String-Matching Cannot be Done by a Two-Head One-Way Deterministic Finite Automaton

Ming Li*
Yaacov Yesha†

83-579
October 1983

Department of Computer Science
Cornell University
Ithaca, New York 14853

*The work of this author was supported in part by NSF grant MCS-8301766.

†The work of this author was supported in part by a Dr. Chaim Weizmann Post-Doctoral Fellowship for Scientific Research.

String-Matching Cannot be Done by a Two-Head One-Way Deterministic Finite Automaton

Ming Li* and Yaacov Yesha**
Department of Computer Science
Cornell University
Ithaca, New York 14853

Abstract

We show that string-matching cannot be performed by a two-head one-way deterministic finite automaton (or even by a Turing machine with two one-way input heads and o(n) storage space). Thus we answer the special case $k=2$ of the open question, due to Galil and Seiferas [GS], whether a k-head one-way deterministic finite automaton can perform string-matching.

* The work of the first author was supported in part by an NSF grant MCS-8301766.
** The work of the second author was supported in part by a Dr. Chaim Weizmann Post-Doctoral Fellowship for Scientific Research.

1. Introduction

The string-matching problem is defined as follows: given a character string x, called the *pattern* and a character string y, called the *text*, find all occurences of x as a subword of y. In [GS], Galil and Seiferas have shown that string-matching can be performed by a six-head *two-way* deterministic finite automaton in linear time. They notice that a multi-head *one-way* deterministic finite automaton must operate in linear time. Motivated by this observation they ask whether a multi-head one-way deterministic finite automaton can perform string-matching. We answer this question for the case of two heads. We actually prove that a two-head one-way deterministic finite automaton cannot solve the *decision problem*: given strings x and y, is x a subword of y? Our result is formalized in the following

Theorem 1.1: Let L be the *string-matching language* defined by:

$$L = \{\#x\$y\not{c} \mid x,y \text{ in } \{0,1,*\}^* \text{ and for some } y', y'' \; y=y'xy''\}$$

Then there is no 2-head one-way deterministic finite automaton which accepts L.

2. Preliminaries and Notation

We shall denote a k-head one-way deterministic finite automaton by k-DFA. A k-DFA M has a finite set Q of states, a subset of Q of accepting states, k heads $h_1, h_2, \cdots, h_k$ which cannot see each other, a transition function, and a one way read only input tape. A state $q_* \in Q$ is distinguished as an initial state. We assume that the input to M is always of the form $\#z\not{c}$ where $z \in \Sigma^*$ for some fixed finite alphabet Σ, and $\#$ and $\not{c}$, which are not in Σ, are the left and right endmarkers, respectively. Initially, all k heads are on the leftmost tape position. At each step, depending on the current state and the ordered k-tuple of symbols seen by the heads h_1, h_2, $\cdots$, h_k, M changes state and moves some of the heads one position to the right. An input $\#z\not{c}$ is *accepted* by M if during the computation of M on $\#z\not{c}$ an accepting state is entered. If during $|Q|+1$ consecutive steps no head is moved we know that no head will ever move again. We can modify M so that whenever this situation occurs, or whenever an accepting state is entered, all heads move to the right until they reach the right end-marker. Hence we assume that on each input all heads will eventually reach the right end-marker, and M halts when this happens. We define

$$L(M) = \{\#z\cent \mid z \in \Sigma^* \text{ and } M \text{ accepts } \#z\cent\}.$$

A ***configuration*** of M ***on*** input m is the $k+2$ tuple: $(m, q, i_1, i_2, \ldots, i_k)$ where q is a state and i_j for $1 \le j \le k$ is the position of the j-th head. Clearly $1 \le i_j \le |m|$ for $1 \le j \le k$. The ***initial configuration*** of M on input m is $(m, q_*, 1, 1, \cdots, 1)$. The ***computation*** of M on m in the sequence of configurations reached by M on input m, starting from the initial configuration.

Let I_1 and I_2 be configurations. We write $I_1 \vdash I_2$ if M, started in configuration I_1, in one step reaches configuration I_2. We write $I_1 \vdash_* I_2$ if $I_1 = I_2$ or if M, started in configuration I_1, reaches configuration I_2 in a finite number of steps. By k-NFA we shall denote the nondeterministic version of a k-DFA.

3. Proof of Theorem 1.1

We shall prove Theorem 1.1 by contradiction. We assume that there exists a 2-DFA M which accepts L. Then, by a series of lemmas, we eventually conclude that an input not in L must be accepted by M, thus reaching a contradiction.

First we fix a positive integer n which is large enough. Later we shall determine how large should n be. We now consider inputs of the form: $\# * w_1 * w_2 * \$ * w_1 * w_2 * \cent$ where $w_1, w_2 \in \{0,1\}^n$. Clearly each such input is in L and is accepted by M. In the sequel we shall make implicit use of the assumption that on any input all heads eventually reach the right end-marker. Because of this fact we have for w_1, w_2 as above:

(1) $(\# * w_1 * w_2 * \$ * w_1 * w_2 * \cent, q_*, 1, 1) \vdash_* (\# * w_1 * w_2 * \$ * w_1 * w_2 * \cent, q, i, j)$ for some $q \in Q$ and i, j such that $\max(i,j) = 3n+7$. This simply means that some head eventually reaches the $*$ following $\$ * w_1$. Let the configuration on the right hand side of (1) be the first one during the computation for which $\max(i,j) = 3n+7$. We shall show that for this configuration, for most w_1, w_2 in $\{0,1\}^n$, $\min(i,j) \ge n+2-O(\log n)$, which means that at this point the leftmost head has "seen" most of w_1. In order to do this let k_0 be an integer satisfying $1 \le k_0 < n+2$. Consider w_1, w_2 in $\{0,1\}^n$ such that in (1) above $\min(i,j) \le k_0$.

We must have

(2) $(\# * w_1 * w_2 * \$ * w_1 * w_2 * \cent, q_*, 1, 1) \vdash_*$

(3) $(\# * w_1 * w_2 * \$ * w_1 * w_2 * \cent, q_1, i_1, j_1) \vdash_*$

(4) $(\# * w_1 * w_2 * \$ * w_1 * w_2 * \cancel{c}, q, i, j) \vdash_*$

(5) $(\# * w_1 * w_2 * \$ * w_1 * w_2 * \cancel{c}, q_2, i_2, j_2)$

where $\max(i_1,j_1)=2n+6$, $\max(i,j)=3n+7$ and $\min(i_2,j_2)=n+3$. We assume that during the computation, the configuration in (3) is the first one for which $\max(i_1,j_1)=2n+6$, the configuration in (4) is the first one for which $\max(i,j)=3n+7$, and the configuration in (5) is the first one for which $\min(i_2,j_2)=n+3$. Now, for each w_1, w_2 in $\{0,1\}^*$ for which $\min(i,j)\leq k_0$ let

$$f(w_1,w_2)=(w,q_1,q,q_2,i_1,i,i_2,j_1,j,j_2)$$

where $w \in \{0,1\}^{k_0-2}$ such that w_1 has w as a prefix and $q_1,q,q_2,i_1,i,i_2, j_1,j,j_2$ are as in (2)-(5) above. We prove:

Lemma 3.1: For any $w_2 \in \{0,1\}^*$ there do not exist two distinct strings w_1, w'_1 in $\{0,1\}^*$ such that $f(w_1,w_2)=f(w'_1, w_2)=(w, q_1, q,q_2, i_1, i,i_2, j_1, j,j_2)$ and $\min(i,j)\leq k_0$.

Proof: Suppose there exist such w_1, w'_1. We must have:

(6) $(\# * w'_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_*, 1,1) \vdash_*$

(7) $(\# * w'_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_1, i_1,j_1) \vdash_*$

(8) $(\# * w'_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q,i,j) \vdash_*$

(9) $(\# * w'_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_2, i_2, j_2)$

Both inputs in (2) and (6) above are in L and are accepted by M. Since $\max(i_1,j_1)=2n+6$ we must also have:

(10) $(\# * w_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_*, 1,1) \vdash_*$

(11) $(\# * w_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_1, i_1, j_1)$.

Also, since w_1 and w'_1 coincide on their first k_0-2 characters, and in the transitions between (7) and (8) no head scans the last $n-(k_0-2)$ characters of w'_1, we must have:

(12) $(\# * w_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_1, i_1, j_1) \vdash_*$

(13) $(\# * w_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q,i,j)$

Finally, by (4), and (5) we must have:

(14) $(\# * w_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q,i,j) \vdash_*$

(15) $(\# * w_1 * w_2 * \$ * w'_1 * w_2 * \cancel{c}, q_2, i_2, j_2)$

Since $\min(i_2, j_2)=n+3$ and from the configuration in (9) M eventually reaches an accepting state, the same is true for the configuration in (15). Hence the input $\# * w_1 * w_2 * \$ * w'_1 * w_2 * \not{c}$ is accepted by M - a contradiction. □

Now for each w_2 in $\{0,1\}^n$ let $A(w_2)$ denote the set of all w_1 in $\{0,1\}^n$ for which in (1) above $\min(i,j) \leq k_0$. From Lemma 3.1 we immediately get

Lemma 3.2: For each w_2 in $\{0,1\}^n$

$$|A(w_2)| \leq 2^{k_0-2} |Q|^3 (4n+9)^6.$$

Proof: Follows from Lemma 3.1 by an obvious upper bound on the number of possibilities for $f(w_1,w_2)$. □

Now let

$$a_0 = \lceil \log(|Q|^3 (4n+9)^6 n) \rceil = \lceil 3\log|Q| + 6\log(4n+9) + \log n \rceil$$

and fix k_0 to be equal to $n+2-a_0$. For this k_0 we have from Lemma 3.2:

(15') $|A(w_2)| \leq \frac{2^n}{n}$

Let

$B=\{(w_1,w_2) \mid w_1,w_2 \text{ in } \{0,1\}^n \text{ and } w_1 \notin A(w_2)\}$, then by (15'):

(15") $|B| \geq 2^{2n}(1-\frac{1}{n})$.

Now consider inputs of the form:

(16) $\# * w_1 * w_2 * \$ * w_1 * * w_1 * w_2 * \not{c} \quad (w_1, w_2) \in B$

By definition of B we must have for each such input:

(17) $(\# * w_1 * w_2 * \$ * w_1 * * w_1 * w_2 * \not{c}, q_*, 1, 1) \vdash_*$

(18) $(\# * w_1 * w_2 * \$ * w_1 * * w_1 * w_2 * \not{c}, q, i, j)$

for some $q \in Q$ and where $\max(i,j)=3n+7$ and $\min(i,j) \geq k_0$. All inputs as in (16) are in L and are accepted by M. We now classify all inputs of (16) according to q,i,j of (18) and the string w_0 formed from the last a_0 characters of w_1. For each (w_1,w_2) in B let $g(w_1,w_2)=(w_0,q,i,j)$. We prove

Lemma 3.3: There exist $\hat{w} \in \{0,1\}^{a_0}$, $\hat{q} \in Q$, and $\hat{i}$, $\hat{j}$ satisfying $\max(\hat{i}, \hat{j})=3n+7$, $\min(\hat{i}, \hat{j}) \geq k_0$ such that $|C| \geq \frac{2^{2n}(1-\frac{1}{n})}{\theta(n,|Q|)}$ where

$C=\{(w_1,w_2) \mid (w_1, w_2) \in B \text{ and } g(w_1,w_2)=(\hat{w}, \hat{q}, \hat{i}, \hat{j})\}$ and θ is a fixed polynomial in n and $|Q|$.

Proof: Follows from (15") by counting the number of possibilities for (w,q,i,j). □

Now we consider inputs of the form:

(19) $\# * w_1 * w_2 * \$ * w_1 * * w_1 * w_2 * \not{c}$ $(w_1, w_2) \in C$

We denote such an input by $m(w_1, w_2)$. For every such input we must have:

(20) $(m(w_1,w_2),q_*,1,1) \vdash_*$

(21) $(m(w_1, w_2),\hat{q}, \hat{i}, \hat{j})$

Clearly every such input is accepted by M. We now use a counting argument to obtain an input not in L which is accepted by M, thus reaching a contradiction. The counting argument is similar to the one used by Yao and Rivest [YR]. They have proved that if the language L_b is defined by

$L_b=\{w_1 * w_2 * \cdots * w_{2b} \mid w_i \in \{0,1\}^*$ and $(w_i=w_{2b+1-i})$ for $1 \leq i \leq 2b\}$ then L_b is recognizable by a k-DFA if $b \leq \binom{k}{2}$, and L_b is not recognizable even by a k-NFA if $b > \binom{k}{2}$. In order to show this they consider the language

$L_b^n=\{w_1 * w_2 * \cdots * w_{2b} \mid w_i \in \{0,1\}^n$ and $(w_i=2_{2b+1-i})$ for $1 \leq i \leq 2b\}$ and show that for n large enough and k such that $b > \binom{k}{2}$, any k-NFA which accepts every string in L_b^n must also accept a string not in L_b. The rest of our proof of Theorem 1.1 resembles their proof for the case $b=2$. For any (w_1,w_2) in C let the ***signature*** of $m(w_1, w_2)$ (called "pattern" in [YR]) be the sequence,

$C_t=(q''_t, i''_t, j''_t)$, $t=0,1,\ldots$, defined as follows:

$C_0=(\hat{q}, \hat{i}, \hat{j})$, and inductively: for $t \geq 0$

$C_{t+1}=(q''_{t+1}, i''_{t+1}, j''_{t+1})$ if

(22) $(m(w_1,w_2),q''_t,i''_t,j''_t) \vdash_*$

(23) $(m(w_1, w_2), q''_{t+1}, i''_{t+1}, j''_{t+1})$

and furthermore, the configuration in (23) is the first one during the computation, starting from the configuration in (22), in which some head has just moved into a position containing "$*$" or "$\not{c}$".

Now, clearly a signature of $m(w_1,w_2)$ is a sequence of at most 12 elements. Hence the number of possible signatures is bounded above by some polynomial $\theta_1(|Q|,n)$. Thus, using Lemma 3.3, we have:

Lemma 3.4: There exists a collection D of inputs $m(w_1,w_2)$, $(w_1, w_2) \in C$, all having the same signature, such that

$|D| \geq \dfrac{2^{2n}(1-\frac{1}{n})}{\theta_2(n, |Q|)}$, where $\theta_2(n, |Q|)$ is a polynomial in n and $|Q|$.

Now consider the suffix

(24) $*w_2*\$*w_1**w_1*w_2*\not c$

of $m(w_1, w_2)$. In [YR] Yao and Rivest mention an observation due to A. Rosenberg which, when applied to our case, implies that for each suffix as in (24) there is an $i_0 \in \{1,2\}$ such that whenever one head of M is scanning one w_{i_0} of this suffix, the other head is not scanning the other w_{i_0} of this suffix at that time. This is true since if the two heads are both on the $*w_1**w_1*$ portion simultaneously, none of them can ever again scan the leftmost w_2 in (24). We now prove

Lemma 3.5: For n large enough there exist two distinct inputs, $m(\hat{w}_1, \hat{w}_2)$ and $m(\hat{w}'_1, \hat{w}'_2)$ in D such that $m(\hat{w}_1, \hat{w}_2)$ and $m(\hat{w}'_1, \hat{w}'_2)$ have the same i_0 and such that $\hat{w}_{i_1} = \hat{w}'_{i_1}$, where $\{i_1\} = \{1,2\} - \{i_0\}$.

Proof: For n large enough there are at least $2^{1.5n}$ inputs in D which have the same i_0. The number of possible w_{i_1} is at most 2^n. □

Now consider the inputs $m(\hat{w}_1, \hat{w}_2)$ and $m(\hat{w}'_1, \hat{w}'_2)$. From Lemma 3.3 we know that:

(25) $(m(\hat{w}_1, \hat{w}_2), q_*, 1, 1) \vdash_* (m(\hat{w}_1, \hat{w}_2), \hat{q}, \hat{i}, \hat{j})$

(26) $(m(\hat{w}'_1, \hat{w}'_2), q_*, 1, 1) \vdash_* (m(\hat{w}'_1, \hat{w}'_2), \hat{q}, \hat{i}, \hat{j})$

Furthermore:

(27) $\max(\hat{i}, \hat{j}) = 3n+7$, $\min(\hat{i}, \hat{j}) \geq k_0$

(28) $\hat{w}_1$, $\hat{w}'_1$ coincide on their last a_0 characters.

(29) $m(\hat{w}_1, \hat{w}_2)$ and $m(\hat{w}'_1, \hat{w}'_2)$ have the same signature and the same i_0.

(30) $m(\hat{w}_1, \hat{w}_2)$ and $m(\hat{w}'_1, \hat{w}'_2)$ are both accepted by M.

Now, let m' be the string obtained from $m(\hat{w}_1, \hat{w}_2)$ as follows: in the suffix $*\hat{w}_1*\hat{w}_2*\not c$ of $m(\hat{w}_1, \hat{w}_2)$ replace $\hat{w}_{i_0}$ by $\hat{w}'_{i_0}$. Clearly,

(30') $(m', q_*, 1, 1) \vdash_* (m', \hat{q}, \hat{i}, \hat{j})$

Obviously m' is not in L. The following lemma contradicts the fact that L=L(M).

Lemma 3.6: The string m' is accepted by M.

Proof: We use the method of [YR]. Let $s_0=(m(\hat{w}_1, \hat{w}_2),\hat{q},\hat{i},\hat{j})$ and ~~let~~ $s_l=(m(\hat{w}_1,\hat{w}_2),q_l,i_l,j_l)$ be the l-th configuration of M on input m($\hat{w}_1, \hat{w}_2$), counting from s_0. Similarly, let $s'_0=(m(\hat{w}'_1,\hat{w}'_2),\hat{q},\hat{i},\hat{j})$, and $s'_l=(m(\hat{w}'_1,\hat{w}'_2),q'_l,i'_l,j'_l)$ be the l-th configuration of M on input m($\hat{w}'_1, \hat{w}'_2$), starting from s'_0. Consider the two sequences:

$d_l=(q_l,i_l,j_l)$, $d'_l=(q'_l,i'_l,j'_l)$, l=1,2,...

Since m($\hat{w}_1, \hat{w}_2$) and m($\hat{w}'_1, \hat{w}'_2$) have the same signature, both d_l, d'_l have the sequence $C_t=(q''_t, i''_t, j''_t)$ as a subsequence (see (22), (23)) and furthermore, the sequence d_l can be divided into blocks as follows:

(31) $C_0\bar{b}_1C_1\bar{b}_2C_2 \cdots C_{v-1}\bar{b}_vC_v$

where $\bar{b}_i$ is the portion of the sequence between C_i and C_{i+1}, and during the corresponding period of the computation on m($\hat{w}_1, \hat{w}_2$) no head advances into a position containing "*" or "¢". Similarly, d'_l can be represented by:

(32) $C_0\bar{b}'_1C_1\bar{b}'_2C_2 \cdots C_{v-1}\bar{b}'_vC_v$

Now let $\hat{s}_l=(m',\hat{q}_l,\hat{i}_l,\hat{j}_l)$ be the sequence of configurations of M on input m', starting from $\hat{s}_0=(m',\hat{q},\hat{i},\hat{j})$. Let $\hat{d}_l=(\hat{q}_l,\hat{i}_l,\hat{j}_l)$. We now prove by induction that for each 0≤h≤v, the sequence $\hat{d}_l$ has the following sequence

(33) $C_0\bar{b}''_1C_1\bar{b}''_2C_2 \cdots C_{h-1}\bar{b}''_hC_h$

as an initial segment, where $\bar{b}''_i=\bar{b}_i$ if during the portion of the computation of M on m($\hat{w}_1, \hat{w}_2$) corresponding to $C_{i-1}\bar{b}_i$, no head is scanning $\hat{w}_{i_0}$ in the suffix $*\hat{w}_1*\hat{w}_2*¢$. Otherwise $\bar{b}''_i=\bar{b}'_i$. Clearly the assertion is true for h=0. Now suppose it is true for all h<h_0. To show that it is true for h=h_0, let $\bar{e}_{h_0}$ be the portion of the sequence $\hat{d}_l$, starting from C_{h_0-1} and until some head advances into a position containing a '*' or a '¢'. We distinguish between two cases. If during the portion $C_{h_0-1}\bar{b}_{h_0}$ of the transition sequence on m($\hat{w}_1, \hat{w}_2$) no head is scanning $\hat{w}_{i_0}$ in the suffix $*\hat{w}_1*\hat{w}_2*¢$ of m($\hat{w}_1, \hat{w}_2$), then clearly $\bar{e}_{h_0}=C_{h_0-1}\bar{b}_{h_0}C_{h_0}$. This is true by the induction hypothesis and the fact that the relevant portion of m' differs from the relevant portion of m($\hat{w}_1, \hat{w}_2$) only on the $\hat{w}_{i_0}$ part of $*\hat{w}_1*\hat{w}_2*¢$. Now consider the case in which during the portion $C_{h_0-1}\bar{b}_{h_0}$ of the transition sequence on m($\hat{w}_1, \hat{w}_2$) some head scans the $\hat{w}_{i_0}$ in the suffix $*\hat{w}_1*\hat{w}_2*\not{c}$. Then during the portion $\bar{\tau}_{k_0}$ the same head is scanning $\hat{w}'_{i_0}$ in m'. Also, during the portion $C_{k_0-1}\bar{\delta}'_{k_0}$ of the transition sequence on m($\hat{w}'_1, \hat{w}'_2$), the same head scans $\hat{w}'_{i_0}$ in the suffix $*\hat{w}'_1*\hat{w}'_2*\not{c}$. By the definition of i_0 no head is scanning the other $\hat{w}'_{i_0}$ in m($\hat{w}'_1, \hat{w}'_2$). Hence during this period M does the same transitions on m' as on m($\hat{w}'_1, \hat{w}'_2$). Hence $\bar{\tau}_{k_0}=C_{k_0-1}\bar{\delta}'_{k_0}C_{k_0}$ in this case.

The lemma now follows by letting h=v. □

Using Lemma 3.6, the proof of Theorem 1.1 is now completed.

4. Further observations and conclusions

We have answered the case k=2 of the open question whether a k-DFA can perform string-matching. We have actually shown that a 2-DFA cannot solve the *decision problem*: "Is the pattern a suffix of the text?" This latter problem can be trivially solved in linear time by a 2-head 2 way DFA or a 2-head 1 -way NFA. It can be even solved by a 3-DFA as follows: head 1 moves to the $ sign; head 1 and head 2 move simultaneously until head 1 reaches the end; then head 3 and head 2 do the matching.

A trivial modification of our proof implies that a 2-DFA cannot perform string-matching even if the pattern is restricted to be of the form awb where $w \in \Sigma^*$ for some finite alphabet Σ and a,b are characters not in Σ. This special case too can be solved in linear time by a two-head two-way DFA. Our proof for the 2-DFA is clearly still valid even if we impose both of the above restrictions.

As a matter of fact, we can further strengthen theorem 1.1 by giving the 2-head 1-way DFA o(n) 2-way storage space. By the same proof, replacing |Q| in the old proof by the number of combinations of storage tape and the internal states ($|Q|2^{o(n)}$ instead of |Q|), we therefore have

Corollary 4.1: Define L_1={#*x*$y¢ | *x* is a suffix of y and * is not in x}. L_1 cannot be accepted by a deterministic Turing machine with two one-way input heads and o(n) storage space.

The language L_1 is in a sense much easier than string-matching. To demonstrate this we next demonstrate a class of machines that cannot do string-matching but can recognize L_1. In [DG] Duris and Galil defined 2sdfa(2) to be a 2-head 2-way DFA with one head "blind" (can only see end-markers), and they proved that 2sdfa(2) cannot perform string-matching. Here we will show that 2sdfa(2) can recognize our L_1 in the following theorem

Theorem 4.2: L_1 can be accepted by a 2sdfa(2) even if the 2 heads cannot see each other. Therefore the class of languages accepted by 2sdfa(2) is not contained in the class of languages accepted by 2-head 1-way DFA's.

Proof: We shall refer to the "blind" head as the *counter* and will call other head the *ordinary head*. To recognize L_1 by a 2-way DFA with one ordinary head and one counter, which cannot see each other, we use the following strategy:

Initialization: The counter stays at the # sign. The ordinary head moves to the right end-marker ¢. Then it moves left one position. If this position does not contain a "*", the input is rejected. Otherwise the ordinary head moves left until another "*" is encountered. The portion of the text which consists of this "*" and the characters to the right of this position is of the form *z* where * is not a character of z.

Step k (k=0,1,2,...): The counter is at the # sign. The ordinary head is at position k to the right of the "*" in *z. The ordinary head remembers the character at this position. Then the ordinary head moves to the left and at the same time the counter moves to the right, until the ordinary head arrives at a "*". At this point clearly the counter is k positions to the right of #. Then the ordinary head moves left until it reaches the # sign. Next, the ordinary head moves right, and at the same time the counter moves left. When the counter reaches the # sign the ordinary head is k-1 positions to the right of the leftmost * in the pattern *x*. Now the ordinary head moves one step forward and performs one character matching. Using the same trick the heads then position themselves for the initial position in step k+ 1.

Thus by *step* $|x|+1$, the 2sdfa(2) can make its decision. (This requires $O(n^2)$ time.) □

Remark 1: This result shows that although a 2sdfa(2) cannot perform normal string-matching, it can do simpler string-matching (accepting L_1) which cannot be done by any 2-head 1-way DFA.

Remark 2: If the pattern is restricted to be over a one letter alphabet, then string matching can be done by a 3-DFA whose heads can see each other. This is done as follows: Suppose the pattern is $x=0^i$. Head 3 moves right one position. Then heads 2 and 3 move simultaneously until head 3 reaches the \$ sign. At this point head 2 is on the last character of the pattern x. Now the following procedure is performed over and over again until 0^i is discovered in the text or head 3 reaches the right end-marker:

(1) Move head 3 to the beginning of the next block of zeroes in the text.

(2) Move heads 1, 2 simultaneously, while keeping the distance between them fixed, until head 2 meets head 3. Then continue to move heads 1, 2 simultaneously as long as head 2 sees a 0. If head 1 reaches the position of head 3 before head 2 reaches a 1, then 0^i has been discovered in the text. Otherwise go to (1).

5. Acknowledgements

The authors would like to thank Professors Stephen A. Cook, Zvi Galil, and Juris Hartmanis for encouragement and helpful discussions.

6. References

[DG] P. Duris and Z. Galil, Fooling a two-way automaton or one pushdown store is better than one counter for two way machines, Proceedings 13th ACM STOC (1981) pp. 177-188.

[GS] Z. Galil and J. Seiferas, Time-space optimal string-matching, Proceedings 13th ACM STOC (1981) pp.106-113.

[HU] J. Hopcroft and J. Ullman, Introduction to automata theory, languages, and computation, Addison-Wesley (1979).

[YR] A. Yao and R. Rivest, k+1 heads are better than k, J. ACM, 25 (1978), pp. 337-340.