), 26-32.

VITÁNYI, P.M.B. (1979), *Multihead and multitape real-time Turing machines*. Techn. Rept. IW 111, Mathematisch Centrum, Amsterdam, The Netherlands, June 1979.

YAO, A. & R. RIVEST (1978), k+1 *heads are better than* k, J ACM 25, 337-340.