THE EQUIVALENCE PROBLEM FOR REGULAR EXPRESSIONS WITH INTERSECTION IS NOT POLYNOMIAL IN TAPE

Harry B. Hunt III†

TR 73-161

March 1973

Department of Computer Science
Cornell University
Ithaca, New York 14850

THE EQUIVALENCE PROBLEM FOR REGULAR EXPRESSIONS WITH INTERSECTION IS NOT POLYNOMIAL IN TAPE

Harry B. Hunt III†

Cornell University
Ithaca, New York

Abstract:

We investigate the complexity of several predicates on regular sets. In particular, we show:

(1) the equivalence and emptiness problem for regular expressions using only the operators $-$ $\cup$, $\cdot$, and $\cap$ are p-complete;

(2) the emptiness problem for regular expressions using the operators $-$ $\cup$, $\cdot$, $\cap$, and $*$ is tape-hard;

(3) the emptiness problem for regular expressions using the operators $-$ $\cup$, $\cdot$, $\cap$, and 2 is tape-hard;

(4) the equivalence problem for regular expressions using the operators $-$ $\cup$, $\cdot$, $\cap$, and $*$ is not polynomial in tape; and

(5) the equivalence problem for regular expressions using the operators $-$ $\cup$, $\cdot$, $\cap$, and 2 requires exponential time.

---

†This research was supported by a National Science Foundation Graduate Fellowship in Computer Science.

§1. Introduction

Before describing our results, we need several definitions. Let $\bar{\Sigma}$ be some countably infinite alphabet.

Definition 1.1: For all positive integers k, ndtape(n^k) is the class of all languages (over $\bar{\Sigma}$) recognized by some nondeterministic n^k-tape bounded Turing machine. We define dtape(n^k) analogously.

Definition 1.2: PTIME is the class of all languages (over $\bar{\Sigma}$) recognized by some deterministic polynomial time bounded Turing machine.

Definition 1.3: NPTIME is the class of all languages (over $\bar{\Sigma}$) recognized by some nondeterministic polynomial time bounded Turing machine.

Definition 1.4: PTAPE is the class of all languages (over $\bar{\Sigma}$) recognized by some deterministic or nondeterministic polynomial tape bounded Turing machine.

Definition 1.5: Let $\Sigma = \{0,1\}$. Let Π be the class of functions from Σ^* into Σ^* computable by some polynomial time bounded Tm. Let $\mathscr{L}$ and $\mathscr{M}$ be languages. Then $\mathscr{L} \underset{\text{Ptime}}{\leq} \mathscr{M}$ ($\mathscr{L}$ is p-reducible to $\mathscr{M}$) if there is a function $f \in \Pi$ such that $x \in \mathscr{L} \iff f(x) \in \mathscr{M}$. A language is p-complete if $\mathscr{M}$ (or $\bar{\mathscr{M}}$) $\in$ NPTIME and every language in NPTIME is p-reducible to M. A language $\mathscr{M}$ is p-hard if every language in NPTIME is p-reducible to it.

<u>Definition 1.6</u>: A language $\mathscr{L}$ is <u>tape-hard</u> if PTAPE $\underset{\text{Ptime}}{\leq} \mathscr{L}$.

A language $\mathscr{L}$ is <u>tape-complete</u> if $\mathscr{L}$ is tape-hard and $\mathscr{L} \in$ PTAPE.

<u>Definition 1.7</u>: $c\backslash\mathscr{L} = \{x \mid c\,x \in \mathscr{L}\}$. $\mathscr{L}/c = \{x \mid x\,c \in \mathscr{L}\}$

<u>Definition 1.8</u>: Let $\mathscr{P}$ be a predicate on $2^{\{0,1\}^*}$.

$\mathscr{P}_L = \bigcup_{x\in\{0,1\}^*} \{x\backslash\mathscr{L} \mid \mathscr{P}(\mathscr{L}) = \text{true}\}$. $\mathscr{P}_R = \bigcup_{x\in\{0,1\}^*} \{\mathscr{L}/x \mid \mathscr{P}(\mathscr{L})$ is true$\}$.

<u>Definition 1.9</u>:

(1) A $(\cup,\cdot)$ - regular expression is defined recursively as follows:

(a) λ, ϕ, 0, and 1 are $(\cup,\cdot)$ - regular expressions;

(b) If A and B are $(\cup,\cdot)$ - regular expressions then (A) $\cup$ (B) and (A)$\cdot$(B); and

(c) A is a $(\cup,\cdot)$ - regular expression iff it satisfies (a) or (b).

(2) A $(\cup,\cdot,\cap)$ - regular expression is defined analogously with (b) replaced by

(b_2) If A and B are $(\cup,\cdot,\cap)$ - regular expressions then so are

(A) $\cup$ (B), (A) $\cdot$ (B), and (A) $\cap$ (B).

(3) A $(\cup,\cdot,\cap,*)$ - regular expression is defined analogously with (b) replaced by

(b_3) If A and B are $(\cup,\cdot,\cap,*)$ - regular expressions then so are

(A) $\cup$ (B), (A)$\cdot$(B), (A) $\cap$ (B), and (A)*.

(4) A $(\cup,\cdot,\cap,2)$ - regular expression is defined analogously with (b) replaced by

(b_4) If A and B are $(\cup,\cdot,\cap,2)$ - regular expressions then so are $(A) \cup (B)$, $(A)\cdot(B)$, $(A) \cap (B)$, and $(A)^2$.

Definition 1.10:

(1) Emptiness$((\cup,\cdot,\cap)) = \{\alpha | \alpha$ is a $(\cup,\cdot,\cap)$ - regular expression and $L(\alpha) = \phi\}$.

(2) Equivalence $((\cup,\cdot,\cap)) = \{(\alpha,\beta) | \alpha$ and β are $(\cup,\cdot,\cap)$-regular expressions and $L(\alpha) = L(\beta)\}$.

Emptiness $((\cup,\cdot))$, Equivalence $((\cup,\cdot))$, Emptiness $((\cup,\cdot,\cap,*))$, Equivalence $((\cup,\cdot,\cap,*))$, Emptiness $((\cup,\cdot,\cap,2))$, and Equivalence $((\cup,\cdot,\cap,2))$ are defined analogously.

In Section 2 we find several new p-complete problems (e.g., equivalence or emptiness of $(\cup,\cdot,\cap)$-regular expressions). In section 3 we show that Emptiness $((\cup,\cdot,\cap,*))$ is tape-hard and that Emptiness $((\cup,\cdot,\cap,2))$ is tape-hard.

In Section 4 we present our most significant single result. We show Equiv$((\cup,\cdot,\cap,*))$ is not polynomial in tape. We find a hierarchy of problems of arbitrarily high polynomial tape complexity. We also show that Equiv$((\cup,\cdot,\cap,2))$ requires exponential time.

§2. P-complete Problems

In [2] we showed that Equiv$((\cup,\cdot))$ is p-complete. For the convenience of the reader we give the proof here.

Theorem 2.1: Equiv$((\cup,\cdot))$ is p-complete.

Proof: Cook [1] has shown that NPTIME $\underset{\text{Ptime}}{\leq}$ D_3-Tautology. Let $f = \bigvee_{i=1}^{m} c_i$ be a D_3-Boolean form. Then each c_i is the product of at most three literals. For each c_i let $\beta_i = \beta_{i1}\cdot\beta_{i2}\cdot\ldots\cdot\beta_{in}$. (The number of variables appearing in f is n.)

$$\beta_{ij} = \begin{cases} (0 \cup 1) & \text{if } x_j \text{ and } \bar{x}_j \text{ are not literals in } c_i, \\ (0) & \text{if } \bar{x}_j \text{ is a literal in } c_i, \text{ and} \\ (1) & \text{if } x_j \text{ is a literal in } c_i. \end{cases}$$

Let $\beta = \bigcup_{i=1}^{m} \beta_i$. Then $y \in L(\beta)$ iff y satisfies some clause c_i. Then, $L(\beta) = \{0,1\}^n$ iff f is a tautology.

Next we show Inequiv$((\cup,\cdot)) = \{(\alpha,\beta) \mid \alpha$ and β are $(\cup,\cdot)$-reg. expr. and $L(\alpha) \neq L(\beta)\} \in$ NPTIME. Given a regular expression β and a string x, we can check if $x \in L(\beta)$ in deterministic time a polynomial in $\max(|\beta|,|x|)$. But, β a $(\cup,\cdot)$-regular expression implies for all $x \in L(\beta)$, $|x| \leq |\beta|$. (Here β is a string over the alphabet $\{(,),\cup,\cdot,0,1\}$.) Hence, given two $(\cup,\cdot)$-regular expressions α and β, to verify $L(\alpha) \neq L(\beta)$, we need only guess a string x of length $\leq \max(|\alpha|,|\beta|)$ such that $x \in L(\alpha) \cap \overline{L(\beta)}$ or $x \in \overline{L(\alpha)} \cap L(\beta)$.

Lemma 2.2: Emptiness$((\cup,\cdot)$ $\in$ PTIME.

Proof: Emptiness for context-free grammars is decidable in deterministic polynomial time.

<u>Theorem 2.3: (Hopcroft and Hunt)</u> Equiv(($\cup,\cdot,\cap$)) is p-complete.

<u>Proof</u>: From 2.1 Equiv(($\cup,\cdot,\cap$)) is trivially seen to be p-hard. We now show Inequiv(($\cup,\cdot,\cap$)) $\in$ NPTIME. As in the proof of 2.1 α a ($\cup,\cdot,\cap$)-regular expression implies $\forall x \in L(\alpha)$, $|x| \le |\alpha|$. Thus, the proof of 2.1 would also prove 2.3 provided given x with $|x| \le \max(|\alpha|,|\beta|)$ we can in det. polynomial time decide if $x \in [L(\alpha) \cap \overline{L(\beta)}] \cup [\overline{L(\alpha)} \cap L(\beta)]$.

Let $x = x_1 \cdot \ldots \cdot x_n$ be a string that differentiates between α and β. $n \le \max(|\alpha|,|\beta|)$. W.ℓ.g. assume $x \in L(\alpha) \cap \overline{L(\beta)}$. There are at most $\frac{n(n+1)}{2}$ proper substrings of x. There are at most $(|\alpha|+|\beta|)$ subregular expressions of α and β. (This is easily seen by induction. $\alpha = \alpha' \cdot \alpha''$, $\alpha' \cup \alpha''$, or $\alpha' \cap \alpha''$ implies the number of subregular expressions of $\alpha \le |\alpha'| + |\alpha''| + 1 = |\alpha|$.)

For each subregular expressions of α and for each subregular expression of β, we construct an array that indicates which substrings of x are in the regular set denoted by the subexpression and which are <u>not</u>. The i,j<u>th</u> element of the array A corresponds to that proper substring of x beginning with the i<u>th</u> and ending with the j-1<u>st</u> element of x, i.e. $x_i x_{i+1} \ldots x_{j-1}$. (Thus $a_{ij} = 0$ for $j < i+1$.)

For each subregular expression A we must know if $\lambda \in L(A)$, where λ denotes the empty word. Clearly given A and B if we know whether $\lambda \in L(A)$ or not and whether $\lambda \in L(B)$ or not, then we know if $\lambda \in L(A \cup B)$, $L(A \cap B)$, or $L(A \cdot B)$.

If |subregular expr| = 1 then the number of nonzero elements of the corresponding array $\le$ n. (It is easy in det. polynomial time to compute this matrix.) If |expr|>1 then expr = $\text{expr}_1 \cup$

$expr_2$, $expr_2 \cap expr_2$, or $expr_1 \cdot expr_2$. Let A, A_1, and A_2 be the corresponding matrices.

<u>Case 1</u>. If $expr = expr_1 \cup expr_2$, then $A = A_1 \vee A_2$ and $\lambda \in L(expr)$ iff $\lambda \in L(expr_1)$ or $\lambda \in L(expr_2)$, i.e., $A^{ij} = A_1^{ij} \vee A_2^{ij}$.

<u>Case 2</u>. If $expr = expr_1 \cap expr_2$, then $A = A_1 \wedge A_2$ and $\lambda \in L(expr)$ iff $\lambda \in L(expr_1)$ and $\lambda \in L(expr_2)$, i.e., $A^{ij} = A_1^{ij} \wedge A_2^{ij}$.

<u>Case 3</u>. If $expr = expr_1 \cdot expr_2$, then $\lambda \in L(expr)$ iff $\lambda \in L(expr_1)$ and $\lambda \in L(expr_2)$.

$$A^{ij} = \begin{cases} \bigcup_{i<k<j} A_1^{ik} \cdot A_2^{kj} & \text{if } \lambda \notin L(expr_1) \text{ and } \lambda \notin L(expr_2), \\ A_2^{ij} \cup \bigcup_{i<k<j} A_1^{ik} \cdot A_2^{kj} & \text{if } \lambda \in L(expr_1) \text{ and } \lambda \in L(expr_2), \\ A_1^{ij} \cup \bigcup_{i<k<j} A_1^{ik} \cdot A_2^{kj} & \text{if } \lambda \notin L(expr_1) \text{ and } \lambda \in \lambda(expr_2), \\ A_1^{ij} \cup A_2^{ij} \cup \bigcup_{i<k<j} A_1^{ik} \cdot A_2^{kj} & \text{if } \lambda \in L(expr_1) \text{ and } \lambda \in L(expr_2). \end{cases}$$

Theorems 2.1 and 2.2 suggest by analogy that Equiv($(\cap,\cdot)$) is p-complete. Reflection upon this should show that Equiv($(\cap,\cdot)$ behaves as Emptiness($(\cup,\cdot)$).

<u>Lemma 2.4</u>: Equiv($(\cap,\cdot)$) $\in$ PTIME.

<u>Proof</u>: α a $(\cap,\cdot)$-regular expression implies $|L(\alpha)| \leq 1$.

<u>Theorem 2.5</u>: Emptiness($(\cup,\cdot,\cap)$) is p-complete.

<u>Proof</u>: D_3-tautology is the same as C_3-satisfiability. Let the variables appearing in f be $\{x_1,\ldots,x_n\}$. Then f is a D_3-Boolean form iff (by def.) $f = \bigvee_{i=1}^{m} c_i$, where each term c_i is the product (and) of at most three literals, i.e., c_i is the product of at most three elements of $\{x_1,\bar{x}_1,x_2,\bar{x}_2,\ldots,x_n,\bar{x}_n\}$.

W.ℓ.g. we assume no pair of literals $x_j,\bar{x}_j$ appears in any term of f. Then f is a tautology iff $\neg f = \bigwedge_{i=1}^{m} (\neg y_{i_1} \vee \neg y_{i_2} \vee \neg y_{i_3})$ is <u>not</u> satisfiable. (Here $c_i = y_{i1}\cdot y_{i2}\cdot y_{i3}$).

Let $\beta = \beta_1 \cap \beta_2 \cap \ldots \cap \beta_m$, where $\beta_i = \beta_i^{\,1} \cup \beta_i^{\,2} \cup \beta_i^{\,3}$ and $\beta_i^{\,j} = \beta_{i1}^{\,j}\cdot\ldots\cdot\beta_{in}^{\,j}$ and

$$\beta_{ik}^{\,j} = \begin{cases} (0 \cup 1) & \text{if } y_{ij} \neq x_k \text{ and } y_{ij} \neq \bar{x}_k, \\ 1 & \text{if } y_{ij} = \bar{x}_k, \text{ and} \\ 0 & \text{if } y_{ij} = x_k. \end{cases}$$

Let $x = t_1 \ldots t_n \in \{0,1\}^n$. Then $x \in L(\beta)$ iff $\neg f(t_1,\ldots,t_n) = \text{true}$, i.e., $\neg f$ is satisfiable.

We have just seen that the addition of intersection is enough to make an otherwise simple emptiness problem difficult. We have also seen that the addition of intersection in some cases does not radically alter the complexity of equivalence problems. We shall investigate these two phenomena repeatedly in the remainder of this paper. Finally, Theorems 2.1 and 2.3 can be used to show that a variety of predicates on the finite regular sets are p-complete.

<u>Theorem 2.6</u>: The following are p-complete.

(1) $\{\beta|\beta$ is a $(\cup,\cdot)$-regular expression and $L(\beta) = L(\beta)^{\underline{rev}}\}$;

(2) $\{\beta|\beta$ is a $(\cup,\cdot,\cap)$-regular expression and $L(\beta) \in \{R_1,\ldots,R_{i_0}\}$, where $\{R_i,\ldots,R_{i_0}\}$ is some fixed finite set of finite sets and $R_1 = \phi\}$;

(3) $\{\beta|\beta$ is a $(\cup,\cdot)$-regular expression and $L(\beta) = \text{Init}(L(\beta))\}$;

(4) $\{\beta|\beta$ is a $(\cup,\cdot)$-regular expression and $L(\beta) = \text{Final}((L(\beta))\}$;

(5) $\forall i \geq 2$ $\{\beta|\beta$ is a $(\cup,\cdot,\cap)$-regular expression and $L(\beta) = L(\beta)^i\}$;

(6) $\{\beta|\beta$ is a $(\cup,\cdot)$-regular expression and $L(\beta) = \gamma(L(\beta))\}$
$[\gamma(R) = \{x|\exists\ y \in R$ and $|x| = |y|\}$; and

(7) (1), (3), (4), and (6) above with $(\cup,\cdot)$ replaced by $(\cup,\cdot,\cap)$.

<u>Proof</u>: Left to the reader.

§3. Emptiness Problems Using Intersection

We show that Emptiness(($\cup,\cdot,\cap,*$)) and Emptiness(($\cup,\cdot,\cap,2$)) are tape-hard. In both cases the proof strongly utilizes the parallel nature of "$\cap$". We feel that it is this parallel structure that causes these emptiness problems to be difficult.

We first note that in either case if we delete "$\cap$", then the corresponding emptiness problem $\in$ PTIME.

Lemma 3.1: Emptiness(($\cup,\cdot,*$)) and Emptiness(($\cup,\cdot,2$)) are both elements of PTIME.

Proof We can convert a regular expression to an equivalent type 3 grammar in det. polynomial time. As is well-known the emptiness problem for context-free grammars is deterministic polynomial in time. (It is interesting to point out that this appears not to be the case for indexed context-free grammars. See [3].)

In the second case if β is a ($\cup,\cdot,2$)-regular expression, then $L(\beta) = \phi$ iff $L(\beta') = \phi$, where β' results from β by deleting all "2's".

We now introduce a certain amount of notation. If the reader is familiar with our work ([2] or [3]) or the work of Meyer and Stockmeyer ([4]), he should skip directly to the statement and proof of the next theorem. Otherwise, he should read the next several paragraphs.

<u>Def. 3.2</u>: Let M be any Tm with tape symbols T and states S. Assume 0,1,b̸ ∈ T, where b̸ denotes the blank tape square. An <u>instantaneous description</u> (i.d.) of M is a word in T*. (SxT)·T*.

<u>Definition 3.3</u>: Given any i.d. x = y·(s x t)·z for y,z ∈ T*, the next i.d., $Next_M(x)$ is defined as follows: if when M is in state s with its read-write head scanning symbol t, M enters state s' and writes symbol t' then $Next_M(x)$ is

1) y·(s' x t')·z if M does not shift its head,
2) y·t'·(s' x u)·w if M shifts its head right and z = u·w for u ∈ T and w ∈ T*,
3) w·(s' x u)·t'·z if M' shifts its head left and y = w·u for u ∈ T and w ∈ T*,

and 4) undefined if (s x t) is a halting condition, or if (s x t) is rightmost symbol of x and M shifts right, or if (s x t) is the leftmost symbol of x and M shifts left.

<u>Definition 3.4</u>: $Next_M(x,0)$ = x if x is an i.d and is undefined otherwise.

$$Next_M(x,n+1) = Next_M(Next_M(x,n)).$$

<u>Definition 3.5</u>: Let # be a symbol not in T ∪ (s x T). The <u>computation</u> $C_M(x)$ of M from x is the following word in ({#} ∪ T ∪ (S x T))*:

$$C_M(x) = \#\cdot Next_M(x,0)\cdot\#\cdot Next_M(x,1)\cdot\#\cdot\ldots\cdot$$
$$\#\cdot Next_M(x,n)\cdot\# .$$

Here, n is the least positive integer such that $(q_f \times t)$ occurs in $Next_M(x,n)$ for some $t \in T$ and designated halting state q_f. The computation is undefined if there is no such n.

Given M as in the preceeding definitions, let $\Sigma = \{\#\} \cup T \cup (S \times T)$. For any i.d. x, let $C_M(i,x)$ be the $i^{\underline{th}}$ symbol of $C_M(x)$ for $1 \leq i \leq |C_M(x)|$. There is a function $f_M: \Sigma^3 \rightarrow \Sigma$ such that for any i.d. x and any integer i, with $|x| + 2 \leq i \leq |C_M(x)|$, $C_M(i,x) = f_M(C_M(i-(|x|+2),x), C_M(i-(|x|+1),x), C_M(i-(|x|,x))$. This follows since the $i^{\underline{th}}$ symbol of $Next_M(y)$ is determined uniquely by the $i\text{-}1^{\underline{st}}$, $i^{\underline{th}}$ and $i\text{+}1^{\underline{st}}$ symbols of y.

<u>Theorem 3.6</u>:

(1) Emptiness $((\cup,\cdot,\cap,*)) \underset{Ptime}{\geq}$ PTAPE.

(2) Emptiness$((\cup,\cdot,\cap,2)) \underset{Ptime}{\geq}$ PTAPE.

<u>Proof</u>:

As shown in [2] we need only simulate all det. LBA's. Let M be a linear bounded automaton with states S, tape alphabet T, designated halting state $q_f \in S$, and designated start state $q_0 \in S$. We assume q_f is <u>final</u>.

Let $\Sigma = \{\#\} \cup T \cup (S \times T)$. Let $x = x_1 \cdot \ldots \cdot x_n$ be a given input to M. We construct a $(\cup,\cdot,\cap,*)$-regular expression β_x such that $L(\beta_x) = \phi$ iff M does not accept x and such that the construction is det. polynomial in time.

β_x is characterized as follows: β_x is the intersection of

1) all words that begin with $\# \cdot (q_0,x_1) \cdot x_2 \cdot \ldots \cdot x_n \cdot \#$,

2) all words that contain exactly one q_f,

3) all words of form $\#[(\Sigma-\#)^n\#]^+$, and

4) all words that do not violate the next-move requirement of M.

$$\beta_x \stackrel{\text{def.}}{=\!=} \#\cdot(q_0,x_1)\cdot x_2\cdot\ldots\cdot x_n\cdot\#\cdot\Sigma^* \quad (1)$$

$$\cap$$

$$(\Sigma - [\bigcup_{t\in T}(q_f,t)])^*\cdot[\bigcup_{t\in T}(q_f,t)]\cdot \quad (2)$$

$$(\Sigma - [\bigcup_{t\in T}(q_f,t)])^* \quad (3)$$

$$\cap$$

$$\#[(\Sigma-\#)^n\#][(\Sigma-\#)^n\#]^*$$

$$\bigcap_{i=0}^{n}\left[\left\{\Sigma^i\left[\bigcup_{\sigma_1,\sigma_2,\sigma_3\in\Sigma}\sigma_1\cdot\sigma_2\cdot\sigma_3\cdot\Sigma^{n-1}\cdot f_M(\sigma_1,\sigma_2,\sigma_3)\cdot\Sigma^{n-(i+1)}\right]\right\}^*\cdot\left\{\#\cdot(\Sigma-\#)^n\cdot\#,\ \#\right\}\right] \quad (4)$$

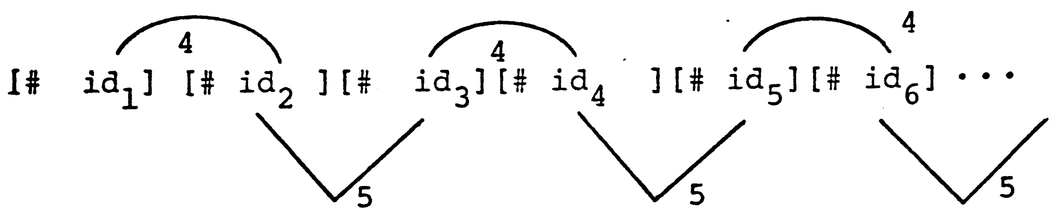
$$\bigcap_{i=0}^{n}\left[\#\cdot(\Sigma-\#)^n\cdot\left\{\Sigma^i\left[\bigcup_{\sigma_1,\sigma_2,\sigma_3,\sigma\in\Sigma}\sigma_1\cdot\sigma_2\cdot\sigma_3\cdot\Sigma^{n-1}\cdot f_M(\sigma_1,\sigma_2,\sigma_3)\cdot\Sigma^{n-(i+1)}\right]\right\}^*\cdot\left\{\#\cdot(\Sigma-\#)^n\cdot\#,\#\right\}\right]. \quad (5)$$

We claim $L(\beta_x) = \begin{cases}\phi, & \text{if M does not accept x and}\\ C_M(x), & \text{otherwise.}\end{cases}$

Clearly from (1), (2), and (3) a string $\alpha \in L(\beta_x)$ only if $\alpha \in \#\ [(\Sigma-\#)^n\#]^k$ for some $k \geq 1$, α begins properly with

$\#\cdot(q_0,x_1)\cdot x_2\cdot\ldots\cdot x_n\cdot\#$, and α has an accepting state. If α satisfies (4) then for each pair of consecutive i.d.'s (id_{2i-1},id_{2i}) starting with (id_1,id_2), id_{2i} follows from id_{2i-1}. Similarly if α satisfies (5) then for each pair of consecutive i.d.'s (id_{2i},id_{2i+1}) starting with (id_2,id_3), id_{2i+1} follows from id_{2i}.

Schematically this is illustrated by



Since id_1 is right by (1) this implies $\alpha = \#id_1\#id_2\#\ldots\#id_k\#$ is a valid computation if $\alpha \in L(\beta_y)$. By noting the size of β_x, it is clear that the reduction is deterministic polynomial time bounded.

(2)

As in the proof of (2) we need only simulate all det. LBA's. Let M be defined as in the proof of (1).

$$\beta_x \overset{\text{def.}}{=} \#\cdot((\Sigma-\#)^n\cdot\#,\Lambda)^{2^{cn}} \cap$$

$$\#\cdot(q_o,x_1)\cdot x_2\cdot x_3\cdot\ldots\cdot x_y\#\cdot((\Sigma-\#)^n\cdot\#,\Lambda)^{2^{cn}-1} \cap$$

$$(\Sigma,\Lambda)^{2^{cn}\cdot(n+1)}\ [\bigcup_{t\in T}(q_f,t)]\cdot(\Sigma,\Lambda)^{2^{cn}\cdot(n+1)} \cap$$

$$\bigcap_{i=0}^{n} \left\{ \Sigma^{i} \cdot \left[\bigcup_{\sigma_1,\sigma_2,\sigma_3,\in \Sigma} \sigma_1\cdot\sigma_2\cdot\sigma_3\cdot \Sigma^{n-1} \cdot f_M(\sigma_1,\sigma_2,\sigma_3) \cdot \Sigma^{n-(i+1)} \right] \cup \Lambda \right\}^{2^{cn-1}} \cdot \left\{ \#\cdot(\Sigma-\#)^{n}\cdot\#,\# \right\}$$

$$\bigcap_{i=0}^{n} \#\cdot(\Sigma-\#)^{n}\cdot \left\{ \Sigma^{i}\cdot\left[\bigcup_{\sigma_1,\sigma_2,\sigma_3 \in \Sigma} \sigma_1\cdot\sigma_2\cdot\sigma_3\cdot \Sigma^{n-1}\cdot f_M(\sigma_1,\sigma_2,\sigma_3)\cdot\Sigma^{n-(i+1)} \right] \cup \Lambda \right\}^{2^{cn-1}} \cdot \left\{ \#\cdot(\Sigma-\#)^{n}\cdot\#,\# \right\} .$$

Clearly using the squaring operator "2" the length of $\beta_x \le c|x|$. Also if a string of length n is accepted by an det. LBA, then the det. LBA makes fewer than 2^{cn} moves for some positive integer c. Thus any valid computation has at most 2^{cn} i.d.'s. The remainder of the proof is identical with that of (1) above.

<u>Theorem 3.7</u>: Let $\mathscr{P}$ be any predicate on the regular sets on $\{0,1\}^*$ s.t.

(1) $\mathscr{P}(\phi)$ is true and

(2) $\mathscr{P}_L = \bigcup_{x\in\Sigma^*} \{x\backslash L \mid \mathscr{P}(L) \text{ is true}\}$

$[\mathscr{P}_R = \bigcup_{x\in\Sigma^*} \{L/x \mid \mathscr{P}(L) \text{ is true}\}] \subsetneq$

regular sets over $\{0,1\}$.

Then $\mathscr{L}^{\mathscr{P}} = \{\beta \mid \beta$ is a $(\cup,\cdot,\cap,*)$-reg. expr. over $\{0,1\}$ and $\mathscr{P}(L(\beta))$ is true$\}$ is tape-hard.

<u>Proof</u>: Standard, see [2] or [3] for details.

§4: Equiv((∪,·,∩,*)) is not Polynomial in Tape.

In this section we show that Equiv((∪,·,∩,*)) is not polynomial in tape. We show that we can in space O(kn) embed all n^k det. (nondet.) tape bounded Turing machine computations. We show that using intersection we can in space O(kn) write a language L_k, such that, roughly $L_k = \Sigma^{n^k}$. We illustrate this idea with several examples.

Example 1: Let $\# \notin \Sigma$. $(\Sigma^n \#) \cap (\Sigma^* \#)^n = (\Sigma^n \#)^n$.
Using ∩, $|(\Sigma^n \#)^* \cap (\Sigma^* \#)^n| = O(n)$, here we mean the length of the string $(\Sigma \cdot \Sigma \cdot \ldots \cdot \Sigma \cdot \#)^* \cap (\Sigma^* \cdot \#) \cdot (\Sigma^* \cdot \#) \cdot \ldots \cdot (\Sigma^* \cdot \#)$.

Example 2: Let $\#, \not{c} \notin \Sigma$. $(\Sigma^n \cdot \{\#, \not{c}\})^* \cap ((\Sigma^* \cdot \#)^n \cdot \Sigma^* \cdot \not{c})^*$ $\cap\ (|\Sigma \cup \{\#\})^* \Sigma^* \not{c})^n = [(\,^n\#)^n \Sigma^n \not{c}]^n$. Using ∩
$|((\Sigma^n \cdot \{\#, \not{c}\})^* \cap (|\Sigma^* \cdot \#)^n \cdot \Sigma^* \cdot \not{c})^* \cap ((\Sigma \cup \{\#\})^* \Sigma^* \cdot \not{c})^n| = O(n)$.

In Lemmas 4.3, 4.4, and 4.5 we define L_k, determine the length of any string in L_k, and determine several of the other properties of L_k. We also show that using intersection in space O(kn) we can define $\{x\$y \mid yx\$ \in L_k\}$, which is essentially the set of cyclic permutations of elements of L_k.

We use essentially the same definitions of i.d.'s, $Next_M(x)$, and $C_M(x)$ as those in Section 3. However, symbols now are triples corresponding to the symbol triples $\sigma_1 \cdot \sigma_2 \cdot \sigma_3$ in Section 3. Thus, every tape square's left and right neighbor is embedded in the symbol denoting it.

Def. 4.1: Let M be a det. Tm with state set S, tape alphabet T, start state q_0 and accepting state q_f. Let $\#_1, \ldots, \#_k \notin T \cup (S \times T)$.

Then

(1) $\Sigma' = T \cup (S \times T) \cup \{\#_1, \ldots, \#_k\}$ and

(2) $\Sigma = [\Sigma' \cup (S \times \{-1,1\}) \times \{\#_1, \ldots, \#_{k-1}\}]^3$.

(First, we could do this directly for nondet. Tm's but this would only complicate the proofs. Second, Σ's seemingly odd definition is due to the fact that the symbols $\{\#_1, \ldots, \#_{k-1}\}$ are only used as hash marks. They are <u>not</u> used as computation tape. Thus, the expression describing the i.d.'s must allow for passing information about change of state and of direction of the move head across them.)

Let $\sigma = (a,b,c) \in \Sigma$. Then $p_1(\sigma) \overset{\text{def.}}{=\!=} a$, $p_2(\sigma) \overset{\text{def.}}{=\!=} b$, and $p_3(\sigma) \overset{\text{def.}}{=\!=} c$.

<u>Def. 4.2</u>: As in Section 3 $C_M(x) = \hat{\#}_k \cdot id_1 \cdot \hat{\#}_k \cdot id_2 \cdot \ldots \cdot \hat{\#}_k \cdot id_n \cdot \hat{\#}_k$ with

(1) each id_j a valid i.d. of the Tm M,

(2) $id_2 = (\widehat{q_0, x_1})\ \hat{x}_2\ \ldots\ \hat{x}_n\ \hat{\#}_1 (\hat{\not{b}}^n \#_1)^{n-2}\ \ldots$,

(3) id_j follows from id_{j-1} by 1 application of a move-rule of M, and

(4) id_n is the first id in $C_M(x)$ in which an accepting state appears.

However, there are several differences.

(1) Each character of $C_M(x)$ is of the form (a,b,c).

(2) If $(a',b'c') \cdot (a,b,c)$ is a proper substring of $C_M(x)$ then $b' = a$.

(3) If $(a,b,c) \cdot (a',b',c')$ is a proper substring of $C_M(x)$ then $c = b'$.

[(2) and (3) guarantee that the left and right contexts are compatible with p_1 and p_3. Thus knowing only (a,b,c) we can calculate the middle coordinate of the corresponding element of the next i.d. (a',b'c'). We cannot, however, tell what a' and c' are given only (a,b,c).]

(4) We allow the Tm head to move left or right over $\#_1,\ldots,\#_{k-1}$'s but not or $\#_k$'s.

[Suppose $(s_1, t_1, s_2, t_2, +1)$ is a move of M, where $s_1, s_2 \in S$ and $t_1, t_2 \in T$. Then the value of $((s_1,t_1),\#_i,c)$ in the next i.d. is $(t_2,(s_{2,1})x\#_i,c)$. The following value is $(t_2,\#,(s_2,c))$.]

Lemma 4.3: Let k be a positive integer $\geq$ 4. We define $\hat{\#}_k \cdot L_k$ to be the intersection of the following:

1) $\hat{\#}_k \cdot [(\Sigma - (\hat{\#}_{k-1},\hat{\#}_k))^* \cdot \hat{\#}_{k-1} \cdot (\Sigma-(\hat{\#}_{k-1},\hat{\#}_k))^*]^{\cdot n-1}\ \hat{\#}_k,$

2) $\hat{\#}_k \cdot [\{(\Sigma(\hat{\#}-(\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k))^* \cdot \hat{\#}_{k-2} \cdot$

$(\Sigma-(\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^*\}^{\cdot n-1}(\hat{\#}_{k-1},\hat{\#}_k)]^+$

3) $\hat{\#}_k \cdot [\{(\Sigma-(\hat{\#}_{k-3}, \hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^* \cdot \hat{\#}_{k-3} \cdot$

$(\Sigma-(\hat{\#}_{k-3}, \hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^*\}^{n-1}(\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k)]^+,$

...

k-1. $\hat{\#}_k\ [\{(\Sigma-(\hat{\#}_1,\hat{\#}_2,\ldots,\hat{\#}_n))^*\ \hat{\#}_1\ (\Sigma-(\hat{\#}_1,\hat{\#}_2,\ldots,\hat{\#}_n))^*\}^{n-1}$
$(\hat{\#}_2,\ldots,\hat{\#}_n)]^+,$

and k $\cdot \#_n \cdot [(\Sigma-(\#_1,\#_2,\ldots,\#_n))^n \cdot (\hat{\#}_1,\ldots,\hat{\#}_k)]^+.$

Then (1) $\hat{\#}_k$ occurs exactly twice;

(2) $\hat{\#}_{k-1}$ occurs exactly n-1 times;

(3) $\hat{\#}_{k-2}$ occurs exactly n(n-1) times;

...

(k) $\hat{\#}_1$ occurs exactly $n^{k-2}(n-1)$ times.

Proof: Clearly (1) and (2) are true.

There are n-1 $\hat{\#}_j$'s for each $\hat{\#}_i$ with $j < i \leq k$ accept for the first $\hat{\#}_k$. Therefore, the number of

$$\hat{\#}_j\text{'s} = (n-1)\cdot[(\sum_{i=j+1}^{k} (\text{number of } \#_i\text{'s}))-1]$$

$$= (n-1)\ [1+(n-1) + n(n-1) + \ldots + n^{k-(j+2)}\ (n-1)].$$

$$= (n-1)\cdot[n + n(n-1) + \ldots + n^{k-(j+2)}\ (n-1)]$$

$$= n\cdot(n-1)\cdot[1 + (n-1)\cdot \sum_{i=0}^{k-(j+3)} n^i]$$

$$= (n-1)\cdot n\cdot[1 + (n-1)\ \frac{[n^{k-(j+2)}]}{(n-1)}\ ]$$

$$= (n-1)\cdot n\cdot n^{k-(j+2)} = (n-1)\cdot n^{k-(j+1)}.$$

We note that only the number of $\hat{\#}_k$'s change if we delete the first occurrence of $\hat{\#}_k$.

Lemma 4.4: $x \in L_k \Rightarrow |x| \geq O(n^k)$.

Proof: For each marker except 1 there are n distinct nonmarkers. Thus the number of nonmarkers = n [1 + (n-1) + n·(n-1) + ... + n

$$= n\cdot[1 + n-1) + n\cdot(n-1) = \ldots + n^{k-2}\cdot(n-1)] =$$

$$n[n+n(n-1)+ \ldots + n^{k-2}(n-1)] = n^2[1 + (n-1)\cdot \sum_{i=0}^{k-3} n^i] = n^2\cdot n^{k-2} = n^k.$$

Lemma 4.5: Let $\tilde{L}_k$ =

1. $[\hat{\Sigma}^{n+1} \cap (\hat{\Sigma}-(\hat{\#}_1,\ldots,\hat{\#}_k))^*\cdot(\hat{\#}_1,\ldots,\hat{\#}_k)\cdot(\hat{\Sigma}-(\hat{\#}_1,\ldots,\hat{\#}_k)|*]^+$

$\cap$

2. $(\Sigma-(\hat{\#}_1,\ldots,\hat{\#}_k))^*\cdot(\hat{\#}_1,\ldots,\hat{\#}_k)\cdot[(\hat{\Sigma}-(\hat{\#}_1,\ldots,\hat{\#}_k))^n\cdot(\hat{\#}_1,\ldots,\hat{\#}_k)]^*\cdot(\hat{\Sigma}-(\hat{\#}_1,\ldots,\hat{\#}_k))^*$

$\cap$

3. $(\hat{\Sigma} - \hat{\#}_k)^* \cdot \hat{\#}_k \cdot (\hat{\Sigma} - \hat{\#}_k)^*$

$\cap$

4. $[(\hat{\Sigma} - \hat{\#}_{k-1})^* \cdot \hat{\#}_{k-1} \cdot (\hat{\Sigma} - \hat{\#}_{k-1})^*]^{n-1}$

$\cap$

5.1 $(\hat{\Sigma} - (\hat{\#}_{k-1}, \hat{\#}_k))^* \cdot (\hat{\#}_{k-1}, \hat{\#}_k) \cdot [\{(\hat{\Sigma} - (\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^* \cdot$
$\hat{\#}_{k-2} \cdot (\hat{\Sigma} - (\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^*\}^{n-1} \cdot (\hat{\#}_{k-1}, \hat{\#}_k)]^* \cdot$
$(\hat{\Sigma} - (\hat{\#}_{k-1}, \hat{\#}_k))^*$

6.1: $[(\Sigma - (\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^* \cdot (\hat{\#}_{k-2}, \Lambda) \cdot (\hat{\Sigma} - (\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^*]^{n-1} \cdot$
$(\hat{\#}_{k-1}, \hat{\#}_k) \cdot \hat{\Sigma}^* \cdot (\hat{\#}_{k-1}, \#_k) \cdot$
$[(\hat{\Sigma} - (\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k))^* \cdot (\hat{\#}_{k-2}, \Lambda) \cdot (\hat{\Sigma} \cdot (\hat{\#}_{k-2} \hat{\#}_{k-1}, \#_k))^*]^{n-1}$,

7.1: $[[(\hat{\Sigma} - \hat{\#}_{k-2})^* \cdot \#_{k-2} \cdot (\hat{\Sigma} - \hat{\#}_{k-2})^*]^n]^*$,

...

5,k-2: $(\hat{\Sigma} - (\hat{\#}_2, \ldots, \hat{\#}_k))^* \cdot (\hat{\#}_2, \ldots, \hat{\#}_k) \cdot [\{(\hat{\Sigma} - (\hat{\#}_1, \ldots, \hat{\#}_k))^* \cdot \hat{\#}_1 \cdot$
$(\hat{\Sigma} - (\hat{\#}_1, \ldots, \hat{\#}_k))^*\}^{n-1} \cdot (\hat{\#}_2, \ldots, \hat{\#}_k)]^* \cdot (\Sigma - (\hat{\#}_2, \ldots, \hat{\#}_k))^*$,

6,k-2: $[(\hat{\Sigma} - (\hat{x}_1, \ldots, \hat{x}_k))^* \cdot (\hat{\#}_1, \Lambda) \cdot (\hat{\Sigma} - (\hat{x}_1, \ldots, \hat{x}_k))^*]^{n-1} \cdot$
$(\hat{x}_2, \ldots, \hat{x}_k) \cdot \hat{\Sigma}^* \cdot (\hat{x}_2, \ldots, \hat{x}_k) \cdot$
$[(\hat{\Sigma} - (\hat{\#}_1, \ldots, \hat{x}_k))^* \cdot (\hat{\#}_1, \Lambda) \cdot (\hat{\Sigma} - (\hat{\#}_1, \ldots, \hat{\#}_k))^*]^{n-1}$, and

7,k-2: $[[(\hat{\Sigma} - (\hat{\#}_1, \ldots, \hat{\#}_k))^* \cdot \hat{\#}_1 \cdot (\hat{\Sigma} - (\hat{\#}_1, \ldots, \hat{\#}_k))^*]^n]^*$.

Then $\tilde{L}_k = \{x \$ y \mid yx\$ \in L_k\}$.

<u>Proof</u>: (a) $\tilde{L}_k \subset \{x\$y \mid xy\$ \in L_k\}$.

From 1 and 2 $\lambda \in \tilde{L}_k \Rightarrow \lambda$ is of the form

$$\underbrace{[\,\alpha_1 \ X \ \beta_1\,]}_{n+1}\underbrace{[\,\alpha_2 \ X \ \beta_2\,]}_{n+1}\underbrace{[\,\alpha_3 \ X \ \beta_3\,]}_{n+1} \cdots \underbrace{[\,\alpha_k \ X \ \beta_k\,]}_{n+1},$$

with $|\leftarrow n \rightarrow|$ spanning $\beta_1\,\alpha_2$, $|\leftarrow n \rightarrow|$ spanning $\beta_2\,\alpha_3$, …, $|\leftarrow n \rightarrow|$ under $[\,\alpha_k$,

where $|\alpha_1| = |\alpha_2| = \ldots = |\alpha_k|$ and $|\beta_1| = |\beta_2| = \ldots = |\beta_k|$ and $|\beta_i| + |\alpha_{i+1}| = n$.

From 3 $\lambda \in \tilde{L}_k$ =>exactly 1 $\hat{\#}_k$ occurs in λ.

From 4 $\lambda \in \tilde{L}_k$ =>exactly n-1 $\hat{\#}_{k-1}$'s occur in λ.

From 3,4,5.1,6.1, and 7.1 exactly n(n-1) $\hat{\#}_{k-2}$'= occur in λ. Furthermore, (a) between any two consecutive $\hat{\#}_k$'s and $\hat{\#}_{k-1}$'s exactly n-1 $\hat{\#}_{k-2}$'s occur and (b) the sum of the number of $\hat{\#}_{k-2}$'s occurring before the first $\hat{\#}_k$ or $\hat{\#}_{k-1}$ and after the last $\hat{\#}_k$ or $\hat{\#}_{k-1}$ equals n-1.

$$\lambda = (\ldots \hat{\#}_{k-2} \ldots)^i \, [\hat{\#}_{k_1} \; \hat{\#}_{k-1}]_f \ldots [\hat{\#}_{k_1} \; \hat{\#}_{k-1}]_L (\ldots \hat{\#}_{k-2} \ldots)$$

no $\hat{\#}_k$' or $\hat{\#}_{k-1}$' — first occurrence in λ of either a $\hat{\#}_{k-1}$ a $\hat{\#}_n$ — last occurrence in $\hat{\lambda}$ of either a $\hat{\#}_{k-1}$ in $\hat{\#}_k$ — no $\hat{\#}_k$'s a $\hat{\#}_{k-1}$'s.

(a) follows immediately from 5.1. From 7.1 we know that the number of $\#_{k-2}$'s occurring in λ is divisible by n. Thus, if there are too few $\#_{k-2}$'s in λ there are at least n too few. Similarly, if there are too many $\#_{k-2}$'s in λ, there are at least n too many. We know that between $\begin{bmatrix}\#_k \\ \#_{k-1}\end{bmatrix}_f$ and $\begin{bmatrix}\#_k \\ \#_{k-1}\end{bmatrix}_L$ there are exactly (n-1)·(n-1) distinct $\#_{k-2}$'s. Hence there are not at least n too few $\#_{k-2}$'s. Suppose there are too many $\#_{k-2}$'s outside $\begin{bmatrix}\#_k \\ \#_{k-1}\end{bmatrix}_f \ldots \begin{bmatrix}\#_k \\ \#_{k-1}\end{bmatrix}_L$, then there are at least n too many. But from 6.1

$i+j \leq 2(n-1) = 2n-2$. But $n+(n-1) = 2n-1 \gneq 2n-2$.

Hence, if $\lambda \in \tilde{L}_k$ and $\lambda \not\in \{x \$ y | y x \$ \in L_k\}$, the error is not in the distribution or number of the $\#_{k-2}$'s.

[We note that in proving that we have the right number and distribution of $\hat{\#}_{k-2}$'s we used only the facts that

(1) we knew that λ had the right number of higher level markers;

(2) between any 2 consecutive higher level markers, we have exactly n-1 $\hat{\#}_{k-2}$'s;

(3) the number of $\hat{\#}_{k-2}$'s is divisible by n; and

(4) the number of $\hat{\#}_{k-2}$'s appearing before $[\hat{\#}_k,\hat{\#}_{k-1}]_f \leq$ n-1 and the number of $\hat{\#}_{k-2}$'s appearing after $[\hat{\#}_k,\hat{\#}_{k-1}]_L \leq n-1$.]

Assume that for all $i > j$ that the $\hat{\#}_i$'s appearing in λ have the right distribution and number. By 5, j-2 between any two consecutive markers $\hat{\#}_{i_1},\hat{\#}_{i_2}$ with $i_1,i_2 > j$, there are exactly (n-1) $\hat{\#}_j$'s. Also there are no more than n-1 $\hat{\#}_j$'s before $[\hat{\#}_k,\ldots,\hat{\#}_{j+1}]_f$, and there are no more than n-1 $\hat{\#}_j$'s after $[\hat{\#}_k,\ldots,\hat{\#}_{j+1}]_L$. By 6, j-2 the number of $\hat{\#}_j$'s appearing in λ is divisible by n. Hence, the number of $\hat{\#}_j$'s occurring between $[\hat{\#}_k,\ldots,\hat{\#}_{j+1}]_f$ and $[\hat{\#}_k,\ldots,\hat{\#}_{j+1}]_L = (n-1)\cdot[(n-1)\cdot \sum_{i=0}^{k-(j+2)} n^i] =$ $(n-1)\ [(n^{k-(j+1)}-1)]$, which is <u>exactly</u> n-1 too few. Hence there are not n too few occurrences of $\hat{\#}_j$ in λ. If there are too many $\hat{\#}_j$'s in λ there are at least n too many. But this implies $i + j \geq n+n-1 = 2n-1$. But $i+j \lneq 2(n-1) = 2n-2$. Hence both the number and distribution of the $\hat{\#}_j$'s appearing in λ is correct. Thus, if $\lambda \in \tilde{L}_k$ and $\lambda \in \{x\ \$\ y \mid y\ x\ \$ \in L_k\}$, it is not because of the number or distribution of any of the markers appearing in

λ. Finally, the number and position of the nonmarkers is correct from 1 and 2.

$\lambda \in \tilde{L}_k \Rightarrow \lambda \in \{x \$ y | y x \$ \in L_k\}$.

(b) $\tilde{L}_k \supset \{x \$ y | x y \$ \in L_k\}$.

From the definition of L_k, $\lambda \in \{x \$ y | y x \$ \in L_k\} \Rightarrow \lambda$ satisfies 1, 2, 3, 4, 5.1, 5.2,..., 5. k-2, and 6.1, 6.2,...., 6.k-2. From Lemma 4.1 λ satisfies 7.1, 7.2,...,7.k-2.

[$\lambda \in \{x \$ y | x y \$ \in L_k\} \Rightarrow \lambda$ is of the form

$(\text{———} \ \hat{\#}_j \ \text{———})^i \ [\hat{\#}_k, \ldots, \hat{\#}_{j+1}]_f \ \ldots \ [\hat{\#}_k, \ldots, \hat{\#}_{j+1}]_L$

$(\text{———} \ \hat{\#}^j \ \text{———})^i$, where $i+j = n-1$,]

<u>Theorem 4.5</u>: For all positive integers k and $\epsilon > 0$, Equiv($(\cup, \cdot, \cap, *)$) requires tape at least $O(n^{k-\epsilon})$.

<u>Proof</u>: $\lambda \in \Sigma^*$ is not a valid computation iff

(i) $\lambda \notin \hat{\#}_k L_k^*$, or

(ii) (a',b'c') (a,b,c) is a proper substring of λ and b' $\neq$ a, or

(iii) (a,b,c)(a',b',c') is a proper substring of λ and c $\neq$ b', or

(iv) λ doesnot start with the right initial i.d., or

(v) λ doesnot have an accepting state, or

(vi) λ makes an error between one i.d. and the next i.d.

We can trivially write out a $(\cup, \cdot, \cap, *)$ - regular expression for (ii),(iii), and (v).

If $\lambda \notin \hat{\#}_k L_k^*$ then

(1.0) $\lambda = \{\Lambda\}$ or $\hat{\#}_k$, or

(1.1) λ doesnot begin with a $\hat{\#}_k$ or end with a $\hat{\#}_k$, or

(1.2) between some 2 consecutive $\hat{\#}_k$'s there are fewer or more than n-1 $\hat{\#}_{k-1}$'s, or

(1.3) between 2 consecutive $\hat{\#}_k$'s or $\hat{\#}_{k-1}$'s there are fewer than or more than n-1 $\hat{\#}_{k-2}$'s, a ...

(1.k) between 2 consecutive $\hat{\#}_2$'s,...,$\hat{\#}_k$'s there are fewer than or more than n-1 $\hat{\#}_1$'s, or

(1.k+1) between 2 consecutive $\hat{\#}_1$'s,...,$\hat{\#}_k$'s there are fewer than or more than n nonmarkers.

$$\beta_i = \{\Lambda\} \cup \{\hat{\#}_k\} \quad \} \qquad \cup \qquad 1.0$$

$$(\Sigma-\{\hat{\#}_k\} \cdot \Sigma^* \cup \Sigma^* \cdot \Sigma-\{\hat{\#}_k\}) \quad \} \qquad 1.1$$

$$\cup$$

$$\left.\begin{array}{l} \Sigma^*\cdot\#_k\cdot[(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1}\})^* \{\hat{\#}_{k-1},\Lambda\}\cdot(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1}\})^*]^{n-2}\cdot\hat{\#}_k\cdot\Sigma^* \\ \cup \\ \Sigma^*\cdot[(\Sigma-\{\hat{\#}_k\})^*\cdot\hat{\#}_{k-1}(\Sigma-\{\hat{\#}_k\})^*]^n\cdot\hat{\#}_k\cdot\Sigma^* \\ \cup \\ \Sigma^*\cdot\#_k\ [(\Sigma-\{\hat{\#}_k\})^*\cdot\hat{\#}_{k-1}\cdot(\Sigma-\{\hat{\#}_k\})^*]^n\cdot\Sigma^* \end{array}\right\} 1.2$$

$$\cup$$

$$\left.\begin{array}{l} \Sigma^*\cdot\{\hat{\#}_k,\hat{\#}_{k-1}\}\cdot[(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1},\hat{\#}_{k-2}\})^*\cdot\{\hat{\#}_{k-2},\Lambda\}\cdot(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1},\hat{\#}_{k-2}\})^*]\cdot \\ \qquad \{\hat{\#}_k,\hat{\#}_{k-1}\}\cdot\Sigma^* \\ \qquad\qquad \cup \\ \Sigma^*\cdot[(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1}\})^*\cdot\hat{\#}_{k-2}\cdot(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1}\})^*]^n\cdot\{\hat{\#}_k,\hat{\#}_{k-1}\}\cdot\Sigma^* \quad \cup \\ \Sigma^*\cdot\{\hat{\#}_k,\hat{\#}_{k-1}\}\cdot[(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1}\})^*\cdot\hat{\#}_{k-2}\cdot(\Sigma-\{\hat{\#}_k,\hat{\#}_{k-1}\})^*]^n\cdot\Sigma^* \end{array}\right\} 1.3$$

$$\cup \ \ldots$$

$$\left.\begin{array}{l} \Sigma^*\cdot\{\hat{\#}_k,\ldots\hat{\#}_1\}\cdot[\Sigma-\{\hat{\#}_k,\ldots,\hat{\#}_1\} \cup \{\Lambda\}]^{n-1}\cdot\{\hat{\#}_k,\ldots,\hat{\#}_1\}\cdot\Sigma^* \\ \qquad \cup \\ \Sigma^*\cdot \{\hat{\#}_k,\ldots,\hat{\#}_1\}\cdot[\Sigma-\{\hat{\#}_k,\ldots,\hat{\#}_1\}]^{n+1}\cdot\Sigma^* \\ \qquad \cup \\ \Sigma^*\cdot [\Sigma-\{\hat{\#}_k,\ldots,\hat{\#}_1\}]^{n+1}\cdot\{\hat{\#}1,\ldots,\hat{\#}1\}\cdot\Sigma^* \end{array}\right\} 1.k$$

λ doesnot begin with the right i.d.

$$\beta_{iv} = \hat{\#}_k \cdot [(\Sigma - \{\hat{x}_1\}) \cup \hat{x}_1 \cdot [(\Sigma - \{x_2\}) \cup \ldots \cup \hat{x}_n [(\Sigma - \{\hat{\#}_1\})] \ldots]] \cdot \Sigma^*$$

$$\cup$$

$$\hat{\#}_k \cdot (\Sigma - \{\hat{\#}_k, \ldots, \hat{\#}_1\})^n \cdot \hat{\#}_1 (\Sigma - \{\hat{\#}_n\})^* \cdot (\Sigma - \{\hat{\#}_k, \ldots, \hat{\#}_1, \not{b}\}) \cdot$$
$$(\Sigma - \{\hat{\#}_k\})^* \cdot \hat{\#}_k \cdot \Sigma^*$$

Since we know that all strings with improper marker-non-marker structure are covered by β_1, the only possibilities for an invalid initial i.d. that β_{iv} need cover are

(1) λ doesnot begin with $\hat{\#}_k \hat{x}_1 \ldots \hat{x}_n \hat{\#}_1 \ldots$ and

(ii) there is a nonmarker nonblank to the right of the first $\hat{\#}_1$ and to the left of the second $\hat{\#}_k$ in λ.

Finally,

$$\beta_{vl} = \bigcup_{a \notin \Sigma} [\Sigma^* \cdot \hat{a} \cdot \tilde{L}_k \cdot [\bigcup_{pr_1 \cdot (\hat{b}) \neq f_M(a)} \hat{b}] \cdot \Sigma^*] .$$

If we choose that string in L_k whose marker - nonmarker structure alligns with that of $\Sigma^* \cdot \hat{a}$ above, then we find any errors that occur. If we choose some string $\beta \in \tilde{L}_k$ s.t. the marker - nonmarker structure of β doesnot match up with that of $\Sigma^* \cdot \hat{a}$ above, then for all $\lambda \in \Sigma^* \cdot \hat{a}\ \beta \cdot \cup \hat{b}\ \ \Sigma^*$, λ is not a valid i.d.

<u>Theorem 4.7</u>: Let $\mathcal{P}$ be any predicate on the regular sets on $\{0,1\}^*$ s.t.

(1) $\mathcal{P}(\{0,1\}^*)$ is true and

(2) $\mathcal{P}_L = \bigcup_{x \in \Sigma^*} \{x \backslash L \mid \mathcal{P}(L) \text{ is true}\} \subsetneq$ regular sets over $\{0,1\}$ or

$\mathcal{P}_R = \bigcup_{x \in \Sigma^*} \{L / x \mid \mathcal{P}(L) \text{ is true}\} \subsetneq$ regular sets over $\{0,1\}$.

Then = $\{\beta | \beta$ is a $(\cup,\cdot,\cap,*)$-regular expression over $\{0,1\}$ and $(L(\beta))$ is true$\}$ is not polynomial in tape.

Proof: Standard, see [2] or [3] for details.

We note that by bounding the depth of intersections that may appear in a $(\cup,\cdot,\cap,*)$ - regular expression we get a polynomial tape heirarchy of problems of arbitrary polynomial tape complexity. Let $(\cup,\cdot,*)_k$ - regular expressions be the set of all r.e.i.'s of the form $\bigcup_{i=1}^{m} R_i$, where each R_i is a r.e.i. with k or fewer occurrences of intersection.

Theorem 4.8: For all nonnegative integers i and for all $\epsilon > 0$, Inequiv$((\cup,\cdot,x)_{2i+1})$ requires nondet. tape $O(n^{i-\epsilon})$.

Proof: We may rewrite $\tilde{L}_i$ in such a way that the number of intersections used is 2i+1. We do this by combining 1 and 3, 5.1 and 6.1, 5.2 and 6.2, etc.

1 and 3 combined is

$$[(\hat{\Sigma}-\{\hat{\#}_1,\ldots,\hat{\#}_k\})^*\cdot\{\hat{\#}_1,\ldots,\hat{\#}_{k-1}\}\cdot(\hat{\Sigma}- \hat{\#}_1,\ldots,\hat{\#}_k\})^*$$
$$\cap\ \hat{\Sigma}^{n+1}]^*\cdot[(\hat{\Sigma}-\{\hat{\#}_1,\ldots,\hat{\#}_k\})^*\cdot\hat{\#}_k\cdot(\hat{\Sigma}-\{\hat{\#}_1,\ldots,\hat{\#}_k\})^*$$
$$\cap\ \hat{\Sigma}^{n+1}]\cdot[(\hat{\Sigma}-\{\hat{\#}_1,\ldots,\hat{\#}_k\})^*\cdot\hat{\#}_k\cdot(\hat{\Sigma}-\{\hat{\#}_1,\ldots,\hat{\#}_k\})^*$$
$$\cap\ \hat{\Sigma}^{n+1}]^*.$$

5.1 and 6.1 are combined together as

$$[(\hat{\Sigma}-\{\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k\})^*\cdot\{\hat{\#}_{k-2},\Lambda\}\cdot(\hat{\Sigma}-\{\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k\})^*]^{n-1}\cdot(\hat{\#}_k,\hat{\#}_{k-1})$$
$$[\{(\hat{\Sigma}-\{\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k\})^*\cdot\hat{\#}_{k-2}\cdot(\hat{\Sigma}-\{\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k\})^*\}^{n-1}\cdot(\hat{\#}_k,\hat{\#}_{k-1})]^*\cdot$$
$$[\{(\hat{\Sigma}-\{\hat{\#}_{k-2}, \hat{\#}_{k-1}, \hat{\#}_k\})^*\cdot\{\hat{\#}_{k-2},\Lambda\}\cdot(\hat{\Sigma}- (\hat{\#}_{k-2},\hat{\#}_{k-1},\hat{\#}_k)^*]^{n-1}.$$

The above construction works for 5.2 and 6.2 , etc.

Thus for each $i \geq 2$ $\tilde{L}_{i+1}$ has 2 more occurrences of $\cap$ then $\tilde{L}_i$. $\tilde{L}_2$ has exactly 5 occurrences of $\cap$, and $\tilde{L}_0$ need have none.

[We note that if we allow $\tilde{L}_k$ to be $O(kn^2)$, then we can write $\tilde{L}_{i+1}$ in such a way that it requires only 1 more $\cap$ then $\tilde{L}_i$.]

But β (of Theorem 4.6) $= R_0 \cup \bigcup R_j$, where R_0 contains no $\cap$'s and the number (depth) of $\cap$'s in each of the remaining R_j's is $2i+1$.

In [5] Meyer and Stockmeyer state the following theorem:

Theorem 4.9: Let L be any language accepted by some non-deterministic time c^n Turing machine for some constant $c > 1$. Then $L \underset{\text{Ptime}}{\leq} \text{Inequiv}((\cup,\cdot,2))$.

Furthermore, there is a $d > 1$ such that $\text{Inequiv}((\cup,\cdot,2))$ is recognized by a nondeterministic time d^n machine.

Proof: We show how to simulate all the invalid computations for all 2^n det. time bounded Turing machines using only $\cup,\cdot$, and ^{2}. A modified version of the same construction will simulate all 2^n nondet. time bounded Tms.

We note that any valid computation is of the form $\#[(\Sigma-\#)^{2^n}\#]^k$, where $k \leq 2^n$. Thus any valid computation is an element of $\#\cdot[(\Sigma-\#)^{2^n}\# \cup \{\Lambda\}]^{2^n}$. (In the nondeterministic case for any valid computation having more than 2^n i.d.s there is a valid computation having less than or equal to 2^n i.d.'s, i.e., we can delete loops.) Thus, the maximum length for a valid computation is $1 + (2^n+1)\cdot 2^n = 2^{2n} + 2^n + 1$. We also demand that a valid computation have at least 1 i.d. Thus all strings of length $\leq 2^n+1$ are invalid computations. Let β_x equal the union of the following:

(1) all strings of length $\leq 2^n+1$;

(2) all strings with the wring initial configuration;

(3) all strings that contain no final state;

(4) all strings that do not end in a "#";

(5) all strings that contain an error; and

(6) all strings that are too long.

[We write (5) as $\bigcup_{\sigma_1,\sigma_2,\sigma_3 \in \Sigma} (\Sigma \cup \{\Lambda\})^{2^{2n}} \sigma_1\sigma_2\sigma_3\Sigma^{2^n-1}$.

$$(\Sigma - f_M(\sigma_1,\sigma_2,\sigma_3)) \cdot (\Sigma \cup \{\Lambda\})^{2^{2n}}.$$

Any string in (5) is of length $\leqq 2^{2n} + 3 + 2^n - 1 + 1 + 2^{2n} = 2^{2n+1} + 2^n + 3$. We define β_x so that $\lambda \in L(\beta_x) \Rightarrow |\lambda| \leq 2^{2n+1} + 2^n + 3$

(1)

$$\beta_1 = (\Sigma \cup \{\Lambda\})^{2^n+1}$$

(2)

$$\beta_2 = ((\Sigma-\#) \cup \#\cdot((\Sigma-(q_0,x_1)) \cup (q_0,x_1)\cdot((\Sigma-x_2) \cup x_2\cdot((\Sigma-x_3) \cup \ldots \cdot(\Sigma-x_n)))\ldots)\cdot[\Sigma \cup \{\Lambda\}]^{2^{2n}+2^n}$$

$$\cup$$

$$\Sigma^{n+1} \cdot (\not{b} \cup \{\Lambda\})^{2^n-n-1}\cdot\#\cdot[\Sigma \cup \{\Lambda\}]^{2^{2n}}$$

$$\cup$$

$$\Sigma^{n+1}\, b\!\!/^{\,2^n-n+1}\cdot[\Sigma \cup \{\Lambda\}]^{2^{2n}}$$

$$\cup$$

$$\Sigma^{2^n+1}\cdot(\Sigma-\#)\cdot[\Sigma \cup \{\Lambda\}]^{2^{2n}-1}$$

(3)

$$\beta_3 = [(\Sigma - \bigcup_{t \in T} (q_0,t)) \cup \{\Lambda\}]^{2^{2n}} + 2^n+1$$

(4)

$$\beta_4 = [\Sigma \cup \{\Lambda\}]^{2^{2n}+2^n} \cdot (\Sigma - \#)$$

(5)

$$\beta_5 = \bigcup_{\sigma_1,\sigma_2,\sigma_3 \in \Sigma} (\Sigma \cup \{\Lambda\})^{2^{2n}} \sigma_1 \sigma_2 \sigma_3 \Sigma^{2^n - 1} \cdot (\Sigma - N(\sigma_1,\sigma_2,\sigma_3)) \cdot (\Sigma \cup \{\Lambda\}^{2^{2n}}$$

(6)

$$\beta_6 = \Sigma^{2^{2n} + 2^n + 2} \cdot (\Sigma \cup \{\Lambda\})^{2^n + 1}.$$

Using 2 the lengths of $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$, and $\beta_6 \leq 5n$ + constant.

Since 2 allows at most logarithmic shrinkings, given a $(\cup, \cdot, {}^2)$-regular expression we can find equivalent $(\cup, \cdot)$-regular expression β' s.t. $|\beta|' \leq 2^{|\beta|}$ by replacing α^2 by $\alpha.\alpha$. Since Inequiv$((\cup,\cdot)) \in$ NPTIME, Inequiv$((\cup,\cdot,2)) \in$ ndtime (2^{cn}) for some $c > 1$.

We can use the proof of Theorem 4.7 to prove the following extended theorem.

<u>Theorem 4.10</u>: Let Inequiv$((\cup,\cdot,op_1,\ldots,op_k))$ be p-complete. Then

(1) $\forall c > 1$, Inequiv$((\cup,\cdot,op_1,\ldots,op_k,{}^2)) \underset{\text{Ptime}}{\geq}$ ndtime(2^{cn});

(2) Inequiv$((\cup,\cdot,op_1,\ldots,op_k,{}^2)) \in$ ndtime (d^n) for some $d > 1$, and

(3) Inequiv$((\cup,\cdot,op_1,\ldots,op_k,{}^2)) \in$ PTAPE iff PTAPE $= \bigcup_{\ell} \bigcup_{c}$ ndtime $(2^{cn^{\ell}})$.

Proof: (1) and (2) follow from the proof of 4.9.

(3) follows since in space $0(cn^{\ell})$ we can describe arbitrary $2^{cn^{\ell}}$ nondet time bounded computations.

Corollary 4.11:

(1) For all positive integers $c \geq 1$, $\text{Inequiv}((\cup,\cdot,\cap,{}^2)) \underset{\text{Ptime}}{\geq}$ ndtime (2^{cn}).

(2) $\text{Inequiv}((\cup,\cdot,\cap,{}^2)) \in$ ndtime (d^n) for some $d > 1$.

(3) $\text{Inequiv}((\cup,\cdot, op_1,\ldots,op_k,{}^2)) \in$ PTAPE iff PTAPE $= \bigcup_{\ell} \bigcup_{c}$ ndtime $(2^{cn^{\ell}})$.

Proof: Immediate from Theorems 2.3 and 4.8.

Finally we observe the following:

(1) $(\cup,\cdot,\cap,\text{rev},{}^2)$ and $(\cup,\cdot,\gamma,{}^2)$ satisfy the conditions of Theorem 4.8 and

(2) the predicates in (1), (2), (3), (4), and (6) in Theorem 2.6 could be substituted for Inequiv in Theorem 4.8.

BIBLIOGRAPHY

[1] Cook, S. A., "The Complexity of Theorem-proving Procedures", Proceedings of the Third ACM Symposium on the Theory of Computing (1971), pp. 151-158.

[2] Hunt, H.B. III, "On the Time and Tape Complexity of Languages, I", Cornell University, Department of Computer Science, Technical Report 73-156.

[3] Hunt, H.B. III, "On the Time and Tape Complexity of Languages I", [to appear in the Proceedings of the Fifth Annual ACM Symposium on the Theory of Computing.]

[4] Meyer, A. R. and Stockmeyer, L. J., "The Equivalence Problem for Regular expressions with Squaring Requires Exponential Space", Proceedings of the Thirteen Annual Symposium on Switching and Automata Theory (1972), pp. 125-129.

[5] Meyer, A. R. and Stockmeyer, L. J., Extended Abstract Word Problems Requring Exponential Time", [to appear in the Proceedings of the Annual ACM Symposium on the Theory of Computing.]