MIT/LCS/TM-123

# AN IMPROVED PROOF OF THE RABIN-HARTMANIS-STEARNS CONJECTURE

Harold M. Perry

January 1979

MIT/LCS/TM-123

# AN IMPROVED PROOF OF THE RABIN-HARTMANIS-STEARNS CONJECTURE

Harold M. Perry

January 1979

This report was prepared with the support of a
National Science Foundation Grant No. MCS77-19754 A02

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

Keywords: Rabin-Hartmanis-Stearns conjecture
real time language recognition,
Turing machine, overlap argument

# AN IMPROVED PROOF OF THE RABIN-HARTMANIS-STEARNS CONJECTURE

by

HAROLD MARC PERRY

Submitted to the Department of Electrical Engineering and Computer Science on May 12, 1978 in partial fulfillment of the requirements for the Degrees of Master of Science and Electrical Engineer.

## ABSTRACT

We offer an improved presentation of Aanderaa's constructive proof of the Rabin-Hartmanis-Stearns conjecture:

For all $k \geq 2$, there exists a language $L_k$ such that $L_k$ can be recognized by a k-worktape real time Turing machine but cannct be recognized by any (k-1)-worktape real time Turing machine.

Name and Title of Thesis Supervisor: Albert R. Meyer

Professor of Computer Science and Engineering

TABLE OF CONTENTS

## INTRODUCTION

All Turing machine models recognize precisely the same class of languages. Intuition tells us however, that a well-equipped model, (e.g., a Turing machine with worktapes which have multiple heads) should have capabilities lacked by a less-equipped model, (e.g., a Turing machine with worktapes which have only one worktape head).

One way to measure such capabilities is by comparing the time it takes each machine model to recognize the same language. This thesis studies the effect of having extra worktapes in the computational process. It addresses such questions as:

i) Does having more worktapes make a machine model more efficient?

ii) If so, how many extra worktapes are needed to achieve a given gain in speed?

We shall answer the first question affirmatively. Moreover, we will prove that only one extra worktape need be added to make the Turing machine model faster and hence conclude:

For all $k \geq 2$, the class of k-worktape Turing machines are more efficient than the class of (k-1)-worktape Turing machines.

Our proof will consist in specifying a language $L_k$ and a k-worktape Turing machine $\mathfrak{M}$ such that $\mathfrak{M}$ can recognize $L_k$ faster than any Turing machine with (k-1)-worktapes.

## HISTORY OF THE PROBLEM

In 1960, Rabin [10] showed that the 2-worktape Turing machine model was more efficient than the 1-worktape model. His proof consisted in specifying a language $L_2$ and showing that it may be recognized faster by a 2-worktape Turing machine $\mathfrak{M}$ than by any 1-worktape Turing machine. $L_2$ was specified as follows:

$L_2 \overset{def}{=} \{UV\#U^R\} \cup \{UV\#V^R\}$ where $U \in \{0,1\}^*$, $V \in \{a,b\}^*$ and $R$ denotes reversal.

Rabin showed that $\mathfrak{M}$ may recognize whether an input was an element of $L_2$ in time equal to the amount of time it took to read the input, i.e., in "real time". (Given an input, $\mathfrak{M}$ need only store U on one worktape and V on the second worktape. After scanning #, $\mathfrak{M}$ would then be able to simultaneously check whether $U^R$ or $V^R$ followed.) It was then shown that no 1-worktape Turing machine could recognize $L_2$ as fast. Intuitively, no 1-worktape Turing machine could operate as fast as $\mathfrak{M}$ and still be able to achieve a configuration after scanning # that would allow it to check simultaneously whether $U^R$ or $V^R$ followed.

In 1965, Hartmanis and Stearns [6] conjectured a generalization of Rabin's result. This conjecture was finally proven by Aanderaa [1] fourteen years after the initial result.

Although Aanderaa's proof was impressive in its result, we found it difficult to read. The ideas and insight which led to the proof did not seem readily apparent. We offer what we believe are simplified proofs of several of the technical results as well as an improved exposition.

## PRELIMINARY DEFINITIONS

We assume the reader is familiar with the multitape Turing machine model. (See [2,7].) We may characterize a multitape Turing machine computation as follows:

Initially, the input word appears with endmarkers on the input tape. The input head is scanning the left endmarker while all the worktapes are blank. The input word is accepted if and only if the multitape Turing machine started in a designated initial state, makes a sequence of moves at the end of which it enters an accepting state. The language accepted is the set of input words so accepted.

We shall find it convenient to number the squares of the worktapes and the input tape. A tape may be numbered by assigning consecutive integers to the squares such that zero denotes the square initially scanned by the tape head, while positive and negative integers correspond to those squares to the right and left, respectively, of the square denoted by zero.

For the following definitions, we fix a multitape Turing machine $\mathfrak{M}$, with input alphabet $\Gamma$, worktape alphabet $\Sigma$, set $Q$ of internal states and k worktapes indexed 1 through k. It will be convenient to refer to the input tape as the 0th worktape. An integer j such that $0 \leq j \leq k$ will be called a **tape index** of $\mathfrak{M}$.

Let $W \in \Gamma^*$ denote an input word. The length of W shall be denoted by $L(W)$. We may therefore denote W as $w_1 w_2 \ldots\ldots w_{L(W)}$ where $w_i \in \Gamma$ for

$1 \leq i \leq L(W)$. The time of the computation of $\mathfrak{M}$ on W shall be denoted by $T_{\mathfrak{M}}(W)$.

For the rest of this section we let $s \in \mathbb{N}$ such that $0 \leq s \leq T_{\mathfrak{M}}(W)$. (We shall use s to refer to a step, i.e., an instantaneous description (i.d.) in the computation of $\mathfrak{M}$ on W.)

DEFINITION 1

Let j be a tape index of $\mathfrak{M}$. **POSN**$(j, W, s)$ is defined to be the number of the worktape square which is scanned by the jth tape head immediately after step s (i.e., the square scanned in the sth i.d. where the initial i.d. is counted as the zeroth) in the computation of $\mathfrak{M}$ on W.

DEFINITION 2

Let j be a tape index of $\mathfrak{M}$. Let $z \in \mathbb{Z}$. **CONT**$(j, z, W, s)$ is defined to be the **contents** of the zth square of the jth worktape immediately after step s in the computation of $\mathfrak{M}$ on W.

DEFINITION 3

**STATE**$(W, s)$ is defined to be the **internal state** of $\mathfrak{M}$ immediately after step s in the computation of $\mathfrak{M}$ on W.

We shall let **ID**$(W, s)$ denote the i.d. immediately after the sth step in the computation of $\mathfrak{M}$ on W. We can easily specify **ID**$(W, s)$ in terms of the previous definitions.

DEFINITION 4

The **instantaneous description** $ID(W,s)$ of $\mathfrak{M}$ on input W immediately after step s is specified as follows:

$ID(W,s) \overset{def}{=} (q, p_0, \ldots, p_k, \sigma, \delta_1, \ldots, \delta_k)$ where:

$$q = STATE(W,s),$$
$$p_j = POSN(j,W,s) \text{ for } 0 \leq j \leq k,$$
$$\sigma = CONT(0, p_0, W, s), \text{ and}$$

for $1 \leq j \leq k$, $\delta_j : \mathbb{Z} \to \Sigma$ such that $\delta_j(z) = CONT(j,z,W,s)$ for $z \in \mathbb{Z}$. We remark that for almost all $z \in \mathbb{Z}$, $\delta_j(z)$ is equal to the blank symbol.

The reader may consult [2, 7] for an explanation of how a Turing machine computation may be formalized as a sequence of instantaneous descriptions and how the time of such a computation may be defined as the index of the last instantaneous description in this sequence.

DEFINITION 5

Consider the computations of $\mathfrak{M}$ on inputs W and $\hat{W}$. Let $s, \hat{s} \in \mathbb{N}$ be such that $0 \leq s \leq T_{\mathfrak{M}}(W)$ and $0 \leq \hat{s} \leq T_{\mathfrak{M}}(\hat{W})$. Let $n \in \mathbb{N}$ and let $j > 0$ be a tape index of $\mathfrak{M}$. We say that $ID(W,s)$ and $ID(\hat{W},\hat{s})$ are **n-equivalent on worktape j** if the following conditions hold:

i) $STATE(W,s) = STATE(\hat{W},\hat{s})$,

ii) $POSN(j,W,s) = POSN(j,\hat{W},\hat{s})$, and

iii) $(\forall m(-n \leq m \leq n))$

$$[CONT(j, POSN(j,W,s)+m, W, s) = CONT(j, POSN(j,\hat{W},\hat{s})+m, \hat{W}, \hat{s})].$$

That is, immediately after step s in the computation of $\mathfrak{M}$ on W and immediately after step $\hat{s}$ in the computation of $\mathfrak{M}$ on $\hat{W}$ we have that:

i) the internal states are equal,

ii) the $j^{th}$ worktape head is scanning the same tape square, and

iii) the information accessible to the $j^{th}$ worktape head in the next n steps is equivalent.

We say that **ID**(W, s) and **ID**($\hat{W}$, $\hat{s}$) are **n-equivalent** if **ID**(W, s) and **ID**($\hat{W}$, $\hat{s}$) are n-equivalent on all worktapes.

We now offer a formalism by which specific parts of an input word may be identified.

DEFINITION 6

Let $l, m \in \mathbb{N}$ such that $0 \leq l \leq m$. An **interval** I is defined to be the set of natural numbers $\{n | l \leq n \leq m\}$, which we also denote as $[l, m]$.

For convenience, we shall define **MIN**(I) to equal l and **MAX**(I) to equal m. The length of interval I, $|I|$, is defined to equal **MAX**(I)-**MIN**(I)+1. For completeness, we shall specify an empty interval equal to $\phi$ such that $|\phi| = 0$ and **MAX**($\phi$) and **MIN**($\phi$) are undefined.

DEFINITION 7

Let I and J be intervals.

i) If **MIN**(J)=**MAX**(I)+1 then I and J are said to be **adjacent**. In this case, $I \cup J$ is equal to the interval, [**MIN**(I),**MAX**(J)].

ii) J is said to be a **subinterval** of I if $J \subset I$.

<u>DEFINITION 8</u>

i) The interval I is said to be an **input interval** of $W = w_1 \cdot \ldots \cdot w_{L(W)}$ if $1 \leq \mathrm{MIN}(I)$ and $\mathrm{MAX}(I) \leq L(W)$.

ii) Let I be an input interval of W. W **at** I (written W@I) is defined to be the word $w_{\mathrm{MIN}(I)} \cdot \ldots \cdot w_{\mathrm{MAX}(I)}$. We note that $W@\phi = \lambda$ (where $\lambda$ denotes the empty string.)

We conclude this section by presenting an information-theoretic concept by which we may characterize the retrieval of stored information. Consider the computation of $\mathfrak{M}$ on input W. Stored information will be retrieved precisely when some worktape head revisits a tape square. We shall refer to such a happening as an **overlap event**. Formally, we define an overlap event as follows:

<u>DEFINITION 9</u>

Let $s, t \in \mathbb{N}$ such that $0 \leq s < t \leq T_{\mathfrak{M}}(W)$. Let j be a tape index of $\mathfrak{M}$. Let $\sigma \in \Sigma \cup \Gamma$ and let $z \in \mathbb{Z}$. The 5-tuple $(s, t, j, z, \sigma)$ is said to be an **overlap event** in the computation of $\mathfrak{M}$ on W if the following conditions hold:

i) $\mathrm{POSN}(j, W, s) = z$ and $\mathrm{POSN}(j, W, s+1) \neq z$,

ii) $\mathrm{CONT}(j, z, W, s+1) = \sigma$, and

iii) $(\min x \mid s < x \leq T_{\mathfrak{M}}(W))\,[\mathrm{POSN}(j, W, x) = z] = t$.

Informally, the overlap event $(s,t,j,z,\sigma)$ occurs if: immediately after step s, the zth square of the jth worktape is being scanned. At the next step, the tape head moves off this square leaving symbol $\sigma$ written. Step t is the first subsequent step where the zth square on the jth worktape is revisited.

We call $\sigma$ the **overlap value** and $(j,z)$ the **overlap location** associated with the overlap event $(s,t,j,z,\sigma)$. Finally, t is called **the step at which the overlap event occurred.**

We shall denote the set of overlap events in the computation of $\mathfrak{M}$ on W by OVERLAP(W).

## REAL TIME LANGUAGE RECOGNITION

### DEFINITION 10

$\mathfrak{M}$ is said to be a **real time Turing Machine** if it scans a new input symbol during each step of its operation. ($\mathfrak{M}$ must halt once the right endmarker is scanned on its input tape.) A language is said to be recognized in **real time** if it is recognizable by a real time Turing machine.

For the rest of this section we shall assume that $\mathfrak{M}$ is a fixed k-worktape real time Turing machine with input alphabet $\Gamma$. Let W be an input word to $\mathfrak{M}$ such that $L(W)>1$.

We make the following trivial observations:

i) The head on the input tape of $\mathfrak{M}$ must be one-way. Thus, no overlap events can occur on the input tape.

ii) $T_{\mathfrak{M}}(W)=L(W)+1$.

iii) $(\forall s \mid 0\leq s\leq T_{\mathfrak{M}}(W))[POSN(0,W,s)=s]$. Thus if I is an input interval of W, the set of indices of steps where $\mathfrak{M}$ is scanning some part of W@I equals I.

We remind the reader that the definitions presented in the previous section apply to all mutitape Turing machine models. The following definitions become relevant under the real time constraint.

### DEFINITION 11

Let I be an input interval of W. Let j be a tape index of $\mathfrak{M}$.

**DSPM**(j,W,I) is defined to be the **displacement** incurred by the jth tape head while $\mathfrak{M}$ was processing W@I. Formally, **DSPM**(j,W,I) is defined to equal $|$**POSN**(j,W,**MIN**(I))-**POSN**(j,W,**MAX**(I)+1)$|$. Notice for example, that **DSPM**(0,W,I)=$|$I$|$.

We now formalize our notion of the **information retrieval activity** of $\mathfrak{M}$.

DEFINITION 12

Let I,J be adjacent input intervals of W. We define the **adjacent overlap events of W during I and J** to be the set

$$\{(s,t,j,z,\sigma)\in\textbf{OVERLAP}(W)\mid \textbf{MIN}(I)\le s\le\textbf{MAX}(I)<t\le\textbf{MAX}(J)\}.$$

We shall denote this set as **ADJ-OVERLAP**(W,I,J). Notice that **ADJ-OVERLAP**(W,I,J) consists of those overlap events where information was first stored while $\mathfrak{M}$ was processing W@I and then retrieved while $\mathfrak{M}$ was processing W@J.

DEFINITION 13

Let I be an input interval of W such that $|$I$|$>1. We define the **DIVIDING POINT of W during I** (denoted by **DP**(W,I)) to be the least natural number l (**MIN**(I)$\le$l<**MAX**(I)), such that $|$**ADJ-OVERLAP**(W,[**MIN**(I),l],[l+1,**MAX**(I)])$|$ is maximized. For convenience, we shall denote the set **ADJ-OVERLAP**(W,[**MIN**(I),**DP**(W,I)], [**DP**(W,I)+1,**MAX**(I)]) as **SP-OVERLAP**(W,I).

In the argument to follow, $|$**SP-OVERLAP**(W,I)$|$ will serve to capture our notion of the **information retrieval activity** of $\mathfrak{M}$ while it processes W@I. It is for this reason that we shall refer to the elements of **SP-OVERLAP**(W, I) as **special overlap events.**

<u>LEMMA 1</u>

Let I be an input interval of W. Then

i) $|\mathbf{OVERLAP}(W)| \leq k\mathbf{L}(W)$, and

ii) $|\mathbf{SP\text{-}OVERLAP}(W,I)| \leq k|I|$.

<u>Proof:</u>

i) For any given worktape and step of $\mathfrak{M}$, at most one overlap event can occur. Since there are at most $\mathbf{L}(W)$ steps for which an overlap event can occur we have that $|\mathbf{OVERLAP}(W)| \leq k\mathbf{L}(W)$.

ii) For any given worktape of $\mathfrak{M}$, there are at most $|I|$ steps for which an overlap event can occur while the input head is scanning some part of W@I. This implies that $|\mathbf{SP\text{-}OVERLAP}(W,I)| \leq k|I|$.□

We next demonstrate that for any set $S \subset \Gamma^N$ for sufficiently large $N \in \mathbb{N}$, while $\mathfrak{M}$ is processing the words of S there must exist some large subinterval I of [1, N] for which:

$$\underset{W\in S}{\text{mean}} \; |\mathbf{SP\text{-}OVERLAP}(W,I)| \text{ is a small fraction of } |I|.$$

That is, on the average $\mathfrak{M}$ will exhibit relatively small information retrieval activity while processing the words of S at I. We shall argue that if $\underset{W\in S}{\text{mean}} \; |\mathbf{SP\text{-}OVERLAP}(W,I)|$ was a large fraction of $|I|$ for all large subintervals I of [1,N], then $\operatorname{mean}_{W\in S}$ |**OVERLAP**(W)| would be greater than kN and thus contradict lemma 1.

Let $N=pn^{2(n-1)}$ for some integer $p\geq 1$ and some integer $n>1$. Let $W\in\Gamma^N$.

We define a set $\mathfrak{I}$ of input intervals of W as follows:

for $0\leq i\leq 2(n-1)$ and $0\leq j<N/(pn^i)$, we define $I_{ij}=[jpn^i+1, (j+1)pn^i]$.

$\mathfrak{I}$ is then defined to be the set of input intervals

$$\{I_{ij} \mid 0\leq i\leq 2(n-1) \text{ and } 0\leq j<N/(pn^i)\}.$$

For convenience, we define for $0\leq i\leq 2(n-1)$ the set $\mathfrak{I}_i$ of input intervals at **level i** to be $\{I_{ij}\in\mathfrak{I} \mid 0\leq j<N/(pn^i)\}$.

We observe the following:

i) Level i is a partition of [1,N] into $N/(pn^i)$ intervals, each of size $pn^i$.

ii) Each interval at level i+1 consists of n consecutive intervals at level i, viz., $I_{i+1,j}$ is the disjoint union of $I_{im}$ such that $nj\leq m<n(j+1)$.

iii) Any two input intervals are either disjoint or one must contain the other.

To aid the reader, we offer an example which illustrates the set $\mathfrak{I}$ of input intervals.

Let n=3. Then for $W \in \Gamma^N$, where N=81p, we may specify the set $\mathfrak{I}$ of input intervals of W as follows:

$$\mathfrak{I} = \bigcup_{i=0}^{4} \mathfrak{I}_i \text{ where:}$$

$$\mathfrak{I}_0 = \{I_{00}, I_{01}, \ldots, I_{080}\} = \{[1,p], [p+1,2p], \ldots, [80p+1,N]\},$$

$$\mathfrak{I}_1 = \{I_{10}, I_{11}, \ldots, I_{126}\} = \{[1,3p], [3p+1,6p], \ldots, [78p+1,N]\},$$

$$\mathfrak{I}_2 = \{I_{20}, I_{21}, \ldots, I_{28}\} = \{[1,9p], [9p+1,18p], \ldots, [72p+1,N]\},$$

$$\mathfrak{I}_3 = \{I_{30}, I_{31}, I_{32}\} = \{[1,27p], [27p+1,54p], [54p+1,N]\}, \text{ and}$$

$$\mathfrak{I}_4 = \{I_{40}\} = \{[1,N]\}.$$

Notice that level i consists of $81/3^i$ distinct input intervals, each of size $p3^i$.

THEOREM 1

Let $\mathfrak{M}$ be a k-worktape real time Turing machine with input alphabet $\Gamma$. Let $N = pn^{2(n-1)}$ for some integer $p \geq 1$ and some integer $n > 1$. Suppose $S \subset \Gamma^N$. Then for some i $(n-1 \leq i \leq 2(n-1))$, there exists an interval $I \in \mathfrak{I}_i$ such that

$$\underset{W \in S}{\text{mean}} \; |\text{SP-OVERLAP}(W,I)| \leq 3k|I|/(n-1).$$

Proof:

Let $W \in S$. We proceed by first proving the following lemma.

## LEMMA 2

$$\sum_{I \in \mathfrak{I}} |\textbf{SP-OVERLAP}(W, I)| \leq 3kN.$$

## Proof:

Let $\hat{\mathfrak{I}} = \{I \in \mathfrak{I} \mid (\forall J \in \mathfrak{I})\ [|J| > |I| \Rightarrow \textbf{DP}(W, J) \notin I]\}$, i.e., $I \in \hat{\mathfrak{I}}$ only if I does not contain the dividing point of any input interval at a greater level. Lemma 2 shall follow immediately from claim 1 and claim 2.

## CLAIM 1

$$\sum_{I \in \hat{\mathfrak{I}}} |\textbf{SP-OVERLAP}(W, I)| \leq kN.$$

## Proof:

Let $I, J \in \hat{\mathfrak{I}}$ such that $I \neq J$. We shall show that $\textbf{SP-OVERLAP}(W, I) \cap \textbf{SP-OVERLAP}(W, J) = \phi$ from which we may conclude

$$\sum_{I \in \hat{\mathfrak{I}}} |\textbf{SP-OVERLAP}(W, I)| \leq |\textbf{OVERLAP}(W)|$$
$$\leq kN \quad \text{(by lemma 1.)}$$

### case 1

Suppose $I \cap J = \phi$. Clearly then, $\textbf{SP-OVERLAP}(W, I) \cap \textbf{SP-OVERLAP}(W, J) = \phi$.

### case 2

Suppose $I \subseteq J$. Since $I \in \hat{\mathfrak{I}}$, $\textbf{DP}(W, J) \notin I$. Thus either $I \subset [\textbf{MIN}(J), \textbf{DP}(W, J)]$ or $I \subseteq [\textbf{DP}(W, J) + 1, \textbf{MAX}(J)]$. Suppose the first alternative holds. But then the special overlap events of W at I occur before step $\textbf{DP}(W, J)$, while the special overlap events of W at J occur after step $\textbf{DP}(W, J)$. Hence, $\textbf{SP-OVERLAP}(W, I) \cap \textbf{SP-OVERLAP}(W, J) = \phi$. The second alternative may be dealt with similarly.

CLAIM 2

$$\sum_{I \in (\mathfrak{I} - \hat{\mathfrak{I}})} |\text{SP-OVERLAP}(W, I)| \leq 2kN.$$

Proof:

Let i be such that $0 \leq i < 2(n-1)$. The maximum number of input intervals at level i such that each contains the dividing point of an interval at a greater level is bounded, a fortiori, by the total number of intervals at a greater level than level i. Thus,

$$|\mathfrak{I}_i \cap (\mathfrak{I} - \hat{\mathfrak{I}})| \leq \sum_{j > i} |\mathfrak{I}_j|$$

$$= \sum_{i < j \leq 2(n-1)} N/(pn^j)$$

$$= (N/p)((1/n^i) - (1/n^{2(n-1)}))/(n-1)$$

$$< (N/p)(1/n^i)/(n-1)$$

$$= |\mathfrak{I}_i|/(n-1).$$

By lemma 1, we have therefore that

$$\sum_{I \in (\mathfrak{I}_i \cap (\mathfrak{I} - \hat{\mathfrak{I}}))} |\text{SP-OVERLAP}(W, I)| \leq |\mathfrak{I}_i| kpn^i/(n-1)$$

$$= kN/(n-1).$$

Thus, by summing over levels 0 through $2(n-1)-1$ we have that

$$\sum_{I \in (\mathfrak{I} - \hat{\mathfrak{I}})} |\text{SP-OVERLAP}(W, I)| \leq 2kN.$$

Lemma 2 follows immediately from claim 1 and claim 2.□

Lemma 2 trivially implies that

$$(\star)\quad \operatorname*{mean}_{W\in S}\ [\sum_{I\in\mathfrak{I}} |\textbf{SP-OVERLAP}(W, I)|] \le 3kN.$$

We now proceed to prove theorem 1.

Suppose to the contrary that for every $I\in\mathfrak{I}_i$ such that $(n-1\le i\le 2(n-1))$,

$$\operatorname*{mean}_{W\in S} |\textbf{SP-OVERLAP}(W, I)| > 3k|I|/(n-1).$$

Then $\operatorname*{mean}_{W\in S}\ [\sum_{I\in\mathfrak{I}} |\textbf{SP-OVERLAP}(W, I)|]$

$$\ge \operatorname*{mean}_{W\in S}\ [\sum_{(n-1)\le i\le 2(n-1)}\ \sum_{0\le j\le (N/(pn^i))-1} |\textbf{SP-OVERLAP}(W, I_{ij})|]$$

$$= \sum_{(n-1)\le i\le 2(n-1)}\ \sum_{0\le j\le (N/(pn^i))-1} [\operatorname*{mean}_{W\in S} |\textbf{SP-OVERLAP}(W, I_{ij})|]$$

$$> 3k/(n-1) \sum_{(n-1)\le i\le 2(n-1)}\ \sum_{0\le j\le (N/(pn^i))-1} |I_{ij}|$$

$$= 3k/(n-1) \sum_{(n-1)\le i\le 2(n-1)} N$$

$$> 3kN$$

which contradicts $(\star)$.□

## THE LANGUAGE $L_k$

In this section, we specify the language $L_k$ for $k \geq 2$ which can be recognized by a k-worktape real time Turing machine but cannot be recognized as fast by any (k-1)-worktape Turing machine. $L_k$ may be derived as follows:

Let $\mathfrak{S}$ be a machine with k pushdown stacks; we use $i \in \{1, \ldots, k\}$ to index the stacks. There are three canonical operations which $\mathfrak{S}$ may perform:

1) push symbol $\alpha$ onto stack i,
2) push symbol $\beta$ onto stack i, and
3) pop stack i.

Such actions may be encoded by supplying specific inputs to $\mathfrak{S}$. We shall let:

$\{a_i \mid 1 \leq i \leq k\}$ denote operation 1),
$\{b_i \mid 1 \leq i \leq k\}$ denote operation 2), and
$\{c_i \mid 1 \leq i \leq k\}$ denote operation 3).

Thus for example, the input $a_1c_3b_2c_1$ would cause $\mathfrak{S}$ to push $\alpha$ onto stack 1, pop stack 3, push $\beta$ onto stack 2, and finally pop stack 1. If stack i is empty when $c_i$ is scanned, then by convention the encoded action is ignored.

Let $\{a_i \mid 1 \leq i \leq k\} \cup \{b_i \mid 1 \leq i \leq k\} \cup \{c_i \mid 1 \leq i \leq k\}$ be denoted by $\Gamma_k$. We define $L_k$ to be those words $W \in \Gamma_k^*$ such that **the last operation encoded by W does <u>NOT</u> involve popping symbol $\alpha$ from some stack.**

Obviously, $L_k$ is accepted by a k-worktape real time Turing machine $\mathfrak{M}$, which simulates the behavior of $\mathfrak{S}$ by simulating a different stack on each worktape. The input alphabet of $\mathfrak{M}$ is $\Gamma_k$ while the worktape alphabet is $\{\not{b},\alpha,\beta\}$ where $\not{b}$ denotes the blank symbol. When $\mathfrak{M}$ scans $a_i$ or $b_i$, it writes $\alpha$ or $\beta$ respectively on the $i^{th}$ worktape and then shifts the $i^{th}$ tape head one square to the right. When $c_i$ is scanned, $\mathfrak{M}$ writes $\not{b}$ on the $i^{th}$ worktape (thus simulating a pop) and then shifts the $i^{th}$ tape head one square to the left. When the right endmarker is scanned, if no worktape head is scanning $\alpha$, the input word is accepted.

$L_k$ may be termed an **information retrieval language.** As long as the input symbols that $\mathfrak{M}$ scans are $a_i$ or $b_i$ it will store information. When $\mathfrak{M}$ scans $c_i$ it will retrieve information.

<u>DEFINITION 14</u>

Let $i\in\{1,\ldots.,k\}$ and let $W\in\Gamma_k^*$. $\mathbf{PROJ}(W,i)$ is defined to be that word which is obtained from W be deleting every $a_j,b_j,c_j$ in W such that $j\neq i$, i.e., the **projection** of W on the $i^{th}$ index.

For example, $\mathbf{PROJ}(c_2a_1b_2a_2a_3c_4,2)=c_2b_2a_2$. We note that $|\mathbf{PROJ}(W,i)|$ equals the number of symbols in W with index i.

We now define a set $\mathbf{S}\subset L_k$ with the following properties:

i) Let $W\in\mathbf{S}$ and let $1\leq i<j\leq k$. Then the number of symbols with index i in W is significantly less than the number of symbols with index j.

ii) All words of **S** encode actions which reference the stacks in the same order.

For integer $m>1$, let $\hat{m}$ denote $\sum_{i=1}^{k} m^i = (m^{k+1}-m)/(m-1)$ and let **PSTRING** (pattern-string) denote $\prod_{i=1}^{k} a_i^{m^i} \in \{a_i \mid 1\leq i\leq k\}^{\hat{m}}$. Thus we have that:

i) $(\forall i \mid 1\leq i\leq k)\,[\,|\mathbf{PROJ}(\mathbf{PSTRING}, i)| = m^i]$, and

ii) $(\forall i,j \mid 1\leq i<j\leq k)\,[\,|\mathbf{PROJ}(\mathbf{PSTRING}, j)| = m^{j-i}|\mathbf{PROJ}(\mathbf{PSTRING}, i)|\,]$.

Therefore by choosing m appropriately large, we may cause the relative frequencies of symbols occurring in **PSTRING** with a specific index to vary as greatly as desired.

Let $N=q\hat{m}$ for some integer $q>0$ and let **SEED** denote the string $(\mathbf{PSTRING})^q \in \{a_i \mid 1\leq i\leq k\}^N$. All the words of **S** shall be derived from **SEED**.

We define $\mathbf{S} \subset (\{a_i \mid 1\leq i\leq k\} \cup \{b_i \mid 1\leq i\leq k\})^N$ by the condition that $W \in \mathbf{S}$ if and only if W is obtainable from **SEED** by replacing arbitrarily selected occurrences of $a_i$ by $b_i$ for $1\leq i\leq k$. Formally, we may specify **S** as follows:
Let $h: \Gamma_k^* \rightarrow \Gamma_k^*$ be a homomorphism defined for $\gamma \in \Gamma_k$ as:

$$h(\gamma)=a_i \text{ if } \gamma=b_i \text{ for } 1\leq i\leq k,$$
$$h(\gamma)=\gamma \text{ otherwise.}$$

Then we define **S** to be $h^{-1}(\mathbf{SEED})$. We offer the following remarks:

i) $\mathbf{S} \subset \mathbf{L}_k$.

ii) $|\mathbf{S}| = 2^N$.

iii) For a given N, **PSTRING** completely specifies the indices of all the symbols of any word in **S**.

iv) Let $X,Y\in\mathbf{S}$ such that $X\neq Y$. Then there exists n, $1\leq n\leq N$, such that X@[n,n] and Y@[n,n] are different symbols with the same stack index i. Thus by the definition of $\mathbf{L}_k$, there exists a string $V\in\mathbf{\Gamma}_k^*$ for which $XV\in\mathbf{L}_k$ if and only if $YV\notin\mathbf{L}_k$. Specifically, we can choose $V=c_i^r$ where r= |**PROJ**(**SEED**@[n,N],i)|.

Suppose $\mathfrak{R}$ is a (k-1)-worktape Turing machine which can simulate the behavior of $\mathfrak{S}$ in real time. Consider how $\mathfrak{R}$ might behave on XV such that $X\in\mathbf{S}$ and $V\in\mathbf{\Gamma}_k^*$. We would expect $\mathfrak{R}$ to use a single worktape 1 to simulate, at least in part, the operation of two stacks indexed i,j (i<j) of $\mathfrak{S}$. By the construction of **S**, $\mathfrak{R}$ must store considerably more information obtained from processing input symbols with index j, (which we shall refer to as "j-information") than from processing input symbols with index i, i.e., "i-information".

Suppose in order to store new j-information, tape head 1 incurs significant displacement away from old i-information. After processing X, $\mathfrak{R}$ must be able to retrieve old i-information fast in case V necessitates the retrieval of i-information. Since $\mathfrak{R}$ operates in real time, tape head 1 would not have enough time to traverse the new j-information in order to retrieve the old i-information. Hence in order to operate properly, tape head 1 must "carry along" old i-information as it stores new j-information. But this can only be achieved if tape head 1 can sustain substantial information retrieval activity.

In the next section we shall invoke Theorem 1 and show there is an input interval of the words in $S$ for which the information retrieval activity is small enough to prevent $\mathfrak{M}$ from proper operation. This shall imply that no (k-1)-worktape Turing machine can recognize $L_k$ in real time.

## A Proof of the Rabin-Hartmanis-Stearns Conjecture

Let $L_k$ be defined as in the previous section for fixed $k \geq 2$. We now proceed to prove by contradiction that there exists no $(k-1)$-worktape Turing machine which can recognize $L_k$ in real time.

Assume that $\mathfrak{M}$ is a $(k-1)$-worktape Turing machine with input alphabet $\Gamma_k$, worktape alphabet $\Sigma$, and set $Q$ of internal states which recognizes $L_k$ in real time.

Let $m \in \mathbb{N}$ be such that $m > \max[4(\log|Q| + \log(2k)), 32k^2 \log|\Sigma|]$. *
As in the previous section, we let $\hat{m} \overset{\text{def}}{=} \sum_{i=1}^{k} m^i = (m^{k+1} - m)/(m-1)$. Let $n \in \mathbb{N}$ be such that $n > 12k(k-1)\hat{m} + 1$ and let $N = k\hat{m}n^{2(n-1)}$.

By the methods outlined previously, we may construct a set $S \subset \Gamma_k^N$ such that $S \subset L_k$. The reader should consult the previous section for the special properties of $S$.

By Theorem 1, there must exist some subinterval $I$ of $[1, N]$ with the following properties:

i) $N^{1/2} < |I| \leq N$,

ii) $|I|$ is a multiple of $k\hat{m}$, and

iii) $\operatorname{mean}_{W \in S} |\text{SP-OVERLAP}(W, I)| \leq 3(k-1)|I|/(n-1)$

$< |I|/(4k\hat{m})$ (by our choice of $n$.)

(* all logs are to the base 2.)

Since $\mathfrak{M}$ has only (k-1) worktapes, we expect $\mathfrak{M}$ to use a single worktape 1 to store, at least in part, the information obtained from processing input symbols with index i and index j, for some i, j $(1 \leq i < j \leq k)$. We shall refer to the information that is stored as a result of processing input symbols with index i and index j as "i-information" and "j-information", respectively. As implied by the discussion at the end of the previous section, we expect that $\mathfrak{M}$ can only process correctly words of the form $X@[1,\mathrm{MAX}(I)]V$ for $X \in S$ and $V \in \Gamma_k^*$, if the computation proceeds with substantial information retrieval activity while $X@[1,\mathrm{MAX}(I)]$ is processed.

Let $S_0 \overset{\mathrm{def}}{=} \{W \in S \mid |\text{SP-OVERLAP}(W,I)| > |I|/(k\hat{m})\}$.

<u>LEMMA 3</u>

$$|S_0| < |S|/4.$$

<u>Proof:</u>

Suppose to the contrary that $|S_0| \geq |S|/4$. Then

$$\begin{aligned} \operatorname*{mean}_{W \in S} |\text{SP-OVERLAP}(W,I)| &\geq |S_0|\,|I|/(|S|k\hat{m}) \\ &\geq |I|/(4k\hat{m}) \end{aligned}$$

which contradicts property iii) of subinterval I.□

Since $\mathfrak{M}$ may incur moderate information retrieval activity while processing the words of $S_0$ at I, we expect that $\mathfrak{M}$ might be able to process correctly words of the form, $X@[1,\mathrm{MAX}(I)]V$ for $X \in S_0$ and $V \in \Gamma_k^*$. On the other hand, we shall argue that $\mathfrak{M}$ cannot process properly all the words of the form $X@[1,\mathrm{MAX}(I)]V$ if $X \in (S-S_0)$. Intuitively, since

$\mathfrak{M}$ must incur little information retrieval activity while processing X at **I**, it will not be able to store the i-information and j-information on worktape 1 in such a manner that would allow it to process correctly any suffix V in real time. This will imply that $\mathfrak{M}$ cannot process correctly all the words of $(\mathbf{S}-\mathbf{S}_0)$ and therefore contradict the assumption that $\mathfrak{M}$ can recognize $\mathbf{L}_k$ in real time.

We proceed by formalizing our intuition. Let $X\in(\mathbf{S}-\mathbf{S}_0)$. We suppose that $\mathfrak{M}$ can only "store properly" either i-information or j-information (but not both!) on worktape 1 while it processes X@**I**. (By store properly, we mean, storing the information in such a way that would allow $\mathfrak{M}$ to retrieve any part of it fast enough so that it may process any suffix in real time.) Since $\mathfrak{M}$ incurs little information retrieval activity while processing X@I, tape head 1 cannot revisit too many tape squares while $\mathfrak{M}$ stores information on worktape 1. Therefore, we expect $\mathfrak{M}$ to use some scheme for storing information on the 1th worktape which approaches that of transcribing information onto consecutive tape squares such that the tape head is always close to the most recent information that is stored. Thus while processing X@**I**, if $\mathfrak{M}$ predominantly uses worktape 1 to store properly i-information, we expect **DSPM**(1,X,I) to be a large fraction of |**PROJ**(X@I,i)| which is the amount of i-information needed to be stored in order to process correctly X@I. Otherwise, we expect **DSPM**(1,X,I) to be a large fraction of |**PROJ**(X@I,j)|.

In order to discover what information has been stored properly, we shall view the computation of $\mathfrak{M}$ on X@I in k stages, indexed 1 through k. For i such that $1\leq i\leq k$, we shall use stages i,i+1,....,k to determine whether $\mathfrak{M}$ has stored properly i-information on any worktape

$j$ $(1\leq j\leq k-1)$. We shall show that for some $i$, no worktape $j$ will store properly the $i$-information during stages $i,i+1,\ldots,k$.

We divide $I$ into $k$ consecutive equal sized subintervals $I_1,\ldots,I_k$, namely,

$$I_i \overset{\text{def}}{=} [((i-1)|I|/k)+1, i|I|/k] \text{ for } 1\leq i\leq k.$$

Thus $I=\bigcup_{i=1}^{k} I_i$. Stage $i$ shall correspond to the computation of $\mathfrak{M}$ on $X@I_i$. Thus, stage $i$ consists of the set of indices of steps equal to $I_i$.

LEMMA 4

Let $X\in S$ and let $h,i\in\{1,\ldots,k\}$. Then

$$|\mathbf{PROJ}(X@I_h,i)|=m^i|I|/(k\hat{m}).$$

PROOF:

By the construction of $S$ it suffices to show that the lemma holds for $X=\mathbf{SEED}$.

$$\begin{aligned}|\mathbf{PROJ}(\mathbf{SEED}@I_h,i)| &= |\mathbf{PROJ}((\mathbf{PSTRING})^{|I|/(k\hat{m})},i)| \\ &= |\mathbf{PROJ}(\mathbf{PSTRING},i)|\,|I|/(k\hat{m}) \\ &= m^i|I|/(k\hat{m}).\square\end{aligned}$$

To aid the reader, we present the function $\mathbf{NUM}:\{1,\ldots,k\}\to\mathbb{N}$ where

$$\mathbf{NUM}(i)\overset{\text{def}}{=}m^i|I|/(k\hat{m}).$$

Thus, $\mathbf{NUM}(i)$ equals the number of symbols with index $i$ in $X$ at any subinterval $I_h$ $(1\leq h\leq k)$ of $I$.

Suppose $X \in (\mathbf{S}-\mathbf{S}_0)$. We note that i-information will not be stored properly on worktape j if either:

i) worktape j predominantly stores $\hat{\imath}$-information for some $\hat{\imath}<i$, or

ii) worktape j predominantly stores $\hat{\imath}$-information for some $\hat{\imath}>i$.

We expect the first alternative to hold if

$$\mathbf{DSPM}(j, X, \mathbf{I}_i) < (4k/m)\,\mathbf{NUM}(i).$$

That is, during stage i, if tape head j does not incur displacement greater than a small fraction of the displacement we expect necessary for the proper storage of i-information, we expect that $\mathfrak{M}$ is primarily storing $\hat{\imath}$-information for some $\hat{\imath}<i$ on worktape j.

We expect the second alternative to hold if

$$\mathbf{DSPM}(j, X, \bigcup_{i<l} \mathbf{I}_l) > k\,\mathbf{NUM}(i).$$

That is, during stages $i+1,\ldots,k$, if tape head j incurs displacement significantly greater than the displacement we expect necessary for the proper storage of i-information, we expect that $\mathfrak{M}$ is primarily storing $\hat{\imath}$-information for some $\hat{\imath}>i$ on worktape j.

This leads us to conjecture that i-information will not be stored properly during the computation of $\mathfrak{M}$ on $X@\mathbf{I}$ if

$$(\forall j \mid 1 \le j \le k-1)\,[\mathbf{DSPM}(j, X, \mathbf{I}_i) < (4k/m)\,\mathbf{NUM}(i) \text{ or } \mathbf{DSPM}(j, X, \bigcup_{i<l} \mathbf{I}_l) > k\,\mathbf{NUM}(i)].$$

For convenience, we shall define for $1 \le i \le k$,

$$\mathbf{y}_i \overset{\text{def}}{=} (4k/m)\,\mathbf{NUM}(i), \text{ and}$$

$$\mathbf{z}_i \overset{\text{def}}{=} k\,\mathbf{NUM}(i).$$

For $1 \le i \le k$,

$$\mathbf{S}_i \overset{\text{def}}{=} \{X \in (\mathbf{S}-\mathbf{S}_0) \mid (\forall j \mid 1 \le j \le k-1)\,[\mathbf{DSPM}(j, X, \mathbf{I}_i) < \mathbf{y}_i \text{ or } \mathbf{DSPM}(j, X, \bigcup_{i<l} \mathbf{I}_l) > \mathbf{z}_i]\}.$$

In particular, we note that $X \in \mathbf{S}_k$ only if $\mathbf{DSPM}(j,X,\mathbf{I}_k) < \mathbf{y}_k$ for all j $(1 \leq j \leq k-1)$.

Intuitively, we expect that $X \in S_i$ only if $\mathfrak{N}$ does not store properly i-information while processing X@I.

Suppose $X \in (\mathbf{S}-\mathbf{S}_0)$. Since there is little information retrieval activity while $\mathfrak{N}$ processes X@I, we expect for some i, that $\mathfrak{N}$ does not store properly i-information while processing X@I, i.e., we expect that $X \in S_i$ for some i. We now confirm our expectations by proving the following two lemmas.

LEMMA 5

Let $X \in \mathbf{S}$ and let J, L be input intervals of X such that $J \subset L$. Let j be a tape index of $\mathfrak{N}$. Suppose that $\mathbf{DSPM}(j,X,J) \geq y$ and $\mathbf{DSPM}(j,X,L) \leq z$ for $y,z \in \mathbb{N}$ such that $y \geq 2z$. Then

$$|\mathbf{SP\text{-}OVERLAP}(X,L)| \geq (y/2)-(z+1).$$

PROOF:

Since $\mathbf{DSPM}(j,X,J) \geq y$ we have that either:

i) $|\mathbf{POSN}(j,X,\mathbf{MAX}(J)+1)-\mathbf{POSN}(j,X,\mathbf{MIN}(L))| \geq y/2$, or

ii) $|\mathbf{POSN}(j,X,\mathbf{MIN}(J))-\mathbf{POSN}(j,X,\mathbf{MIN}(L))| \geq y/2$.

Suppose the first alternative holds. Since $\mathbf{DSPM}(j,X,L) \leq z$, the tape square denoted by $\mathbf{POSN}(j,X,\mathbf{MAX}(L))$ must be within $(z+1)$ tape squares of the square denoted by $\mathbf{POSN}(j,X,\mathbf{MIN}(L))$. Therefore after step $\mathbf{MAX}(j)+1$, tape head j must revisit at least $(y/2)-(z+1)$ tape squares which implies that $|\mathbf{SP\text{-}OVERLAP}(X,L)| \geq (y/2)-(z+1)$. The second alternative may be dealt with similarly.$\square$

We note that an improved lower bound of $(y-(z+1))/2$ may be established by a more complicated proof. The weaker bound shall suffice for our purposes. We also remark that lemma 5 is a direct consequence of the linear structure of the worktapes of $\mathfrak{M}$.

LEMMA 6

$$(\mathbf{S}-\mathbf{S}_0) = \bigcup_{i=1}^{k} \mathbf{S}_i.$$

PROOF:

Clearly it suffices to show that $(\mathbf{S}-\mathbf{S}_0) \subseteq \bigcup_{i=1}^{k} \mathbf{S}_i$. Let $X \in (\mathbf{S}-\mathbf{S}_0)$ and suppose to the contrary that $X \notin \mathbf{S}_i$ for all $i$ $(1 \leq i \leq k)$. Then by the definition of $\mathbf{S}_i$ for $i>0$, we have that

$$(\forall i \mid 1 \leq i \leq k)(\exists j \mid 1 \leq j \leq k-1)[\mathbf{DSPM}(j,X,\mathbf{I}_i) \geq \mathbf{y}_i \text{ and } \mathbf{DSPM}(j,X,\bigcup_{i<l} \mathbf{I}_l) \leq \mathbf{z}_i].$$

But then for some $1 \leq i_1 < i_2 \leq k$, there exists $\hat{\jmath}$ for which:

$$\mathbf{DSPM}(\hat{\jmath},X,\mathbf{I}_{i_1}) \geq \mathbf{y}_{i_1} \text{ and } \mathbf{DSPM}(\hat{\jmath},X,\bigcup_{i_1<l} \mathbf{I}_l) \leq \mathbf{z}_{i_1}, \text{ and}$$
$$\mathbf{DSPM}(\hat{\jmath},X,\mathbf{I}_{i_2}) \geq \mathbf{y}_{i_2} \text{ and } \mathbf{DSPM}(\hat{\jmath},X,\bigcup_{i_2<l} \mathbf{I}_l) \leq \mathbf{z}_{i_2}.$$

By lemma 5, the second and third of these bounds on **DSPM** imply that

$$\begin{aligned}
|\mathbf{SP\text{-}OVERLAP}(X,\bigcup_{i_1<l} \mathbf{I}_l)| &\geq (\mathbf{y}_{i_2}/2)-(\mathbf{z}_{i_1}+1) \\
&= (2k/m)\mathbf{NUM}(i_2)-k\mathbf{NUM}(i_1)-1 \\
&\geq 2k\mathbf{NUM}(i_1)-k\mathbf{NUM}(i_1)-1 \\
&> \mathbf{NUM}(1) \\
&> |\mathbf{I}|/(k\hat{m}).
\end{aligned}$$

But this implies that $X \in \mathbf{S}_0$ and thus contradicts our choice of $X$.$\square$

We call attention to the fact that lemma 6 is the only part of the proof which relies directly on $\mathfrak{N}$ having only (k-1)-worktapes.

We have shown therefore that $\mathbf{S}=\bigcup_{i=0}^{k}\mathbf{S}_i$. We shall eventually prove that $\sum_{i=1}^{k}|\mathbf{S}_i|<|\mathbf{S}|/2$ which along with lemma 3 shall contradict the assumption that $\mathfrak{N}$ can recognize $\mathbf{L}_k$ in real time.

DEFINITION 15

Let $X,Y\in\mathbf{S}$ and let $i\in\{1,\ldots,k\}$. $X\equiv_i Y$ if and only if

$(\forall n\in[1,N])\,[X@[n,n]=Y@[n,n]$ or $(n\in\mathbf{I}_i$ and $\{X@[n,n],Y@[n,n]\}=\{a_i,b_i\})]$.

Thus $X\equiv_i Y$ if and only if X and Y are identical except for possible differences involving symbols with index i in $X@\mathbf{I}_i$ and $Y@\mathbf{I}_i$. Clearly then, $\equiv_i$ is an equivalence relation on $\mathbf{S}$.

We define $E_i^X$ to be the equivalence class of X with respect to $\equiv_i$, that is,

$$E_i^X \overset{\text{def}}{=} \{Y\in\mathbf{S}\,|\,Y\equiv_i X\}.$$

We note that $|E_i^X|=2^{\mathrm{NUM}(i)}$.

Suppose $X\equiv_i Y$ for some $i\in\{1,\ldots,k\}$ such that $X\neq Y$. By the definition of $\mathbf{L}_k$ there exists $V\in\Gamma_k^*$ such that $X@[1,\mathbf{MAX}(\mathbf{I})]V\in\mathbf{L}_k$ if and only if $Y@[1,\mathbf{MAX}(\mathbf{I})]V\notin\mathbf{L}_k$. Since X and Y are identical except for differences at $\mathbf{I}_i$ involving symbols with index i, we expect that $\mathfrak{N}$ could distinguish between $X@[1,\mathbf{MAX}(\mathbf{I})]V$ and $Y@[1,\mathbf{MAX}(\mathbf{I})]V$ only if it stored properly the i-information obtained during stage i of the computations. Thus in order for $\mathfrak{N}$ to operate properly, we expect that very few words of $E_i^X$ are also contained in $\mathbf{S}_i$. We now show that this is indeed the case.

## LEMMA 7

Let $X \in \mathbf{S}$ and let $i \in \{1,\ldots,k\}$. Then

$$|E_i^X \cap \mathbf{S}_i| < |E_i^X|/(2k).$$

## PROOF:

Let $X, Y \in \mathbf{S}_i$ for $i > 0$, be distinct and suppose $X \equiv_i Y$. Let $V \in \Gamma_k^*$. We shall denote $X@[1,\mathbf{MAX}(\mathbf{I})]$ by $\hat{X}$ and $Y@[1,\mathbf{MAX}(\mathbf{I})]$ by $\hat{Y}$.

Consider the computations of $\mathfrak{M}$ on $\hat{X}V$ and $\hat{Y}V$. We shall show that if the i.d.'s immediately after step $\mathbf{MAX}(\mathbf{I}_i)+1$, i.e., immediately after stage i, are very equivalent then the subsequent i.d.'s immediately after step $\mathbf{MAX}(\mathbf{I})+1$, i.e., immediately after stage k, become even more equivalent. If the equivalence after stage k has become too great, we shall show that $\mathfrak{M}$ could not process correctly a suffix V which distinguishes between $\hat{X}$ and $\hat{Y}$. This shall imply that $|E_i^X \cap \mathbf{S}_i|$ is bounded by some number of non-equivalent i.d.'s that $\mathfrak{M}$ can achieve immediately after stage i. We proceed by first presenting claim 1 and claim 2.

## CLAIM 1

Suppose $\mathbf{ID}(\hat{X}V, \mathbf{MAX}(\mathbf{I}_i)+1)$ and $\mathbf{ID}(\hat{Y}V, \mathbf{MAX}(\mathbf{I}_i)+1)$ are $(y_i+|\mathbf{I}|/(k\hat{m}))$-equivalent. Then $\mathbf{ID}(\hat{X}V, \mathbf{MAX}(\mathbf{I})+1)$ and $\mathbf{ID}(\hat{Y}V, \mathbf{MAX}(\mathbf{I})+1)$ are $(y_i+z_i+|\mathbf{I}|/(k\hat{m}))$-equivalent.

PROOF:

Consider the computations of $\mathfrak{M}$ on $\hat{X}V$ and $\hat{Y}V$. Since the input tape is one-way and $X \equiv_i Y$, the subsequent computations after stage i can differ only if some tape head scans different i-information which was first stored while $\mathfrak{M}$ processed the inputs during stage i. Let $j>0$ be a tape index of $\mathfrak{M}$. We shall proceed by showing that worktape j must satisfy one of the following conditions:

i) The information stored on worktape j during stage i is identical for each computation, i.e.,

$$(\forall z \in \mathbb{Z})\,[\mathbf{CONT}(j,z,\hat{X}V,\mathbf{MAX}(I_i)+1)=\mathbf{CONT}(j,z,\hat{Y}V,\mathbf{MAX}(I_i)+1)].$$

ii) The information scanned on worktape j during stages $i+1,\ldots,k$, is identical for each computation. In addition, immediately after stage k of each computation, tape head j is displaced at least $(y_i+z_i+|I|/(k\hat{m}))$ tape squares away from any possibly different i-information.

Claim 1 will follow immediately from case 1 and case 2. We note that $\mathbf{DSPM}(j,\hat{X}V,I_i)$ necessarily equals $\mathbf{DSPM}(j,\hat{Y}V,I_i)$ and that the displacement is in the same direction.

case 1

Suppose $\mathbf{DSPM}(j,\hat{X}V,I_i)<y_i$. We proceed by showing that worktape j must satisfy condition 1. (Intuitively, this case implies that during stage i, worktape j was used predominantly for storing $\hat{\imath}$-information for some $\hat{\imath}<i$. Because of the substantial equivalence immediately after stage i, we shall be able to argue that no different

i-information could have been stored on worktape j during stage i.)

Suppose to the contrary that different i-information was stored on worktape j during stage i, that is,

$$(\exists z\in\mathbb{Z})\,[\mathbf{CONT}(j,z,\hat{X}V,\mathbf{MAX}(I_i)+1)\neq\mathbf{CONT}(j,z,\hat{Y}V,\mathbf{MAX}(I_i)+1)].$$

Since $\mathbf{ID}(\hat{X}V,\mathbf{MAX}(I_i)+1)$ and $\mathbf{ID}(\hat{Y}V,\mathbf{MAX}(I_i)+1)$ are $(\mathbf{y}_i+|I|/(k\hat{m}))$-equivalent on worktape j and $\mathbf{DSPM}(j,\hat{X}V,I_i)<\mathbf{y}_i$, this implies without loss of generality that $|\mathbf{SP\text{-}OVERLAP}(\hat{X}V,I_i)|>|I|/(k\hat{m})$. But then $|\mathbf{SP\text{-}OVERLAP}(X,I)>|I|/(k\hat{m})$ which contradicts $X\in\mathbf{S}_i$ for $i>0$.

<u>case 2</u>

Suppose $\mathbf{DSPM}(j,\hat{X}V,I_i)\geq\mathbf{y}_i$. We note that $i<k$ for this case to apply. (Intuitively, this case implies that during stage i, worktape j was used predominantly for storing $\hat{\imath}$-information for some $\hat{\imath}>i$. However, it is quite possible that different i-information was also stored. Because of the substantial equivalence immediately after stage i, and because $\mathfrak{M}$ must store additional $\hat{\imath}$-information during stages $i+1,\ldots,k$, we shall be able to argue that worktape j must satisfy condition 2.)

Suppose to the contrary that during stages $i+1,\ldots,k$, worktape j scanned different information. Clearly then, this information was first stored while $\mathfrak{M}$ was processing the inputs during stage i. But since $\mathbf{ID}(\hat{X}V,\mathbf{MAX}(I_i)+1)$ and $\mathbf{ID}(\hat{Y}V,\mathbf{MAX}(I_i)+1)$ are $(\mathbf{y}_i+|I|/(k\hat{m}))$-equivalent, worktape j can only scan different information during stages $i+1,\ldots,k$, if without loss of generality $|\mathbf{SP\text{-}OVERLAP}(\hat{X}V,\bigcup_{l=i}^{k}I_l)|>(\mathbf{y}_i+|I|/(k\hat{m}))$, which implies that $|\mathbf{SP\text{-}OVERLAP}(X,I)|>|I|/(k\hat{m})$ which contradicts $X\in I_i$ for $i>0$.

Since $X, Y \in S_i$, case 2 implies that $\mathbf{DSPM}(j, \hat{X}V, \bigcup_{i<l} I_l) > z_i$ and $\mathbf{DSPM}(j, \hat{Y}V, \bigcup_{i<l} I_l) > z_i$. The displacement incurred during stages $i+1, \ldots, k$, of the computations must be in the same direction as the displacement incurred during stage i. (Otherwise, there would be too many special overlap events.) But since $\mathbf{ID}(\hat{X}V, \mathbf{MAX}(I_i)+1)$ and $\mathbf{ID}(\hat{Y}V, \mathbf{MAX}(I_i)+1)$ are $(y_i+|I|/(k\hat{m}))$-equivalent, immediately after stage k, tape head j must be at least $y_i+z_i+|I|/(k\hat{m}))$ tape squares away from any possibly different i-information.

CLAIM 2

$|E_i^X \cap S_i|$ is bounded by the number of non-$(y_i+|I|/(k\hat{m}))$-equivalent i.d.'s that $\mathfrak{M}$ can achieve immediately after stage i, that is, immediately after step $\mathbf{MAX}(I_i)+1$.

PROOF:

Immediately after stage k, i.e., immediately after step $\mathbf{MAX}(I)+1$, at most $(k-i+1)\mathbf{NUM}(i)$ pop operations are needed to retrieve the different i-information that $\mathfrak{M}$ stored while processing $\hat{X}$ and $\hat{Y}$ during stage i. Thus for some $r \leq (k-i+1)\mathbf{NUM}(i)$, we have that

$$(\star) \quad \hat{X}c_i^r \in L_k \text{ if and only if } \hat{Y}c_i^r \notin L_k.$$

Suppose that $\mathbf{ID}(\hat{X}c_i^r, \mathbf{MAX}(I_i)+1)$ and $\mathbf{ID}(\hat{Y}c_i^r, \mathbf{MAX}(I_i)+1)$ are $(y_i+|I|/(k\hat{m}))$-equivalent. Then by claim 1, we have that $\mathbf{ID}(\hat{X}c_i^r, \mathbf{MAX}(I)+1)$ and $\mathbf{ID}(\hat{Y}c_i^r, \mathbf{MAX}(I)+1)$ are $(y_i+z_i+|I|/(k\hat{m}))$-equivalent. But since

$$r \leq (k-i+1)\mathbf{NUM}(i)$$
$$< \mathbf{y}_i + \mathbf{z}_i + |\mathbf{I}|/(k\hat{m}),$$

we have that

$$\hat{X}c_i^r \in \mathbf{L}_k \text{ if and only if } \hat{Y}c_i^r \in \mathbf{L}_k$$

which contradicts (★). Claim 2 immediately follows.

We now proceed to prove lemma 7. Claim 2 implies that

$$|\mathbf{E}_i^X \cap \mathbf{S}_i| \leq |\mathbf{Q}|\,|\mathbf{\Sigma}|^{(k-1)(2(y_i+|I|/(k\hat{m}))+1)}(2|\mathbf{I}_i|+1)^{(k-1)}$$

We shall prove that $|\mathbf{E}_i^X \cap \mathbf{S}_i| < |\mathbf{E}_i^X|/(2k)$ by showing that

$$\log|\mathbf{Q}| + (k-1)(2(\mathbf{y}_i + |\mathbf{I}|/(k\hat{m}))+1)\log|\mathbf{\Sigma}| + (k-1)\log(2|\mathbf{I}_i|+1) + \log(2k) < \log|\mathbf{E}_i^X|$$
$$= \mathbf{NUM}(i).$$

1) $\log|\mathbf{Q}| + \log(2k) < m/4$ (by our choice of m)

$< \mathbf{NUM}(i)/4.$

2) $(k-1)(2(\mathbf{y}_i + |\mathbf{I}|/(k\hat{m}))+1)\log|\mathbf{\Sigma}| < 4k\mathbf{y}_i\log|\mathbf{\Sigma}|$

$= 16k^2\log|\mathbf{\Sigma}|\mathbf{NUM}(i)/m$

$< \mathbf{NUM}(i)/2$ (by our choice of m.)

3) $(k-1)\log(2|\mathbf{I}_i|+1) < k\log|\mathbf{I}|$

$< m^i|\mathbf{I}|/(4k\hat{m})$ (since $\log|\mathbf{I}|/|\mathbf{I}| < m^i/(4k^2\hat{m})$ by our choice of n)

$= \mathbf{NUM}(i)/4.$

Lemma 7 follows from 1), 2), and 3).□

LEMMA 8

$$\sum_{i=1}^{k} |S_i| < |S|/2.$$

PROOF:

Let $i \in \{1,\ldots,k\}$. Since $\{E_i^X | X \in S\}$ partitions $S$, we have that:

$$\begin{aligned} |S_i| &= \Big|\bigcup_{X\in S} (E_i^X \cap S_i)\Big| \\ &< 1/(2k) \Big|\bigcup_{X\in S} E_i^X\Big| \quad \text{(by lemma 7)} \\ &= |S|/(2k). \end{aligned}$$

Lemma 8 follows immediately.□

Lemma 6 implies that:

$$\begin{aligned} |S| &= |S_0| + \sum_{i=1}^{k} |S_i| \\ &< |S|/4 + |S|/2 \quad \text{(by lemma 3 and lemma 8)} \\ &< |S| \end{aligned}$$

which is a contradiction. Hence we conclude that $\mathfrak{M}$ cannot recognize $L_k$ in real time.

INDEX

| | |
|---|---|
| overlap event | 12 |
| OVERLAP(W) | 13 |
| real time Turing machine | 14 |
| real time | 14 |
| DSPM(j, W, I) | 15 |
| adjacent overlap events | 15 |
| ADJ-OVERLAP(W, I, J) | 15 |
| dividing point | 15 |
| DP(W, I) | 15 |
| SP-OVERLAP(W, I) | 15 |
| information retrieval activity | 15 |
| special overlap event | 16 |
| $a_i$ | 22 |
| $b_i$ | 22 |
| $c_i$ | 22 |
| $\Gamma_k$ | 22 |
| $L_k$ | 22 |
| PROJ(W, i) | 23 |
| PSTRING | 24 |
| SEED | 24 |
| S | 24 |
| $\mathfrak{N}$ | 27 |
| $m$ | 27 |
| $\hat{m}$ | 27 |
| $N$ | 27 |
| i-information | 28 |
| $S_0$ | 28 |

## REFERENCES

1) Aanderaa, S. O., *"On k-Tape Versus (k-1)-Tape Real Time Computation"*, SIAM-AMS PROC. vol 7, 1974.

2) Aho, A. V., Hopcroft, J. E., and Ullman, J. D., The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

3) Cook, S. A., *"On the Minimum Computation Time of Functions"*, Doctoral thesis, Harvard Univ., Cambridge, Mass. 1966.

4) Cook, S. A., and Aanderaa, S. O., *"On the Minimum Computation Time of Functions"*, Trans. Amer. Math. Soc. 142 (1969), 291-314.

5) Fischer, M. J., Meyer, A. R., and Paterson, M. S., *"An Improved Overlap Argument for On-Line Multiplication"*, Project MAC Technical Memo 40, M. I. T., 1974.

6) Hartmanis, J., and Stearns, R. E., *"On the Computational Complexity of Algorithms"*, Trans. Amer. Math. Soc. 117 (1965), 285-306.

7) Hopcroft, J. E., and Ullman, J. D., Formal Languages and Their Relation to Automata, Addison-Wesley, 1969.

8) Howe, S., and Anderson, J., *"Close To The Edge"*, Atlantic, SD7224, 1972.

9) Perry, H. M., *"An Improved Overlap Argument for On-Line Multiplication"*, unpublished manuscript, M. I. T. Dept. of Elec. Engr. and Comp. Sci., April, 1977.

10) Rabin, M. O., *"Real Time Computation"*, Israel J. Math. 1 (1963), 203-221.